Міністерство освіти і науки України
Державний університет інформаційно-комунікаційних технологій

Кисіль Т.М., Зінченко О.В., Чичкарьов Є.А., Фесенко М.А.

АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ

ЧАСТИНА 1

Методичні рекомендації до виконання практичних завдань для здобувачів ступеня бакалавра освітньої програми «Штучний інтелект» за спеціальністю 122 «Комп'ютерні науки»

Київ - 2023

Міністерство освіти і науки України

Державний університет інформаційно-комунікаційних технологій

Кисіль Т.М., Зінченко О.В., Чичкарьов Є.А., Фесенко М.А.

АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ

ЧАСТИНА 1

Методичні рекомендації до виконання практичних завдань для здобувачів ступеня бакалавра освітньої програми «Штучний інтелект» за спеціальністю 122 «Комп'ютерні науки»

Київ – 2023

УДК 004.43

Рецензенти: **Гайдур Г.І.**, доктор технічних наук, професор, завідувач кафедри Інформаційної та кібернетичної безпеки Державного університету інформаційно-комунікаційних технологій.
Іларіонов О.Є, кандидат технічних наук, доцент, завідувач кафедри інтелектуальних технологій Київського національного університету імені Тараса Шевченка.

Кисіль Т.М., Зінченко О.В., Чичкарьов Є.А., Фесенко М.А.
Алгоритмізація та програмування: Частина 1. – Методичні рекомендації до виконання практичних завдань для здобувачів ступеня бакалавра освітньої програми «Штучний інтелект» за спеціальністю 122 «Комп'ютерні науки». – К: ДУІКТ, 2023. – 250 с.

Методичні рекомендації містять практичні завдання по алгоритмізації та програмуванню: завдання та приклади їх виконання, тексти лабораторних та практичних робіт (контрольні питання, завдання індивідуального виконання). Методичні рекомендації використовуються на заняттях дисципліни «Алгоритмізація та програмування» для студентів спеціальності 122 «Комп'ютерні науки» освітньої програми «Штучний інтелект»

Рекомендовано на засіданні вченої ради Навчально-наукового інституту інформаційних технологій (Протокол № 2 від 13.09.2023 року)

ЗМІСТ

ПЕРЕДМОВА

«Алгоритмізація та програмування» є ключовою дисципліною в комп'ютерних науках, яка забезпечує основними знаннями та навичками студентів у вирішенні прикладних задач та створенні програм, що базуються на алгоритмах та структурах даних. Дисципліна включає не лише технічні навички, а й розвиває абстрактне мислення, креативність та уміння вирішувати проблемні завдання. Дана дисципліна створює міцний фундамент для подальшого росту студентів в сфері комп'ютерних наук та програмування. Навчальна дисципліна "Алгоритмізація та програмування" викладається студентам першого року підготовки бакалаврського рівня вищої освіти за освітньою програмою «Штучний інтелект» спеціальності "Комп'ютерні науки".

Методичні рекомендації «Алгоритмізація та програмування» призначено для проведення лабораторних та практичних занять в середовищах програмування C++ та Python. Методичні рекомендації містять: теоретичний матеріал з основ алгоритмізації; алгоритми та задачі на мовах програмування C++, Python; приклади програм з детальними коментарями, а також завдання до самостійного виконання; основні аспекти та техніки програмування, що полегшує засвоєння матеріалу; індивідуальні задання, спрямовані на самостійне опрацювання та закріплення матеріалу.

До методичних рекомендацій включено практичні завдання першого навчального модуля дисципліни «Алгоритмізація та програмування», кожне з них супроводжується необхідним для виконання теоретичним матеріалом та типовими прикладами відповідної теми, переліком індивідуальних та контрольних запитань. Основні завдання циклу практичних занять полягають в тому, щоб студенти закріпили матеріал лекцій і отримали практичні навички розробки алгоритмів і проектування програм з використанням сучасних технологій структурного програмування.

Робочою програмою курсу "Алгоритмізація та програмування" передбачено два види практичних завдань:

— завдання, на яких розглядаються та аналізуються типові алгоритми;
— завдання, на яких студенти виконують роботи комп'ютерного практикуму за варіантами, пов'язані з програмуванням і налагодженням програм у відповідних програмних середовищах.

Методичні рекомендації «Алгоритмізація та програмування» базуються на курсі лекцій та інструкцій до виконання лабораторних робіт за відповідною дисципліною, яка викладається на кафедрі штучного інтелекту «Державного університету інформаційно-комунікаційних технологій». Надані методичні рекомендації можна використовувати для самостійного вивчення дисципліни.

Дисципліна "Алгоритмізація та програмування" базується на знаннях, отриманих студентами при вивченні таких дисциплін, як «Прикладне програмування–JAVA» та «Дискретні структури». Компетенції, отримані студентами в процесі вивчення даної дисципліни застосовуються, в подальшому, в навчальних дисциплінах: «Прикладне програмування Python», «Об'єктно-орієнтоване програмування» та «Мови програмування інтелектуальних систем».

Практичне завдання 1.
Лінійні та розгалужені алгоритми мовами програмування C++, Python

Мета роботи: набути практичних навичок розробки базових алгоритмів мовами програмування **C++**, **Python;** створювати консольні проекти; редагувати та компілювати програмні коди; виявляти синтаксичні та логічні помилки на етапі трансляції програм; придбати практичні навички з проектами по виконанню лінійних, розгалужених, комбінованих алгоритмів; реалізовувати алгоритми графічно в онлайн-сервісах **LucidChart**, **Draw.io**.

Завдання 1. Розробити програму мовою **C++** та/або **Python** лінійної структури відповідно до варіанту (***табл. 1***), в якій необхідно забезпечити:

— виведення на екран заданої за варіантом функції;
— виведення на екран значень параметра ***a*** і змінної ***x***;
— заданим змінним послідовно присвоїти значення одного типу даних, різних типів, випадковими величинами;
— вивести розраховані значення функцій у відповідному форматі;

Реалізовану програму **C++** та/або **Python** протестувати з вхідними даними різного типу: цілого та дійсного. Передбачити в реалізованому алгоритмі відповідні коментарі.

Таблиця 1. – Лінійні рівняння функції

№ вар.	*Функція для розрахунку*	*№ вар.*	*Функція для розрахунку*
1.	$y = \frac{2x-a}{3+\sqrt{x-1}} - 3\cos\lvert x\rvert$	**2.**	$y = \frac{\sqrt{\lg x+5}}{3x-a^2} + \lvert x+5\rvert$
3.	$y = \sqrt{\frac{x+2}{2-ax}} + 4\lvert log_2 x - 1\rvert$	**4.**	$y = \frac{5x-a}{1-\sqrt{x+1}} - ax^4$
5.	$y = \frac{2x+a}{\sqrt{x}-3} - log_3\lvert x-4\rvert$	**6.**	$y = \frac{7x-a}{3+log_2 x} - \sqrt{\lvert x+3\rvert}$
7.	$y = \frac{\sqrt{\log x+2}}{4x-a} + \sin\lvert x+4\rvert$	**8.**	$y = \frac{\sqrt{\lg(x+1)}}{5x-a} + 2ax$
9.	$y = \sqrt{\frac{x+a}{5x-3}} + \lvert\sin x\rvert$	**10.**	$y = \sqrt{\frac{5x-a}{x^2-1}} + \cos\lvert x-1\rvert$
11.	$y = \frac{\sqrt{\ln x+5}}{3x+a} + log_3\lvert x+5\rvert$	**12.**	$y = \sqrt{\frac{x-a}{2x+1}} + \lvert\sin x-3\rvert$
13.	$y = \frac{5x-a}{1-\sqrt{x+1}} - \mathrm{ctg}\lvert x-5\rvert$	**14.**	$y = \frac{\sqrt{x-2}}{3-ax} + \lvert\cos x-5\rvert$
15.	$y = \frac{7x-a}{3+log_2 x} + \sqrt{\lvert x+3\rvert}$	**16.**	$y = \sqrt{\frac{x+a}{2x+5}} + \lvert\sin x-a\rvert$
17.	$y = \frac{\sqrt{\ln(x+1)}}{5x-a} + \sin\lvert x+5\rvert$	**18.**	$y = \sqrt{\frac{lnx+1}{5x+a}} + \lvert x+5x\rvert$
19.	$y = \sqrt{\frac{5x-a}{x^2-1}} + \cos\lvert x-1\rvert$	**20.**	$y = \sqrt{\frac{4x-a}{x-2}} + \log\lvert x+2\rvert$

№ вар.	Функція для розрахунку	№ вар.	Функція для розрахунку
21.	$y = \sqrt{\frac{x-1}{2-ax}} + \lvert x - tgx \rvert$	22.	$y = \sqrt{\frac{x-a}{3x+1}} + \lvert \sin x - 2 \rvert$
23.	$y = \frac{\sqrt{x+2}}{2-ax} + \lvert \cos(x) + 5 \rvert$	24.	$y = \frac{\sqrt{3x+5}}{x-a} + \cos\lvert x + 5 \rvert$

Завдання 2. Розробити програму мовою **C++** та/або **Python**, відповідно до варіанту (***табл. 2***), розгалужених процесів з послідовною перевіркою умов та застосуванням повної/неповної форми умовного оператора, в якому необхідно в якому необхідно забезпечити:

— виведення на екран заданої за варіантом функції;
— виведення на екран вхідних параметрів ***a*** та змінної ***x***;
— заданим змінним послідовно присвоїти значення одного типу даних, різних типів даних, випадкових величин;
— вивести розраховані значення функцій у відповідному форматі;

Реалізовану програму на мові **C++ (Python)** протестувати з вхідними даними різного типу: цілого та дійсного. В програмі передбачити перевірку можливих помилок:

— ділення на нуль;
— підкореневий від'ємний вираз;
— логарифми з від'ємними значеннями;
— невизначеність функцій (вважається, що функція визначена на заданому проміжку, а поза її межами - не визначена);
— забезпечити своєчасне виведення повідомлень до відповідних помилок;
— результат обчислень заданої функції вивести на кінцевому етапі алгоритму.

Таблиця 2. – Системи нерівностей

№ вар.	Функція для розрахунку	№ вар.	Функція для розрахунку
1	$f(x) = \begin{cases} \sin 3x & \text{при } x \geq 5 \\ 2e^{ax-1} & \text{при } x < 5 \end{cases}$	2	$f(x) = \begin{cases} bx^2 + x - 6 & \text{при } x \geq 1 \\ \sqrt{ax+b} & \text{при } x < 1 \end{cases}$
3	$f(x) = \begin{cases} \lvert ax + 7 \rvert & \text{при } x \leq 15 \\ 5\sqrt{bx+1} & \text{при } x > 15 \end{cases}$	4	$f(x) = \begin{cases} \sqrt{5x^2 - a} & \text{при } x \geq 3 \\ bx + 8 & \text{при } x < 3 \end{cases}$
5	$f(x) = \begin{cases} \sin x & \text{при } x \geq -1 \\ 3x^2 + bx - 3 & \text{при } x < -1 \end{cases}$	6	$f(x) = \begin{cases} \cos x & \text{при } x \geq 0 \\ (\sqrt{ax-b})^{-1} & \text{при } x < 0 \end{cases}$
7	$f(x) = \begin{cases} \sqrt{3x^2 - a} & \text{при } x \geq -3 \\ -bx + 3 & \text{при } x < -3 \end{cases}$	8	$f(x) = \begin{cases} \sin x & \text{при } x \geq 2 \\ 2e^{ax} & \text{при } x < 2 \end{cases}$
9	$f(x) = \begin{cases} \sqrt{x+b} & \text{при } x \geq 11 \\ ax^2 + 14 & \text{при } x < 11 \end{cases}$	10	$f(x) = \begin{cases} \sqrt{x^3 + a} & \text{при } x \geq 7 \\ -bx - 3 & \text{при } x < 7 \end{cases}$
11	$f(x) = \begin{cases} \sin(x+1) & \text{при } x \geq 7 \\ ax^2 + 2bx - 4 & \text{при } x < 5 \end{cases}$	12	$f(x) = \begin{cases} \sin x & \text{при } x \geq -5 \\ 3e^{ax} + 1 & \text{при } x < -5 \end{cases}$
13	$f(x) = \begin{cases} \cos x & \text{при } x \geq 5 \\ \sqrt{ax+b} & \text{при } x < 5 \end{cases}$	14	$f(x) = \begin{cases} \lvert ax + b \rvert & \text{при } x > -8 \\ 5\sqrt{bx+3} & \text{при } x \leq -8 \end{cases}$

№ вар.	Функція для розрахунку	№ вар.	Функція для розрахунку
15	$f(x)=\begin{cases}\cos x & \text{при } x>9\\ 3e^{ax}+1 & \text{при } x\le 9\end{cases}$	**16**	$f(x)=\begin{cases}\lvert ax+x^2\rvert & \text{при } x\ge 4\\ \sqrt{bx-1}+3 & \text{при } x<4\end{cases}$
17	$f(x)=\begin{cases}\cos x & \text{при } x\ge -4\\ 3x^2+bx & \text{при } x<-4\end{cases}$	**18**	$f(x)=\begin{cases}\cos 2x & \text{при } x\ge -6\\ x^3-bx+1 & \text{при } x<-6\end{cases}$
19	$f(x)=\begin{cases}\sqrt{ax-3} & \text{при } x\ge -8\\ \lvert x^2-bx+7\rvert & \text{при } x<-8\end{cases}$	**20**	$f(x)=\begin{cases}2x+7 & \text{при } x>3{,}5\\ 3e^{ax} & \text{при } x\le 3{,}5\end{cases}$

Завдання 3. Розробити програму мовою **C++** та/або **Python**, відповідно до варіанту (***табл. 3***), розгалужених процесів з послідовною перевіркою умов та застосуванням повної/неповної форми умовного оператора, в якому необхідно в якому необхідно забезпечити:

— виведення на екран заданої за варіантом функції;
— виведення на екран вхідних параметрів ***a*** та змінної ***x***;
— заданим змінним послідовно присвоїти значення одного типу даних, різних типів даних, випадкових величин;
— вивести розраховані значення функцій у відповідному форматі;

Реалізовану програму на мові **C++, Python** протестувати з вхідними даними різного типу: цілого та дійсного. В програмі передбачити перевірку можливих помилок:

— ділення на нуль;
— підкореневий від'ємний вираз;
— логарифми з від'ємними значеннями;
— невизначеність функцій (вважається, що функція визначена на заданому проміжку, а поза її межами - не визначена);
— забезпечити своєчасне виведення повідомлень до відповідних помилок;
— результат обчислень заданої функції вивести на кінцевому етапі алгоритму.

Таблиця 3. – Системи нерівностей

№ вар.	Функція для розрахунку	№ вар.	Функція для розрахунку
1	$f(x)=\begin{cases}\sin 3x, & \text{при } x\in[0,5)\\ 2e^{ax-1}+1, & \text{при } x=5\\ (bx-a)^{-1}, & \text{при } x\in(5,12]\end{cases}$	**2**	$f(x)=\begin{cases}bx^2+x-6, & \text{при } x\in[0,1)\\ \left(\sqrt{ax+b}\right)^{x-1}, & \text{при } x=1\\ \cos x, & \text{при } x\in(1,9)\end{cases}$
3	$f(x)=\begin{cases}\lvert ax+7\rvert, & \text{при } x\in[-\infty,9)\\ 5\sqrt{bx+1}, & \text{при } x\in[9,15)\\ \sin x, & \text{при } x=15\end{cases}$	**4**	$f(x)=\begin{cases}\sqrt{5x^3-a}, & \text{при } x\in(-\infty,3]\\ bx+8, & \text{при } x=3\\ \cos x, & \text{при } x\in(3,11)\end{cases}$
5	$f(x)=\begin{cases}\sin x, & \text{при } x\in[-1,6)\\ 3x^4+bx-3, & \text{при } x=6\\ \lg(bx+a), & \text{при } x\in(6,+\infty)\end{cases}$	**6**	$f(x)=\begin{cases}\cos x, & \text{при } x\in[0,5)\\ \left(\sqrt{ax-b}\right)^{-1}, & \text{при } x=5\\ bx^2+3x-2, & \text{при } x\in(5,8)\end{cases}$
7	$f(x)=\begin{cases}\sqrt{3x^2-a} & \text{при } x\in(-3,3)\\ -bx+3, & \text{при } x=3\\ \cos(x-4), & \text{при } x\in(3,8)\end{cases}$	**8**	$f(x)=\begin{cases}\sin x, & \text{при } x\in[0,2)\\ 2e^{ax}, & \text{при } x=2\\ (bx+a)^{-x}, & \text{при } x\in(2,8)\end{cases}$

№ вар.	*Функція для розрахунку*	*№ вар.*	*Функція для розрахунку*
9	$f(x)=\begin{cases}\sqrt{x+b}, & \text{при } x\in[0,11)\\ ax^3-3x+14, & \text{при } x\in[11,15)\\ e^{-ax}, & \text{при } x=15\end{cases}$	**10**	$f(x)=\begin{cases}\sqrt{x^4+a}, & \text{при } x\in(-\infty,7)\\ -bx-3, & \text{при } x=7\\ \sin x, & \text{при } x\in(7,10)\end{cases}$
11	$f(x)=\begin{cases}\sin(x+1) & \text{при } x\in[0,1)\\ ax^2+2bx-4, & \text{при } x\in[1,7)\\ (ax+b), & \text{при } x=12\end{cases}$	**12**	$f(x)=\begin{cases}\cos x, & \text{при } x\in[0,5)\\ 3e^{ax}+1, & \text{при } x=5\\ (b\,x-10)^2, & \text{при } x\in(5,11)\end{cases}$
13	$f(x)=\begin{cases}log(ax), & \text{при } x\in[-2,4)\\ \sqrt{ax+b}, & \text{при } x\in[4,9)\\ x^3-bx+3, & \text{при } x=9\end{cases}$	**14**	$f(x)=\begin{cases}\lvert ax+b\rvert, & \text{при } x\in(-\infty,8)\\ 5\sqrt{bx+3}, & \text{при } x=8\\ \cos x, & \text{при } x\in(8,10)\end{cases}$
15	$f(x)=\begin{cases}tg\,x, & \text{при } x\in[0,9)\\ 3e^{ax}+1, & \text{при } x=9\\ \ln(bx-1), & \text{при } x\in(9,11)\end{cases}$	**16**	$f(x)=\begin{cases}\lvert ax+x^3\rvert, & \text{при } x\in[-\infty,4)\\ \sqrt{bx-1}+3, & \text{при } x=4\\ \cos ax, & \text{при } x\in(4,10)\end{cases}$
17	$f(x)=\begin{cases}cos\,2x, & \text{при } x\in[0,4)\\ 3x^2+bx, & \text{при } x=4\\ \ln(ax+b), & \text{при } x\in(4,9)\end{cases}$	**18**	$f(x)=\begin{cases}\cos x^3, & \text{при } x\in[-\infty,4)\\ x^3-bx+1, & \text{при } x=4\\ \sqrt{ax}, & \text{при } x\in(4,7)\end{cases}$
19	$f(x)=\begin{cases}\sqrt{ax-3}, & \text{при } x\in(-8,3)\\ \lvert x^2-bx+7\rvert, & \text{при } x=3\\ \cos(x+1), & \text{при } x\in(3,+\infty)\end{cases}$	**20**	$f(x)=\begin{cases}2x+7, & \text{при } x\in[0,5)\\ 3e^{ax}, & \text{при } x=5\\ \lg(bx+a), & \text{при } x\in(5,8)\end{cases}$

Завдання 4. Розробити програму мовою **C++** та/або **Python** відповідно до варіанту (***табл. 4***), розгалужених процесів з послідовною перевіркою умов та застосуванням повної/неповної форми умовного оператора, в якому необхідно в якому необхідно забезпечити:

— виведення на екран відповідних даних;

— вивести розраховані значення на екран у відповідному форматі;

— передбачити перевірку можливих помилок, які можуть виникати при введенні даних.

Таблиця 4. – Задачі економічного спрямування

№ вар.	*Задача для реалізації*
1	Обчислити вартість введеного товару та розрахувати його знижку у 10%.
2	Збільшити ціну введеного товару на 10% та розрахувати його вартість.
3	Обчислити вартість введеного товару та розрахувати його знижку у 12%.
4	Зменшити ціну введеного товару на 10% та розрахувати його вартість.
5	Обчислити вартість введеного товару та розрахувати його знижку у 15%.
6	Збільшити ціну введеного товару на 15% та розрахувати його вартість.
7	Розрахувати загальну вартість товарів за наступної умови: якщо кількість товару більше 50 одиниць, то надається знижка в 10%
8	Розрахувати загальну вартість товарів за наступної умови: якщо кількість товару більше 20 одиниць, то надається знижка в 5%, якщо кількість більше 100 одиниць, то знижка складає 10%
9	Розрахувати загальну вартість товарів за наступної умови: якщо кількість товару більше 30 одиниць, то надається знижка в 7%, якщо кількість більше 50 одиниць, то знижка складає 12%, якщо кількість складає 70 одиниць, знижка - 15%
10	Розрахувати заробітну плату робітника, якщо його місячний оклад складає К-тисяч гривень. Вивести на екран суму окладу, сплаченого податку та розмір заробітної плати.
11	Збільшити оклад працівника на 10% та розрахувати його заробітну плату

№ вар.	*Задача для реалізації*
12	Визначити суму сплаченого податку за наступної умови: якщо оклад працівника складає більше 20 000 гривень, то нараховується податок в 25%
13	Зменшити оклад працівника на 15% та розрахувати його заробітну плату
14	Визначити суму сплаченого податку за наступної умови: якщо оклад працівника складає більше 5000 гривень, то нараховується податок в 21%, якщо більше 10 000, то податок 22%
15	Визначити суму сплаченого податку за наступної умови: якщо оклад працівника складає більше 5000 гривень, то нараховується податок в 21%, якщо більше 10 000, то податок 22%, якщо оклад менше 5000 гривень – податок не нараховується
16	Розрахувати заробітну плату робітника, якщо його погодинний тариф складає К-гривень. Вивести на екран зароблену суму працівником за день, суму сплаченого податку та розмір виплаченої заробітної плати.
17	Зменшити погодинний тариф працівника на 10% та розрахувати його заробітну плату за день
18	Визначити суму сплаченого податку за наступної умови: якщо працівник відпрацював більше 8-ми годин за день, то нараховується податок в 18%
19	Збільшити погодинний тариф працівника на 20% та розрахувати його заробітну плату за день
20	Визначити суму сплаченого податку за наступної умови: якщо працівник відпрацював більше 8-ми годин за день, то нараховується податок в 18%, якщо більше 10-ти годин, то податок складає 16%

Завдання 5. За умовою поданих завдань ***1*** та ***2*** розробити програму мовою С++, відповідно до якого програма структурується на блоки: блок введення даних, блок обробки і блок виведення результатів (вся видача повідомлень про помилки і результат обчислення здійснюється в кінці програми; щоб реалізувати таку видачу, треба використати робочу змінну, яка вказує номер повідомлення). За умовою поданих завдань ***1*** та ***2***, розробити програму з множинним вибором розрахунку лінійних та розгалужених функцій, в якому необхідно забезпечити:

— виведення на екран відповідного меню вибору функцій;
— структурувати на блоки: блок введення даних, блок обробки і блок виведення результатів;
— виведення на екран вхідних параметрів ***a*** та змінної ***x***;
— заданим змінним послідовно присвоїти значення одного типу даних та різних типів даних;
— вивести розраховані значення функцій у відповідному форматі.

Реалізовану програму протестувати з вхідними даними різного типу: цілого та дійсного. Передбачити в реалізованому алгоритмі відповідні коментарі.

Зміст звіту

1. Титульна сторінка, тема та мета практичного завдання.
2. Завдання відповідно до номеру наданого варіанту
3. Опис виконаних завдань, лістинги реалізованих програм, результати виконання програм, блок-схеми, структурограми.

4. Реалізований алгоритм подати наступними способами: природною мовою в структурованому вигляді; формульно-словесною мовою; псевдокодом на мові С++. блок-схемами, реалізовані онлайн-сервісами ***Lucidchart, Draw.io, MS Visio***, тощо.
5. Висновки, які відображають результати виконаних завдань та їх аналіз.

Теоретичні відомості

Базові алгоритми мовою С++

С++ є універсальною мовою програмування, яка забезпечує можливості для написання програм на різних рівнях складності - від низькорівневих операцій з пам'яттю до високорівневих абстракцій. С++ підтримує об'єктно-орієнтований підхід, що дозволяє робити код більш структурованим, зменшує повторюваність та сприяє повторному використанню коду за допомогою класів та об'єктів.

С++ - це мова з високою продуктивністю, оскільки надає прямий доступ до пам'яті та має можливості оптимізації. Вона часто використовується для розробки операційних систем, ігор, додатків реального часу та вбудованих систем. Розглянемо приклади базових алгоритмів.

Приклад 1. Розробити схему алгоритму і створити програмний проект у С++ для обчислення заданої функції, де x − довільна змінна, яка задається користувачем.

$$y = \frac{0{,}2x^2 - x}{(\sqrt{3} + x)(1 + 2x}$$

Розв'язок. Послідовність виконання завдання:

1) Запустити онлайн-компілятор **https://www.onlinegdb.com/.**
2) Вибрати мову програмування *С++*. Виконати програмний код.
3) Для *запуску* програми виконати команду меню Run або скористатись функціональною клавішею F9 (рис. 2).

Текст програмного коду та схема алгоритму (рис. 1)

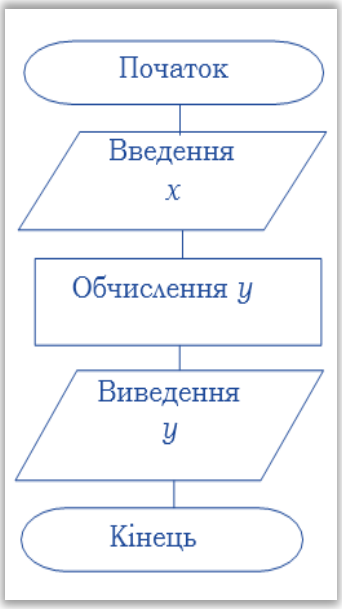


Рисунок 1. – Блок-схема лінійного алгоритму.

```
#include <iostream>
#include <cmath>
using namespace std;
        int main()
        {
                setlocale(0, ".1251");
                double y, x;
                cout << "Введіть значення x= ";
                cin >> x;
                y = (0.2 * x * x - x) / ((sqrt(3.) + x) * (1 + 2. * x));
                cout << "Результат y= " << y << endl; }
```

Рисунок 2. – Лістинг лінійного алгоритму з результатом виконання.

Приклад 2. Розробити схему алгоритму і створити програмний проект на мові С++ для обчислення функції, де x – випадкова змінна.

$$y = \frac{0{,}2x^2 - x}{(\sqrt{3} + x)(1 + 2x}$$

Складемо програму для обчислення заданої функції, програмний код якої наведено на рис. 3.

```
#include <iostream>
#include <cmath>
#include <cstdlib> // для функцій rand() та srand()
using namespace std;
    int main()
    {
        setlocale(0, ".1251");
        //srand(4541);
        srand(time(NULL));
        double y, x;
        x=rand();
        cout << "Для значення x= "<<x<< endl;
        y = (0.2 * x * x - x) / ((sqrt(3.) + x) * (1 + 2. * x));
        cout << "Результат y= " << y << endl;
        return 0;
}
```

Рисунок 3. – Програмний код лінійного алгоритму випадкових величин

Приклад 3. Розрахувати лінійну функцію виду: **f=cos(a)+cos(x)**. Протестувати створену програму за різними типами даних.

Складемо програму мовою С++ (рис. 4) розв'язку лінійної функції. Результат виконання програми наведено на рис. 5.

```
ConsoleApplication2                                    (Глобальная область)
1   // ConsoleApplication2.cpp : Этот файл содержит
2   #include <iostream>
3   #include <locale>
4   #include <cmath>
5   using namespace std;
6
7   int main()
8   {
9       setlocale(LC_CTYPE, "ukr");
10      int a, x,f;
11      cout << "Введіть значення a" << endl;
12      cin >> a;
13      cout << "Введіть значення x" << endl;
14      cin >> x;
15      f = cos(a)+cos(x);
16      cout << "Функція f= "<<f<<endl;
17  }
```

Рисунок 4.- Лістинг лінійної функції мовою С++

```
Консоль отладки Microsoft Visual Studio
Введіть значення a
2
Введіть значення x
3
Функція f= -1
```

Рисунок 5.- Результат виконання програми заданої лінійної функції

Приклад 4. Розрахувати розгалужену функцію виду:

$$f = \begin{cases} \cos(a) + \cos(x), & x > 5 \\ a + x, & x \le 5 \end{cases}$$

Складемо програму мовою С++ (рис. 6) розв'язку розгалуженої функції. Результат виконання програми наведено на рис. 7.

```
ConsoleApplication2                                    (Глобальная область)
2   #include <iostream>
3   #include <locale>
4   #include <cmath>
5   using namespace std;
6
7   int main()
8   {
9       setlocale(LC_CTYPE, "ukr");
10      int a, x;
11      float f;
12      cout << "Введіть значення a" << endl;
13      cin >> a;
14      cout << "Введіть значення x" << endl;
15      cin >> x;
16      if(x>5)
17      f=cos(a)+cos(x);
18      else
19      f=a+x;
20      cout << "Функція f= "<<f<<endl;
21  }
```

Рисунок 6.- Лістинг програми розгалуженої функції мовою С++

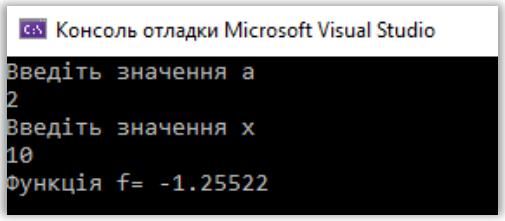
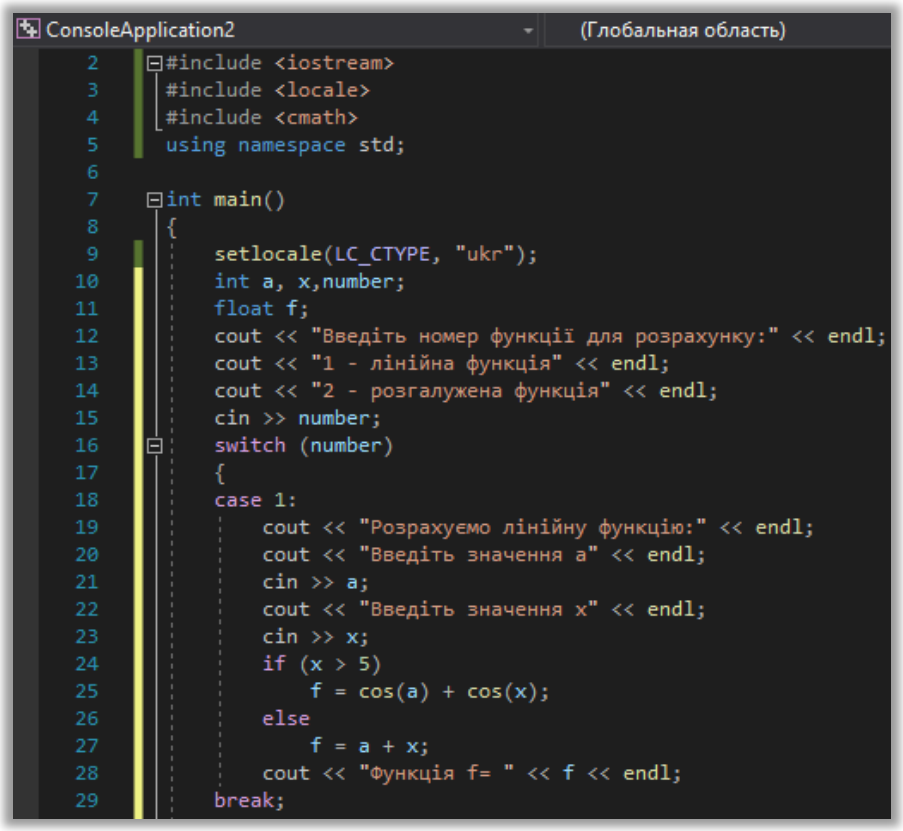


Рисунок 7.- Результат виконання програми розгалуженої функції

Приклад 5. За умовою прикладів 1 та 2 розробити програму мовою С++ з використанням оператору множинного вибору для розрахунку лінійної та розгалуженої функції.

Складемо програму мовою С++ (рис. 8) розв'язку лінійної функції. Результат виконання програми наведено на рис. 9.



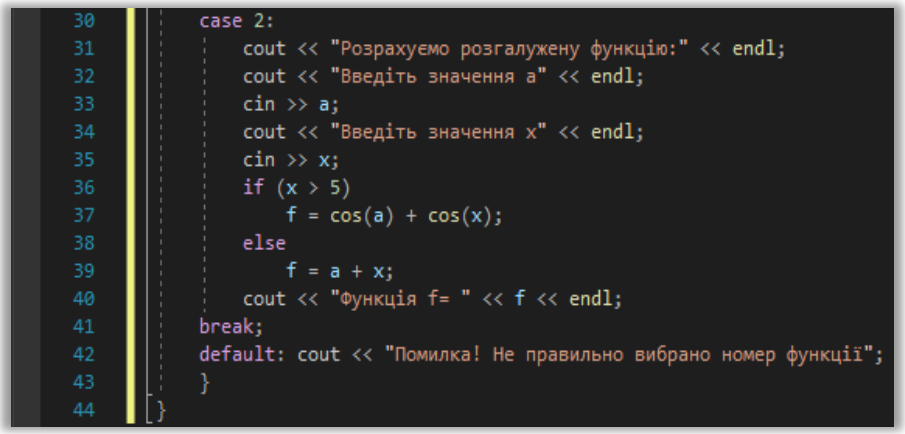


Рисунок 8.- Лістинг програми з використання оператору множинного вибору

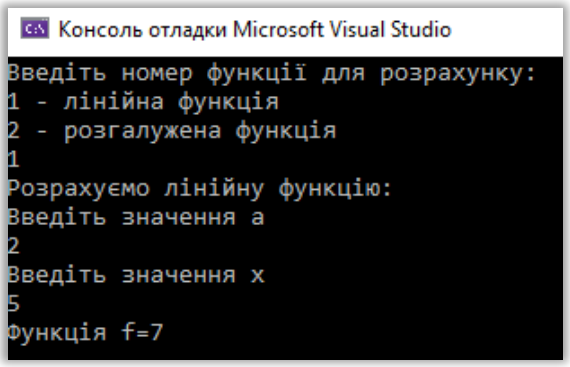


Рисунок 9.- Результат виконання програми оператору множинного вибору

Базові алгоритми мовою Python

Python - це високорівнева, інтерпретована, загального призначення мова програмування з простим синтаксисом та потужними функціональними можливостями. Python підтримує об'єктно-орієнтоване програмування, функціональне програмування та інші парадигми, що дозволяють створювати більш структурований та гнучкий код. Python використовується у багатьох областях, включаючи веб-розробку, наукові дослідження, аналіз даних, штучний інтелект, машинне навчання, розробку ігор, автоматизацію та багато іншого. Розглянемо приклади базових алгоритмів.

Приклад 6. Розрахувати лінійну функцію виду:

$$y = \frac{\sqrt{x+2}}{2-ax} + |\cos x + 5|$$

Складемо програму мовою **Python** для розв'язку лінійної функції. Результат виконання програми наведено на рис. 10.

Текст програмного коду

```
from math import*
# ввод даних
a=int(input('Введіть ціле число a='))
x=int(input('Введіть ціле число x='))
y=sqrt(x+2)/(2-a*x)+abs(cos(x)+5)
print('Значення y=',y)
```

```
Введіть ціле число a=-2
Введіть ціле число x=3
Значення y= 4.28951600058702
```

Рисунок 10.- Результат виконання програми розв'язку лінійної функції

Приклад 7. Розрахувати розгалужену функцію виду:

$$f = \begin{cases} \cos(a) + \cos(x), & x > 5 \\ a + x, & x \le 5 \end{cases}$$

Складемо програму мовою **Python** розв'язку розгалуженої функції. Результат виконання програми наведено на рис. 11.

Текст програмного коду

```
from math import*
# ввод даних
a=int(input('Введіть ціле число a='))
x=int(input('Введіть ціле число x='))
if x>5:
  y=cos(a)+cos(x)
else:
  y=a+x
print('Значення y=',y)
```

```
Введіть ціле число a=3
Введіть ціле число x=6
Значення y= -0.029822209950079448
```

Рисунок 11.- Результат розв'язку розгалуженої функції

Приклад 8. Розрахувати розгалужену функцію виду:

$$f = \begin{cases} 2x + 7, & x \in [0,5) \\ 3e^{ax}, & x = 5 \end{cases}$$

Складемо програму мовою **Python** розв'язку розгалуженої функції. Результат виконання програми наведено на рис. 12.

Текст програмного коду

```
from math import*
# ввод даних
a=int(input('Введіть ціле число a='))
x=float(input('Введіть ціле число x='))
if x>=0 and x<5:
  y=2*x+7
  print('Значення y=',y)
elif x==5:
  y=3*exp(a*x)
  print('Значення y=',y)
else:
  print('Функція y не визначена')
```

```
Введіть ціле число a=5
Введіть ціле число x=7.5
Функція y не визначена
```

Рисунок 12.- Результат виконання програми розгалуженої функції

Приклад 9. Зменшити вартість товару на 10% , якщо задана його ціна та кількість.

Складемо програму мовою **Python** для розв'язку поставленої задачі. Результат виконання програми наведено на рис. 13.

Текст програмного коду

```
print('Введіть ціну товару')
A = float(input())
print('Введіть кількість товару')
B = int(input())
cost1 = A * B
```

```
print('Вартість товару', cost1, 'грн.')
print('Зменшена вартість товару на 10% складає', cost1*0.9, 'грн.')
```

```
Введіть ціну товару
2.5
Введіть кількість товару
10
Вартість товару 25.0 грн.
Зменшена вартість товару на 10% складає 22.5 грн.
```

Рисунок 13.- Результат виконання програми поставленої задачі

Приклад 10. Розрахувати розгалужену функцію виду:

$$f = \begin{cases} 2x + 7, & x \in [0,5) \\ 3e^{ax}, & x = 5 \end{cases}$$

Значення величини **a** задано цілим випадковим числом в інтервалі від 0 до 10; значення ***x*** задано дійсним випадковим числом в інтервалі від -25 до 25.

Складемо програму мовою **Python** для розв'язку поставленої задачі. Результат виконання програми наведено на рис. 14.

Текст програмного коду

```
from math import*
from random import randint
import random
# ввод даних
a= randint(0, 10)
print("В інтервалі від 0 до 10 задано значення a=",a)
x=random.uniform(-25.0, 25.0)
print("В інтервалі від -25 до 25 задано значення x=",x)
if x>=0 and x<5:
  y=2*x+7
  print('Значення y=',y)
elif x==5:
  y=3*exp(a*x)
  print('Значення y=',y)
else:
  print('Функція y не визначена')
```

```
В інтервалі від 0 до 10 задано значення a= 0
В інтервалі від -25 до 25 задано значення x= 3.478763696027876
Значення y= 13.957527392055752
```

Рисунок 14.- Результат виконання поставленої задачі

Контрольні запитання

1. Що таке ідентифікатор? Які правила запису ідентифікаторів?
2. Що таке змінна? Для чого використовуються змінні? Яка відмінність змінної від константи?
3. Які є форми запису дійсних чисел у мові С++?
4. Які прості типи даних мови С++ Ви знаєте?
5. Як здійснюється виведення на екран в мові С++?

6. Що таке керуюча послідовність? Наведіть приклади цих послідовностей.
7. Які операції в мові С++ Ви знаєте?
8. Які є форми запису операцій інкременту та декременту?
9. Перерахуйте відомі Вам операції мови С++?
10. Що таке бібліотека стандартних функцій? Для чого їх використовують?
11. Що таке алгоритм? Які є форми запису алгоритмів?
12. З яких базових конструкцій може складатися довільний алгоритм?
13. Які блоки можна використовувати на блок-схемах?
14. Як записується і працює умовний оператор if/else в С++?
15. Які особливості вкладених структур if/else?
16. Що таке складений оператор? Коли він використовується?
17. Як реалізовано оператор вибору в С++?
18. Для чого використовується оператор goto?
19. Яким чином можна згенерувати випадкове число?

Практичне завдання 2.
Розробка і реалізація циклічних алгоритмів з використанням операторів циклу: передумови while, постумови do-while та з параметром for. Реалізація комбінованих алгоритмів з вкладеними циклами.

Мета заняття: опрацювання числових послідовностей за допомогою циклів; набуття практичних навичок реалізації циклічних процесів з використанням оператора циклу з параметром **for**, циклів з передумовою **while** та постумовою **do-while** та створення алгоритмів із вкладеними циклами мовами програмування **C++, Python**; реалізовувати алгоритми графічно в онлайн-сервісах **Lucidchart**, **Draw.io.**

Завдання 1. Розробити алгоритми для обчислення функції $f(x)$, відповідно до варіанту (***табл. 1***), (***табл. 2***), (***табл. 3***), використовуючи: оператор циклу з параметром **for** та/або цикл з передумовою **while** та/або постумовою **do-while.** В алгоритмах необхідно забезпечити:

— виведення на екран заданої за варіантом функції;
— виведення на екран значень параметра ***a*** та **10-ть** значень змінної ***x***;
— заданим змінним послідовно присвоїти значення одного типу даних, різних типів, випадковими величинами;
— вивести розраховані значення функцій на екран у відповідному форматі;
— при організації циклів застосувати унарні операції: інкремента (++), декремента (--)

Забезпечити своєчасне виведення повідомлень до відповідних помилок:
— ділення на нуль;
— підкореневий від'ємний вираз;
— логарифми з від'ємними значеннями;

Передбачити в реалізованому алгоритмі перевірку формальних та неформальних умов, результат обчислень заданої функції вивести на кінцевому етапі алгоритму.

Реалізовану програму **C++, Python** протестувати з вхідними даними різного типу: цілого та дійсного. Передбачити в реалізованому алгоритмі відповідні коментарі (**обов'язково! 20-25%** загального алгоритму).

Завдання 4. Розробити алгоритми для обчислення суми заданої функції $f(x)$ за різними операторами циклу: з параметром **for**, з передумовою **while** та постумовою **do-while** з врахуванням зразка наведеної блок-схеми на ***рис. 15***.

$$f(x) = \sum_{k=1}^{5} \frac{x^{k+1}}{2^k + k}$$

В алгоритмах необхідно забезпечити:
— виведення на екран вхідних параметрів та значень змінної ***x***;
— вивести розраховані значення функцій у відповідному форматі;
— при організації циклів застосувати унарні операції: інкремента (++), декремента (--)

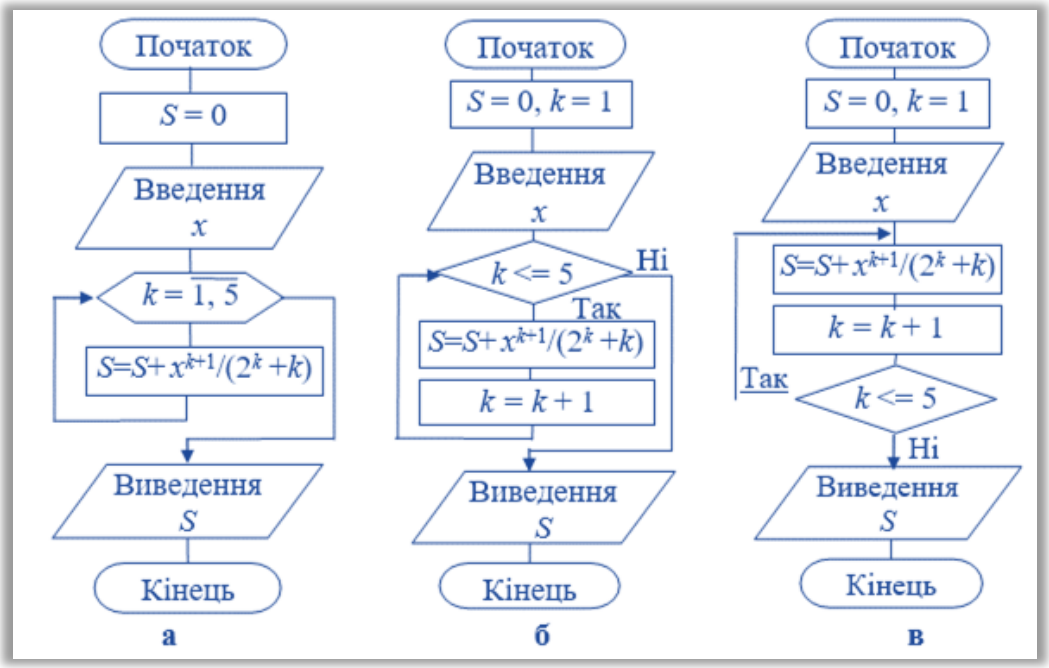


Рисунок 15. - Схеми алгоритмів з використанням операторів циклу:
а) **for**; б) **while**; в) **do-while**

Реалізовану програму **C++, Python** протестувати з вхідними даними різного типу: цілого та дійсного. Передбачити в реалізованому алгоритмі відповідні коментарі.

Завдання 2. Розробити алгоритми для обчислення функції $f(x)$, відповідно до варіанту (***табл. 5***), використовуючи: оператор циклу з параметром **for**, циклу з передумовою **while** та постумовою **do-while.**

В алгоритмах необхідно забезпечити:

— виведення на екран вхідних параметрів та значень змінної ***x***;

— вивести розраховані значення функцій на екран у відповідному форматі;

— при організації циклів застосувати унарні операції: інкремента (++), декремента (--)

Реалізовану програму **C++, Python** протестувати з вхідними даними різного типу: цілого та дійсного. Передбачити в реалізованому алгоритмі відповідні коментарі (**обов'язково! 20-25%** загального алгоритму).

Таблиця 5. – Математичні функції до виконання

№ вар.	*Функція f(x)*	*№ вар.*	*Функція f(x)*
1	$f(x) = \sum_{k=1}^{12} \frac{\sin(kx) + k}{\sqrt[k]{x + 0{,}1} + 6k}$	**2**	$f(x) = \sum_{k=1}^{7} \frac{2^k \sin(x + k)}{(x + 1)^k}$
3	$f(x) = \sum_{k=1}^{9} \frac{x^{k+1}}{(k + 1)^x}$	**4**	$f(x) = \sum_{k=1}^{9} \frac{\sin(2kx) + 0{,}2}{2k + 5}$
5	$f(x) = \sum_{k=1}^{9} \frac{\ln(x + 1)}{(x + k)^k}$	**6**	$f(x) = \sum_{k=1}^{7} \frac{kx \cos(x + k)}{\ln(2 + x) + 2k}$

№ вар.	Функція f(x)	№ вар.	Функція f(x)
7	$f(x)=\sum_{k=1}^{6}\frac{\sin(0{,}17x^k)}{2k+x}$	8	$f(x)=\sum_{k=1}^{8}\frac{5\ln(2kx)}{arctg(2x)+k^2}$
9	$f(x)=\sum_{k=1}^{9}\frac{\mathrm{tg}(x)-\frac{x^2}{k}}{k^2-1}$	10	$f(x)=\sum_{k=1}^{7}\frac{\sin(x^k-\pi)}{ln\,k^2+0{,}3}$
11	$f(x)=\sum_{k=1}^{12}\frac{\cos(kx)}{k}$	12	$f(x)=\sum_{k=1}^{8}\frac{\sin(x^k)}{4k}$
13	$f(x)=\sum_{k=1}^{12}\frac{\cos(kx)}{k}$	14	$f(x)=\sum_{k=1}^{7}\frac{\ln(3x^2)}{(2+x)^k}$
15	$f(x)=\sum_{k=1}^{8}\sqrt[k]{\ln(x+1)}$	16	$f(x)=\sum_{k=6}^{1}\frac{x^k}{k^3+x^{k+2}}$
17	$f(x)=\sum_{k=1}^{11}\frac{\sin(x^k-1)}{4k^2+1}$	18	$f(x)=\sum_{k=1}^{8}\frac{\ln x^{2k-1}}{2^k(2k-1)}$
19	$f(x)=\sum_{k=2}^{9}\frac{\mathrm{tg}(e^x)}{3k^2+1}$	20	$f(x)=\sum_{k=3}^{10}\frac{x^{k-1}\cos x}{12^k-1}$
21	$f(x)=\sum_{k=3}^{11}\frac{cos^{2+k}\,x}{2k-1}$	22	$f(x)=\sum_{k=1}^{18}(k+\cos(x+2))$
23	$f(x)=\sum_{k=2}^{10}\frac{arctg^3(2kx)}{1{,}2\ln(k+x)}$	24	$f(x)=\sum_{k=1}^{11}\frac{\sin(x^k)+0{,}3}{(2^k)}$
25	$f(x)=\sum_{k=1}^{11}\frac{\sin(x^k)+0{,}3}{(2^k)}$	26	$f(x)=\sum_{k=1}^{6}\frac{k^2\sin(x/k)-kx^2}{e^{kx}}$
27	$f(x)=\sum_{k=1}^{12}\frac{\cos(x^k)}{(x+5)^k+k}$	28	$f(x)=\sum_{k=2}^{9}\frac{\sin(x+1)+1{,}5}{\lg(5xk)-2{,}1}$
29	$f(x)=\sum_{k=1}^{7}\frac{2(x+1)^{3-k}}{(k+1)^x+k^3}$	30	$f(x)=\sum_{k=1}^{10}\cos\left(k^3\frac{kx}{5}\right)$
31	$f(x)=\sum_{k=1}^{7}\frac{x\sin(x-k)}{e^{2+k}+k}$	32	$f(x)=\sum_{k=2}^{6}\frac{x\cdot\sin(x-k)}{e^{2+x}+k}$

Завдання 3. Розробити алгоритми для визначення найбільшого числа із 10-ти введених дійсних значень використовуючи: оператор циклу з параметром **for**, циклу з передумовою **while** та постумовою **do-while** відповідно до зразка наведених блок-схем (рис. 16).

В алгоритмах необхідно забезпечити:

— виведення на екран вхідних параметрів та значень змінної ***x***;

— вивести розраховані значення функцій на екран у відповідному форматі;

— при організації циклів застосувати унарні операції: інкремента (++),

декремента (--)

Реалізовану програму **C++, Python** протестувати з вхідними даними різного типу: цілого та дійсного.

Рисунок 16. - Схема алгоритму з використанням оператору циклу **for**.

Завдання 4. Розробити алгоритми із застосуванням оператора циклу для опрацювання числових послідовностей відповідно до варіанту, поданих у ***табл. 6.*** В алгоритмах необхідно забезпечити:

— виведення на екран вхідних параметрів та **відповідних** значень ***x***;
— вивести розраховані значення функцій на екран у відповідному форматі;
— при організації циклів застосувати унарні операції: інкремента (++), декремента (--)

Таблиця 6. - Варіанти до завдання 4

№ вар.	*Завдання для виконання*
1	Ввести 7 дійсних чисел та обчислити добуток елементів цієї послідовності, значення яких є менше за 6
2	Ввести 10 дійсних чисел та обчислити кількість додатних елементів
3	Ввести 6 дійсних чисел та обчислити суму від'ємних елементів
4	Ввести 5 дійсних чисел і визначити найменше та найбільше серед них
5	Ввести 8 дійсних чисел та обчислити середнє арифметичне ненульових
6	Ввести 9 дійсних чисел та обчислити суму елементів, абсолютне значення яких не перевищує число 5
7	Ввести 11 дійсних чисел та обчислити кількість елементів послідовності, значення яких є більше за значення першого елемента
8	Ввести 6 дійсних чисел та обчислити добуток елементів послідовності, значення яких перебувають у діапазоні [3, 6]
9	Ввести 8 дійсних чисел та обчислити середнє арифметичне додатних

№ вар.	*Завдання для виконання*
10	Ввести 7 дійсних чисел та обчислити суму квадратів тих чисел, модуль яких не перевищує число 3
11	Ввести 14 цілих чисел та обчислити кількість ненульових елементів
12	Ввести 9 дійсних чисел та визначити мінімальний елемент послідовності
13	Ввести 6 цілих чисел та обчислити добуток ненульових елементів
14	Ввести 10 цілих чисел та обчислити середнє арифметичне елементів послідовності, значення яких перебувають у діапазоні [10, 20]
15	Ввести 8 дійсних чисел та обчислити кількість елементів, значення яких перебувають у діапазоні [5, 10]
16	Ввести 7 цілих чисел та визначити суму модулів усіх від'ємних елементів
17	Ввести 9 дійсних чисел та обчислити добуток додатних елементів, значення яких не перевищує число 4
18	Ввести 12 дійсних чисел та обчислити кількість додатних і кількість від'ємних елементів послідовності
19	Ввести 8 цілих чисел та обчислити середнє арифметичне абсолютних (за модулем) значень усіх елементів послідовності
20	Ввести 6 дійсних чисел та віднайти максимальний і мінімальний елементи та визначити наскільки максимальний елемент є більшим за мінімальний
21	Ввести 11 цілих чисел та обчислити суму тільки двоцифрових елементів
22	Ввести 9 цілих чисел та обчислити добуток непарних елементів
23	Ввести 14 цілих чисел та обчислити кількість елементів, кратних до числа 3
24	Ввести 7 цілих чисел та обчислити середнє арифметичне парних елементів
25	Ввести 6 дійсних чисел та обчислити суму елементів, значення яких є меншим за значення першого елемента послідовності
26	Ввести 9 цілих чисел та обчислити добуток одноцифрових елементів
27	Ввести 8 цілих чисел та визначити найменший з непарних додатних елементів цієї послідовності
28	Ввести 11 цілих чисел та обчислити середнє арифметичне елементів, кратних до числа 3
29	Ввести 7 цілих чисел та обчислити добуток елементів, кратних до числа 5
30	Ввести 10 цілих чисел та визначити найбільший з парних додатних елементів цієї послідовності

Реалізовану програму **C++, Python** протестувати з вхідними даними різного типу: цілого та дійсного. Передбачити в реалізованому алгоритмі відповідні коментарі (**обов'язково! 20-25%** загального алгоритму).

Завдання 5. Розробити програму мовою **C++**, **Python** відповідно до варіанту (***табл. 4***), використовуючи: оператор циклу з параметром **for**, циклу з передумовою **while** та постумовою **do-while.** В алгоритмах необхідно забезпечити:

— виведення на екран початкових даних (не менше **10-ти** відповідних значень);

— заданим змінним послідовно присвоїти значення одного типу даних, різних типів, випадковими величинами;

— вивести розраховані значення функцій на екран у відповідному форматі;

— при організації циклів застосувати унарні операції: інкремента (++), декремента (--).

Передбачити в реалізованому алгоритмі перевірку формальних та неформальних умов, результати проміжних та кінцевих обчислень вивести в процесі виконання алгоритму.

Реалізовану програму **C++, Python** протестувати з вхідними даними різного типу: цілого та дійсного. Передбачити в реалізованому алгоритмі відповідні коментарі.

Завдання 6. З використанням вкладених циклів, розробити алгоритми для обчислення суми ряду:

$$S = \sum_{k=1}^{7} \frac{2x^{2i-1}}{3(2i-1)!}$$

де i = 1, 2, …, 7 відповідно до наведеного зразка реалізованої блок-схеми (рис. 17). В алгоритмах необхідно забезпечити:

— виведення на екран вхідних параметрів та відповідних значень ***x***;

— вивести розраховані значення функцій на екран у відповідному форматі;

— при організації циклів застосувати унарні операції: інкремента (++), декремента (--)

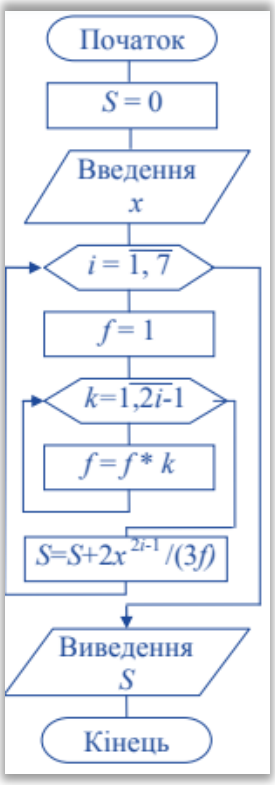


Рисунок 17. – Блок-схема алгоритму визначення суми відповідного ряду.

Реалізовану програму **C++, Python** протестувати з вхідними даними різного типу: цілого та дійсного. Передбачити в реалізованому алгоритмі відповідні коментарі.

Завдання 7. Розробити алгоритм для обчислення суми ряду відповідно до варіанту (***табл. 7***), використовуючи вкладені цикли. В алгоритмах необхідно забезпечити:

— виведення на екран вхідних параметрів та відповідних значень ***x***;
— вивести розраховані значення функцій у відповідному форматі;
— при організації циклів застосувати унарні операції: інкремента (++), декремента (--)

Таблиця 7. - Варіанти функцій до завдання

№ вар.	*Функція f(x)*	*№ вар.*	*Функція f(x)*
1	$y = \sum_{k=1}^{10} \frac{x^k}{(k+1)!}$	**2**	$y = \sum_{i=1}^{6} \frac{(-1)^i x^{2i}}{(3i-1)!}$
3	$y = \sum_{i=1}^{10} \frac{(-1)^{i+1} i!}{2^{2i-1} \sin x}$	**4**	$y = \sum_{i=1}^{5} \frac{(2i-1)!}{x^{2i-1}}$
5	$y = \sum_{i=1}^{5} (-1)^{i+1} \frac{\sin x^i}{(2i-1)!}$	**6**	$y = \sum_{k=1}^{7} \frac{k! \cos(\pi k - x)}{\ln x}$
7	$y = \prod_{k=1}^{10} \frac{x^k}{(2k)!}$	**8**	$y = \sum_{k=1}^{7} \frac{(2k-1)!}{2^k x^{k-1}}$
9	$y = \sum_{i=1}^{5} (-1)^{i-1} \frac{\cos x^i}{(2i)!}$	**10**	$y = \sum_{i=1}^{3} \frac{x^{2i}(2i-1)!}{2^i}$
11	$y = \sum_{i=1}^{6} \frac{(-1)^{i+1} x^{2i}}{(2i-1)!}$	**12**	$y = \sum_{i=1}^{9} \frac{(-1)^i tg x^2}{(2i-1)!}$
13	$y = \sum_{i=1}^{11} \frac{(-1)^i x^i}{i! \cos(i + \pi/4)}$	**14**	$y = \sum_{k=1}^{6} \frac{k!}{(1+x)^k}$
15	$y = \prod_{k=1}^{5} \frac{(k+1)!}{x^{k+2}}$	**16**	$y = \prod_{k=1}^{5} \frac{x^{k+2}}{k!}$
17	$y = \prod_{k=1}^{5} \frac{(-1)^k (2k)!}{4{,}5 x^{2k-1}}$	**18**	$y = \sum_{i=1}^{6} \frac{tg(x-\pi)^2}{i!}$
19	$y = \sum_{k=1}^{8} \frac{(-1)^k x^{2k-1}}{2^k k!}$	**20**	$y = \sum_{k=1}^{5} \frac{(-1)^k x^{3k-2}}{(k+1)!}$
21	$y = \sum_{i=1}^{11} \frac{(-1)^i x^{3i}}{(2i-1)!}$	**22**	$y = \sum_{k=1}^{8} \frac{2k! \cos(\pi - x)}{x^{2k-1}}$
23	$y = \sum_{k=1}^{7} \frac{(-1)^k \cdot k!}{\sqrt{x} + \sin x}$	**24**	$y = \prod_{k=1}^{5} \frac{(-1)^k (k)!}{x^{k+2}}$
25	$y = \sum_{k=1}^{6} \frac{(-1)^k \cdot x^{4k+1}}{(2k)!}$	**26**	$y = \sum_{k=1}^{5} \frac{(-1)^k \cdot x^{k+2}}{(k)!}$
27	$y = \sum_{k=1}^{6} (-1)^k \frac{x^k}{k!}$	**28**	$y = \sum_{i=1}^{10} \frac{(2i-1) x^{i+1}}{2i!}$
29	$y = \sum_{i=1}^{7} \frac{(i+1) x^i}{i!}$	**30**	$y = \sum_{i=1}^{5} \frac{(-1)^i x^{2i}}{(i+1)! \cos x}$

Реалізовану програму **C++, Python** протестувати з вхідними даними різного типу: цілого та дійсного. Передбачити в реалізованому алгоритмі відповідні коментарі.

Завдання 8. За умовою завдання 1-7 розробити комбіновані алгоритми з множинним вибором:

— циклічних алгоритмів розрахунку лінійної функції;
— циклічних алгоритмів розрахунку розгалужених функцій;
— обчислення суми заданої функції;
— числових послідовностей;
— економічної задачі;
— обчислення суми заданого ряду.

Реалізовану програму протестувати з вхідними даними різних типів, вивести на екран проміжні та кінцеву результати розрахунків.

Зміст звіту

1. Титульна сторінка, тема та мета практичного завдання.
2. Завдання відповідно до номеру наданого варіанту
3. Опис виконаних завдань, лістинги реалізованих програм, результати виконання програм, блок-схеми, структурограми.
4. Реалізований алгоритм подати наступними способами: природною мовою в структурованому вигляді; формульно-словесною мовою; псевдокодом на мові C++. блок-схемами, реалізовані онлайн-сервісами ***Lucidchart, Draw.io, MS Visio***, тощо.
5. Висновки, які відображають результати виконаних завдань та їх аналіз.

Теоретичні відомості

Циклічні алгоритми в мові програмування C++

Оператори циклу. Цикл задає багаторазове проходження по тому самому коду програми (ітерації). Він має точку входження, перевірочну умову і (необов'язково) точку виходу. Цикл, що не має точки виходу, називається **нескінченним**. Для нескінченного циклу перевірочна умова завжди є дійсним значенням.

Перевірка умови може здійснюватися перед виконанням (цикли **for**, **while**) або після закінчення (**do-while**) тіла циклу. Цикли можуть бути вкладеними один в одного довільним чином.

Цикл for. Оператор циклу for використовують, як правило, у тому випадку, коли відома кількість повторень циклу.

Синтаксис циклу for:

```
for ([вираз1] ; [вираз2] ; [вираз3])
                                оператор;    // Тіло циклу
```

Цикл **for** оператор працює в такий спосіб:

- спочатку виконується **вираз1**, якщо він присутній у конструкції;
- потім, на початку кожної ітерації, обчислюється **вираз2** (якщо він присутній) і, якщо отриманий результат прийняв значення ***true***,

виконується *тіло циклу* (оператор або блок операторів). А якщо ні, то виконання циклу припиняється й здійснюється перехід до оператора, що розташований безпосередньо за тілом циклу. У якості **вираз2** звичайно використовують вираз логічного типу;

- після виконання *тіла циклу* наприкінці кожної ітерації обчислюється **вираз3**, якщо він є в конструкції, і здійснюється перехід до пункту обчислення **вираз2**.

Вираз1 найчастіше служить у якості ініціалізації якої-небудь змінної, що виконує роль лічильника ітерацій.

Вираз2 використовується як перевірочна умова, на практиці часто містить вираз з операторами порівняння. За замовчуванням величина **вираз2** приймає дійсне значення.

Вираз3 служить найчастіше для збільшення значення лічильника циклів або містить вираз, що впливає на перевірочну умову.

Усі три вирази не обов'язково повинні бути присутні у конструкції, однак синтаксис не допускає пропуску символу крапка з комою (**;**). Тому найпростіший приклад нескінченного циклу **for** (виконується постійно до примусового завершення програми) виглядає в такий спосіб:

```
for ( ; ; ) cout << "Нескінченний цикл...'\n'";
```

Якщо в циклі повинні синхронно змінюватися декілька змінних, які залежать від змінної циклу, обчислення їх значень можна помістити в оператор **for**, скориставшись оператором "кома".

Типова помилка програмування циклів **for** – зміна значення лічильника як у конструкції (**вираз3**), так і в тілі циклу. Це може приводити до таких негативних наслідків, як "випадання" ітерацій.

Приклад 1. Знайти суму десяти цілих чисел, починаючи з 10.

```
int s = 0;
  for (int i=10; i<20; i++) s += i;
cout << "sum = " << s;
```

Як видно із прикладу, при ініціалізації був оголошений лічильник **i** з початковим значенням 10, що збільшується з кожною ітерацією на одиницю. Тіло циклу в цьому випадку складається з одного єдиного оператора, що додає до результуючої величини **s** значення лічильника **i** у даній ітерації. Код реалізації циклу має наступний вигляд мовою С++:

```
for (int s = 0, i = 10; i<20; s += i, i++);
```

АБО

```
for (int s = 0, i = 10; i < 20; s += i++);
```

Таким чином, спочатку ініціалізуються змінні **i** і **s**, а потім при кожній з 10-ти ітерації буде обчислюватися нове значення змінних **s** і **i**. Після завершення циклу на екран виводиться значення суми.

Цикл while. Оператор циклу while використовують, як правило, у тому випадку, коли невідомо, скільки ітерацій необхідно виконати. Оператор циклу while виконує оператор або блок доти, доки перевірочна умова (вираз) залишається дійсною.

Синтаксис циклу while:

while (*вираз***)**

оператор**;** *// Тіло циклу*

Якщо вираз є ненульовою константою, тіло циклу буде виконуватися завжди, отже, ми маємо справу з нескінченним оператором. Цикл також виявиться нескінченним, коли умова істинна і ніде далі в тілі циклу не змінюється. Якщо ж перевірочна умова повертає ***false***, здійсниться вихід із циклу й тіло оператора **while** буде пропущено.

Досить часто в якості виразу використовується оператор присвоєння. Тому що при цьому вертається деяке число, в операторі **while** фактично проводиться порівняння отриманого значення з нулем (слід нагадати, що нуль – еквівалент ***false***) з подальшим прийняттям рішення про вихід із циклу або його продовження.

Як і для оператора **for**, якщо в циклі повинні синхронно змінюватися декілька змінних, які залежать від змінної циклу, обчислення їх значень можна помістити в перевірочний вираз оператора **while**, скориставшись оператором ",".

Приклад 2. Обчислити значення функції **y = ax^2-sinx**, якщо **a = 10.5 та x є [-1; 2]; h_x = 0,5**.

Текст програмного коду

```
#include <iostream>
#include <cmath>
#include <conio.h>                    // для функції getch()
using namespace std;

int main()
{
   float a = 10.5, x, y;
   x = -1 ;
   while (x <= 2)
   {
    y=a *pow(x,2) - sin(x);          // розрахунок функції y = a* x* x- sin(x);
    cout << "x= " << x << "\t y= " << y << endl;
    x += 0.5;
   }
   getch();                          // для затримки виводу на екран
   return 0;
}
```

В результаті отримаємо значення заданої функції (рис.18) в інтервалі від -1 до 2 реалізованої в циклі з передумовою.

```
x= -1    y= 11.3415
x= -0.5  y= 3.10443
x= 0     y= 0
x= 0.5   y= 2.14557
x= 1     y= 9.65853
x= 1.5   y= 22.6275
x= 2     y= 41.0907
```

Рисунок 18. – Результат виконання програми за прикладом 2.

Цикл do-while. На відміну від оператора while, цикл do-while спочатку виконує тіло (оператор або блок), а потім уже здійснює перевірку виразу на істинність. Така конструкція гарантує, що тіло циклу буде обов'язково виконане хоча б один раз.

Синтаксис циклу do-while:

```
do
оператор;    // Тіло циклу
while (вираз);
```

Одне із часто використовуваних застосувань даного оператора – запит до користувача на продовження виконання програми.

Приклад 3. Реалізувати циклічний процес виконання програми з підтвердженням користувача (Y/N), використовуючи цикл з постумовою.

Текст програмного коду

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
   char answer;
   do
   {                    //Тіло програми
   cout << "Продовжити виконання?\n";
   cin >> answer ;
   }                    //якщо користувач натиснув клавішу Y – продовження циклу
   while ( answer=='Y' || answer=='y' );
   getch();
   return 0;
}
```

Таким чином, тіло програми буде повторюватися доти, поки користувач на запитання "*Продовжувати виконання*?" не відповість введенням будь-якого символу, крім 'Y' або 'y'. При цьому, у якості змінної, яка зберігає символьну величину, виступає змінна **answer**.

Оператор break. Для дострокового виходу з оператору циклу, використовується оператор break. Даний оператор може зустрічатися в тілі циклу скільки завгодно раз і, як і у випадку switch-case, передає керування поза тілом конструкції. На практиці оператор break часто використовують для виходу з нескінченного циклу.

```
for ( ; ; )
{
   // Тіло циклу
   if (<умова1>) break; if (<умова2>) break;
}
```

У даному прикладі, якщо хоч один із умовних операторів поверне ***true*** відповідний оператор **break** виведе керування з тіла конструкції нескінченного циклу **for**.

Оператор continue. Так само, як і ключове слово break, оператор continue перериває виконання тіла циклу, але на відміну від першого, він пропонує програмі перейти на наступну ітерацію циклу.

Як приклад використання оператора **continue** пропонується програма знаходження простих чисел (*просте число* – число, яке ділиться на 1 і саме на себе).

Приклад 4. Вивести на друк прості числа в інтервалі від 2 до 20

Текст програмного коду

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
   bool simple;                    // прапор простого числа
                                   // для всіх чисел від 2 до 20
   for (int i = 2; i < 20; i++)
   {
   simple = true ;
                                   // для всіх дільників від 2 до i-1
   for (int d = 2; d < i; d++)
   {
   if (i % d)                      // якщо і не ділиться націло на d
   continue;                       // завершення ітерації
   else                            // якщо і ділиться націло на d
   {
   simple = false;                 // число і – непросте
   break; } }                      // вихід із внутрішнього циклу з параметром d
   if (simple)                     // якщо число просте
   cout <<i<<'\t';                 // вивід на екран числа і
   }
   getch();
   return 0; }
```

Програма організована у вигляді двох вкладених циклів таким чином, що здійснюється перебір і перевірка залишку від розподілу пари чисел, перше з яких змінюється від 2 до 20, а друге – від 2 до значення першого числа. На початку кожної ітерації в зовнішньому циклі прапор простого числа **simple** установлюється в стан ***true***. Якщо залишок від розподілу не буде нульовим, здійснюється продовження внутрішнього циклу по оператору **continue**. У випадку, якщо залишок від розподілу склав 0, виконується вихід із внутрішнього циклу з установкою прапора простого числа **simple** у стан ***false***. По закінченню внутрішнього циклу проводиться аналіз логічної змінної **simple** і вивід простого числа.

Оператор переходу goto і мітки. ***Мітка*** являє собою ідентифікатор з розташованим за ним символом двокрапки (:). Мітками позначають який-небудь оператор, на який надалі повинен бути здійснений безумовний перехід. Безумовна передача керування на мітку проводиться за допомогою оператора goto. Оператор goto може здійснювати перехід (адресуватися) до міток,

обов'язково розташованих в одному з ним тілі функції.

Синтаксис оператора goto:

goto мітка; мітка : оператор**;**

Даний оператор – дуже потужний і небезпечний засіб керування поведінкою програми. Використовувати його потрібно обережно, тому що, наприклад, перехід всередину циклу (в обхід ініціалізації) може привести до помилок, які важко локалізувати. За допомогою операторів **if** і **goto** можна реалізувати будь-який алгоритм, але читати й проводити налагодження коду, перенасиченого операторами переходу й мітками, надзвичайно важко. Тому використання оператора переходу вважається поганим стилем програмування.

Приклад 5. Застосування оператора **goto** для організації циклу, у якому підсумовується значення чисел від 10 до 20.

Текст програмного коду

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
   int i = 10; int s = 0;
   label:
      s += i;
      i++;
      if (i < 21) goto label;
   cout << "s=" << s <<'\n';
   getch();
   return 0; }
```

Як тільки виконання програми досягне оператора **goto**, керування буде передано оператору, що стоїть за міткою **label**.

Загалом, використання структурного і об'єктно-орієнтованого підходів до програмування дозволяє повністю відмовитися від застосування операторів безумовного переходу. Однак на практиці часто бувають випадки, коли **goto** значно спрощує код програми. Особливо це твердження стосується вкладених конструкцій **switch-case** і **if-else**.

Циклічні алгоритми в мові програмування Python

Досить часто і в житті, і при написанні програм існує необхідність повторення деякої дії певної кількості раз. Тобто при написанні програм може знадобитися певна конструкція, за якою можна буде організувати повторне виконання операторів. Таку конструкцію називають конструкцією повторення або циклом. А кожну повторену дію – кроком циклу або ітерацією. Отже, можна зазначити, що цикл у програмуванні – це повторюване виконання одних і тих самих простих або складених операторів.

Всі цикли складаються з заголовку та тіла циклу. Заголовок циклу відповідає за налагодження циклу, тобто умову повторення циклу. Тіло ж циклу відповідає за самі дії, які мають повторно виконуватися. Так наприклад, уявімо собі першокласника, який дуже полюбляє морозиво. Йому мама видала певну

суму грошей на морозиво. Зрозуміло, він побіг його купувати, але згадав, що множити та ділити він не вміє. Як же йому вирішити цю проблему? Спочатку він перевірить, чи вистачить йому грошей на купівлю пачки морозива. Якщо так, то він її купить і знову погляне на залишок грошей. В цьому прикладі можна виділити заголовок циклу – поки грошей достатньо, і тіло циклу – купівля морозива.

Цикл з передумовою (Цикл while)

Цикл з передумовою є одним з самих універсальних циклів в мові **Python**, але достатньо повільний. Цикл є циклом з передумовою, оскільки умова записується і перевіряється до тіла циклу. Цикл з передумовою ще називають циклом **While**, оскільки саме з цього ключового слова він починається.

Синтаксис оператора циклу while:

```
while Логічний_вираз:
    Блок_інструкцій
```

Логічний вираз також називають умовою виконання циклу, а блок інструкцій – тілом циклу, яке може містити довільні оператори. За циклом while виконується вказаний набір інструкцій до тих пір, поки умова циклу істинна.

При виконанні циклу while спочатку обчислюється значення логічного виразу, якщо це значення є істинним (істинність умови визначається так само як і в операторі if), то виконується тіло циклу і відбувається повернення до перевірки логічного виразу. Процес продовжується доти, поки значення логічного виразу не стане хибним. Після цього робота циклу завершиться і відбувається перехід до інструкції після тіла циклу while.

Якщо при першому обчисленні значення логічного виразу є хибним, тіло циклу не виконується жодного разу Щоб цикл закінчив роботу, в його тілі повинен бути оператор, що впливає на значення логічного виразу. Окрім того логічний вираз має бути коректним, тобто його значення повинно бути визначеним ще до першої перевірки.

Як правило цикл while використовується, коли неможливо визначити точне значення кількості проходів використання циклу.

Приклад 6. Обчислити значення виразу з точністю до 10^{-6}.

$$\sum_{n=1}^{\infty} \frac{1}{n^2}$$

В даному прикладі накопичення суми виконується до тих пір, поки черговий доданок не стане менше заданої точності обчислень.

Текст програмного коду

```
s=0
n=1
x=1
while abs(x)>=1e-6:
  s=s+x
  n=n+1
  x=1/n**2
print('%.3f' % s) #формат числа, 3 знаки після коми
```

В результаті отримаємо накопичену суму: 1.644

Приклад 7. Визначити кількість цифр введено натурального числа n.

Текст програмного коду

```
n = int(input())
length = 0
while n > 0:
  n = n // 10
  length = length + 1
print(length)
```

Приклад 8. Обчислити суму непарних додатних чисел, менших за введене значення n.

Текст програмного коду

```
n=int(input())
s=0
i=1
while i<n:
  s=s+i
  i=i+2
print('Сума непарних чисел =',s)
```

В результаті отримаємо суму непарних чисел наведених на рис. 19

```
28
Сума непарних чисел = 196
```

Рисунок 19. – Результат виконання програми за прикладом 6.

Тип діапазон (range). Тип діапазон (range) є незмінюваною послідовністю цілих чисел. Для задання діапазону призначені функції:

- **range(stop)** – задання послідовності цілих чисел від 0 до stop-1 з кроком 1.
- **range(start, stop[, step])** - задання послідовності, яка є арифметичною прогресію від start до stop-1 з кроком step. Якщо параметр step опущений, він за замовчуванням дорівнює 1.

За функцією range(5) отримаємо діапазон в з елементів 0, 1, 2, 3, 4

За функцією range(1, 5) отримаємо діапазон в з елементів 1, 2, 3, 4

За функцією range(0, 10, 3) отримаємо діапазон в з елементів 0, 3, 6, 9 Для отримання діапазону, в якому значення будуть зменшуватися, необхідно використовувати функцію range з трьома параметрами. Третій параметр має бути від'ємним, а перший більшим ніж другий.

За функцією range(0, -5, -2) отримаємо діапазон в з елементів 0, -2, -4 Проте вивести на екран елементи утвореного діапазону звичайними методами неможливо (рис. 20).

```
r = range(1, 5)
r
range(1, 5)

range(1, 5)
```

Рисунок 20. – Результат виведення діапазону даних функцією range.

Можна перевірити належність деякого числа до діапазону (рис. 21), використовуючи оператор **in**.

Рисунок 21. – Результат перевірки належності діапазону.

Цикл for. Окрім циклу з передумовою, в мові Python є цикл **for**, за яким надається можливість перебору всіх елементів з деякого набору (послідовності, бінарної послідовності, рядка, множини, словника, файлу). В загальному можна зазначити, що використовуваним набором в циклі **for**, може будь який набір, що підтримує ітерації. Перебір елементів можна пояснити так. У нас є набір, що складається з ряду елементів. При переборі ми спочатку беремо з даного набору перший елемент, і в тілі циклу виконуємо над ним визначені дії. Потім беремо другий елемент, і над ним знову виконуємо ті ж дії. І так далі продовжуємо над всіма елементами набору. При такому опрацюванні не потрібно турбуватися про індекси елементів і їх кількість.

Іншими словами, можна зазначити, цикл for являє собою формальний запис інструкції виду: «Виконати операцію X для всіх елементів, що входять в набір M». Тому цикл for інколи називають циклом перегляду.

Синтаксис оператора циклу for:

for Ідексна_змінна **in** Послідовність:

Блок_інструкцій

На початку індексній змінній надається значення першого елемента послідовності, потім виконується тіло циклу (блок інструкцій) і індексній змінній надається значення наступного елемента послідовності. Так продовжується доти, поки індексній змінній послідовно не будуть надані значення всіх елементів послідовності. Тобто індексна змінна буде пробігати всі елементи послідовності.

Як правило, цикли for використовуються для виконання операцій над всіма елементами послідовності або виконання операцій таку кількість разів, яка відповідає кількості елементів в послідовності.

Цикл **for** дещо складніший і менш універсальний, але виконується значно швидше циклу **while**.

Приклад 9. Вивести на екран квадрати введених додатних цілих чисел, менших за значення n.

Текст програмного коду

```
n=int(input())
for i in range(1,n):
  print(i**2)
```

Якщо значення змінної **n** буде рівне нулю або від'ємне, то в тілі цикл не виконається жодного разу.

Приклад 10. Для заданого користувачем значення n обчислити суму заданого ряду:

$$\sum_{i=1}^{n} \frac{i}{2i+1}$$

Текст програмного коду

```
n=int(input())
s=0
for i in range(1,n+1):
  s=s+i/(2*i+1);
print('Сума =', '%.3f' % s)
```

Після виконання програми отримаємо результат:

```
7
Сума = 2.989
```

Приклад 11. Обчислити суму непарних додатних чисел з проміжку [n; m].

Текст програмного коду

```
n=int(input())
if n%2==0:n=n+1
m=int(input())
s=0
for i in range(n,m+1,2):
  s=s+i
print('Сума непарних чисел =',s)
```

Після виконання програми отримаємо результат, наведений на рис. 22.

```
10
20
Сума непарних чисел = 75
```

Рисунок 22. – Результат обчислення суми непарних додатних чисел.

В даній програмі спочатку уточнюється початок проміжку таким чином, щоб початком було перше непарне число з заданого проміжку. Потім за функцією **range(n,m+1,2)** формується діапазон всіх непарних чисел з заданого проміжку і відбувається їх підсумовування.

Інструкції управління циклами Оператор continue. Оператор ***continue*** призначений для переривання поточної ітерації циклу і переходу до наступної. Тобто оператори, що будуть іти в тілі циклу після виклику **continue**, на даному кроці виконуватися не будуть.

Приклад 12. Обчислити суму непарних додатних чисел з проміжку **[n; m],** не включати до суми числа, які діляться без залишку на 5

Текст програмного коду

```
n=int(input())
if n%2==0:n=n+1
m=int(input())
s=0
for i in range(n,m+1,2):
  if i%5==0:
    continue
```

```
s=s+i
print('Сума непарних чисел =',s)
```

Після виконання програми отримаємо результат, наведений на рис. 23.

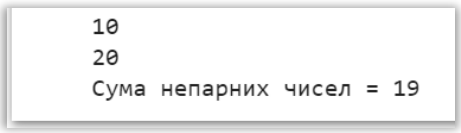


Рисунок 23. – Результат виконання програми за прикладом 12

При виконанні тіла циклу, коли значення змінної і буде ділитися націло на 5 (залишок від цілочисельного ділення і на 5 дорівнює 0), відбувається перехід на наступний елемент послідовності.

Оператор break. Оператор ***break*** призначений для дострокового припинення роботи циклу (**for** або **while**), тобто зупинки виконання тіла циклу, навіть якщо умова виконання циклу ще не набула значення **False** або послідовність елементів не закінчилась.

Зрозуміло, інструкцію **break** варто викликати тільки всередині інструкції **if**, тобто вона повинна виконуватися тільки при виконанні якоїсь особливої умови.

Приклад 13. Обчислити суму непарних додатних чисел з проміжку **[n; m]**. При додаванні до суми числа, яке ділиться без залишку на 5, необхідно припинити підрахунок суми.

Текст програмного коду

```
n=int(input())
if n%2==0:n=n+1
m=int(input())
s=0
for i in range(n,m+1,2):
  s=s+i
  if i%5==0:
    break
print('Сума непарних чисел =',s)
```

Після виконання програми отримаємо результат, наведений на рис. 24.

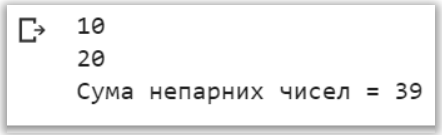


Рисунок 24. – Результат виконання програми за прикладом 13

Блок else в циклах. Блок ***else*** може використовуватися як додатковий блок циклів while та for, сфера застосування якого досить схожа до застосування однойменного блоку в конструкціях обробки винятків (**try-except**) та умовного оператора (**if-else**), тобто «якщо цього виконати не можна, то (інакше) виконати це».

Синтаксис операторів циклу з блоком else:

while Логічний_вираз:
    Блок_інструкцій_1

```
else:
    Блок_інструкцій_2
```

або

```
for Ідексна_змінна in Послідовність:
    Блок_інструкцій_1
else:
    Блок_інструкцій_2
```

Інструкція всередині блоку else (Блок_інструкцій_2) виконується в тому випадку, коли цикл завершився згідно з умовою повторення циклу, вказаною в заголовку циклу (для циклу while у випадку, коли умова циклу стала хибною, для циклу for – коли були перебрані всі елементи послідовності).

Іншим варіантом завершення циклу є вихід з циклу за виконанням оператору break і в такому випадку інструкції блоку else не виконуються. Отже використання else доцільне тільки разом з інструкцією break.

Приклад 14. Обчислити суму непарних додатних чисел з проміжку [n; m]. При додаванні до суми числа, яке ділиться націло на 5, припинити розрахунок суми. Вивести додаткове повідомлення, якщо результуюча сума містить доданок, який ділиться націло на 5.

Текст програмного коду

```
n=int(input())
if n%2==0:n=n+1
m=int(input())
s=0
for i in range(n,m+1,2):
  s=s+i
  if i%5==0:
    break
else:
  print('Сума не містить доданку, що ділиться на 5')
print('Сума непарних чисел =',s)
```

Після виконання програми отримаємо результат, наведений на рис. 25

```
1
4
Сума не містить доданку, що ділиться на 5
Сума непарних чисел = 4
```

Рисунок 25. – Результат виконання програми за прикладом 14

Вкладені цикли. В усіх операторах циклу мови Python оператор, який є тілом циклу, може сам бути оператором циклу або містити у собі оператор циклу. Утворена конструкція називається вкладеним циклом.

Приклад 15. За заданим натуральним числом **n ≤ 9** вивести проходження шляху по драбині з **n** сходинок, де і-а сходинка складається з чисел від 1 до **i**.

Текст програмного коду

```
n=int(input('Вкажіть кількість сходинок n='))
for i in range(1,n+1):
```

```
for j in range(1,i+1):
  print(j,end='')
print()
```

Після виконання програми отримаємо результат, наведений на рис. 26.

```
Вкажіть кількість сходинок n=5
1
12
123
1234
12345
```

Рисунок 26. – Результат виконання програми за прикладом 15

Приклад 16. Обчислити суму ряду заданого виразу:

$$\sum_{i=1}^{10}\sum_{j=1}^{5}\frac{1}{i+j^2}$$

Текст програмного коду

```
s=0
for i in range(1,11):
 for j in range(1,6):
   s=s+1/(i+j**2);
print('Сума =', '%.5f' % s)
```

Після виконання програми отримаємо результат: `Сума = 4.71139`

Контрольні запитання

1. Зазначте всі типи операторів циклу в мові С++. Які бувають цикли? У чому їх відмінність?
2. З якою метою використовується в циклі оператор break в мові С++?
3. З якою метою використовується в циклі оператор continue в мові С++?
4. Чим відрізняється оператор do while від інших операторів циклу в мові програмування С++?
5. Формат запису циклу while в мові С++.
6. Формат запису циклу do-while в мові С++.
7. Поясність поняття та застосування інкремента в мові С++.
8. Які існують унарні операції в мові програмування С++?
9. Поясніть сутність постфіксної форми в мові С++.
10. Поясність поняття та застосування декремента в мові С++.
11. Поясніть сутність префіксної форми в мові С++.
12. Що таке цикл, для чого його використовують в мові Python?
13. Як описується та виконується циклічна інструкція while мові Python?
14. Формат запису циклу for мові Python.
15. Формат запису циклу while мові Python.
16. Як можна організувати нескінченні цикли мові Python? Наведіть декілька прикладів і поясніть їх.
17. Для чого служать оператори переривання break та continue мові Python?

Практичне завдання 3.
Реалізація алгоритмів з використанням лінійних структур даних

Мета заняття: набути практичних навичок по обробці лінійних структур даних мовами програмування **C++**, **Python;** ознайомитися з особливостями визначення та використання одновимірних, двовимірних та багатовимірних масивів, їх структурною організацією та способами доступу до відповідних елементів; створювати консольні проекти; редагувати та компілювати програмні коди; реалізовувати алгоритми графічно в онлайн-сервісах **Lucidchart**, **Draw.io.**

Завдання 1. Розробити алгоритми, за допомогою яких в одновимірний масив будуть додаватись **10-ть** елементів: сталими величинами / користувачем в процесі виконання програми / випадковими величинами.

В алгоритмах необхідно забезпечити:

— виведення на екран заданого одновимірного масиву;

— виведення на екран значень параметра ***a*** та результативного одновимірного масиву обчисленої функції відповідно до варіанту (***табл. 1***).

Розроблені програми на мові **C++ (Python)** протестувати з вхідними даними різного типу. В програмах передбачити перевірку можливих помилок:

— ділення на нуль;

— підкореневий від'ємний вираз;

— логарифми з від'ємними значеннями;

— невизначеність функцій (вважається, що функція визначена на заданому проміжку, а поза її межами - не визначена);

— забезпечити своєчасне виведення повідомлень до відповідних помилок;

— результат обчислень заданої функції вивести на кінцевому етапі алгоритму.

Завдання 2. Розробити алгоритм реалізації та розрахунку значень в одновимірному масиві n-ї розмірності відповідно до заданого в ***табл. 8*** варіанту.

Таблиця 8. - Варіанти задач до завдання 2

№ вар.	*Умова задачі*
1.	Знайти і надрукувати кількість додатних елементів масиву
2.	Підрахувати і надрукувати кількість додатних елементів, які розташовані на парних місцях в масиві
3.	Знайти мінімальний елемент масиву
4.	Вивести на друк середнє арифметичне від'ємних елементів масиву.
5.	Надрукувати суму від'ємних елементів, які стоять на парних місцях в масиві.
6.	Надрукувати середнє арифметичне невід'ємних елементів масиву, які стоять на непарних місцях
7.	Знайти та надрукувати кількість додатних елементів масиву
8.	Обчислити добуток додатних елементів масиву
9.	Знайти суму всіх елементів масиву, значенняяких менше 0,25.

№ вар.	*Умова задачі*
10.	Обчислити добуток модулів значень елементів масиву
11.	Визначити кількість елементів масиву значення яких більші за 0,99.
12.	Визначити кількість від'ємних елементів масиву
13.	Обчислити добуток елементів масиву, значення яких більше 2.0.
14.	Надрукувати порядкові номери від'ємних елементів масиву
15.	Підрахувати кількість елементів масиву, значення яких більше 2,3.
16.	Обчислити добуток елементів масиву, значення яких більші 5,4.
17.	Обчислити суму значень від'ємних елементів масиву
18.	Визначити номери додатних елементів масиву
19.	Обчислити добуток всіх елементів масиву
20.	Визначити мінімальний елемент масиву та його порядковий номер.
21.	Надрукувати номер першого від'ємного елемента масиву
22.	Визначити номер максимального елемента масиву
23.	Вивести парні номери від'ємних елементів масиву
24.	Знайти квадрат мінімального елементу масиву
25.	Визначити максимальний по модулю елемент масиву

В алгоритмі необхідно забезпечити:

— виведення на екран початково введеного одновимірного масиву;

— виведення на екран n-ї розмірності масиву та результативних функцій.

Завдання 3. Розробити алгоритм реалізації та розрахунку значень в двовимірному масиві відповідно до заданого варіанту.

В алгоритмі необхідно забезпечити:

— виведення на екран початково введеного одновимірного масиву;

— виведення на екран n-ї розмірності масиву та результативних функцій.

Таблиця 9. - Варіанти задач до завдання 3

№ вар.	*Умова задачі*
1.	У цілочисельній матриці A(n,n) (n<=5) знайти найменший спільнийдільник головної діагоналі.
2.	У матриці A(m,n) (m<=5, n<=7) визначити середні арифметичні значення елементів стовпців.
3.	У матриці A(m,n) (m<=5, n<=4) визначити максимальні елементикожного стовпця та їх номери.
4.	У матриці A(m,n) (m<=5, n<=5) визначити добуток максимального тамінімального елементів.
5.	У матриці A(m,n) (m<=5, n<=8) визначити різниці максимального тамінімального елементів кожного стовпця.
6.	У матриці A(m,n) (m<=5, n<=7) знайти добуток елементів стовпця, вякому знаходиться максимальний елемент.
7.	У матриці A(m,n) (m<=7, n<=4) визначити найменший та найбільшийелементи кожного рядка.
8.	У матриці A(m,n) (m<=5, n<=5) знайти добуток ненульових елементів,які лежать на головній діагоналі.

№ вар.	Умова задачі
9.	У матриці A(m,n) (m<=7, n<=5) знайти суму максимальних елементів їїрядків та їх індекси.
10.	У цілочисельній матриці A(m,n) (m<=5, n<=6) визначити мінімальніелементи кожного стовпця та їх індекси.
11.	Замінити всі парні елементи цілочисельної матриці A(m,n) (m<=3,n<=6) нулями.
12.	У матриці A(m,n) (m<=7,n<=5) визначити суми та добутки елементіврядків.
13.	Замінити всі елементи матриці A(m,n) (m<=5, n<=4), сума індексіваких парна, добутками відповідних індексів.
14.	Піднести до квадрату всі від'ємні елементи матриці A(m,n) (m<=5,n<=5), які лежать вище головної діагоналі.
15.	Знайти середнє арифметичне додатних елементів матриці A(m,n)(m<=6, n<=6), які лежать нижче головної діагоналі.
16.	Знайти суму елементів масиву A(m,n) (m<=3, n<=5), які мають хоча б один непарний індекс.
17.	Визначити номери елементів масиву A(m,n) (m<=6, n<=2), модуль якихбільший 5.
18.	Визначити мінімальний по модулю елемент та його індекси в масиві A(m,n) (m<=7, n<=5).
19.	У матриці A(n,n) (n<=5) знайти максимальні елементи кожного стовпця.
20.	У матриці A(n,n) (n<=7) знайти кількість від'ємних елементів, що лежать нижче головної діагоналі.
21.	У матриці A(m,n) (m<=5, n<=5) знайти суму елементів головної та побічної діагоналей.
22.	Знайти суму елементів матриці A(m,n) (m<=5, n <=5), які стоять над і під головною діагоналлю.
23.	Просумувати елементи кожного стовпця матриці A(n,n) (n<=4). Отримані суми роздрукувати, занести в масив і вибрати серед них максимальне.
24.	У матриці A(n,n) (n<=5) впорядкувати другий рядок за зростанням.Вивести стару та нову матриці.
25.	Задана матриця A (n,n) (n<=5). Отримати нову матрицю шляхом ділення всіх її елементів на найбільший по модулю від'ємний елемент.

Реалізовані програми на мові **C++ (Python)** протестувати з вхідними даними різного типу.

Завдання 4. Відповідно до варіанту, розробити алгоритм реалізації та розрахунку даних заданого тензора. Дані в тензор заносяться: сталими величинами / користувачем в процесі виконання програми; / випадковими величинами.

Таблиця 10. - Варіанти задач до завдання 4

№ вар.	Умова задачі
1.	В тензорі T(2x2x3) знайти суму всіх додатних елементів.
2.	Дано тензор T(2x3x2), в якому необхідно знайти квадрат найбільшого елемента. $$T(2x3x2) = \left\{\begin{pmatrix}1 & 0 & 2\\ 3 & -1 & 1\end{pmatrix}, \begin{pmatrix}3 & 0 & 4\\ 5 & 2 & 3\end{pmatrix}\right\}$$
3.	Дано тензор T(3x3x2). Визначити суму всіх елементів тензора.
4.	В тензорі знайти різницю між максимальним та мінімальним елементом. $$T(2x3x2) = \left\{\begin{pmatrix}1 & 0 & 2\\ 3 & 1 & -4\end{pmatrix}, \begin{pmatrix}3 & 0 & 4\\ 5 & -2 & 3\end{pmatrix}\right\}$$

№ вар.	*Умова задачі*
5.	Дано тензор T(2x3x2), в якому необхідно вивести суму додатних та від'ємних елементів, підрахувавши при цьому їх кількість. $T(2x3x2) = \left\{\begin{pmatrix} -1 & 0 & 8 \\ 2 & -1 & 1 \end{pmatrix}, \begin{pmatrix} 3 & -5 & 1 \\ 7 & 2 & -3 \end{pmatrix}\right\}$
6.	Дано тензор T(2x3x2), в якому необхідно знайти суму добутків елементів кожного рядка тензора. $T(2x3x2) = \left\{\begin{pmatrix} 5 & 1 & 0 \\ 8 & 2 & -2 \end{pmatrix}, \begin{pmatrix} 2 & 5 & 4 \\ -4 & 1 & -3 \end{pmatrix}\right\}$
7.	Дано тензор T(2x3x2), в якому необхідно піднести до квадрату всі елементи тензора. $T(2x3x2) = \left\{\begin{pmatrix} 10 & 1 & 0 \\ 6 & 2 & -2 \end{pmatrix}, \begin{pmatrix} 2 & 5 & 1 \\ -4 & 3 & -3 \end{pmatrix}\right\}$
8.	В тензорі T(2x3x2) піднести до кубу всі елементи, значення яких більші за 3. $T(2x3x2) = \left\{\begin{pmatrix} 8 & 5 & 3 \\ -2 & 2 & -2 \end{pmatrix}, \begin{pmatrix} 2 & -4 & 1 \\ 0 & 3 & -7 \end{pmatrix}\right\}$
9.	Дано тензор T(2x3x2). Піднести до кубу додатні елементи тензора.
10.	Дано тензор T(2x3x2). Піднести до кубу від'ємні елементи тензора. $T(2x3x2) = \left\{\begin{pmatrix} -1 & 0 & -8 \\ -5 & 3 & -2 \end{pmatrix}, \begin{pmatrix} 8 & 5 & 1 \\ 3 & 0 & -3 \end{pmatrix}\right\}$
11.	В тензорі T(3x3x4) піднести до квадрату всі елементи, значення яких менші за 5.
12.	В тензорі знайти добуток елементів його діагоналей. $T(2x3x2) = \left\{\begin{pmatrix} -2 & 17 & 2 \\ 3 & 28 & 1 \end{pmatrix}, \begin{pmatrix} 3 & 0 & 4 \\ 5 & 25 & 3 \end{pmatrix}\right\}$
13.	Дано тензор T(2x3x2). Визначити добуток всіх елементів тензора. $T(2x3x2) = \left\{\begin{pmatrix} 9 & 7 & 5 \\ 3 & -1{,}5 & -4 \end{pmatrix}, \begin{pmatrix} 3 & 0 & 4 \\ 5 & -2{,}5 & 3 \end{pmatrix}\right\}$
14.	Дано тензор T(2x3x2). Визначити добуток всіх непарних елементів тензора. $T(2x3x2) = \left\{\begin{pmatrix} -1{,}5 & 0 & 2 \\ 3{,}2 & -1 & 1 \end{pmatrix}, \begin{pmatrix} 3 & 0{,}5 & 4 \\ 5 & 2{,}5 & 3 \end{pmatrix}\right\}$
15.	Дано тензор T(3x2x2). Визначити суму парних елементів тензора.
16.	В заданому тензорі визначити кількість додатних елементів. $T(2x3x2) = \left\{\begin{pmatrix} 5 & 1 & -2 \\ 8 & 2 & 4{,}7 \end{pmatrix}, \begin{pmatrix} 3 & -0{,}3 & 4 \\ 5 & 2{,}2 & 3 \end{pmatrix}\right\}$
17.	В заданому тензорі визначити кількість від'ємних елементів. $T(2x3x2) = \left\{\begin{pmatrix} 10 & 2{,}7 & 2 \\ 3 & -1{,}5 & 1 \end{pmatrix}, \begin{pmatrix} 3{,}2 & 0 & 4 \\ 5 & 25 & 3 \end{pmatrix}\right\}$
18.	В заданому тензорі T(2x3x2) знайти кількість нульових значень. $T(2x3x2) = \left\{\begin{pmatrix} -8 & 5{,}5 & 2 \\ 3{,}2 & -1 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 10 & -4 \\ 0 & 3 & -7 \end{pmatrix}\right\}$
19.	В тензорі T(4x4x4) знайти кількість елементів, значення яких більші за 7.
20.	В тензорі T(3x3x3) знайти кількість додатних елементів.

В алгоритмах необхідно забезпечити:

— виведення на екран початково введеного тензора;

— виведення на екран результативного тензора та результативних функцій.

Реалізовані програми на мові **C++ (Python)** протестувати з вхідними даними різного типу.

Завдання 5. Розробити програму мовою **C++, Python** відповідно до варіанту (***табл. 4***), в якому необхідно оголосити одновимірний та/або багатовимірний масив сталих / випадкових / символьних величин розмірністю

10-ть елементів. В даній програмі забезпечити:

— виведення на екран відповідних вхідних даних;

— вивести на екран проміжні та кінцеві результати розрахованих значень у відповідному форматі.

Передбачити в реалізованому алгоритмі відповідні коментарі, перевірку на коректність введених даних.

Завдання 6. За умовою поданих завдань розробити комбіновані алгоритми з множинним вибором:

— роботи з одновимірними масивами;

— роботи з двовимірними масивами;

— роботи з тензором.

В даному алгоритмі необхідно забезпечити:

— виведення на екран відповідного меню вибору задач;

— структурувати на блоки: блок введення даних, блок обробки і блок виведення результатів;

— виведення на екран вхідних параметрів ***a***, змінних ***x,*** масивів;

— заданим змінним послідовно присвоїти значення одного типу даних та різних типів даних;

— вивести розраховані значення функцій на екран у відповідному форматі;

Реалізовані програми на мові **C++** та/або **Python** протестувати з вхідними даними різного типу.

Завдання 7. Ввести матрицю дійсних чисел розмірності 5x5 та за допомогою користувацької функції замінити елементи головної діагоналі на середнє арифметичне відповідного рядка.

Розробити програму мовою **C++**, **Python**, в якій необхідно забезпечити:

— виведення на екран відповідних вхідних даних;

— виведення на екран проміжних та кінцевих результатів розрахованих значень у відповідному форматі;

— користувацьку функцію реалізувати з прототипом/без прототипу (мова C++).

Передбачити в реалізованому алгоритмі відповідні коментарі, можливі помилки при введенні даних.

Завдання 8. Розробити програму мовою **C++**, **Python** відповідно до варіанту (***табл. 11***) з використанням користувацьких функцій для обробки багатовимірних масивів.

В даних програмних реалізаціях необхідно забезпечити:

— виведення на екран відповідних вхідних даних;

— виведення на екран проміжних та кінцевих результатів розрахованих значень у відповідному форматі;

— користувацьку функцію реалізувати з прототипом/без прототипу (**мова C++**).

Передбачити в реалізованому алгоритмі відповідні коментарі, можливі помилки при введенні даних.

Таблиця 11. - Варіанти задач до завдання 8

№ вар.	*Розмір масиву*	*Тип даних*	*Індивідуальне завдання*
1	5x5	цілий	Обчислити кількість від'ємних елементів матриці
2	4x4	дійсний	Обчислити суму елементів головної діагоналі матриці
3	6x4	цілий	Визначити найменший елемент матриці
4	3x3	дійсний	Обчислити добуток ненульових елементів матриці
5	4x5	цілий	Обчислити середнє арифметичне мінімального та максимального елементів матриці
6	3x5	дійсний	Обчислити кількість елементів, значення яких більше за значення першого елемента матриці
7	5x3	цілий	Обчислити середнє арифметичне елементів матриці
8	3x4	цілий	Визначити найменший серед парних додатних елементів
9	5x3	цілий	Обчислити суму та кількість парних елементів матриці
10	5x5	дійсний	Обчислити суму та кількість трицифрових елементів
11	4x6	цілий	Обчислити суму та кількість елементів матриці, які кратні 3
12	5x4	дійсний	Обчислити модуль суми всіх від'ємних елементів матриці
13	3x5	цілий	Визначити розміщення (індекси) мінімального та максимального елементів матриці
14	4x3	дійсний	Обчислити середнє арифметичне від'ємних елементів
15	6x4	цілий	Обчислити кількість ненульових елементів матриці
16	5x5	дійсний	Визначити максимальний і мінімальний елементи матриці
17	4x5	дійсний	Обчислити середнє арифметичне елементів матриці, значення яких належать проміжку [10, 20]
18	3x5	цілий	Визначити найменший елемент матриці
19	5x3	цілий	Обчислити добуток одноцифрових елементів матриці
20	3x4	дійсний	Обчислити суму модулів всіх від'ємних елементів матриці
21	3x3	цілий	Обчислити суму та кількість двоцифрових елементів
22	5x5	цілий	Обчислити середнє арифметичне елементів, кратних 5
23	4x6	цілий	Визначити найбільший з парних додатних елементів
24	5x4	дійсний	Обчислити суму додатних елементів і кількість від'ємних елементів матриці
25	3x4	цілий	Обчислити суму елементів матриці, значення яких належать проміжку [3, 6]
26	3x3	дійсний	Обчислити визначник матриці
27	6x4	дійсний	Обчислити кількість елементів, що перевищують середнє арифметичне всіх елементів
28	5x5	дійсний	Обчислити середнє арифметичне елементів неголовної діагоналі матриці
29	4x5	цілий	Обчислити суму елементів парних стовпців матриці
30	3x5	дійсний	Визначити мінімальний із додатних елементів матриці

Завдання 9. Розробити програму мовою **C++**, **Python** відповідно до варіанту (***табл. 12***) з використанням користувацьких функцій для обробки багатовимірних масивів.

В даних програмних реалізаціях необхідно забезпечити:

— виведення на екран відповідних вхідних даних;

— виведення на екран проміжних та кінцевих результатів розрахованих значень у відповідному форматі;

— користувацьку функцію реалізувати з прототипом/без прототипу (**мова C++**).

Таблиця 12. - Варіанти задач до завдання 9

№ вар.	*Розмір масиву*	*Тип даних*	*Індивідуальне завдання*
1	4x3	цілий	Замінити парні за значенням (не за індексом) елементи числом 0
2	6x4	дійсний	Поміняти місцями мінімальний і максимальний елементи
3	4x4	цілий	Поміняти місцями елементи головної та неголовної діагоналей матриці
4	4x5	дійсний	Замінити всі від'ємні елементи на значення мінімального
5	3x5	цілий	Обчислити суму додатних непарних елементів і замінити кутові елементи матриці на цю суму
6	5x3	цілий	Замінити всі нульові елементи значенням мінімального елемента
7	5x5	цілий	Замінити нулями всі елементи від початку і до найбільшого елемента
8	5x3	цілий	Замінити всі непарні елементи матриці одиницями
9	3x4	дійсний	Замінити нульові елементи на середнє арифметичне найменшого і найбільшого елементів
10	3x5	цілий	Замінити від'ємні елементи матриці нулями
11	4x6	дійсний	Замінити нулями ті елементи матриці, які більші за середнє арифметичне
12	5x5	цілий	Поміняти місцями елементи першого рядка матриці з елементами її неголовної діагоналі
13	4x4	дійсний	Транспонувати матрицю
14	3x5	дійсний	Замінити найменший та найбільший елементи на нулі
15	4x3	цілий	Обчислити суму додатних непарних елементів і замінити парні елементи масиву на цю суму
16	6x4	дійсний	Замінити мінімальний і максимальний елементи значенням середнього арифметичного всіх елементів
17	5x3	цілий	Замінити парні елементи на значення найменшого елемента
18	4x5	цілий	Замінити парні за значенням елементи матриці на 0
19	3x5	цілий	Поміняти місцями максимальний елемент з першим
20	5x5	дійсний	Розмістити елементи головної діагоналі у зворотному порядку
21	3x4	цілий	Замінити елементи кратні 5-ти на значення найбільшого елемента
22	3x6	дійсний	Поміняти місцями елементи першого й останнього стовпців
23	5x5	цілий	Замінити на максимальне значення матриці всі його нульові елементи

№ вар.	*Розмір масиву*	*Тип даних*	*Індивідуальне завдання*
24	4x6	цілий	Замінити всі парні елементи масиву на значення останнього елемента матриці
25	5x4	дійсний	Поміняти місцями елементи першого й останнього рядків
26	3x4	цілий	Поміняти місцями два найбільші за значенням елементи
27	3x3	дійсний	Замінити елементи головної діагоналі матриці на значення середнього арифметичного її елементів
28	6x4	цілий	Замінити на значення мінімального елемента ті елементи матриці, які менші за середнє арифметичне
29	5x3	дійсний	Замінити від'ємні елементи в непарних рядках матриці на нулі, а парних рядках – на одиниці
30	4x5	цілий	Поміняти місцями два найменші за значенням елементи

Передбачити в реалізованому алгоритмі відповідні коментарі, можливі помилки при введенні даних.

Зміст звіту

1. Титульна сторінка, тема та мета практичного завдання.
2. Завдання відповідно до номеру наданого варіанту
3. Опис виконаних завдань, лістинги реалізованих програм, результати виконання програм, блок-схеми, структурограми.
4. Реалізований алгоритм подати наступними способами: природною мовою в структурованому вигляді; формульно-словесною мовою; псевдокодом на мові С++. блок-схемами, реалізовані онлайн-сервісами ***Lucidchart, Draw.io, MS Visio***, тощо.
5. Висновки, які відображають результати виконаних завдань та їх аналіз.

Теоретичні відомості

Обробка лінійних структур даних мовою С++

Організація одновимірних масивів. **Масив** у програмуванні – це сукупність однотипних елементів. При оголошенні масивів у квадратних дужках зазначається кількість елементів, а нумерація елементів завжди розпочинається з нуля. Одновимірний масив оголошується в програмі таким чином:

<тип_даних><ім'я_масиву> [<розмір_масиву>];

Приклади оголошення масивів:

int a[125];

double vector[100];

Кожен елемент масиву однозначно можна визначити за ім'ям масиву й індексами. ***Індекси*** визначають місцезнаходження елемента в масиві і записують після імені масиву в квадратних дужках, тобто використовуючи ім'я масиву та індекс, можна звертатися до елементів масиву:

<ім'я_масиву> [<значення_ індексу>]

Значення індексів повинні бути в діапазоні від нуля до величини, на одиницю меншу, ніж розмір масиву, визначений при його оголошенні,

оскільки в С++ нумерація індексів розпочинається з нуля.

Наприклад, команда ***int A[10];*** оголошує масив з ім'ям А, який містить 10 цілих чисел: А[0] – перший елемент, А[1]– другий, А[9]– останній.

Приклад 1. Розробити алгоритм та програмний проект мовою С++ розрахунку заданої функції $y = ax^2 - sin(x)$, де a − довільна змінна, яка задається сталою величиною, значення x – записуються в одновимірний масив користувачем в процесі виконання програми.

Текст програмного коду створеного алгоритму:

```
#include <iostream>
#include <cmath>
#include <conio.h>
using namespace std;
int main()
{
   const int n = 7;
   float x[n], y, a(10.5);
   int i;
   for(i = 0;i< n;i++)
   {
   cin>>x[i];       //введення елементів масиву
   y = a * x[i] * x[i] - sin(x[i]);
//-------------------------------------- виведення результату
  cout << " x["<<i<<"] ="<<x[i]<<" y = " << y << endl;
 }
return 0;}
```

Виконання створеного алгоритму визначення заданої функції наведено на рис. 27

```
1 2 3 4 5 6 7
 x[0] =1 y = 9.65853
 x[1] =2 y = 41.0907
 x[2] =3 y = 94.3589
 x[3] =4 y = 168.757
 x[4] =5 y = 263.459
 x[5] =6 y = 378.279
 x[6] =7 y = 513.843
```

Рисунок 27. - Результати роботи алгоритму поставленої задачі.

Приклад 2. Знайти суму додатних елементів одновимірного масиву n-ї розмірності. Дані одновимірного масиву задаються сталими величинами.

Текст програмного коду створеного алгоритму:

```
#include <iostream>
using namespace std;
int main()
{  int i;
   float C[6]={2, 3, 2.5, 1.2,-1,};
   float s=0;
   for(i=0; i<6; i++)
   {
   cout <<"C["<< i << "]="<<C[i]<<endl;
   }
```

```
for (i=0; i<6; i++)
 if (C[i]>0)
 s+=C[i];
cout << "Значеня суми S=" << s;
return 0;}
```

Виконання створеного алгоритму визначення заданої функції наведено на рис. 28

```
C[0]=2
C[1]=3
C[2]=2.5
C[3]=1.2
C[4]=-1
C[5]=0
Значе?я суми S=8.7
```

Рисунок 28. - Результати роботи алгоритму поставленої задачі.

Приклад 3. Знайти середнє арифметичне від'ємних елементів одновимірного масиву n-ї розмірності. Дані в одновимірний масив заносяться випадковими величинами.

Текст програмного коду створеного алгоритму:

```
#include <iostream>
#include <cmath>
#include <cstdlib>
using namespace std;
int main()
{
   setlocale(0, ".1251");
   srand(time(NULL));
   int n, i, k=0;
   float C[10];
   cout <<"Вкажіть розмір масиву n (<=9) = ";
   cin >> n;
   for(i=0; i<n; i++)
   {
   C[i]=rand()%10;
   cout << "C[" << i << "]="<<C[i]<<endl;
   }
   float s=0;
   for (i=0; i<n; i++)
         if (C[i]>0)
          {
         s+=C[i]; k++;
         }
          if (k>0)
                  cout << "середнє арифметичне = " << s/k;
          else
                  cout << "додатніх елементів не знайдено";
   return 0; }
```

Виконання створеного алгоритму визначення заданої функції наведено на рис. 29

```
Вкажіть розмір масиву n (<=9) = 5
C[0]=1
C[1]=2
C[2]=6
C[3]=9
C[4]=1
середнє арифметичне = 3.8
```

Рисунок 29. - Результати роботи алгоритму поставленої задачі.

Приклад 4. Знайти суми кубів додатних, добутку квадрату від'ємних та кількість нульових елементів одновимірного масиву n-ї розмірності. Дані в одновимірний масив заносяться користувачем в процесі виконання програми.

Текст програмного коду створеного алгоритму:

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
   setlocale(0, ".1251");
   int i, k=0; float C[6];
   for(i=0; i<6; i++)
   {
   cout << "C[" << i << "]=";
   cin >> C[i];
   }
   float s=0, d=1;
   for (i=0; i<6; i++)
   if (C[i]>0)
      s+=pow(C[i], 3);
      else if(C[i]<0)
      d*=pow(C[i], 2);
   else
   k++;
   cout << "сума кубів додатних =" << s << endl;
   cout << "добутку від'ємних =" << d << endl;
   cout << "кількість нульових елементів =" << k << endl;
   return 0;}
```

Виконання створеного алгоритму визначення заданої функції наведено на рис. 30

```
C[0]=1
C[1]=2
C[2]=3
C[3]=4
C[4]=5
C[5]=6
сума кубів додатних =441
добутку від'ємних =1
кількість нульових елементів =0
```

Рисунок 30. - Результати роботи алгоритму поставленої задачі.

Організація багатовимірних масивів. Вимірність масиву визначається кількістю індексів. Елементи одновимірного масиву (вектора) мають один індекс, двовимірного масиву (матриці, таблиці) – два індекси: перший з них –

номер рядка, другий – номер стовпця. Кіль- кість індексів у масивах є необмежена. При розміщуванні елементів масиву в пам'яті комп'ютера першою чергою змінюється крайній правий індекс, потім решта – справа наліво.

Багатовимірний масив оголошується у програмі в такий спосіб:

<тип> <ім'я> [<розмір1>] [<розмір2>] … [<розмірN>];

Кількість елементів масиву дорівнює добутку кількості елементів за кожним індексом. Наприклад, матриця з 3-х рядків і 4-х стовпців оголошена як: ***int a[3][4];*** має 12 елементів цілого типу:

a[0][0], a[0][1], a[0][2], a[0][3],
a[1][0], a[1][1], a[1][2], a[1][3],
a[2][0], a[2][1], a[2][2], a[2][3];

Під масив надається пам'ять, потрібна для розміщення усіх його елементів. Елементи масиву один за одним, з першого до останнього, запам'ятовуються у послідовно зростаючих адресах пам'яті так само, як і елементи одновимірного масиву. Наприклад, масив ***int a[3][4]*** зберігається у пам'яті у спосіб, який наведено на рис. 31.

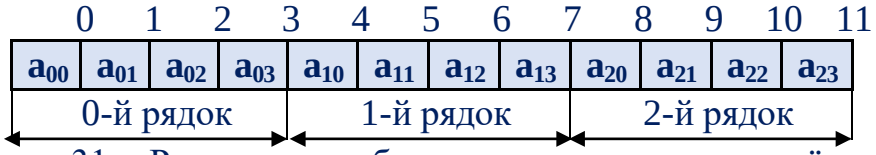


Рисунок 31. - Результати роботи алгоритму поставленої задачі.

Наведемо приклади оголошення масивів:

int x[5][5]; *// Матриця 5x5=25 елементів цілого типу*
char S[10][3]; *// Двовимірний масив з 10x3=30 елементів символьного типу*
double z[4][5][4]; *//Тривимірний масив з 4x5x4=80 елементів дійсного типу*

Здійснювати введення-виведення значень елементів масиву можна лише по-елементно, для чого слід організовувати цикли, в яких послідовно змінюватимуться значення індексів елементів.

Введення-виведення матриць у консольному режимі:

а) **Введення у консолі** матриці **a** розміром **3x4**:

1 **спосіб:**

```
for(i=0; i<3; i++) for(j=0; j<4; j++)
{ cout<<"Введіть елемент "<<i+1<<"-го рядка "<<j+1<<"-го стовпця: ";
cin>>a[i][j]; }
```

2 **спосіб:**

```
for(i=0; i<3; i++)
{ cout<<"Введіть 4 елементи "<<i+1<<"-го рядка "<<endl; for(j=0; j<4;
j++) cin>>a[i][j]; }
```

3 **спосіб:**

```
cout<<"Введіть матрицю з 3-х рядків і 4-х стовпців:"<<endl;
for(i=0; i<3; i++)
for(j=0; j<4; j++) cin>>a[i][j];
```

б) Для **виведення у консолі** двовимірного масиву **a** розміром 3x4 у вигляді матриці елементи слід розмістити у рівні стовпці, можна виконати за допомогою символу табуляції ***"\t":***

```
for(i=0; i<3; i++)
{
for(j=0; j<4; j++) cout << a[i][j] << "\t"; cout << endl;
}
```

При виведенні елементів матриць дійсних чисел командою cout<< доречним є використання маніпуляторів форматування:

- **setprecision**, який обмежує кількість знаків після десяткової крапки;
- **fixed**, який задає формат з фіксованою крапкою;
- **setw**, який дозволяє задавати ширину (мінімальну кількість символьних позицій) кожного виведеного числа.

Якщо виведене число має ширину меншу, аніж зазначену у модифікаторі setw, перед ним будуть виведені пропуски:

```
for(i=0; i<3; i++)
{ for(j=0; j<6; j++) cout<<fixed<<setprecision(2)<<setw(5)<<a[i][j]/3.<<"\t";
cout << endl; }
```

Для можливості використання цих маніпуляторів слід долучити заголовний файл ***iomanip*** командою ***#include <iomanip>***.

Приклад 5. Знайти суму додатних та від'ємних елементів, підрахувати кількість нульових елементів в двовимірному масиві n, m-ї розмірності. Дані в двовимірний масив заносяться користувачем в процесі виконання програми.

Текст програмного коду створеного алгоритму:

```
#include <iostream>
using namespace std;
int main()
{
   setlocale(0, ".1251");
   int i, j;
   float C[3][4];
   for(i=0; i<3; i++)
      for(j=0; j<4; j++)
      {
      cout << "C[" << i << "][" << j << "]=";
      cin >> C[i][j];
      }
   float sd=0, sv=0;
   int kn=0;
   for (i=0; i<3; i++)
      for (j=0; j<4; j++)
         if (C[i][j]<0)
         sv+=C[i][j];
         else if (C[i][j]>0)
         sd+=C[i][j];
         else
         kn++;
   cout << "сума додатних =" << sd << endl;
   cout << "сума від'ємних =" << sv << endl;
   cout << "кількість нульових елементів =" << kn << endl;
return 0; }
```

Виконання алгоритму наведено на рис. 32.

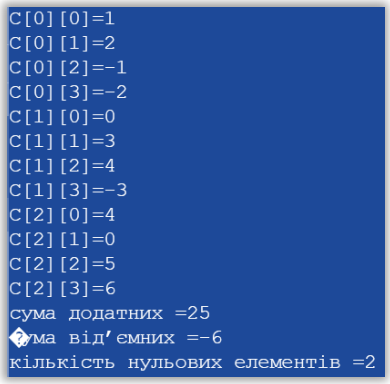


Рисунок 32. - Результати роботи алгоритму поставленої задачі.

Приклад 6. Знайти в матриці A(n, m), (n, m <=9) суму всіх додатних елементів, крім тих, що лежать на головній діагоналі. Дані в двовимірний масив заносяться сталими величинами;

Текст програмного коду створеного алгоритму:

```
#include <iostream>
using namespace std;
int main()
{
   setlocale(0, ".1251");
   int i, j, n;
   float C[9][9]={{1.2, 2.5, -1, 2,},{3,4,7.2,-2,}};
   cout << "Введіть розмір матриці n (<=9) = ";
   cin >> n;
   for(i=0; i<n; i++)
      for(j=0; j<n; j++)
      {
      cout << "C[" << i << "][" << j << "]="<<C[i][j]<<endl;
      }
   float sd=0;
   for (i=0; i<n; i++)
      for (j=0; j<n; j++)
         if (i!=j)
         if (C[i][j]<0)
         sd+=C[i][j];
       cout << "сума =" << sd << endl;
   return 0; }
```

Виконання створеного алгоритму наведено на рис. 33

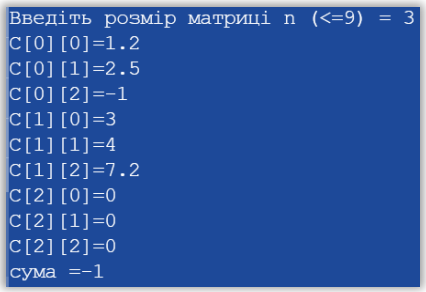


Рисунок 33. - Результати роботи алгоритму поставленої задачі.

Приклад 7. Матриця A(m,n) (m<=9, n<=9) містить додатні та від'ємні елементи. Сформувати одновимірні масиви: B, який містить лише додатні елементи масиву A(m,n), масив C, який містить лише від'ємні елементи масиву A(m,n). Підрахувати кількість елементів в даних масивах. Дані в двовимірний масив заносяться випадковими величинами.

Текст програмного коду створеного алгоритму:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
   setlocale(0, ".1251");
   srand(time(NULL));
   int i, j, m, n;
   float A[9][9];
   float B[81], C[81];
   cout << "Введіть m (<=9) = ";
   cin >> m;
   cout << "Введіть n (<=9) = ";
   cin >> n;
   for(i=0; i<m; i++)
      for(j=0; j<n; j++)
      {
      A[i][j]=1+rand()%21-10;
      cout << "A[" << i << "][" << j << "]="<<A[i][j]<<endl; }
   int kd=0, kv=0;
   for (i=0; i<m; i++)
      for (j=0; j<n; j++)
         if (A[i][j]>0)
         {
         B[kd] = A[i][j];
         kd++; }
         else if (A[i][j]<0)
         {
         C[kv] = A[i][j];
         kv++; }
      cout << "кількість додатних =" << kd << endl;
      cout << "кількість від'ємних =" << kv << endl;
return 0;}
```

Виконання створеного алгоритму наведено на рис. 34

```
Введіть m (<=9) = 3
Введіть n (<=9) = 3
A[0][0]=7
A[0][1]=3
A[0][2]=-1
A[1][0]=6
A[1][1]=3
A[1][2]=-1
A[2][0]=0
A[2][1]=-2
A[2][2]=6
кількість додатних =5
кількість від'ємних =3
```

Рисунок 34. - Результати роботи алгоритму поставленої задачі.

Приклад 8. Дано тензор Т розмірністю 2х3х2. Знайти суму всіх парних елементів тензора. Дані в тензор заносяться користувачем в процесі виконання програми.

Текст програмного коду створеного алгоритму:

```
#include <iostream>
using namespace std;
int main()
{
   setlocale(0, ".1251");
   int i, j, k;
   float T[2][3][2];
   for(i=0; i<2; i++)
      for(j=0; j<3; j++)
       for(k=0; k<2; k++)
       {
       cout << "T[" << i << "][" << j << "][" << k << "]=";
       cin >> T[i][j][k];
       }
   int s=0;
   for(i=0; i<2; i++)
      for(j=0; j<3; j++)
         for(k=0; k<2; k++)
            if (i%2==0 && j%2==0 && k%2==0)
            s+=T[i][j][k];
   cout << "сума =" << s << endl;
return 0; }
```

Виконання створеного алгоритму наведено на рис. 35

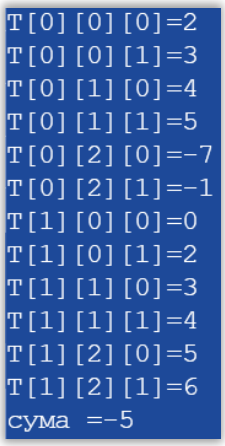


Рисунок 35. - Результати роботи алгоритму поставленої задачі.

Приклад 9. Дано тензор Т розмірністю 2х3х2. Знайти суму всіх парних елементів тензора. Дані тензора задаються фіксованими значеннями.

Текст програмного коду створеного алгоритму:

```
#include <iostream>
using namespace std;
int main()
{
   setlocale(0, ".1251");
   int i, j, k;
   float T[2][3][2]=
```

```
{
   {
      {1,2},
      {4,5},
      {2,8}
   },
   {
      {1,2},
      {4,5},
      {2,5}
   }
};
for(i=0; i<2; i++)
   for(j=0; j<3; j++)
   for(k=0; k<2; k++)
   {
cout << "T[" << i << "][" << j << "][" << k << "]="<<T[i][j][k]<<endl;
   }
int s=0;
for(i=0; i<2; i++)
   for(j=0; j<3; j++)
      for(k=0; k<2; k++)
         if (i%2==0 && j%2==0 && k%2==0)
         s+=T[i][j][k];
   cout << "сума =" << s << endl;
return 0; }
```

Виконання створеного алгоритму наведено на рис. 36

```
T[0][0][0]=1
T[0][0][1]=2
T[0][1][0]=4
T[0][1][1]=5
T[0][2][0]=2
T[0][2][1]=8
T[1][0][0]=1
T[1][0][1]=2
T[1][1][0]=4
T[1][1][1]=5
T[1][2][0]=2
T[1][2][1]=5
сума =3
```

Рисунок 36. - Результати роботи алгоритму поставленої задачі.

Обробка одновимірних масивів у функціях. При опрацюванні масивів у функціях передавання їх у якості аргументів завжди здійснюється за адресою, тобто передається адреса першого елемента (початок масиву), а доступ до кожного з елементів масиву здійснюється як певний зсув (обчислюваний через індекси) від початку масиву.

Передавання одновимірних масивів до функцій у якості аргументів можна організувати одним з трьох способів:

double fun (int a[10]);
double fun (int a[]);
double fun (int *a);

За першим способом явно зазначено кількість елементів масиву. У другій синтаксичній формі константний вираз у квадратних дужках є відсутній. За третього способу передається вказівник як посилання на масив, явно визначений в основній програмі. Оскільки для другого і третього способів інформація про кількість елементів у прототипі відсутня, то такі форми припустимі, коли кількість елементів масиву є глобальними змінними чи константами. Але доцільніше передавати розмірність одновимірного масиву до функції окремим параметром, оскільки це зробить таку функцію більш універсальною, адже вона зможе коректно опрацьовувати масиви з різною кількістю елементів.

double fun (int a[], int n); double
fun (int *a, int n);

Хоча наведені записи мають різний синтаксис, насправді вони є рівнозначними, оскільки авторами стандарту С для більшої ясності було вирішено, що масив оголошений як параметр функції є **вказівником**. При цьому навіть явно зазначене число у квадратних дужках ігнорується.

Якщо функція для опрацювання елементів масиву не повинна передавати результат як одне значення, а лише переставляє елементи масиву місцями або змінює їхні значення, то таку функцію оголошують з типом результату ***void*** (нема величини, яка повертається). Оскільки масив передається до функції за посиланням, то будь-які змінювання значень елементів масиву у функції буде видно і в основній програмі, яка викликає цю функцію.

Приклад 10. Ввести матрицю дійсних чисел розмірності 5x5 та за допомогою користувацької функції замінити елементи головної діагоналі на середнє арифметичне відповідного рядка.

Текст програмного коду створеного алгоритму:

```
#include <iostream>
using namespace std;
void zamina(double a[5][5])
{
   for(int i=0; i<5; i++)
   {
   double s=0;
   for(int j=0; j<5; j++)
   s+=a[i][j]/5; a[i][i]=s;
   }
}

int main ()
{
   setlocale(0,".1251");
   double a[5][5];
   int i, j;
cout<<"Введіть матрицю з 5-ти рядків і 5-ти стовпців:"<<endl;
   for(i=0; i<5; i++)
      for(j=0; j<5; j++)
      cin >> a[i][j];
```

```
            zamina(a);
            cout<< "\n Матриця, в якій елементи головної діагоналі поміняли \n на середнє арифметичне відповідного рядка:" << endl;
          for(i=0; i<5; i++)
          {
            for(j=0; j<5; j++)
            cout << a[i][j] << "\t";
            cout << endl;
          }
        return 0;}
```

Виконання створеного алгоритму визначення заданої функції наведено на рис. 37

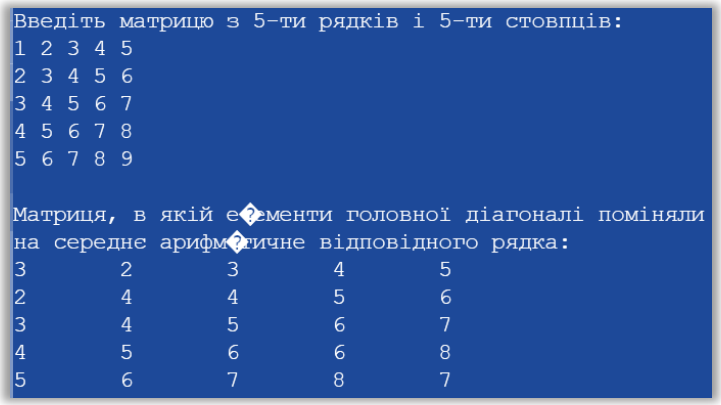


Рисунок 37. - Результати роботи алгоритму поставленої задачі.

Обробка лінійних структур даних мовою Python

Одновимірні масиви в Python. Найчастіше у програмах необхідно зберігати та обробляти велику кількість даних про об'єкти одного типу. І тут зручно використовувати масиви. ***Масив*** – це набір об'єктів одного типу під загальним ім'ям (***ім'я масиву***). Кожен об'єкт (***елемент масиву***) має свій номер (***індекс***), за допомогою якого можна звертатись до відповідного елементу масиву.

Оголошення масиву в Python відомого розміру. Масив з певним числом елементів ***N*** оголошується за наступною формою, при цьому всім елементам масиву надається нульове значення:

Назва масиву = [0] * N

Задати значення елементам масиву можна під час оголошення масиву:

Назва масиву = [елемент №1, елемент №2, елемент №3,…]

Назва масиву [індекс елемента масиву] = значення елемента

При цьому масив має фіксований розмір у відповідності вказаної кількості елементів. Оголосити масив можна двома способами:

- ***Спосіб №1.*** ***a = [0, 1, 2, 3, 4]***
- ***Спосіб №2.*** ***a[0] = 0***
  a[1] = 1
  a[2] = 2
  a[3] = 3
  a[4] = 4

При роботі з масивами зручно використовувати цикли для перебору відповідних елементів масиву.

a = [0] * розмір масиву
for i in range(розмір масиву):
a[i] = вираз

Розмір масиву можна визначити за допомогою команди ***len(ім'я масиву)***. Наприклад, якщо вводити значення масиву з клавіатури, то використаємо команду ***len(ім'я масиву)*** в циклі. При виведенні елементів масиву в один рядок, використовуємо параметр ***end =" "*** в операторі: ***print(a[i], end = " ")***

Створення списків на Python

Створити список можна декількома способами.:

1. ***Отримання списку через присвоювання конкретних значень***
   - Ініціалізація в Python порожнього списку:
     ```
     s = [ ]        # Порожній список
     ```
   - Приклади створення списків із відповідними значеннями:
     ```
     l = [25, 755, -40, 57, -41]                    # список цілих чисел
     l = [1.13, 5.34, 12.63, 4.6, 34.0, 12.8]       # список з дійсних чисел
     l = [ "Sveta", "Sergei", "Ivan", "Dasha"] # список з рядків
     l = [ "Київ", "Іваненко", 12, 124]          # змішаний список
     l = [ [0, 0, 0], [1, 0, 1], [1, 1, 0] ]          # вкладений список
     l = [ 's', 'p', [ 'isok'], 2]                    # список  зі значень і списку
     ```
   - Списки можна додавати/об'єднувати за допомогою «+»:
     ```
     l = [1, 3] + [4, 23] + [5] # Результат: # L = [1, 3, 4, 23, 5]
     ```
2. ***Списки, створені за допомогою функції List ()***
   - Отримуємо список за допомогою функції List ()
     ```
     l = list ( 'spysok')# 'spysok' - рядок
     print (l)          # [ 's', 'p', 'i', 's', 'o', 'k'] - результат - список
     ```
3. ***Створення списку за допомогою функції Split ().*** Використовуючи функцію ***split*** в Python можна отримати з рядка список. Розглянемо приклад:
   ```
   stroka = "Hello, world" # stroka - рядок
   lst = stroka.split ( ",")   # lst - список
   lst                         # [ 'Hello', 'world']
   ```
4. ***Генератори списків.*** В Python створити список можна також за допомогою генераторів:
   - *Перший спосіб*. Додавання однакових елементів в списку замінюється множенням:
     ```
     # Список з 10 елементів, заповнений одиницями
     l = [1] * 10
     # Список l = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
     ```
   - *Другий спосіб*. Додавання елементів списку в циклі:
     ```
     l = [i for i in range (10)]      # Список l = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
     ```
     або:
     ```
     c = [c * 3 for c in 'list']
     print (c)                        # [ 'lll', 'iii', 'sss', 'ttt']
     ```
     Наприклад, можна заповнити список квадратами чисел від 0 до 9,

використовуючи генератор списку.

```
l = [i * i for i in range (10)]
```

або в інший спосіб:

```
l = [(i + 1) + i for i in range (10)]
print (l)       # [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

- *Випадкові числа в списку:*

```
from random import randint
l = [randint (10,80) for x in range (10)]
# 10 чисел, згенерованих випадковим чином в діапазоні (10,80)
```

Введення списків (масивів) в мові Python

1. Для введення елементів списку використовується цикл ***range***:

```
for i in range (N):
  print ( "L [", i, "] =", sep =
  "", end = "")L [ i ] = int
  (input ())
```

2. Більш простіший варіант введення/виведення списку і його виведення:

```
L = [int (input ()) for i in range (N)]
```

Функція ***int*** використовується для того, щоб рядок, введений користувачем, перетворювався в цілочисельний тип даних.

3. Список можна виводити цілим і по-елементно:

```
# Виведення цілого списку (масиву)
print (L)
# Поелементне  виведення списку (масиву)
for i in range (N):
   print (L [ i ], end = "")
```

Розглянемо приклади реалізації з використання двовимірних масивів мовою ***Python*** та імпортування бібліотеки по роботі з масивами ***array.***

Приклад 11. Розробити алгоритм та програмний проект мовою ***Python*** розрахунку заданої функції $y = ax^2 - \sin(x)$, де a – довільна змінна, яка задається сталою величиною, значення *x* – фіксовані значення одновимірного масиву.

Текст програмного коду створеного алгоритму:

```
from array import *
from math import*
a=2
array1 = array('i',[11, 12, 5, 2, 15, 6, 10, 8, 12, 5])
for x in array1:
  y=a*x**2-sin(x)
  print('При x=',x,' функція y=%.3f' % y)
```

Виконання створеного алгоритму наведено на рис. 38

```
При x= 11  функція y=243.000
При x= 12  функція y=288.537
При x= 5  функція y=50.959
При x= 2  функція y=7.091
При x= 15  функція y=449.350
При x= 6  функція y=72.279
При x= 10  функція y=200.544
При x= 8  функція y=127.011
При x= 12  функція y=288.537
При x= 5  функція y=50.959
```

Рисунок 38. - Результати роботи алгоритму поставленої задачі.

Приклад 12. Розробити алгоритм та програмний проект мовою ***Python*** розрахунку заданої функції $y = ax^2 - \sin(x)$, де a – довільна змінна, яка задається користувачем, значення *x* – записуються в одновимірний масив користувачем в процесі виконання програми. Розмірність одновимірного масиву задається безпосередньо користувачем.

Текст програмного коду створеного алгоритму:

```
a=int(input('Введіть ціле число a='))
n=int(input('Вкажіть розмір одновимірного масиву n='))
mas = [i] * n
for i in range(len(mas)):
    i = str(i + 1)
    print("Введіть елемент массиву " + i, end = " ")
    i = int(i)
    i = i - 1
    mas[i] = int(input())
print("")
for i in range(len(mas)):
    mas[i] = a*mas[i]**2-sin(mas[i])
print("Масив розрахованих значень y:")
for i in range(len(mas)):
    print('y',i,'= %.3f' % mas[i], end = " ")
```

Виконання створеного алгоритму визначення заданої функції наведено на рис. 39

```
Введіть ціле число a=2
Вкажіть розмір одновимірного масиву n=3
Введіть елемент массиву 1 2
Введіть елемент массиву 2 4
Введіть елемент массиву 3 6

Масив розрахованих значень y:
y 0 = 7.091 y 1 = 32.757 y 2 = 72.279
```

Рисунок 39. - Результати роботи алгоритму поставленої задачі.

Приклад 13. Розробити алгоритм та програмний проект мовою ***Python*** розрахунку заданої функції $y = ax^2 - \sin(x)$, де a – довільна змінна, яка задається користувачем, значення *x* – записуються в одновимірний масив випадковими величинами. Розмірність одновимірного масиву задається безпосередньо користувачем.

Текст програмного коду створеного алгоритму:

```
a=int(input('Введіть ціле число a='))
n=int(input('Вкажіть розмір одновимірного масиву n='))
mas = [i] * n
for i in range(0,n):
  mas[i] = randint(1,30)
print(mas,"")
for i in range(0,n):
   mas[i] = a*mas[i]**2-sin(mas[i])
print("Масив розрахованих значень y:")
for i in range(0,n):
   print('y',i,'= %.3f' % mas[i], end = " ")
```

Виконання створеного алгоритму наведено на рис. 40

```
Введіть ціле число a=2
Вкажіть розмір одновимірного масиву n=3
[7, 12, 29]
Масив розрахованих значень y:
y 0 = 97.343 y 1 = 288.537 y 2 = 1682.664
```

Рисунок 40. - Результати роботи алгоритму поставленої задачі.

Приклад 14. Розробити алгоритм та програмний проект мовою ***Python*** для визначення максимального та мінімального значення в одновимірному масиві. Значення в масив заносяться користувачем в процесі виконання програми.

Текст програмного коду створеного алгоритму:

```
a = [i] * 5
for i in range(len(a) - 1):
   i = str(i + 1)
   print("Вкажіть елемент масиву " + i, end = " ")
   i = int(i)
   a[i] = int(input())
min = a[0]
max = a[0]
for i in range(len(a)):
   if (a[i ]< min):
      min = a[i]
   if (a[i] > max):
      max = a[i]
min = str(min)
max = str(max)
print("Мінімальне значення в масиві = " + min)
print("Максимальне значення в масиві = " + max)
```

Виконання створеного алгоритму наведено на рис. 41

```
Вкажіть елемент масиву 1 2
Вкажіть елемент масиву 2 8
Вкажіть елемент масиву 3 12
Вкажіть елемент масиву 4 7
Мінімальне значення в масиві = 2
Максимальне значення в масиві = 12
```

Рисунок 41. - Результати роботи алгоритму поставленої задачі.

Двовимірні масиви в Python. Двовимірні масиви (матриці) можна розглядати як таблицю, якаскладається з n-рядків і m-стовпців (перший індекс задає кількість рядків, а другий – кількість стовпців). ***Описати двовимірний масив*** *можна наступним чином*:

Dim ім'я (n,m) As Type

Наприклад, ***Matr(5,3) As Integer***

При зверненні до елемента двовимірного масиву потрібно вказати ідентифікатор змінної-масиву, а також два індекси, які вказують на місце знаходження необхідного значення в масиві. Індекси записується біля ідентифікатора змінної-масиву в круглих дужках через кому.

Наприклад, **MATR(I , j)** — це елемент ***масиву MATR***, який знаходиться ***наперетині і-го рядка і j-го стовпця.***

Елементи матриці у пам'яті ***розташовуються по рядках*** (рис. 42), дії над масивами виконуються ***поелементно***. Для двовимірних масиві використовуються вкладені цикли.

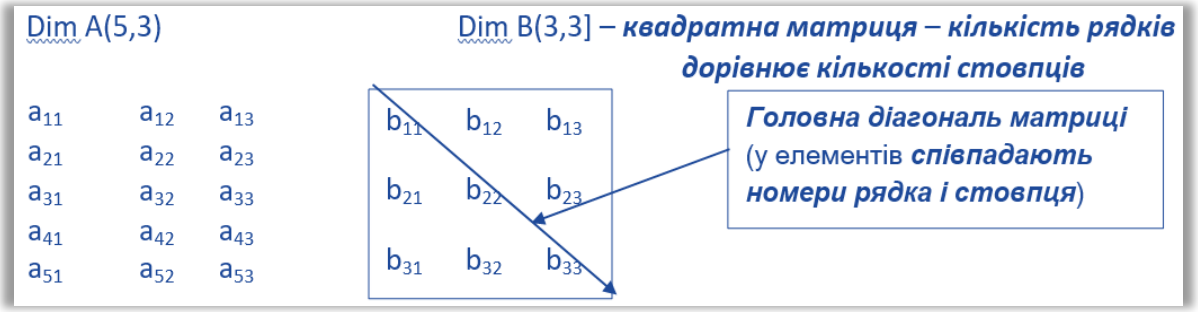


Рисунок 42. - Результати роботи алгоритму поставленої задачі.

Матриці в Pyhton. Матриця в ***Python*** - це таблиця об'єктів одного типу, із загальним ім'ям. Двовимірна матриця складається з рядків та стовпців, щоб звернутися до елемента матриці необхідно вказати ім'я матриці та у квадратних дужках номер рядка та номер стовпця (рис. 43).

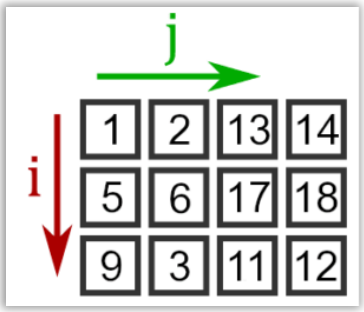


Рисунок 43. – Розміщення елементів в двовимірній матриці.

Задати матрицю в Python можна за допомогою бібліотеки ***numpy*** командою ***import numpy***

Назва матриці = numpy.zeros((кількість рядків, кількість стовпців))

Можна командою ***numpy.zeros*** задати матрицю, всі значення якої дорівнюють нулю. Наприклад, вказати матрицю розміром 6 х 8 можна наступним чином:

a = numpy.zeros((6, 8))

Щоб почати роботу з матрицями в ***Python***, необхідно вказати значення всім елементам з використанням циклу ***for***. Нижче наведено шаблон коду, для створення матриці з нульовими значеннями та визначенням їх індексів.

```
import numpy
Ім'я матриці = numpy.zeros((кількість рядків, кількість стовпців))
# переглядаємо всі рядки
for i in range(кількість рядків):
# переглядаємо всі стовпці
   for j in range(кількість стовпців):
      # працюємо з елементом у рядку і у стовпці j
      Ім'я матриці[i][j]
```

Приклад 15. Заповнити матрицю розміром 5×10 випадковими числами. Випадкові числа вибираються у діапазоні, заданому користувачем.

Текст програмного коду створеного алгоритму:

```
import numpy
import random
a = numpy.zeros((5, 10))
lowerBound = int(input("Введіть нижню межу. "))
upperBound = int(input("Введіть верхню межу. "))
for i in range(5):
   for j in range(10):
      a[i][j] = random.randint(lowerBound, upperBound)
print(a)
```

Виконання створеного алгоритму наведено на рис. 44

```
Введіть нижню межу. 3
Введіть верхню межу. 8
[[7. 5. 6. 3. 6. 5. 3. 5. 7. 4.]
 [5. 3. 8. 4. 7. 5. 7. 5. 3. 8.]
 [4. 8. 5. 8. 8. 3. 8. 5. 7. 3.]
 [7. 6. 8. 6. 6. 4. 5. 6. 8. 3.]
 [4. 4. 3. 3. 3. 3. 7. 8. 3. 3.]]
```

Рисунок 44. - Результати роботи алгоритму поставленої задачі.

Приклад 16. Задати матрицю розміром 3×3, з клавіатури та знайти найбільше значення серед всіх елементів матриці.

Для пошуку максимальної кількості введемо змінну ***max***. Спочатку вкажемо змінній ***max = a [0] [0].*** Переглянемо всі елементи матриці у вкладеному циклі ***for***, порівнюючи елементи матриці із значенням ***max***. Якщо поточний елемент більший за ***max***, то присвоюємо змінній ***max*** значення даного елемента.

Текст програмного коду створеного алгоритму:

```
import numpy
a = numpy.zeros((3, 3))
max = a[0][0]
# переглядаємо всі рядки
for i in range(0, 3):
# переглядаємо всі стовпці
   for j in range(0, 3):
      # працює з елементом у рядку і у стовпці j
```

```
        a[i][j] = int(input("Введення даних матриці "))
        if (a[i][j] > max):
            max = a[i][j]
max = str(max)
print("")
print("Найбільше значення "+max)
```

Виконання створеного алгоритму наведено на рис. 45

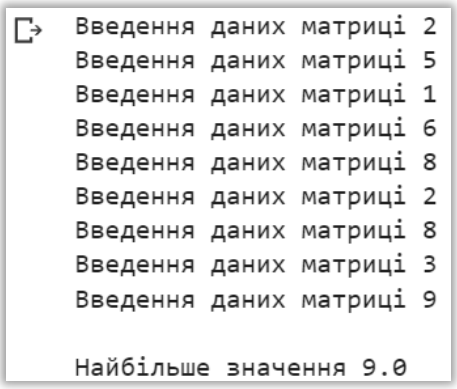


Рисунок 45. - Результати виконання поставленої задачі.

Приклад 17. Двовимірний масив n, m -ї розмірності містить додатні та від'ємні елементи. Знайти суму додатних та від'ємних елементів в масиві, підрахувати їх загальну кількість. Дані в двовимірний масив заносяться сталими величинами.

Текст програмного коду створеного алгоритму:

```
from array import *
T = [[11, 0, 5, -2], [15, 0,10], [10, 0, 12, -5], [12,15,8,-6]]
for r in T:
    for c in r:
        print(c,end = " ")
    print()
sd=0
sv=0
kn=0
for r in T:
    for c in r:
        if(c>0):
          sd=sd+c
        elif(c<0):
          sv=sv+c
        else:
          kn=kn+1
print('Сума додатніх s=',sd)
print('Сума відємних s=',sv)
print('Кількість нулів k=',kn)
```

Виконання створеного алгоритму наведено на рис. 46

```
11 0 5 -2
15 0 10
10 0 12 -5
12 15 8 -6
Сума додатніх s= 98
Сума відємних s= -13
Кількість нулів k= 3
```

Рисунок 46. - Результати роботи алгоритму поставленої задачі.

Приклад 18. Двовимірний масив n, m -ї розмірності містить додатні та від'ємні елементи. Знайти суму додатних та від'ємних елементів в масиві, підрахувати їх загальну кількість. Дані в двовимірний масив заносяться користувачем в процесі виконання програми.

Текст програмного коду створеного алгоритму:

```
n=int(input('Вкажіть розмір двовимірного масиву n='))
a = [[0] * n for i in range(n)]
for i in range(len(a)):
    for j in range(len(a[i])):
      a[i][j] = int(input('Вкажіть значення елементу масиву '))
print(a)
sd=0; sv=0; kn=0
for i in a:
    for j in i:
        if(j>0):
          sd=sd+j
        elif(j<0):
          sv=sv+j
        else:
          kn=kn+1
print('Сума додатніх s=',sd)
print('Сума відємних s=',sv)
print('Кількість нулів k=',kn)
```

Виконання створеного алгоритму розрахунку заданої функції наведено на рис. 47

```
Вкажіть розмір двовимірного масиву n=2
Вкажіть значення елементу масиву 1
Вкажіть значення елементу масиву 0
Вкажіть значення елементу масиву -1
Вкажіть значення елементу масиву 2
[[1, 0], [-1, 2]]
Сума додатніх s= 3
Сума відємних s= -1
Кількість нулів k= 1
```

Рисунок 47. - Результати роботи алгоритму поставленої задачі.

Приклад 19. Двовимірний масив n, m -ї розмірності містить додатні та від'ємні елементи. Знайти суму додатних та від'ємних елементів в масиві, підрахувати їх загальну кількість. Дані в двовимірний масив заносяться випадковими величинами.

Текст програмного коду створеного алгоритму:

```
n=int(input('Вкажіть кількість рядків в масиві n='))
m=int(input('Вкажіть кількість стовпців в масиві m='))
a = [[0] * m for i in range(n)]
for i in range(len(a)):
    for j in range(len(a[i])):
      a[i][j] = randint(-10,10)
    print(a[i])
sd=0; sv=0; kn=0
for i in a:
    for j in i:
        if(j>0):
          sd=sd+j
        elif(j<0):
          sv=sv+j
        else:
          kn=kn+1
print('Сума додатніх s=',sd)
print('Сума відємних s=',sv)
print('Кількість нулів k=',kn)
```

Виконання створеного алгоритму наведено на рис. 48

```
Вкажіть кількість рядків в масиві n=2
Вкажіть кількість стовпців в масиві m=3
[8, -8, 9]
[-5, -3, -6]
Сума додатніх s= 17
Сума відємних s= -22
Кількість нулів k= 0
```

Рисунок 48. - Результати роботи алгоритму поставленої задачі.

Реалізація тензорів в Python. Тензор (матриця третього рангу) – це масив, який в Python ініціалізується з використання бібліотеки ***numpy*** як:

```
import numpy as np
a = np.array([[
[10, 11, 12],
[14, 15, 16],
[18, 19, 20]],
[[22, 23, 24],
[26, 27, 28],
[30, 31, 32]],
[[34, 35, 36],
[38, 39, 40],
[42, 43, 44]]])
```

Вказаний тензор має 3 осі (рис. 49): ***матриця*** (пласт, прошарок), ***рядок***, ***стовпець***. На рисунку позначені індекси: i - матриці, j - рядки, k - стовпці:

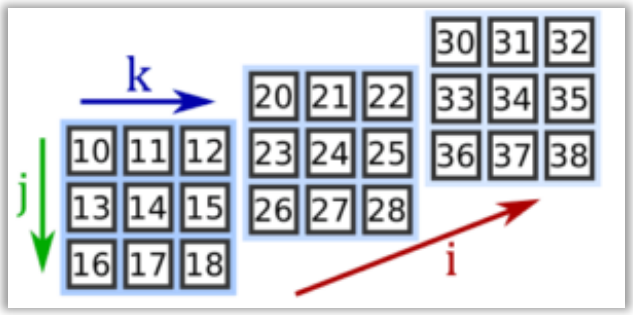


Рисунок 49. – Розміщення елементів е тензорі.

Доступ до елементів тензора.

Для доступу до елементу тензора, вказуються індекси матриці, рядка, стовпця: ***a[1, 2, 0]***

Результат виконання: ***30***

Для доступу до матриці тензора, достатньо вказати його індекс у квадратних дужках: ***a[2]***

Результат виконання: ***array([[34, 35, 36], [38, 39, 40], [42, 43, 44]])***

Для доступу до рядка матриці (рис. 50), задаються в квадратних дужках елементи рядка матриці: ***a[2, 1]***

Результат виконання: ***array([38, 39, 40])***

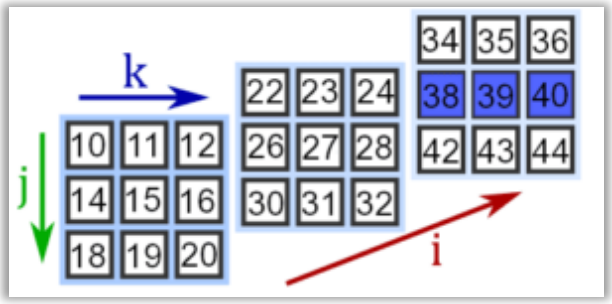


Рисунок 50. - Виділення рядків тензора ***numpy***

Виділення рядків.

Для виділення стовпця матриці (рис. 51), вказується номер матриці, вибираються всі рядки матриця, потім номер стовпця: ***a[0, :, 2]***

Результат виконання: ***array([12, 16, 20])***

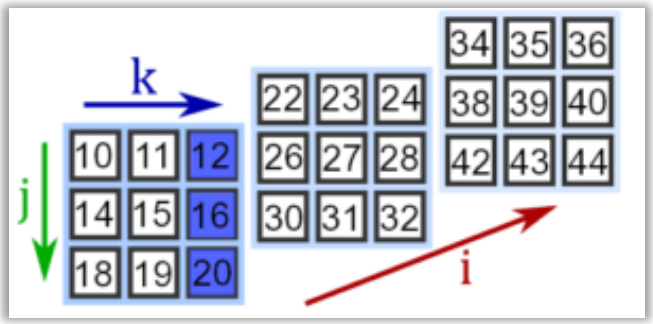


Рисунок 51. - Виділення стовпців тензора ***numpy***

Виділення стовпців.

Виключення елементів усіх матриць відбувається через двокрапку. Наприклад, отримаємо елементи матриць (рис. 52), що стоять на 1-му рядку, 0-му стовпці: ***a[:, 1, 0]***

Результат виконання: ***array([14, 26, 38])***

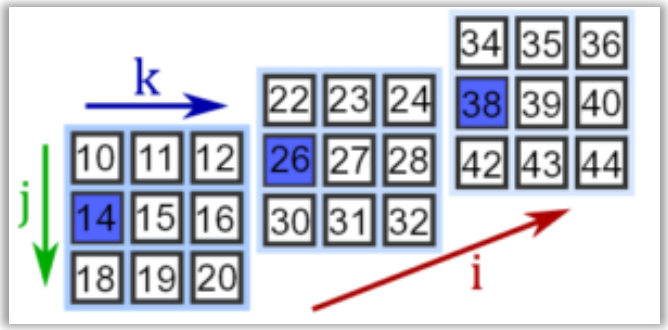


Рисунок 52. - Виділення елемента другого рядка, нульового стовпця кожної матриці тензора ***numpy***

Тензори можна використовувати як представлення чорно-білих зображень в Machine Learning, де кожна матриця представляє одне зображення з елементами відповідного розміру в пікселям.

Контрольні запитання

1. Синтаксис оголошення масиву мовою C++.
2. Як компілятор розглядає ім'я масиву в мові C++?
3. Як визначити розмір одновимірного масиву мови програмування C++?
4. Як передати одновимірний масив у функцію на мові C++?
5. Як передати багатовимірний масив у функцію на мові C++?
6. Як заборонити зміну в функції елементів масиву, переданого в якості параметру в мові C++?
7. Як присвоїти вказівнику адресу одновимірного масиву у мові C++?
8. Як присвоїти вказівнику адресу двовимірного масиву на мові C++?
9. Як звернутись до елементу одновимірного масиву без використання індексу в мові C++?
10. Як звернутись до елементу двовимірного масиву без використання індексу в мові C++?
11. Назвіть принципи роботи зі списками у мові Python.
12. Що таке одновимірний масив?
13. Для чого використовують одномірні масиви? Як їх описують в Python?
14. Як в програмі Python використати значення конкретного елемента одновимірного масиву?
15. Для чого в програмах використовуються двовимірні масиви? Як їх описують в Python?
16. Скільки індексів характеризують конкретний елемент двовимірного масиву в Python?
17. Чи може список в Python складатись із елементів різних типів?
18. Як в Python видалити елемент зі списку за значенням?
19. Як додати елемент у вказану позицію списку в Python?
20. Для чого призначена функція range в Python?
21. Назвіть методи впорядкування елементів у списку Python.
22. Як створити функцію у мові Python?

23. Що таке модулі та пакети?
24. Як бувають способи підключення модулів та пакетів?
25. Що таке анонімні функції та інструкція lambda?
26. Який синтаксис оголошення функції?
27. Що таке позиційні та непозиційні аргументи функції?
28. Як створити функцію зі значеннями за замовчуванням?
29. Який порядок передачі аргументів у функцію, якщо вона містить позиційні та непозиційні аргументи та аргументи із значеннями за замовчуванням?
30. Для чого икористовується інструкція return? Чи обов’язково вона присутня у функції?
31. Що повертає функція, якщо в її тілі відсутня інструкція return?

Практичне завдання 4.
Алгоритми сортування статичних одновимірних та багатовимірних масивів

Мета заняття: набуття практичних навичок по реалізації алгоритмів сортування даних в статичних одновимірних та багатовимірних масивах; ознайомлення з основними функціональними можливостями алгоритмів сортування: обміну, вибору та вставки мовами програмування **C++**, **Python** в одновимірних та багатовимірних статичних масивах; реалізовувати алгоритми графічно в онлайн-сервісах **Lucidchart**, **Draw.io**.

Завдання 1. Розробити алгоритми сортування ***статичного*** одновимірного масиву ***методом обміну, вибору та вставки*** мовами програмування ***C++, Python*** відповідно до варіанту, наведеного в ***табл. 14***. Дані в масив додаються: сталими величинами / користувачем в процесі виконання програми / випадковими величинами.

В алгоритмах необхідно забезпечити:

— виведення заданого одновимірного масиву;

— виведення відсортованого масиву та проміжних розрахунків.

Таблиця 14. – Варіанти задач до завдання 1

№ вар.	*Індивідуальне завдання*
1.	Визначити, скільки елементів одномірного масиву A(9) більше, ніж число 3. Заданий масив відсортувати за спаданням.
2.	Заданий одномірний масив *A(8)*. Знайти суму всіх елементів масиву. Заданий масив відсортувати за зростанням.
3.	Заданий одномірний масив *A(I0)*. Знайти найменший по модулі елемент масиву. Заданий масив відсортувати за зростанням та спаданням.
4.	Заданий одномірний масив *A(I0)*. Знайти добуток негативних елементів масиву. Заший масив відсортувати за зростанням.
5.	Заданий одномірний масив *A(10)*. Знайти суму всіх елементів масиву, що мають парні індекси. Заданий масив відсортувати за зростанням та спаданням.
6.	Заданий одномірний масив *A(10)*. Знайти найбільший з елементів масиву, що мають непарні індекси. Заданий масив відсортувати за спаданням.
7.	Заданий одномірний масив *A(I0)*. Знайти кількість від'ємних елементів масиву. Заданий масив відсортувати за спаданням.
8.	Визначити, скільки елементів одномірного масиву *A(9)* менше, ніж число 6. Заданий масив відсортувати за зростанням.
9.	Заданий одномірний масив *A(I0)*. Знайти добуток всіх невід'ємних елементів масиву. Заданий масив відсортувати за зростанням.
10.	Заданий одномірний масив *A(I0)*. Знайти середнє арифметичне всіх елементів масиву. Заданий масив відсортувати за спаданням.
11.	Заданий одномірний масив *A(8)*. Знайти добуток $a_1*2a_2*3a_3*..*8a_8$. Заданий масив відсортувати за спаданням.
12.	Заданий одномірний масив *A(I0)*. Знайти найменший елемент у масиві. Заданий масив відсортувати за зростанням.

№ вар.	*Індивідуальне завдання*
13.	Заданий одномірний масив *A(H)*. Знайти кількість позитивних елементів масиву. Заданий масив відсортувати за спаданням.
14.	Заданий одномірний масив *A(I0)*. Знайти найменший елемент у масиві. Заданий масив відсортувати за зростанням та спаданням.
15.	Заданий одномірний масив *A(8)*. Знайти кількість позитивних елементів масиву. Заданий масив відсортувати за спаданням.

Завдання 2. Розробити алгоритми сортування ***статичного*** двовимірного масиву ***методом обміну, вибору та вставки*** мовами програмування ***C++, Python*** відповідно до варіанту, наведеного в ***табл. 15***. Дані в масив додаються: сталими величинами / користувачем в процесі виконання програми / випадковими величинами. В алгоритмах необхідно забезпечити:

— виведення заданого одновимірного масиву;

— виведення відсортованого масиву та проміжних розрахунків.

Таблиця 15. – Варіанти задач до завдання 2

№ вар.	*Індивідуальне завдання*
1.	Існує масив A вимірністю m на n. Відсортувати його за зростанням. Початковий і кінцевий масиви вивести на екран.
2.	Існує двовимірний масив довжиною NxN, де N = 32. Відсортувати за зростанням ті елементи масиву, які розташовуються на непарних позиціях.
3.	Існує двовимірний масив довжиною NxN, де N = 27. Упорядкувати масив таким чином, щоб елементи, які знаходяться на парних позиціях розташовувались за спаданням, а на непарних позиціях - по спаданням
4.	Існує двовимірний масив розмірністю NxN, де N = 8. Відсортувати всі рядки так, щоб елементи в них розташовувалися за зростанням.
5.	Існує двовимірний масив довжиною NxN, де N = 25. Упорядкувати масив таким чином, щоб елементи, які знаходяться на парних позиціях розташовувалися за зростанням, а на непарних позиціях - за спаданням.
6.	Існує двовимірний масив довжиною NxN, де N = 26. Відсортувати за спаданням ті елементи масиву, які є непарними числами.
7.	Існує двовимірний масив розмірністю NxN, де N = 10. Відсортувати всі стовпці так, щоб елементи в них розташовувалися за спаданням.
8.	Існує двовимірний масив розмірністю NxN, де N = 9. Відсортувати парні елементи в масиві за зростанням.
9.	Існує двовимірний масив розмірністю NxN, де N = 10. Відсортувати всі рядки так, щоб елементи в них розташовувалися за спаданням.
10.	Існує двовимірний масив розмірністю NxN, де N = 8. Відсортувати всі рядки так, щоб елементи в них розташовувалися за зростанням.
11.	Існує двовимірний масив розмірністю NxN, де N = 9. Відсортувати всі непарні елементи масиву за спаданням.
12.	Існує двовимірний масив довжиною NxN, де N = 70. Відсортувати за зростанням ті елементи масиву, які розташовуються на непарних позиціях.
13.	Існує двовимірний масив довжиною NxN, де N = 38. Упорядкувати масив за зростанням та спаданням.

№ вар.	*Індивідуальне завдання*
14.	Існує двовимірний масив довжиною NxN, де N = 45. Відсортувати за спаданням ті елементи масиву, які є парними числами.
15.	Існує двовимірний масив розмірністю NxN, де N = 10. Відсортувати всі стовпці так, щоб непарні елементи в них розташовувалися за зростанням.

Завдання 3. Розробити алгоритми сортування ***динамічних*** одновимірних та двовимірних масивів **методом вставки, обміну та вибору** мовами програмування **C++**, **Python**. Дані в масив додаються: сталими величинами / користувачем в процесі виконання програми / випадковими величинами.

В алгоритмах необхідно забезпечити:

— виведення заданого одновимірного масиву;

— виведення відсортованого масиву та проміжних розрахунків.

Завдання 4. Розробити алгоритми сортування ***терзорів*** **методом вставки, обміну та вибору** мовами програмування **C++**, **Python**. Дані в масив додаються: сталими величинами / користувачем в процесі виконання програми / випадковими величинами.

В алгоритмах необхідно забезпечити:

— виведення заданого одновимірного масиву;

— виведення відсортованого масиву та проміжних розрахунків.

Завдання 5. Розробити алгоритми сортування ***одновимірних, двовимірних масивів*** та ***тензорів*** ***методами злиття, випадкового сортування, швидкого сортування, шейкерного сортування, сортування частинами*** на мовах програмування **C++**, **Python**. В алгоритмах необхідно передбачити:

— виведення на екран заданих початкових даних;

— виведення на екран проміжних розрахунків та відсортованих масивів за зростанням/спаданням.

Зміст звіту

1. Титульна сторінка, тема та мета практичного завдання.
2. Завдання відповідно до номеру наданого варіанту
3. Опис виконаних завдань, лістинги реалізованих програм, результати виконання програм, блок-схеми, структурограми.
4. Реалізований алгоритм подати наступними способами: природною мовою в структурованому вигляді; формульно-словесною мовою; псевдокодом на мові C++. блок-схемами, реалізовані онлайн-сервісами ***Lucidchart, Draw.io, MS Visio***, тощо.
5. Висновки, які відображають результати виконаних завдань та їх аналіз.

Теоретичні відомості

Методи сортування мовою програмування C++

Метод сортування вставками. Нехай потрібно упорядкувати масив X(n) за зростанням пропонується використовувати наступний підхід: для $i=1,2,3,\ldots,n$, кожен елемент x_i будемо вставляти в потрібне місце серед упорядкованих раніше елементів x_1, x_2, x_3, …, x_{i-1}, розсовуючи їх (рис. 53). Цей

метод у явному виді часто не використовується на практиці, однак покладена в його основу ідея добре працює, коли потрібно вставити новий елемент у вже упорядкований масив.

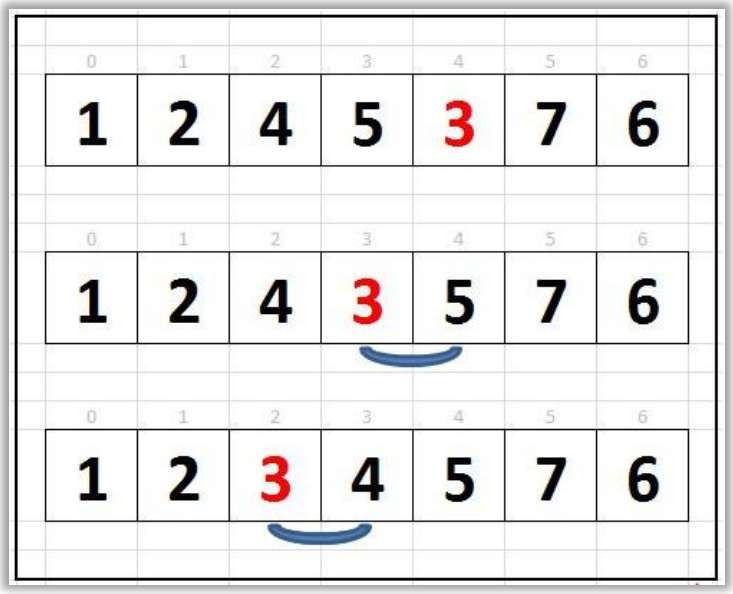


Рисунок 53. – Сортування одновимірного масиву методом вставки

Приклад 1. Відсортувати одновимірний статичний масив сталих величин методом вставки.

Програмна реалізація методу сортування вставками:

```
int main()
{
 const int N = 10;
 int a[N] = { 12, 5, 3, 2, 45, 96, 6, 8, 11, 24 };
  int buff = 0; // для збереження зміненого значення при вставці
 int i, j; // для циклів
  /************* Початок сортування за зростанням******************/
 for (i = 1; i < N; i++)
 {
 buff = a[i]; // запам'ятовується елемент обробки та  починається заміна зліва на право, до тих пір поки занесений в буфер елемент не буде меншим від переглянутих елементів.
 for (j = i - 1; j >= 0 && a[j] > buff; j--)
 a[j + 1] = a[j];
  a[j + 1] = buff; // та поставимо елемент з буфера обміну на відповідне місце
 }
 /************* Кінець сортування ******************/
  for (int i = 0; i < N; i++) // виведення відсортованого масиву на екран
 cout << a[i] << '\t';
 cout << endl;}
```

Метод сортування вибором. Метод сортування простим вибором, у певному сенсі є протилежним до методу сортування простими включеннями: під час сортування простими включеннями на кожному кроці розглядається тільки один черговий елемент вхідної послідовності та всі елементи готового масиву для знаходження місця включення. Під час сортування простим вибором

розглядаються всі елементи вхідного масиву для знаходження елементів з найменшим ключем, і цей один наступний елемент відправляється до готової послідовності.

Цей метод ґрунтується на такому правилі:

1) обирається елемент із найменшим ключем;

2) обраний елемент міняється місцем із першим елементом a1.

Наведені операції (рис. 54) повторюються з n–1 елементами, що залишилися, потім із n–2 елементами, поки не залишиться лише один елемент – найбільший. Ідея методу полягає в створенні відсортованої послідовності шляхом приєднання до неї елементів одного за іншим в правильному порядку.

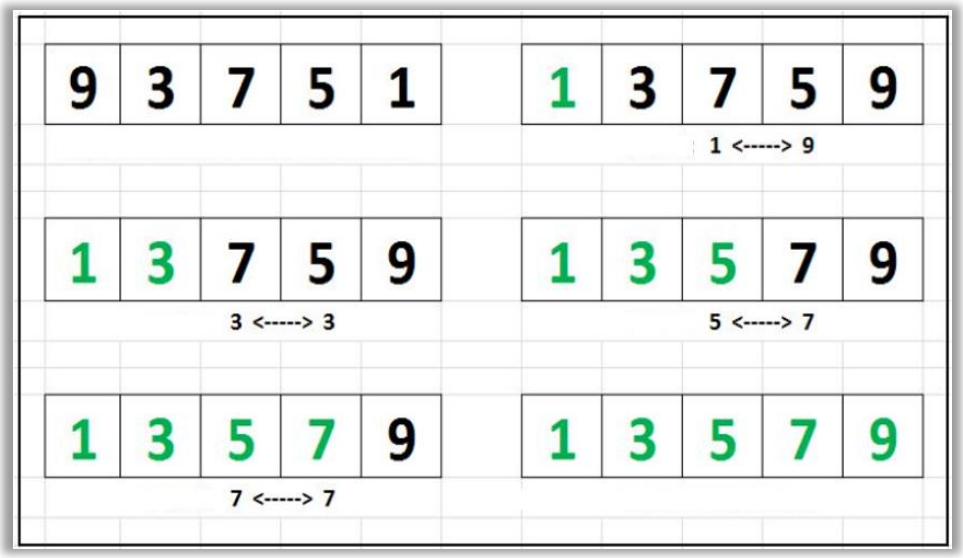


Рисунок 54. – Сортування одновимірного масиву методом вибору

Приклад 2. Відсортувати за методом вибору статичний одновимірний масив, дані якого вводяться з клавіатури.

Програмна реалізація методу сортування вибором:

```
int main()
{
srand(time(NULL));
const int N = 10;
int a[N] = {};
int min = 0; // для запису мінімального значення
int buf = 0; // для обміну даних
for (int i = 0; i < N; i++) //введення даних користувачем в масив
{
 cin>>a[i];
cout << a[i] << '\t';
}
cout << endl;
cout << endl;
/********** Початок сортування за спаданням**********/
for (int i = 0; i < N; i++)
{
 min = i; // запам'ятовується номер поточної комірки
// в циклі знайдемо номер комірки з мінімальним значенням
for (int j = i + 1; j < N; j++)
```

```
min = ( a[j] > a[min] ) ? j : min;
// виконуємо перестановку знайденого мінімального елемента, міняючи його місцями
з поточним елементом
if (i != min)
{
buf = a[i];
a[i] = a[min];
a[min] = buf;
}
}
/********** Кінець сортування за спаданням **********/
for (int i = 0; i < N; i++) //виведення на екран відсортованого масиву
cout << a[i] << '\t';
cout << endl;
}
```

Метод обміну (бульбашки). Для опису основної ідеї ***бульбашкового сортування*** (***Bubble Sort***) необхідно записати ключі в масиві, які сортуються, розташованими вертикально.

Алгоритм, що реалізує метод обміну полягає у здійсненні повторюваних проходжень по масиву, що сортується. За кожен прохід елементи послідовно порівнюються попарно і, якщо порядок у парі є невірним, виконується обмін елементів місцями. Проходження по масиву (рис. 55) повторюються N–1 разів (N – кількість елементів масиву) або доти, доки під час чергового проходження не виявиться, що обміни більше не потрібні, а це означає – заданий масив відсортовано.

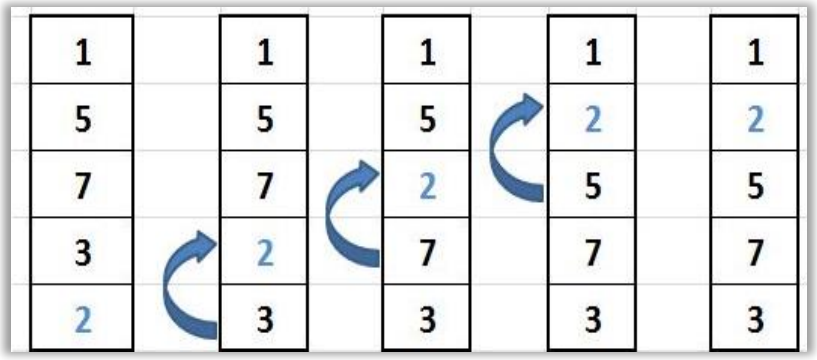


Рисунок 55. – Сортування одновимірного масиву за методом обміну

Під час кожного проходження алгоритму із внутрішнього циклу черговий найбільший елемент масиву ставиться на своє місце в кінці масиву поряд із попереднім «найбільшим елементом», а найменший елемент переміщується на одну позицію до початку масиву («спливає» до потрібної позиції, як бульбашка на воді, звідси й походить назва алгоритму).

Приклад 3. Відсортувати статичний двовимірний масив сталих величин за методом обміну (бульбашки).

Програмна реалізація методу сортування обміном:

```
int main()
{
int k,l,temp,i,j;
int x[10][10]={{2,3,},{-2,3,},{2,-3,},};
```

```
//виведення з пам'яті початкового масиву
cout<<"виведення з памяті початкового масиву"<<endl;
for(i=0;i<10;i++)
{
for(j=0;j<10;j++)
{cout<<x[i][j]<<" ";
}
cout<<endl;
}
//сортування за зростанням
for(k=0;k<10;k++)
{
  for(l=0;l<10;l++)
  {
    for(i=0;i<10;i++)
    {
       for(j=0;j<10;j++)
       {
          if (i==10-1 && j==10-1){  //якщо рядок останній і справа немає значень, то
сортування не виконується.
                    continue;  }
          if (x[i][j] > x[i][j+1])
          { //Якщо елемент не на своїй позиції
               temp = x[i][j];        //то здійснюється обмін значень місцями
               x[i][j] = x[i][j+1];
               x[i][j+1] = temp; }}}}}
//виведення відсортованого масиву на екран
cout<<endl;
cout<<"виведення відсортованого масиву за зростанням"<<endl;
for(i=0;i<10;i++)
{
   for(j=0;j<10;j++)
   {
     cout<<x[i][j]<<" ";  }  }
```

Методи сортування мовою програмування Python

Метод обміну. Алгоритм полягає в циклічних проходах по масиву, за кожен прохід елементи масиву попарно порівнюються і, якщо їх порядок неправильний, то здійснюється обмін. Обхід масиву повторюється до тих пір, поки масив не буде впорядкований.

Цей підхід полягає у тому, що перебір здійснюється за списком і з порівнянням сусідніх елементів. Вони міняються місцями в тому випадку, якщо порядок неправильний. І так триває до тих пір, поки всі елементи не розташуються в потрібному порядку. В методі обміну сортування складність в гіршому випадку - O (n^2), через велику кількість повторень (рис. 56).

Рисунок 56. – Сортування одновимірного масиву методом обміну

Приклад 4. Відсортувати одновимірний масив цілих чисел за методом обміну. Розмір масиву та його дані задаються користувачем.

Програмна реалізація методу сортування обміном:

```
# сортування бульбашкою за зростанням
def bubble_sort_z(array):
    length = len(array)
    for i in range(0, length):
        for j in range(0, length - i - 1):
            if array[j] > array[j + 1]:
                temp = array[j]
                array[j] = array[j + 1]
                array[j + 1] = temp
print("Сортування бульбашкою")
arr = []
n = int(input("Введіть довжину масиву: "))
for i in range(0, n):
    element = int(input("arr[" + str(i + 1) + "] = "))
    arr.append(element)
bubble_sort_z(arr)
print("Відсортований масив за зростанням: ")
print(arr)
```

Виконання створеного алгоритму сортування методом обміну наведено на рис. 57

```
Сортування бульбашкою
Введіть довжину масиву: 5
arr[1] = -2
arr[2] = 0
arr[3] = 3
arr[4] = -4
arr[5] = 7
Відсортований масив за зростанням:
[-4, -2, 0, 3, 7]
```

Рисунок 57. – Результати сортування одновимірного масиву методом обміну за зростанням та спаданням.

Метод сортування вставками (включенням). Сортування включенням (*insertion sort*), або *сортування вставками* - це алгоритм сортування, в якому всі елементи масиву почергово переглядаються, при цьому кожен елемент переміщається у відповідне місце серед раніше впорядкованих значень.

Опис алгоритму сортування включенням

Алгоритм роботи сортування включенням наступний:

- на початку роботи впорядкована частина порожня;
- додаємо в неї перший елемент масиву з не впорядкованих даних;
- переходимо до наступного елементу в невідсортованих даних, і знаходимо йому правильну позицію у відсортованій частині масиву, цим ми розширюємо область впорядкованих даних;
- повторюємо попередній крок для всіх елементів, що залишилися.

Сортування вставками швидке і просте. На кожній ітерації програма бере один з елементів і підшукує для нього місце у вже відсортованому списку (рис.

58). Так відбувається до тих пір, поки не залишиться жодного невикористаного елемента.

Рисунок 58. – Сортування одновимірного масиву методом включення.

Приклад 5. Відсортувати одновимірний масив цілих чисел за методом включення (вставками). Розмір масиву та його дані задаються користувачем.

Програмна реалізація методу сортування вставками:

```
def insertion_sort(array):
    length = len(array)
    for i in range(1, length):
        key = array[i]
        j = i
        while (j - 1 >= 0) and (array[j - 1] > key):
            array[j - 1], array[j] = array[j], array[j - 1]
            j = j - 1
        array[j] = key

print("Сортування включенням")
arr = []
length = int(input("Введіть довжину масиву: "))
for i in range(0, length):
    element = int(input("arr[" + str(i + 1) + "] = "))
    arr.append(element)
insertion_sort(arr)
print("Відсортований масив: ")
print(arr)
```

Виконання створеного алгоритму сортування методом включення наведено на рис. 59

```
Сортування включенням
Введіть довжину масиву: 5
arr[1] = 3
arr[2] = 8
arr[3] = 0
arr[4] = -4
arr[5] = 1
Відсортований масив:
[-4, 0, 1, 3, 8]
```

Рисунок 59. – Результати сортування одновимірного масиву методом включення за зростанням.

Метод Selection Sort (сортування вибором). Сортування вибором також доволі простий алгоритм, але більш ефективний у порівнянні з бульбашковим сортуванням.

У цьому алгоритмі список (або масив) ділиться на дві частини: список з відсортованими елементами та список з елементами, які потрібно відсортувати. Спочатку визначається найменший елемент у другому списку. Він додається в

кінець першого. Таким чином алгоритм поступово формує список від меншого до більшого (рис. 60). Так відбувається до тих пір, поки не буде готовий відсортований масив.

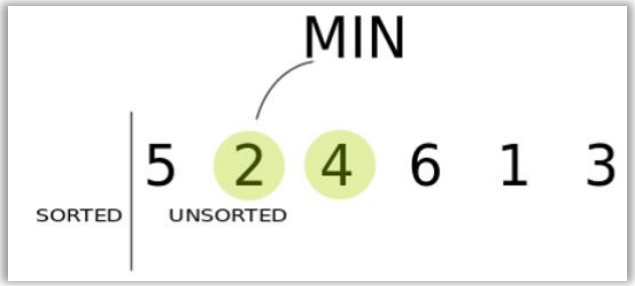


Рисунок 60. – Сортування одновимірного масиву методом вибору.

Приклад 6. Відсортувати одновимірний масив цілих чисел за методом вибору. Розмір масиву та його дані задаються користувачем.

Програмна реалізація методу сортування вибором:

```
def selection_sort(arr):
  for i in range(len(arr)):
    minimum=i
    for j in range(i+1, len(arr)):
      #вибір найменшого значення
      if arr[j]<arr[minimum]:
        minimum=j
    #вставка мінімального значення перед кінцем відсортованого масиву
    arr[minimum], arr[i]=arr[i], arr[minimum]
  return arr
print("Сортування вставками")
arr = []
length = int(input("Введіть довжину масиву: "))
for i in range(0, length):
    element = int(input("arr[" + str(i + 1) + "] = "))
    arr.append(element)
selection_sort(arr)
print("Відсортований масив: ")
print(arr)
```

Виконання створеного алгоритму сортування методом вставки наведено на рис. 61

```
Сортування вставками
Введіть довжину масиву: 5
arr[1] = -2
arr[2] = 1
arr[3] = 8
arr[4] = 0
arr[5] = 4
Відсортований масив:
[-2, 0, 1, 4, 8]
```

Рисунок 61. – Результат сортування одновимірного масиву методом вставки.

Контрольні запитання

1. Які існують методи сортування алгоритмів?
2. Які існують вимоги до алгоритмів сортування?
3. Як можна змінити алгоритм пошуку найбільшого із введених чисел, щоб знайти найменше число?
4. Розробіть алгоритм злиття масивів a і b з упорядкованими ділянками довжини d.
5. Розробіть алгоритм злиття двох масивів з упорядкованими ділянками довільної довжини. Як визначити кінець упорядкованої ділянки?
6. Чи зміниться складність алгоритму злиття, якщо зливати не по два, а по три масива?
7. Що таке метод розробки (проектування) алгоритмів?
8. У чому полягає особливість методу зменшення розміру задачі?
9. Оцінка часу роботи алгоритму сортування вставками в найкращому разі, середньому, гіршому.
10. Недоліки методу сортування вставками.
11. Оцінка часу роботи алгоритму в найкращому разі, середньому, гіршому.
12. Недоліки методу сортування злиттям.

Практичне завдання 5.
Алгоритми пошуку даних в одновимірних та багатовимірних масивах

Мета заняття: набуття практичних навичок по реалізації алгоритмів пошуку даних в статичних одновимірних та багатовимірних масивах; їх реалізація за допомогою користувацьких функцій; ознайомлення з принципами роботи алгоритмів пошуку в одновимірних та двовимірних динамічних масивах.

Завдання 1. Реалізуйте програму, яка буде знаходити індекс елементу масиву за заданим ключем двома методами – методом ***лінійного пошуку*** та ***бінарного***. Порівняйте швидкодію алгоритмів, зробіть висновки про найкращі та найгірші випадки вхідних даних.

Завдання 2. Реалізуйте програму, яка буде знаходити входження слова в тексті двома методами – прямим алгоритмом пошуку послідовності та алгоритмом Кнута-Морріса-Пратта. Результатом роботи алгоритмів повинен бути індекс першого символу, з якого починається співпадіння. Порівняйте швидкодію алгоритмів, зробіть висновки про найкращі та найгірші випадки вхідних даних.

Завдання 3. Реалізуйте програму, яка буде знаходити входження слова в тексті з використанням алгоритму Боєра-Мура. Порівняйте швидкодію алгоритму з алгоритмами, реалізованими в Завданні 2, зробіть висновки про найкращі та найгірші випадки вхідних даних.

Завдання 4. Мовами програмування C++ та Python скласти алгоритм ***інтерполяційного пошуку*** в:

1. одновимірному масиві розмірністю від 0 до 100. Дані в масив додаються: сталими величинами / користувачем в процесі виконання програми / випадковими величинами.
2. двовимірному масиві розмірністю MxN, де M-номер студента в списку групи+10, N- номер студента в списку групи+15.
3. в тензорі довільної розмірності.

Необхідно реалізувати програму, яка буде знаходити індекс елементу масиву за заданим ключем та визначити їх швидкодію. Зробіть висновки про найкращі та найгірші випадки вхідних даних.

Зміст звіту

1. Титульна сторінка, тема та мета практичного завдання.
2. Завдання відповідно до номеру наданого варіанту
3. Опис виконаних завдань, лістинги реалізованих програм, результати виконання програм, блок-схеми, структурограми.
4. Реалізований алгоритм подати наступними способами: природною мовою в структурованому вигляді; формульно-словесною мовою; псевдокодом на мові C++. блок-схемами, реалізовані онлайн-сервісами ***Lucidchart, Draw.io, MS Visio***, тощо.
5. Висновки, які відображають результати виконаних завдань та їх аналіз.

Теоретичні відомості

Алгоритми пошуку в С++

Лінійний пошук в С++. Лінійний або ***послідовний*** пошук – найпростіший з алгоритмів пошуку елементів в масиві.

Алгоритм полягає в обході всіх елементів масиву, як правило, зліва направо, і порівняння їх із шуканим значенням. Якщо значення елементу і ключа співпадають, то пошук повертає індекс(позицію) елементу.

Оскільки лінійний алгоритм обходить масив послідовно, то він досить повільний.

Тим не менш, цей метод використовується для пошуку:

- на невеликих масивах даних;
- в потоковій обробці даних;
- пошуку мінімального та максимального значення масиву;
- на одиночних невпорядкованих великих масивах.

Лінійний пошук належить до найбільш простих способів пошуку даних. Мета лінійного пошуку – знайти потрібний елемент (ключ) в масиві даних. В алгоритмі лінійного пошуку кожен елемент масиву послідовно порівнюється з ключем до тих пір, поки не буде знайдено потрібний елемента або не буде переглянуто весь масив.

В контексті лінійного пошуку можуть виникати наступні задачі:

- визначити наявність заданого елементу в масиві (наборі даних);
- визначити кількість входжень заданого елементу в масиві;
- визначити номер позиції або масив номерів позицій розміщення заданого елементу в масиві.

Реалізація лінійного пошуку може бути розширена для використання в багатовимірних масивах.

Це найпростіша техніка пошуку, і її також простіше застосувати. При лінійному пошуку ключ, який потрібно шукати, порівнюється лінійно з кожним елементом збору даних. Цей прийом ефективно працює на лінійних структурах даних. Розглянемо одновимірний масив, який наведено на рис. 62.

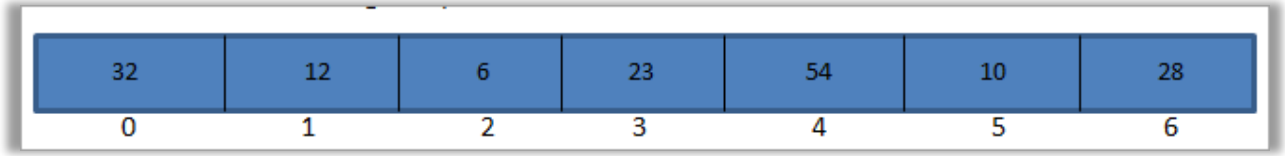


Рисунок 62. – Одновимірний масив із 7-ми елементів

В даному масиві міститься сім елементів. Якщо ми хочемо шукати ключ = 23, то починаючи з 0-го елемента, ключове значення буде порівняно з кожним елементом. Як тільки ключовий елемент збігається з елементом у масиві, тоді буде повернуто саме це місце. У цьому випадку місцеположення повертається 4, оскільки ключ-значення відповідає значенню в цьому місці. Наведемо приклад лінійного пошуку мовою програмування С ++

Приклад 1. В статичному одновимірному масиві, використовуючи лінійний пошук, знайти довільне число.

Програмна реалізація методу пошуку:

```
#include <iomanip>
#include <ctime>
using namespace std;
//прототипи функцій
int linSearch(int arr[], int requiredKey, int size); // Лінійний пошук
void showArr (int arr [], int size); // показ масиву
int main()
{
 setlocale(LC_ALL, "rus");
 const int arrSize = 50;
 int arr[arrSize];
 int requiredKey = 0; // Шукане значення (ключ)
 int nElement = 0; // Номер елементу масиву
 srand(time(NULL));
 // Запис випадкових чисел у масив від 1 до 50
 for (int i = 0; i < arrSize; i++)
 {
 arr[i] = 1 + rand() % 50;
 }
 showArr(arr, arrSize);
 cout << "Яке число необхідно шукати?";
 cin >> requiredKey; // Введення шуканого числа
  //Пошук шуканого числа та запис номера елемента
 nElement = linSearch(arr, requiredKey, arrSize);
  if (nElement! = -1)
 {
 //якщо у масиві знайдено шукане число - виводимо індекс елемента на екран
 cout << "Значення " << requiredKey << " знаходиться в комірці з індексом: " <<
nElement << endl;
 }
 else
 {
 //якщо у масиві не знайдено шукане число
 cout << "У масиві немає такого значення" << endl;
 }
 return 0;
}
 //Виведення масиву на екран
void showArr(int arr[], int arrSize)
{
 for (int i = 0; i < arrSize; i++)
 {
 cout << setw(4) << arr[i];
 if ((i + 1) % 10 == 0)
 {
 cout << endl;
 }
  }
 cout << endl << endl;
}
//лінійний пошук
```

```
int linSearch(int arr[], int requiredKey, int arrSize)
{
 for (int i = 0; i < arrSize; i++)
 {
 if (arr[i] == requiredKey)
 return i;
 }
 return -1;
}
```

Двійковий пошук в C++. Бінарний пошук - це техніка, яка використовує техніку «розділи і владай» для пошуку ключа. Він працює над відсортованим лінійним списком елементів. Відсортований список - основна вимога для роботи двійкового пошуку.

У двійковому методі пошуку список неодноразово ділиться навпіл і ключовий елемент знаходиться в обох половинах списку, поки не буде знайдений ключ. Наприклад, візьмемо наступний відсортований масив із 10 елементів (рис. 63).

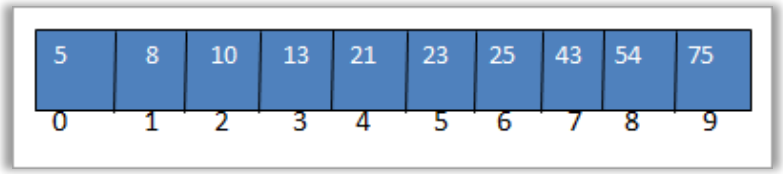


Рисунок 63. – Одновимірний масив з 10-ти елементів.

Якщо потрібно знайти ключ = 21 в масиві, то обчислимо середнє розташування масиву. Середина = 0 + 9/2 = 4 (рис. 64)

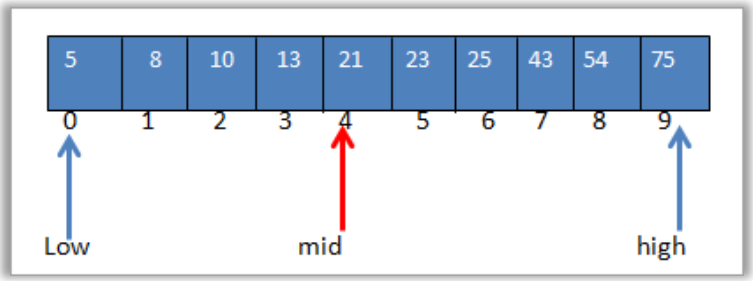


Рисунок 64. – Результат пошуку в одновимірному масиві.

Наприклад, візьмемо наступний відсортований масив із 10 елементів (рис. 65). Ключ = 21. Спочатку порівняємо значення ключа з елементом [mid] та визначимо, що значення елемента в середині = 21. Таким чином, ми знаходимо, що ключ = [середина] із значенням 21. Звідси ключ знайдений.

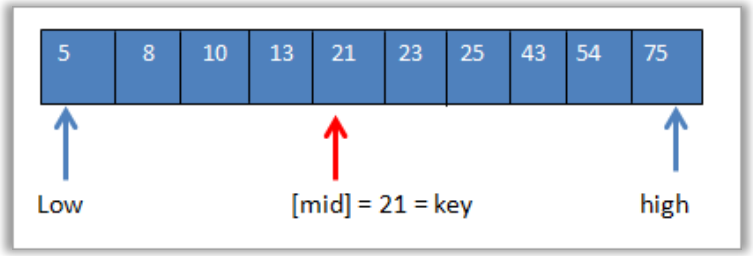


Рисунок 65. – Визначення середини заданого масиву.

Якщо задано ключ=25, то спочатку порівнюємо ключове значення із середнім. Отже (21<25). Тоді будемо шукати ключ у верхній половині масиву (рис. 66).

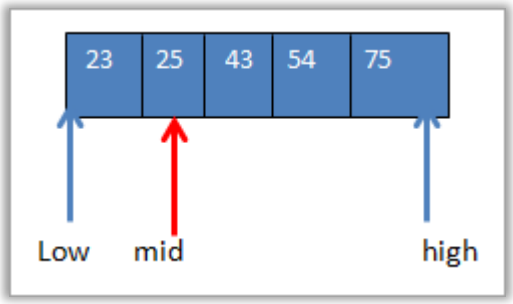


Рисунок 66. – Пошук в масиві за заданим ключем.

Тепер знайдемо середину для верхньої половини масиву. Середина = 4 + 9/2 = 6. Значення в місці [середина] = 25 (рис. 67)

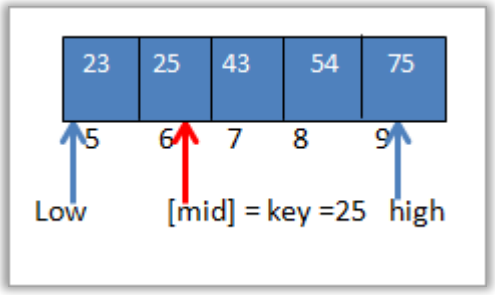


Рисунок 67. – Результат пошуку в одновимірному масиві.

Надалі порівняємо ключовий елемент із середнім елементом. Отже (25 = 25). Отже, ключ знайдено у відповідному місці. В процесі пошуку неодноразово ділиться масив і, порівнюється із ключовий елементом. Наведено алгоритм двійкового пошуку мовою С.

Приклад 2. В статичному одновимірному масиві, використовуючи бінарний пошук, знайти довільне число.

Програмна реалізація методу пошуку:

```
#include <iostream>
using namespace std;
// Функція з алгоритмом двійкового пошуку
int Search_Binary (int arr[], int left, int right, int key)
{
 int midd = 0;
 while (1)
 {
 midd = (left + right) / 2;
 if (key < arr[midd]) // якщо шукане менше значення в комірці
 right = midd - 1; // зміщуємо правий кордон пошуку
 else if (key > arr[midd]) // якщо шукане більше значення в комірці
 left = midd + 1; // зміщуємо ліву межу пошуку
 else // інакше (значення рівні)
 return midd; // функція повертає індекс комірки
 if (left > right) // якщо межі зімкнулися
 return -1;  } }
```

```
int main()
{
 setlocale (LC_ALL, "rus");
  const int SIZE = 12;
 int array[SIZE] = {};
 int key = 0;
 int index = 0; // Індекс осередку з шуканим значенням

 for (int i = 0; i < SIZE; i++) // заповнюємо та показуємо масив
 {
 array[i] = i + 1;
 cout << array[i] << " | ";
 }
  cout << "\n\nВведіть будь-яке число: ";
 cin >> key;
  index = Search_Binary (array, 0, SIZE, key);
  if (index >= 0)
 cout << "Вказане число знаходиться в осередку з індексом: " << index << "\n\n";
 else
 cout << "У масиві немає такого числа!\n\n";
  return 0;
}
```

Інтерполяційний пошук в C++. Інтерполяційний пошук працює лише у впорядкованому масиві. Заснований на тому самому принципі, що і алгоритм швидкого пошуку, але більш інтелектуально підходить до поділу масиву на частини. У швидкому пошуку масив на кожному кроці ділиться рівно на 2 половини, а в інтерполяційному - враховуються значення крайніх елементів з шуканим елементом.

Припустимо, що необхідно знайти число **3** у масиві: 1, 3, 5, 55, 66, 77, 567, 545, 12222, 43323. Алгоритм швидкого пошуку знайде потрібний елемент на ***O(Log(n))***, за рахунок того, що вже на першому кроці відтинає половину списку. Число 567 більше ніж 3, отже, числа 3 не може бути в правій частині списку (список же впорядкований).

Інтерполяційний пошук знайде результат ще швидше, оскільки він враховує значення елементів. Насправді число 3 значно менше ніж 43323, значить число 3 з більшою ймовірністю знаходиться десь на початку списку. Цей алгоритм враховує значення елементів вибору опорного елемента та використовує таку формулу:

$$mid = left + \frac{(value - A_{left}) \cdot (right - left)}{A_{right} - A_{left}}$$

Обчислювальна складність такого алгоритму в середньому випадку log(log(n)), проте при невдалих вхідних даних становитиме O(n). Блок-схема алгоритму інтерполяційного пошуку наведена на рис. 68.

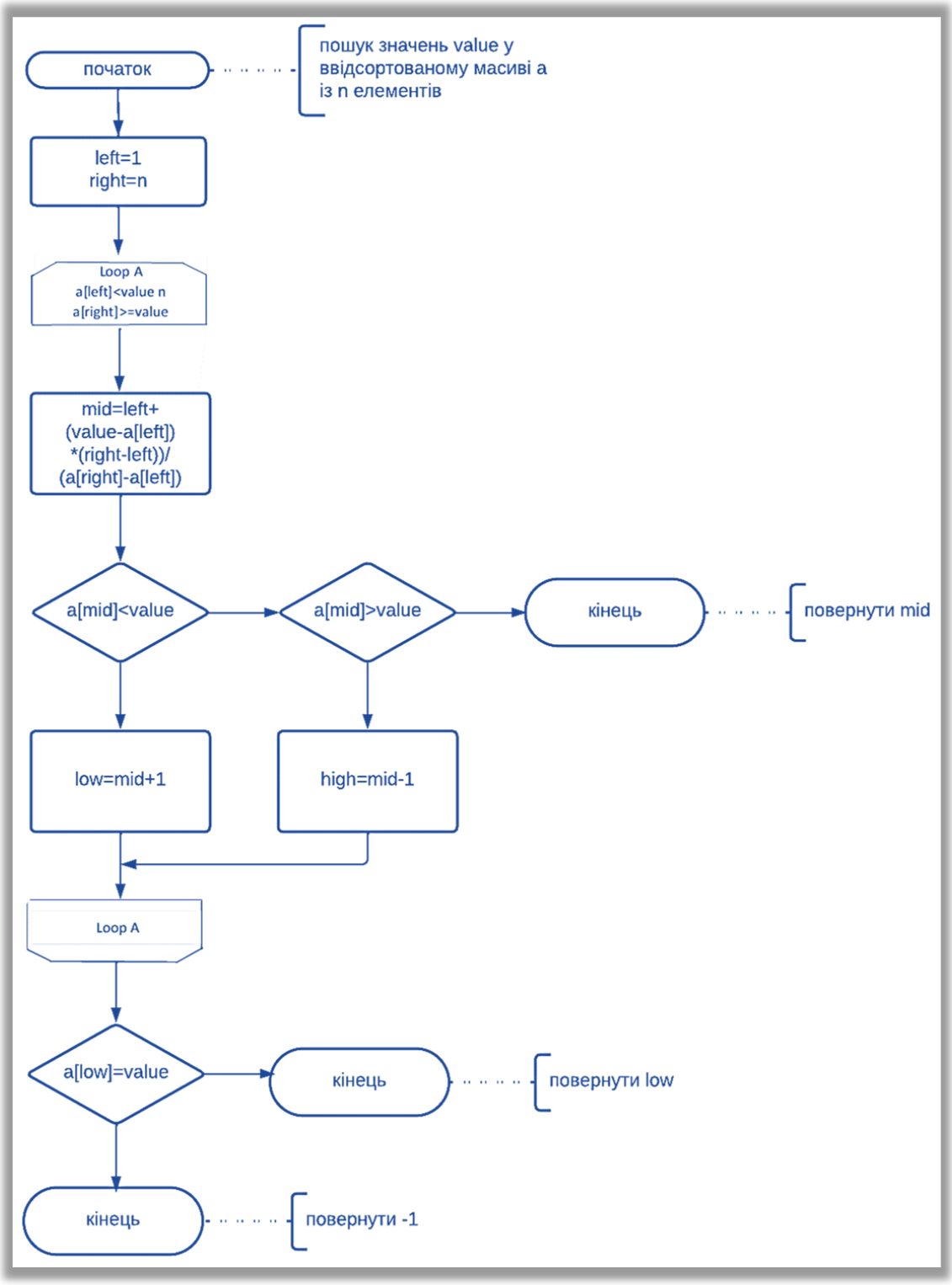


Рисунок 68. – Блок-схема виконання інтерполяційного пошуку.

Приклад 3. В статичному одновимірному масиві, використовуючи інтерполяційний пошук, знайти число 123.

Програмна реалізація методу пошуку:

```
#include <iostream>
using namespace std;
int main()
{
 //Масив значень у якому піде пошук
```

```
int MyArray[] { 1, 2, 4, 6, 7, 89, 123, 231, 1000, 1235};
 int x = 0; //Поточна позиція масиву, з яким порівнюється шукане
int a = 0; //Лівий кордон області, де ведеться пошук
int b = 9; //Правий кордон області, де ведеться пошук
 int WhatFind = 123; //Значення, яке потрібно знайти
bool found; //Зміна-прапор, що приймає True якщо шукане знайдено
 /********** Початок інтерполяції ******************************/
 //Цикл пошуку за масивом, поки не знайдено шукане
//або межі пошуку ще існують
for (found = false; (MyArray[a] < WhatFind) && (MyArray[b] > WhatFind) && !found;
)
{
//Обчислення інтерполяцією наступного елемента, який порівнюватиметься з
шуканим
x = a + ((WhatFind - MyArray [a]) * (b - a)) / (MyArray [b] - MyArray [a]);
//Отримання нових кордонів області, якщо шукане не знайдено
if (MyArray[x] < WhatFind)
a = x +1;
else if (MyArray[x] > WhatFind)
b = x – 1;
else
found = true;
}
 /************** Кінець інтерполяції *************************/
 //Якщо шукане знайдено на межах області пошуку, показати на якому кордоні воно
міститься
if (MyArray[a] == WhatFind)
cout << WhatFind << " founded in element " << a << endl;
else if (MyArray[b] == WhatFind)
cout << WhatFind << " founded in element " << b << endl;
else
cout << "Sorry. Not found" << endl;
 return 0; }
```

Решето Ератосфена в C++. Решето Ератосфена - алгоритм знаходження всіх простих чисел до деякого цілого числа N, який приписують давньогрецькому математику Ератосфен Кіренському. Назва алгоритму свідчить про принципі його роботи, тобто решето передбачає фільтрацію, у разі фільтрацію всіх чисел крім простих. Принаймні обробки масиву чисел необхідні числа (прості) залишаються, а непотрібні (складові) виключаються.

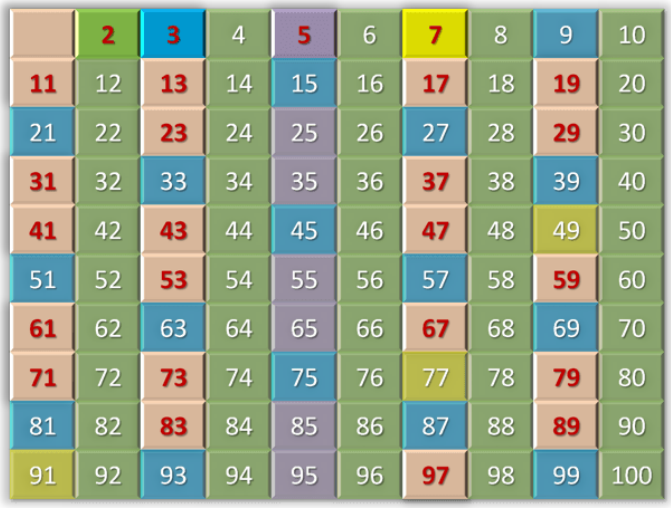
Сама проблема отримання простих чисел займає ключове місце у математиці, на ній засновані деякі криптографічні алгоритми, наприклад, RSA.

Для знаходження всіх простих чисел не більше заданого числа N потрібно виконати наступні кроки:

- Заповнити масив N елементів цілими числами поспіль від 2 до N.
- Присвоїти змінної p значення 2 (першого простого числа).
- Видалити з масиву числа від p2 до N з кроком p (це числа кратні p: p^2, p^2+p, p^2+2p і т. д.).
- Знайти перше невіддалене число в масиві, більше p, і привласнити значення змінної p це число.

- Повторювати два попередні кроки поки що це можливо.

Усі числа, що залишилися в масиві є простими числами від 2 до N. На рис. 69 проілюстровано алгоритм пошуку простих чисел. Числа, позначені білим шрифтом, є видаленими з масиву.



	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Рисунок 69. – Алгоритм роботи Решето Ератосфена.

Приклад 4. Використовуючи алгоритм решето Ератосфена, в динамічному масиві знайти всі прості числа.

Програмна реалізація методу пошуку:

```
#include <iostream>
using namespace std;
 int main()
{
 int n = 100; //Вважати числа до цього
  //Запитуємо масив
 int * a = new int [n + 1];
 //Наповнюємо його числами для решета
 for (int i = 0; i <= n; i++)
 a[i] = i;
  //********************************************** **********
 //Проводимо головний цикл. - Початок роботи решета
 for (int i = 2; i * i <= n; i++)
 {
 if (a[i])
 //Якщо поточне число не дорівнює 0 - починаємо від нього шукати складні
 for (int j = i * i; j <= n; j + = i)
 //І обнулюємо їхні комірки, щоб більше не перевіряти їх у циклі
 a[j] = 0;
 }
 // В масиві залишилися лише прості числа
 //**********************************************
  //Виводимо ненульові – прості числа
 for (int i = 2; i < n; i++)
 {
 if (a[i])
 {
 cout << a[i] << '';  }  }
```

```
cout << endl << endl;
delete [] a; //І звільняємо масив
return 0; }
```

Алгоритм пошуку підрядка в рядку в С++. Розберемо на прикладах, як може виглядати алгоритм пошуку підрядка у рядку. Приклади будуть ґрунтуватися на функціях стандартних бібліотек, адже саме у таких функціях і виявляються всі зручності написання програм. А ось класичний розбір алгоритму, заснований на циклах та порівняннях, також досить примітний. Сам алгоритм дуже простий. Є два рядки. Наприклад "Hello world" та "lo". Застосуємо два цикли:

- Перший буде виконувати прохід по всьому рядку, і шукатиме місце розташування першої літери рядка ("lo").
- Другий, починаючи зі знайденої позиції першої літери – звіряти, які літери стоять після неї і скільки їх поспіль збігаються.

На перших двох ітераціях циклу літери (рис. 70), що порівнюються, не збігаються (виділено червоним). На третій ітерації шукана літера (перший символ шуканого слова) збіглася із символом у рядку, де відбувається пошук. За такого збігу в роботу включається другий цикл.

Він спроможний відраховувати кількість символів після першого в рядку, який необхідно знайти та збігаються з символами в вихідному рядку. Якщо один із таких символів не збігається – цикл завершує свою роботу. Немає сенсу надалі переглядати рядок - після першої розбіжності зрозуміло, що шукане значення відсутнє. Проілюструємо пошук підрядка у рядку:

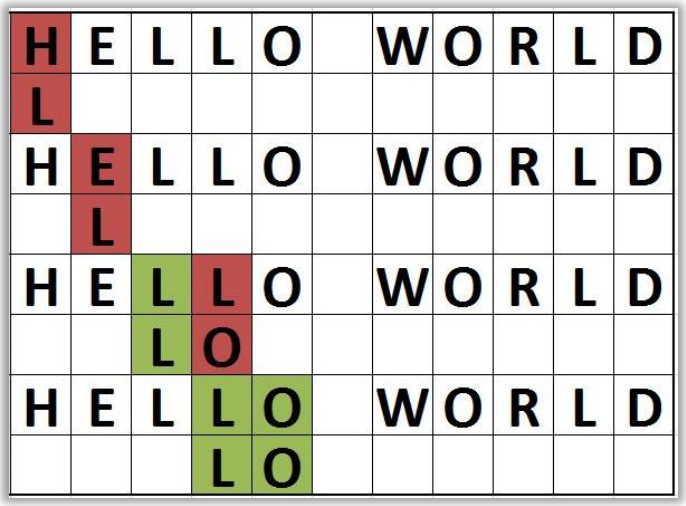


Рисунок 70. – Результат роботи алгоритму пошуку підрядка в рядку.

На третій ітерації збігся тільки перший символ рядка, що шуканого, а ось другий вже не збігається. Доведеться першому циклу продовжити роботу. Четверта ітерація дає необхідні результати - збігаються всі символи рядка з частиною вихідного рядка. А якщо всі символи збіглися – підрядок знайдено. Роботу алгоритму можна закінчити. Розглянемо класичний алгоритм пошуку підрядка в рядку на мові С++.

Приклад 5. Використовуючи алгоритм пошуку підрядка в рядку, знайти довільний рядок у символьному масиві.

Програмна реалізація методу пошуку:

```
#include <iostream>
 // Функція для пошуку підрядка у рядку
// + пошук позиції, з якої починається підрядок
int pos (char * s, char * c, int n)
{
 int i, j; // Лічильники для циклів
 int lenC, lenS; // Довжини рядків
  //Знаходимо розміри рядка вихідного та шуканого
 for (lenC = 0; c [lenC]; lenC ++);
 for (lenS = 0; s [lenS]; lenS ++);
  for (i = 0; i <= lenS - lenC; i++) // Поки є можливість пошуку
 {
 for (j = 0; s[i + j] == c[j]; j++); // Перевіряємо збіг посимвольно
 // Якщо посимвольно збігається за довжиною шуканого
 // Повернемо з функції номер комірки, звідки починається збіг
 // Враховувати 0-термінатор ( '\0' )
 if (j - lenC == 1 && i == lenS - lenC && !(n - 1)) return i;
 if (j == lenC)
 if (n - 1) n--;
 else return i;
 }
 //Інакше повернемо -1 як результат відсутності підрядка
 return -1;
}
int main()
{
 char * s = "parapapa";
 char * c = "pa";
  int i, n = 0;
  for (i = 1; n! = -1; i++)
 {
 n = pos(s, c, i);
  if (n> = 0)
 std:: cout << n << std:: endl;  } }
```

Два цикли виконують кожен своє завдання. Один переглядає рядок у пошуку заданого слова (перший символ). Другий з'ясовує, чи є після знайденого підрядка символи у рядку. Причому перевіряє, чи міститься кінець заданого рядка. Тобто, чи не є довжина знайденого слова на одиницю більше довжини рядка, якщо враховувати, що в цю одиницю потрапляє нуль-термінатор ('\0').

В результаті бачимо, що програма знайшла початок підрядка ***pa*** у комірках символьного масиву з індексом 0 і 4. Але в слові ***parapapa*** 3 такі підрядки. Вся справа в '\0'. Однак слід звернути увагу на множинність пошуку: скільки разів шукане слово зустрічається у рядку та в яких місцях. Саме третій параметр – int n – контролює номер входження в рядок. Якщо поставити туди одиницю – він знайде перший збіг шуканого. Якщо двійку, він змусить перший цикл пропустити знайдене перше значення і шукати друге. Якщо трійку – шукати третє і так далі. З кожним знайденим словом, яке потрібно знайти, лічильник зменшується на одиницю. Це і дозволяє описати пошук у циклі:

```
for (i = 1; n! = -1; i++)
{
n = pos(s, c, i);
if (n> = 0)
std:: cout << n << std:: endl;
}
```

Тобто знайти перший, другий, третій, четвертий та інші збіги до тих пір, поки функція не поверне -1, або вкаже на відсутність N-ного шуканого в рядка. Для порівняння, виконаємо пошук підрядки у рядку C++ з хедером ***string***.

Програмна реалізація методу пошуку:

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
setlocale(LC_ALL, "rus");
string s = "Hello world";
cout << "Знайдено в позиції " << s.find("lo") << endl;
}
```

Клас ***string*** в C++ забезпечений методом ***find()***, що повертає номер комірки, з якого починається тіло рядка, який потрібно знайти у вихідному рядку. Розглянемо програму і з врахуванням множинності.

Програмна реалізація методу пошуку:

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
string s = "parapapa";
int i = 0;
for ( i = s.find("pa", i++); i != string::npos; i = s.find("pa", i + 1))
cout << i << endl;
}
```

Функція find() приймає другим параметром номер символу, з якого розпочати пошук. Тобто. знайшовши перше входження, значення збільшується на одиницю і find() продовжує пошук з наступного символу після знайденої голови. Все це є в C++, і сам клас string досить зручний для роботи з рядками саме як рядками, а не просто з масивом символів.

Алгоритми пошуку в Python

Пошук даних, що зберігаються в різних структурах даних, є важливою частиною практично кожної окремої програми. Існує безліч різних алгоритмів, доступних для використання при пошуку, і кожен з них має різні реалізації та покладається на різні структури даних, щоб виконати свою роботу.

Вміння вибрати конкретний алгоритм для конкретної задачі є ключовою навичкою для розробників і може означати різницю між швидким, надійним та стабільним додатком та додатком, що руйнується від простого запиту.

- Оператори членства
- Лінійний пошук
- Бінарний пошук
- Пошук стрибка
- Пошук Фібоначчі
- Експонентний пошук
- Інтерполяційний пошук

Оператори членства. Алгоритми розвиваються та оптимізуються з часом у результаті постійної еволюції та необхідності пошуку найбільш ефективних рішень проблем, що лежать в їх основі, в різних галузях.

Однією з найпоширеніших проблем у галузі інформатики є пошук у колекції та визначення того, присутній даний об'єкт у колекції чи ні.

Майже кожна мова програмування має власну реалізацію базового алгоритму пошуку, зазвичай як функції, яка повертає логічне значення True чи False , коли елемент знайдено у цій колекції елементів.

В Python найпростіший спосіб пошуку об'єкта – використовувати оператори Membership – названі таким чином, тому що вони дозволяють нам визначити, чи є цей об'єкт членом колекції.

Ці оператори можуть використовуватися з будь-якою ітеративною структурою даних у Python, включаючи рядки, Списки та кортежі.

in – Повертає True , якщо елемент є частиною структури.

not in – Повертає True , якщо елемент не є частиною структури.

```
>>> 'apple' in ['orange', 'apple', 'grape']
True
>>> 't' in 'stackabuse'
True
>>> 'q' in 'stackabuse'
False
>>> 'q' not in 'stackabuse'
True
```

Оператори членства достатні, коли все, що нам потрібно зробити, це знайти, чи існує підрядок у даному рядку, чи визначити, чи перетинаються два рядки, списки чи кортежі з погляду об'єктів, які вони містять.

У більшості випадків нам потрібне положення елемента в послідовності, на додаток до визначення того, чи існує він чи ні; оператори членства не задовольняють цю вимогу.

Існує безліч алгоритмів пошуку, які не залежать від вбудованих операторів і можуть бути використані для більш швидкого та ефективного пошуку значень. Крім того, вони можуть дати більше інформації, наприклад, положення елемента в колекції, а не просто визначити його існування.

Лінійний пошук в Python. Лінійний пошук – один із найпростіших алгоритмів пошуку та найпростіший для розуміння. Алгоритм складається з ітерації за масивом та повернення індексу першого входження елемента після його знаходження:

```
def LinearSearch(lys, element):
    for i in range (len(lys)):
        if lys[i] == element:
            return i
    return -1
```

Отже, якщо використати функцію для обчислення:

print(LinearSearch([1,2,3,4,5,2,1], 2))

То після виконання коду отримаємо результат: **1.** Це індекс першого елемента, який відшукується.

Тимчасова складність лінійного пошуку дорівнює O(n) , що означає, що час, витрачений виконання, збільшується зі збільшенням кількості елементів у нашому вхідному списку ***lys***.

Лінійний пошук не часто використовується на практиці, тому що така ефективність може бути досягнута за допомогою вбудованих методів або існуючих операторів. Цей алгоритм не такий швидкий та ефективний, як інші алгоритми пошуку.

Лінійний пошук підходить для тих випадків, коли потрібно знайти перше входження елемента в несортованому масиві, тому що, на відміну від інших алгоритмів пошуку, цей алгоритм не вимагає сортування масиву до початку пошуку.

Бінарний пошук. Бінарний пошук швидший від лінійного, але вимагає, сортування масиву перед початком пошуку.

Припускаючи, що шукається значення ***val*** у відсортованому масиві, то алгоритм порівнює ***val*** із значенням середнього елемента масиву ***mid***.

Якщо ***mid*** – елемент, який відшукується, то повертається його індекс. В іншому випадку визначається з якого боку **mid | vail** з більшою ймовірністю буде знаходитись val, ґрунтуючись на тому, що менше або більше **mid**, і відкидається інша сторона масиву. Надалі рекурсивно чи ітеративно виконуються ті самі кроки, вибираючи нове значення для mid порівнюючи його з **val** і відкидаючи половину можливих збігів на кожній ітерації алгоритму.

Алгоритм бінарного пошуку може бути записаний рекурсивно чи ітеративно. Рекурсія зазвичай повільніша в Python, так як вимагає виділення нових комірок стека.

Оскільки алгоритм пошуку має бути максимально швидким і точним, розглянемо ітеративну реалізацію бінарного пошуку.

Програмна реалізація методу пошуку:

```
from random import randint
 # Створення списку та його сортування за зростанням
# виведення списку на екран
a = []
for i in range(10):
    a.append(randint(1, 50))
a.sort()
print(a)
 # шукане значення
value = int(input())
 mid = len(a) // 2
```

```
low = 0
high = len(a) - 1
 while a[mid] != value and low <= high:
   if value > a[mid]:
      low = mid + 1
   else:
      high = mid - 1
   mid = (low + high) // 2
 if low > high:
   print("No value")
else:
   print("ID =", mid)
```

В результаті виконання алгоритму пошуку отримаємо:

[8, 15, 20, 21, 24, 35, 39, 44, 48, 50]
21
ID = 3

Перерахуємо наступні дії алгоритму, які виконуються на кожній ітерації під час виконання алгоритму пошуку:

- Повернення індексу поточного елемента
- Пошук по лівій половині масиву
- Пошук у правій половині масиву

Можна вибрати тільки одну можливість кожної ітерації, і наш пул можливих збігів ділиться на дві в кожній ітерації. Це додає тимчасову складність алгоритму бінарного пошуку O(log n).

Бінарний пошук досить часто використовується на практиці, так як він є ефективним і швидким, порівняно з лінійним пошуком. Існують інші алгоритми пошуку, які є похідними від бінарного пошуку. Розглянемо деякі з них.

Метод пошуку Jump Search. Jump Search схожий на бінарний пошук в тому, що він працює з відсортованим масивом та використовує аналогічний підхід ***divide and conquer*** для пошуку по ньому.

Його можна класифікувати як вдосконалення алгоритму лінійного пошуку. У відсортованому масиві, замість того, щоб поступово шукати елементи масиву, задаємо значення так званого стрибка. Отже, у вхідному списку ***list***, якщо розмір стрибка ***jump***, алгоритм буде розглядати елементи в порядку lys[0], lys[0+jump], lys[0+2jump], lys[0+3jump] і т. д.

З кожним стрибком зберігається попереднє значення та його індекс. Коли знайдено набір значень, де lys[i] lys[i+jump], застосовується лінійний пошук з lys[i] як найлівіший елемент і lys[i+jump] як найправіший елемент у наборі пошуку.

Програмна реалізація методу пошуку:

```
import math
def JumpSearch (lys, val):
    length = len (lys)
    jump = int(math.sqrt(length))
    left = 0
    right = 0
    while left < length and lys[left] <= val:
       right = min(length - 1, left + jump)
       if lys[left] <= val and lys[right] >= val:
```

```
            break
        left+=jump;
    if left >= length or lys[left] > val:
        return -1
    right = min(length - 1, right)
    i = left
    while i <= right and lys[i] <= val:
        if lys[i] == val:
            return i
        i += 1
    return -1
```

Розглянемо покрокове обчислення методу ***Jump Search*** за заданим пошуком: print(JumpSearch([1,2,3,4,5,6,7,8,9], 5))

Jump Search спочатку визначить розмір масиву, обчислюючи math.sqrt(len(lys)). Оскільки в масиві є 9 елементів, розмір стрибка дорівнює $\sqrt{n}$.

Надалі обчислюється значення змінної ***right***, яке є мінімумом довжини масиву мінус 1, або значення ***left+jump***. Оскільки 3 менше 8, то використовується 3 як значення right. Тепер перевіряється, чи знаходиться пошуковий елемент 5 між lys[0] та lys[3] . Оскільки 5 не знаходиться між 1 та 4, то пошук продовжується.

Знову проводиться обчислення та перевіряється, чи знаходиться пошуковий елемент між lys[3] та lys[6]. Оскільки 5 знаходиться між 4 і 7, то застосовується лінійний пошук елементів між lys[3] і lys[6] та повертаємо індекс шуканого елемента: 4

Тимчасова складність пошуку ***Jump*** дорівнює O($\sqrt{n}$) , де $\sqrt{n}$– розмір стрибка, а ***n*** – довжина списку. Пошук ***Jump*** за ефективністю лежить між алгоритмами лінійного та бінарного пошуку. Важливою перевагою стрибкового пошуку в порівнянні з двійковим є те, що ним не використовується оператор розподілу (/). Щоб модернізувати алгоритм пошук ***Jump*** за швидкодією, в ньому можна використовувати двійковий пошук або інший внутрішній алгоритм пошуку, замість лінійний пошук.

Пошук Фібоначчі. Пошук Фібоначчі – алгоритм, який базується на принципі "розділяй і володарюй", має подібність з бінарним пошуком та ***Jump*** пошуком. Даний алгоритм отримав таку назву, так як у розрахунку використовує числа Фібоначчі для обчислення розміру блоку або діапазону пошуку на кожному кроці.

Числа Фібоначчі починаються з нуля і наслідують шаблон 0, 1, 1, 2, 3, 5, 8, 13, 21… де кожен елемент є додаванням двох чисел, які безпосередньо передують йому. Алгоритм працює з трьома числами Фібоначчі одночасно. Назвемо три числа ibm, ibm_minus_1 та ibm_minus_2 де fibM_minus_1 та ibm_minus_2 –два числа безпосередньо перед fibM у послідовності:

fibM = fibM_minus_1 + fibM_minus_2

При ініціалізації значень 0,1 і 1 або перші три числа в Фібоначчі-послідовності, у випадку уникнення помилки індексу, коли пошуковий масив ***lys*** містить невелику кількість елементів. Надалі вибирається найменше число Фібоначчі-послідовності, яке більше або дорівнює числу елементів пошукового масиві ***lys***, як значення ***fibM***, а наступні два числа Фібоначчі як значення

fibM_minus_1 та ***fibM_minus_2.*** Якщо в масиві залишилися елементи більші за одиницю, то порівнюється ***val*** зі значенням блоку в діапазоні до ***ibm_minus_2*** та повертається індекс знайденого елемента.

Якщо значення більше, ніж шуканий елемент, то переміщуються значення ***fibM***, ***fibM_minus_1*** та ***fibM_minus_2*** на два кроки вниз у Фібоначчі-послідовності і скидається індекс на індекс елемента.

Коли значення менше, ніж шуканий елемент, то переміщається значення ***fibM***, ***ibm_minus_1*** і ***fibM_minus_2*** на один крок вниз у Фібоначчі-послідовності. Розглянемо реалізацію даного алгоритму на Python:

```
def FibonacciSearch(lys, val):
    fibM_minus_2 = 0
    fibM_minus_1 = 1
    fibM = fibM_minus_1 + fibM_minus_2
    while (fibM < len (lys)):
        fibM_minus_2 = fibM_minus_1
        fibM_minus_1 = fibM
        fibM = fibM_minus_1 + fibM_minus_2
    index = -1;
    while (fibM > 1):
        i = min(index + fibM_minus_2, (len(lys)-1))
        if (lys[i] < val):
            fibM = fibM_minus_1
            fibM_minus_1 = fibM_minus_2
            fibM_minus_2 = fibM - fibM_minus_1
            index = i
        elif (lys[i] > val):
            fibM = fibM_minus_2
            fibM_minus_1 = fibM_minus_1 - fibM_minus_2
            fibM_minus_2 = fibM - fibM_minus_1
        else :
            return i
    if(fibM_minus_1 and index < (len(lys)-1) and lys[index+1] == val):
        return index+1;
    return -1
```

Для пошуку скористаємось викликом створеної функції пошуку Фібоначчі: ***print(FibonacciSearch([1,2,3,4,5,6,7,8,9,10,11], 6))***

Розглянемо покроковий процес такого пошуку:

1. Визначення найменшого числа Фібоначчі, більшого або рівного довжині списку як ibm; у цьому випадку найменше число Фібоначчі дорівнює 13.

2. Надалі перевіряється елемент lys[4], де 4-мінімум -1 +5. Оскільки значення lys[4] дорівнює 5, що менше значення шуканого елемента, то переміщаються числа Фібоначчі на один крок вниз у даній послідовності.

3. Надалі перевіряється елемент lys[7], де 7-мінімум 4+3. Оскільки значення lys[7] дорівнює 8, що більше шуканого значення, тоді переміщаються числа Фібоначчі на два кроки вниз у послідовності.

4. Тепер перевіряється елемент lys[5], де 5-мінімум 4+1. Значення lys[5] дорівнює 6, що і є шуканим значенням.

Тимчасова складність пошуку Фібоначчі, як і в бінарному пошуку, дорівнює O(log n). Це означає, що в більшості випадків алгоритм працює швидше, ніж лінійний пошук та ***Jump*** пошук.

Пошук Фібоначчі може бути використаним на великій кількості елементів. Додатковою перевагою пошуку Фібоначчі є можливість внесення великих за розміром вхідних масивів в кеші процесора або оперативної пам'яті, оскільки пошук елементів здійснюється з кроковими розмірами, а не з фіксованими.

Експоненційний пошук. Експонентний пошук залежить від бінарного пошуку для виконання остаточного порівняння значень. Алгоритм працює наступним чином:

- визначається діапазон, в якому, ймовірно, буде знаходитись шуканий елемент;
- використовується двійковий пошук в масиві для визначення точного індексу елемента

Реалізація алгоритму експоненційного пошуку на Python записується наступним чином:

```
def ExponentialSearch(lys, val):
    if lys[0] == val:
        return 0
    index = 1
    while index < len(lys) and lys[index] <= val:
        index = index * 2
    return BinarySearch( arr[:min(index, len(lys))], val)
```

Якщо викликати створену функцію для знаходження певного значення, то отримаємо відповідний результат:

```
>>> print(ExponentialSearch([1,2,3,4,5,6,7,8],3))
```

Даний алгоритм виконує пошук наступним чином:

1. Перевіряється, чи відповідає перший елемент у списку шуканому значенню – оскільки lys[0] дорівнює 1, то знаходиться значення 3. Встановлюється індекс рівним 1 і пошук далі виконується.

2. Переглядаються всі елементи у масиві, і поки елемент на індексній позиції менше або дорівнює шуканому значенню, експоненційно збільшуємо значення index кратне двом:

- індекс lys[1] дорівнює 2, що менше 3, тому індекс множиться на 2 та встановлюється рівним 2.
- індекс lys[2] дорівнює 3, що дорівнює 3, тому індекс множиться на 2 та встановлюється рівним 4.
- індекс lys[4] дорівнює 5, що більше ніж 3; у цій точці цикл переривається.

3. Виконується двійковий пошук шляхом поділу списку: are[:4]. В Python це означає, що вкладений список буде містити всі елементи до 4-го елемента, тому викликається: ***BinarySearch([1,2,3,4], 3)***

В результаті буде поверне: ***2***. Дане значення є індексом знайденого елемента у вихідному списку, та у списку, який передається алгоритму двійкового пошуку.

Експонентний пошук виконується за ***O(log i)*** час, де ***i*** - індекс шуканого елемента. Тимчасова складність алгоритму дорівнює ***O(log n)***, де ***n*** – довжина масиву. Експоненційний пошук має вищу швидкодію ніж двійковий пошук. На практиці часто використовуємо експоненційний пошук, тому що це найбільш ефективних алгоритм пошуку для необмежених та нескінченних масивів.

Інтерполяційний пошук. Інтерполяційний пошук - алгоритм типу "розділяй і володарюй", аналогічний до бінарного пошуку. На відміну від бінарного пошуку, даний алгоритм починає пошук даних з середини масиву. Інтерполяційний пошук обчислює ймовірне положення шуканого елемента за формулою:

index = low + [(val-lys[low])*(high-low) / (lys[high]-lys[low])]

де, ***lys*** –вхідний масив;

val – шуканий елемент;

index - ймовірний індекс елемента пошуку;

low – початковий індекс масиву;

high – кінцевий індекс масиву.

Алгоритм здійснює пошук шляхом обчислення значення індексу:

1. Якщо збіг знайдено (коли ***lys[index]***), то повертається значення індексу.
2. Якщо значення ***val*** менше ***lys[index]***, то значення індексу перераховується за формулою для лівого підмасиву.
3. Якщо значення ***val*** більше, ніж ***lys[index]***, то значення індексу перераховується за формулою для правого підмасиву

Реалізуємо алгоритм інтерполяційного пошуку у Python:

```
def InterpolationSearch(lys, val):
    low = 0
    high = (len (lys) - 1)
    while low <= high and val >= lys[low] and val <= lys[high]:
        index = low + int(((float(high - low) / (lys[high] - lys[low])) * (val - lys[low])))
        if lys[index] == val:
            return index
        if lys[index] < val:
            low = index + 1;
        else:
            high = index-1;
    return -1
```

Якщо викликати створену функцію для заданого масиву:

```
>>> print(InterpolationSearch([1,2,3,4,5,6,7,8], 6))
```

Оскільки lys[5] дорівнює 6, що і є шуканим значенням, то припиняється виконання алгоритму і повертається результат: ***5***

Якщо задана велика кількість елементів, і індекс не може бути обчисленим за одну ітерацію, то продовжується перерахунок значень для ***index*** після коригування значень ***high*** і ***low*** у формулі.

Тимчасова складність інтерполяційного пошуку становить ***O(log(log n))*** за рівномірного розподілу значень. Якщо значення розподілені нерівномірно, то часова складність, як і в лінійному пошуку дорівнює ***O(n)***.

Інтерполяційний пошук найкраще працює на рівномірно розподілених та відсортованих масивах. У той час як двійковий пошук починає пошук з середини і завжди ділиться навпіл, то інтерполяційний пошук обчислює можливе положення елемента і перевіряє індекс, що робить його вірогіднішим для пошуку елемента за меншу кількість ітерацій.

Рекомендації до застосування алгоритмів пошуку. Існує безліч можливих способів пошуку елемента в масивах. Вибір певного алгоритму ґрунтується на даних, які повинні знаходитись у вхідному масиві за наступних умов:

- Якщо здійснюється пошук в несортованому масиву або потрібно знайти перше входження змінної - найкращим варіантом є ***лінійний пошук***.
- Якщо виконується пошук у відсортованому масиву, то варто застосувати найпростіший і найшвидший метод - ***двійковий пошук***.
- Якщо у відсортованому масиві не застосовувати оператор поділу, то можна скористатись алгоритмом ***пошуку Фібоначчі***.
- Якщо шуканий елемент ймовірно буде розміщеним на початку масиву, то варто скористатись алгоритмом ***експоненційного пошуку***.
- Якщо відсортований масив рівномірно розподілений, то найшвидшим та найефективнішим алгоритмом пошуку буде ***інтерполяційний пошук***.

Якщо ви не впевнені, який алгоритм використовувати у відсортованих масивах, то можна почергово використати кожен із алгоритмів та вибрати той, який найкраще і найшвидше працює з певним набором даних.

Контрольні запитання

1. По яких параметрах можна оцінювати алгоритм?
2. Як обчислюється часова складність алгоритму?
3. Які алгоритми сортування називаються стійкими?
4. Чим можна пояснити різноманіття алгоритмів пошуку в лінійних структурах?
5. У чому переваги пошуку з бар'єром порівняно з послідовним пошуком?
6. Знаходження якого по порядку елемента в лінійній множині гарантує алгоритм прямого пошуку? Як в цьому випадку повинен бути виконаний перегляд?
7. Чим можна пояснити різноманіття алгоритмів пошуку в лінійних структурах?
8. У чому переваги алгоритму пошуку з бар'єром порівняно з алгоритмом послідовного пошуку?
9. Знаходження якого за порядком елемента у лінійній множині гарантує алгоритм бінарного пошуку? Відповідь обґрунтуйте.
10. Як виконується пошук за алгоритмом інтерполяційного пошуку?
11. Яка розрахункова формула застосовується при використанні алгоритму експоненційного пошуку?
12. Поясніть принцип пошуку за алгоритмом ***Jump Search***.

Практичне завдання 6.
Розробка програм з рядковими змінними та функціями користувача

Мета: Навчитися працювати з текстовими даними, отримати знання і навички, необхідні для програмування на основі створення і використання користувацьких функцій, та навчитися використовувати їх на практиці в процесі розроблення програм мовою програмування С++

Завдання 1. Результати екзаменаційної сесії студентів 1-го курсу подані у вигляді таблиці (***табл. 16***). Необхідно сформувати дані відповідно до наведеного варіанту (***табл. 17***). Необхідно сформувати текстові масиви, опрацювати їх та вивести кінцеві результати.

Таблиця 16. Результати екзаменаційної сесії студентів

№	Прізвище	Іноземна мова	Вища математика	Фізика	Програмування
1.	Іванчук С.О.	4	3	3	4
2.	Пащенко І.А.	5	4	4	5
3.	Зайцев О.М.	3	4	4	4
4.	Вербицький П.О.	4	3	3	3
5.	Сінченко В.Р.	2	3	3	2
6.	Кравець З.І.	3	5	4	5
7.	Якусевич Р.Н.	5	4	4	3
8.	Зінченко П.М.	4	2	3	3
9.	Бовсуновський Г.С.	4	5	5	5
10.	Демченко Н.С.	5	5	4	4

Таблиця 17. Варіанти до виконання завдання 1

№п/п	Завдання до виконання
1.	Надрукувати таблицю, що містить номери, прізвища та кількість “5”, “4”, “3”,”2” у кожного студента групи, а також підрахувати загальну кількість “5”, “4”, “3”, “2” в групі.
2.	Надрукувати таблицю, що містить номери, прізвища, оцінки та середній бал тих студентів групи, середній бал яких більше 4, а також підрахувати кількість таких студентів у групі.
3.	Надрукувати таблицю, що містить номери, прізвища та оцінки студентів, які мають хоча б одну “3”, а також підрахувати кількість таких студентів у групі.
4.	Надрукувати таблицю, що містить номери, прізвища та оцінки студентів, які не мають жодної “5”. Підрахувати кількість таких студентів.
5.	Надрукувати таблицю, що містить номери, прізвища та оцінки кожного студента, а в кінці вказати середній бал групи з кожної дисципліни.
6.	Надрукувати таблицю, що містить номери, прізвища, оцінки та середній бал кожного студента групи.
7.	Надрукувати таблицю, що містить номери, прізвища та оцінку студентів з вищої математики, а також підрахувати середній бал групи з цього предмета.
8.	Надрукувати таблицю, що містить прізвища та оцінки тих студентів, які мають найбільший та найменший середній бал у групі.
9.	Надрукувати таблицю, що містить номери, прізвища, оцінки та середній бал студентів групи, середній бал яких менше 4.

№п/п	*Завдання до виконання*
10.	Надрукувати таблицю, що містить номери, прізвища, оцінки студентів, які мають тільки добрі та відмінні оцінки.
11.	Надрукувати таблицю, що містить номери, прізвища, оцінки та кількість “3” в оцінках кожного студента.
12.	Надрукувати таблицю, що містить номери, прізвища та оцінки тих студентів, які отримали з інформатики добрі та відмінні оцінки, а також підрахувати кількість таких студентів.
13.	Надрукувати таблицю, що містить номери, прізвища та оцінки тих студентів, які отримали з вищої математики задовільну або незадовільну оцінку, а також підрахувати кількість таких студентів.
14.	Надрукувати таблицю, що містить номери, прізвища та екзаменаційні оцінки студентів. В кінці вказати дисципліну, середній бал якої максимальний.
15.	Надрукувати таблицю, що містить номери, прізвища та оцінки студентів, які отримали хоча б одну незадовільну оцінку.
16.	Надрукувати кількість “2”, “3”, “4”, “5” з кожної дисципліни.
17.	Надрукувати таблицю, що містить номери, прізвища і кількість “2”, “3”, “4”, “5” в оцінках кожного студента.
18.	Надрукувати таблицю, що містить номери, прізвища і оцінки студентів з предметів “ Вища математика” і “ Іноземна мова”.
19.	Надрукувати таблицю, яка містить середні екзаменаційні бали студента по кожному предмету.
20.	Надрукувати таблицю, що містить номери, прізвища, оцінки та кількість позитивних оцінок кожного студента.
21.	Надрукувати таблицю, що містить номери, прізвища та оцінки студентів, які мають позитивні оцінки з іноземної мови.
22.	Надрукувати таблицю, що містить номери, прізвища та оцінки студентів, які мають позитивні оцінки з вищої математики. Надрукувати кількість таких студентів.
23.	Надрукувати таблицю, що містить номери, прізвища та оцінки студентів, які мають оцінки “добре” та “відмінно” з інформатики. Надрукувати кількість таких студентів.
24.	Надрукувати таблицю, що містить номери, прізвища та оцінки студентів, які мають позитивні оцінки з фізики. Надрукувати кількість таких студентів.
25.	Надрукувати таблицю, що містить номери, прізвища та оцінки студентів, які мають оцінки “добре” та “відмінно” з програмування. Надрукувати кількість таких студентів.
26.	Надрукувати таблицю, що містить прізвища та оцінки тих студентів, які мають найбільший та найменший середній бал у групі.
27.	Надрукувати таблицю, що містить номери, прізвища, оцінки та середній бал студентів групи, середній бал яких менше 3.
28.	Надрукувати таблицю, що містить номери, прізвища, оцінки студентів, які мають тільки відмінні оцінки.
29.	Надрукувати таблицю, що містить номери, прізвища, оцінки та кількість “2” в оцінках кожного студента.
30.	Надрукувати таблицю, що містить номери, прізвища та оцінки тих студентів, які отримали з програмування добрі та відмінні оцінки, а також підрахувати кількість таких студентів.

Завдання 2. Виконати завдання відповідно до наведеного в ***табл. 18*** варіанту. Необхідно сформувати текстові масиви, опрацювати їх та вивести кінцеві результати.

Таблиця 18. Варіанти до виконання завдання 2

№п/п	*Завдання до виконання*
1.	Дано текстовий масив А(10). Знайти і надрукувати елементи найменшої довжини. Вивести на друк даний елемент, його порядковий номер і довжину (кількість символів).
2.	В текстовому масиві В(12) відшукати елемент з найбільшою довжиною, вивести його на друк разом з номером і довжиною.
3.	В текстовому масиві С(15) знайти суму довжин елементів з найменшою та найбільшою довжиною.
4.	З елементів текстового масиву В(20) сформувати масиви, елементи яких мають однакову довжину.
5.	В текстовому масиві А(15) поміняти місцями елементи з найменшою та найбільшою довжинами.
6.	В текстовому масиві А(13) поміняти місцями:1-й елемент з 13-м, 2-й з 12-м, і т. д. Вивести на друк початковий та перетворений масиви.
7.	Дано масив А(10) вивести на друк елементи в зростаючому порядку їх довжини.
8.	Масив В(10) містить прізвища студентів. Впорядкувати його в алфавітному порядку.
9.	Дано текстовий масив: папір, вода, башта, канал, висота, об'єм. Об'єднати 2-й і 4-й елементи масиву і отриману текстову змінну поставити на друге місце, 4-й елемент масиву знищити.
10.	Дано текстовий масив В(12). Відсортувати його в порядку спадання довжин його елементів.
11.	Дано числовий масив оцінок: 3, 4, 4, 5, 2, 3, 3, 4. Сформувати текстовий масив оцінок, замінивши: 3 на задовільно, 4 на добре, і т. д. Надрукувати отриманий масив.
12.	Дано числовий масив оцінок студентів: 3, 4, 4, 5, 2, 3, 3, 4. Підрахувати середній бал кожного студента.
13.	Дано текстовий масив А(10). Вивести на друк його елементи в зростаючому порядку їх довжин
14.	Дано текстовий масив А(10). Вивести на друк його елементи вспадаючому порядку їх довжин.
15.	В текстовому масиві А(8) даних, що містить 8 слів, підрахуватисуму довжин елементів що стоять на парних місцях.
16.	В текстовому масиві F(10) підрахувати суму довжин перших 7-миелементів.
17.	В текстовому масиві з 9-ти елементів знайти сумарну довжину елементів з 2 по 6.
18.	Дано текстовий масив В(12). Відсортувати його в порядку спадання довжин його елементів і записати отриманий масив в А(12).
19.	Дано масив текстових змінних В(10). Створити масив с(10), щомістить елементи масиву В(12) записані у зворотному порядку.
20.	Дано текстовий масив А(10). Знайти і надрукувати елементи найбільшої довжини. Вивести на друк даний елемент, його порядковий номер і довжину (кількість символів).
21.	В текстовому масиві С(15) знайти різницю довжин елементів з найменшою та найбільшою довжиною. Надрукувати ці елементи.
22.	Дано текстовий масив В(12). Відсортувати його в порядку зростаннядовжин його елементів.
23.	В текстовому масиві з 10-ти елементів знайти суму довжин елементів з 3 по 9.
24.	Дано масив текстових змінних В(10). Створити масив С(10), що містить елементи масиву В(12), збільшені на 3 кожний.

№п/п	Завдання до виконання
25.	В текстовому масиві A(15) поміняти місцями елементи з найменшою та найбільшою довжинами.
26.	В текстовому масиві A(15) поміняти місцями: 1-й елемент з 15-м, 2-й з 14-м, і т. д. Вивести на друк початковий та перетворений масиви.
27.	Дано текстовий масив A(20). Знайти і надрукувати елементи найбільшої довжини. Вивести на друк даний елемент, його порядковий номер і довжину (кількість символів).
28.	В текстовому масиві B(14) відшукати елемент з найбільшою довжиною, вивести його на друк разом з номером і довжиною.
29.	В текстовому масиві F(12) підрахувати суму довжин перших 6-ти елементів.
30.	В текстовому масиві B(14) відшукати елемент з найменшою довжиною, вивести його на друк, його індекс та довжиною.

Зміст звіту

1. Титульна сторінка, тема та мета практичного заняття.
2. Підготувати звіт по виконаній роботі, у якому представити наступні матеріали:
   - ✓ зміст завдання і варіант;
   - ✓ лістинг програми.
   - ✓ схему ієрархії класів програмного результати виконання – вигляд екрану при виконанні програми;
   - ✓ результати роботи розробленого програмного засобу;
3. Висновки, які відображають результати виконаних завдань та їх аналіз.

Теоретичні відомості

Символи та рядки. Змінні символьного типу оголошуються за допомогою ключового словаchar і займають у пам'яті 1 байт. Тип char є цілочисельним типом і може задаватися зі знаком або без знаку. Спосіб інтерпретації змінних типу char може задаватися неявно або явно. Неявна форма типу char визначається опцією компілятора. В інтегрованому середовищі ця опція задається за допомогою меню Options/Compile/Code Generation. Явна форма визначається за допомогою модифікаторів типу signed або unsigned.

Приклади оголошень:

char c; unsigned char t;signed
char v;

Значення змінної типу char визначає код одного із 256 символів кодової таблиці.

Якщо тип char розглядається як signed; то старший біт коду визначає знак. В цьому випадку діапазон значень типу char становить від - 128 до 123. Для типу unsigned char діапазон значень коду становить від 0 до 255. Ініціалізація змінних типу char може здійснюватися неявно або явно.

Неявно статичні та глобальні змінні типу char ініціалізуються значенням '\0'. Локальні змінні, які не є статичними, приймають невизначене значення. Явна ініціалізація змінних типу char може здійснюватися при їх оголошенні або використанням операції присвоєння чи функцій вводу.Змінній типу char можна

присвоїти числове або символьне значення. Символьна константа задається в апострофах явно або своїм вісімковим чи шістнадцятковим кодом, перед яким повинен йти символ \, наприклад:

char c1='A'; char c1='x41';

char c3,c4=0x41;

В усіх випадках змінні приймуть значення 0x41 (або десяткове 65), яке, в залежності від контексту використання, можна інтерпретувати як число, або символ з відповідним кодом.

Масиви символів. Масив символів - це послідовність елементів типу char, розміщених у неперервній області пам'яті. Приклад оголошення масиву символів, наприклад: ***char buffer[10];***

У масив символів можуть входити латинські букви, символи кирилиці, знаки пунктуації і керуючі символи. Керуючі символи задаються своїм мнемонічним позначенням або значенням ASCII-коду, перед яким повинен записуватися символ \.

Ініціалізація масиву символів може здійснюватися одним з наступних способів:

- ✓ по змовчуванню глобальні та статичні масиви ініціалізуються символом '0';
- ✓ явна посимвольна ініціалізація під час оголошення;
- ✓ посимвольна ініціалізація за допомогою операції присвоєння або посимвольного вводу.

Якщо масив символів ініціалізується під час його оголошення, то кількість елементів може вказуватися явно, наприклад:

- ***char bufrer[10]={'V','i','s','u','a','l',' ','C++'};***

або неявно:

- ***char buffer[]={'V','i','s','u','a','l',' ','C++'};***

В першому прикладі задається масив з 10 елементів. Першим 7 з них будуть присвоєні значення, а наступні 3 приймуть значення '0', якщо масив є глобальним або статичним, і невизначені значення - в інших випадках. Кількість елементів списку ініціалізації не повинна перевищувати заданого значення.

В другому прикладі кількість елементів масиву визначається неявно по їх списку, В результаті буде створений масив з 7 символів, які приймуть значення із заданого списку ініціалізації.

Доступ до елементів масиву може бути здійснений за допомогою індексу або вказівника. Індекс масиву - це вираз цілого типу, який записується в квадратних дужках після імені масиву. Індекс першого елемента масиву дорівнює 0. Наприклад, для звертання до символу 'r' раніше визначеного масиву, необхідно записати buffer[2]. При доступі до символів за допомогою індексу необхідно пам'ятати, що компілятор не здійснює контроль його допустимого значення. Некоректне використання індексу може призвести до звертання за межі масиву без видачі зауваження або повідомлення про помилку.

Враховуючи, що ім'я масиву є вказівник-константа на перший по порядку байт цього масиву (елемент buffer[0]), для доступу до - і-го елемента масиву

символів можна використати операцію *, наприклад;

***(buffer+i)**

Посимвольна ініціалізація за допомогою операції присвоєння здійснюється шляхом звертання до окремих елементів масиву одним із приведених нижче способів:

buffer[1]='U';

***(buffer+2)='R';**

По-елементний ввід масиву символів організується за допомогою оператора циклу, наприклад:

for(i=0;i<10;i++)

getchar(buffer[i]);

Для вводу символів можна використати спеціально призначені бібліотечні функції getchar, getch, getche; або функцію форматованого вводу scar.

Масиви символів інакше називаються буферами символів. Для роботи з буферами символів призначені спеціальні функції, прототипи яких приведені у файлі mem.h.

Рядки символів. Рядок символів (string) на мові С - це масив символів, що закінчується символом '\0' (NULL), Оголошення рядків здійснюється так само, як масивів символів, наприклад::

char line[20]; */* оголошений рядок із 20 символів */*

В дійсності в такий масив можна записати 19 символів, а 20-й символ є признаком кінця рядка '\0', під який обов'язково необхідно зарезервувати пам'ять. Кожен символ рядка займає в пам'яті 1 байт. Максимальна дов- жина рядка залежить від вибраної моделі пам'яті.

Початкова ініціалізація рядків символів може здійснюватися двома способами. При першому способі символи задаються як елементи масиву, наприклад: ***charstr[6]= {'w','h','i','l','e','\0'};***

При такому способі оголошення рядка символ '\0' повинен вказуватися явно в кінці масиву елементів. Якщо кількість перерахованих у фігурних дужках символів є меншою від вказаної розмірності, то рядок доповнюється справа до визначеної довжини символами '\0'.

Якщо символ '\0' в кінці списку ініціалізації не вказаний, то в пам'яті буде сформований не рядок, а масив символів.

При другому способі ініціалізації рядок не розділяється на окремі символи, а задається в лапках, наприклад:

char str[] = "while";

При такому способі ініціалізації компілятор сам додасть символ '\0' в кінець рядка. Рядок символів може бути пустим. Такий рядок складається тільки з одного символа '\0' і ініціалізується наступним чином:

char str[]="";

Звертання до символа масиву здійснюється по його порядковому номеру, наприклад: ***char c, str[]="while";c=str[2]; //c='i'***

Ім'я рядка символів є вказівником на нульовий елемент (str ==&str[0]), тобто приймає константне значення, присвоєне йому на етапі компіляції. Тому не

дозволяється модифікація цього значення, наприклад операція str++ є недопустимою. Для оголошення рядка символів можна використовувати вказівник, наприклад:

char *str1 = "while";

або

char *str1; str1="while";

Змінна str1 містить адресу рядка "while". На відміну від попереднього опису рядка як масиву символів, допускається її модифікація, наприклад операція str1++ допустима і нове значення змінної str1 вказує на наступний елемент.

Вираз *(srt1+i) забезпечує звертання до і-го елементу рядка символів, наприклад:

c=*(str1+2) */*c='i' */*

Масиви символьних рядків. Рядки символів можна об'єднувати в масиви довільної розмірності. На практиці найбільш поширена робота з двомірними масивами символів (одномірними масивами рядків символів).

В залежності від опису рядки символів зберігаються в неперервній області пам'яті як двомірні символьні масиви або в різних областях пам'яті у вигляді окремих рядків. Перший спосіб опису:

charmasstr[4][10]={"green", "red", "white", "blue"};

визначає прямокутний масив символів. Всі чотири рядки матимуть однакову довжину. Короткі рядки доповнюються справа символами '\0' до вказаної довжини (до 10 символів).

Опускаючи другий індекс, будемо мати адреси кожного з рядків, наприклад, запис masstr[2] визначає вказівник на рядок "white", а запис *(masstr[2]+j) визначає j-й символ цього рядка (j = 0,...,9).

Для доступу до символів можна використовувати також спосіб, характерний для числових масивів, наприклад, запис masstr[0][1] визначає символ 'r' рядка "green".

Другий спосіб опису:

char *prtstr[4]={"green", "red", "white", "blue"};

визначає масив із 4-х вказівників на рядки різної довжини. Значення рядків, можливо, будуть зберігатися не підряд, а в різних областях пам'яті.

Для доступу до j-го символу і-го рядка можна скористатися наступним виразом; *(ptrstr[i]+j).

Організація стандартного виводу та вводу символьних даних.

Форматоване введення даних здійснюється за допомогою функції printf, яка має змінну кількість аргументів. Прототип функції приведений вище у таблиці. Звертання до функції:

printf ("форматований рядок", арг1,арг2,...);

Форматний (керуючий) рядок використовується для того, щоб задати кількість та типи аргументів і може включати в себе:

1. звичайні символи, які виводяться на екран дисплея;

2. специфікації перетворення даних, кожна з яких викликає на екран значення наступного аргумента із списку,
3. керуючі символьні константи:
   - ***\a - викликає звуковий сигнал;***
   - ***\n - перехід на новий рядок;***
   - ***\b - повернення на позицію вліво;***
   - ***\r - перехід на початок біжучого рядка;***
   - ***\f - перехід на нову сторінку;***
   - ***\t - горизонтальна табуляція;***
   - ***\v - вертикальна табуляція;***
   - ***\ddd - вісімковий код символу;***
   - ***\' - апостроф;***
   - ***\xddd - шістнадцятковий код символу;***
   - ***\" - подвійні лапки;***
   - ***\0 - нульовий символ (пусто);***
   - ***\\ - зворотна коса риска;***

Специфікація перетворення має наступний формат:

- ***%[вирівнювання][ширина][точність]символ перетворення***

Квадратні дужки не є символами специфікації, а лише вказують, що дане поле може бути опущене. Специфікація перетворення починається знаком % і закінчується символом перетворення (форматом), між якими можуть бути:

1. Знак "-" (мінус), який показує, що перетворений параметр повинен бути вирівняний вліво у своєму полі (по змовчуванню вирівнюється вправо);
2. Рядок цифр - мінімальна ширина поля. Якщо значення змінної перевищує ширину поля, то виводиться стільки символів, скільки потрібно;
3. Рядок цифр - максимальне число символів, які необхідно вивести для типу char. Аргументами функції printf можуть бути змінні, константи, вирази, виклики функцій. Значення аргументів повинні відповідати заданій специфікації.

Виведення символів. Специфікація перетворення:

%[-][ширина]c

Вивід символів рядка здійснюється доти, поки не зустрінеться символ '\0', або до вказаної точності. Якщо довжина рядка більша від заданої точності, то залишок рядка відкидається.

Здійснюється за допомогою функції scanf, яка може мати змінну кількість аргументів. Функція scanf призначена в основному для зчитування сукупності даних різних типів. Формат функції:

scanf("форматний рядок", арг1, арг2, ...);

Характерною особливістю даної функції є те, що її аргументи повинні бути вказівниками на значення. Для кожного аргумента у форматному рядку задається своя специфікація перетворення.

Введення символу. Специфікація перетворення:

%[*][ширина]c

Ширина визначає число символів, які повинні бути прочитані із вхідного

потоку і присвоєні масиву символів. Якщо ширина опущена, то вводиться один символ. По даній специфікації можна вводити пусті символи.

Приклад:

```
char a[5], b; /* Вхідний потік: 1234567890 */
scanf("%5c", a); a[4]='\0';      /* Результат a = 1234\0 */
scanf("%c", &b); /* Результат b = 5  */
```

Введення рядка символів. Специфікація перетворення:

%[*][ширина]s

Ширина задає максимальну довжину ввідного рядка. Рядки у вхідному потоці повинні розділятися пустими символами. Ведучі пусті символи ігноруються. Зчитування відбувається до першого пустого символу (пропуска, табуляції переходу на новий рядок), або до закінчення вказаної ширини. В пам'яті у кінець рядка додається символ '\0' для заповнення оголошеної довжини.

Приклад:

```
char a[5], b[6]; /* Вхідний потік; 1234567890 */
scanf("%3s", a); /* Результат a = 123\0\0 */
scanf("%5s", b); /* Результат b = 45678\0 */
```

Функції безформатованого виводу. Виведення символів. Функція putchar використовується у програмі для відображення символу на екрані відео-термінала. Звернення до функції:

putchar(символ);

Після виводу символа курсор залишається в рядку виводу. Якщо вивідний рядок заповнений, то при виводі наступного символу відбувається перехід на початок нового рядка. Приклади виводу символів:

```
char ch='b';
putchar('a');  /* a */
putchar('\n'); /* перехід на новий рядок */
putchar('\007'); /* звуковий сигнал */
putchar(ch);   /*b */
putchar(getchar()); /* вивід введеного символу */
```

Виведення рядка символів. Для виводу рядка символів використовується функція puts. Звертання до функції:

puts(вказівник на рядок);

Функція puts зупиняє вивід символів, якщо зустріне символ '\0'. Вивід цього символу не здійснюється. Рядок символів, що виводиться функцією puts завжди починається з нового рядка на екрані. Приклади використання функції puts:

```
char str1 [] = "abcdefgh";
char *str2 = "1234567890";
 puts("Вивід повідомлення");
puts(str1);   /*abcdefgh*/
puts(str2); /* 1234567890 */
puts(&str1[4]); /*efgh*/
puts(str2+6);        /* 7890 */
```

Функції безформатованого вводу. Введення символів. Для вводу символів використовуються функції getchar, getch та getche. Прототип функції getchar знаходиться у файлі stdio.h, а прототипи функцій getch та getche - у файлі conio.h.

Функція getchar призначена для зчитування символа з клавіатури, відображення його на екрані та передачі у програму. Функція реалізує буферизований ввід - передача символу у програму відбувається після його набору на клавіатурі та натискання клавіши <Enter>.

Звертання до функції getchar; змінна = getchar();

Змінна, якій присвоюється значення функції, повинна мати тип char.

Наприклад:

char ch; ch=getchar();

while((ch=getchar())!='*') *{/* тіло циклу */}*

Функція getch реалізує небуферизований ввід - передача символу у програму відбувається зразу після його набору на клавіатурі без натискання клавіші <Enter>. Характерною особливістю даної функції є також те, що введений символ не відображається на екрані.

Звертання до функції getch:

змінна = getch();

Функція getche теж реалізує небуферизований ввід символу, але з відображенням його на екрані. Звертання до функції getche:

змінна = getche();

Якщо у вхідному потоці функції getchar(), getch() або getche() зустрінуть код кінця файлу, то вони повертають признак EOF. Оголошення змінної EOF приведено у стандарному файлі stdio.h. Ознака кінця файлу імітується одночасним натисканням двох клавіш "Ctrl+Z" (при введенні символа з клавіатури). Це можна використати як умову виходу з циклу, якщо введений символ використовується в тілі циклу, наприклад:

char ch; while((ch=getchar())!=EOF)

{ }

Введення рядків символів. Для вводу рядка символів використовується функція gets. Дана функція читає символи з вхідного потоку доти, поки не зустрінеться символ нового рядка '\n', який формується при натисканні клавіші <Enter>. Символ '\n' не включається в кінець рядка, а замість нього автоматично формується символ '\0', на що необхідно зарезервувати додатковий байт пам'яті.

Перед використанням функції gets необхідно виділити пам'ять для рядка символів. Звертання до функції gets:

gets(вказівник на рядок);

Наприклад:

char name[81];gets(name);

Наступний приклад демонструє неправильне використання функції gets:

static char *name;gets(name);

Хоча аргумент функції вказаний правильно, як вказівник на символьний тип, але не зарезервована пам'ять під рядок символів. Розміщення рядка в пам'яті по даній адресі може привести до затирання іншої інформації і, в результаті, до

непередбачених наслідків.

На відміну від функції scanf, функція gets дає можливість вводити пусті символи, наприклад, пропуски. Функція gets повертає вказівник на введений рядок символів. При неправильному зчитуванні даних або при зустрічі у вхідному потоці коду кінця файлу функція gets повертає вказівник NULL, наприклад: ***while(gets(name)!==NULL) {}***

Обробка рядків символів. При обробці рядків символів найчастіше необхідно виконувати операції визначення довжини рядка, копіювання, конкатенації та порівняння рядків. Для роботи з рядками призначені функції, прототипи яких приведені у файлі string.h.

Довжина рядка визначається за допомогою функції strlen(), яка має наступний прототип:

unsigned strlen(char *str);

Функція повертає кількість символів рядка до нуль-символа завершення рядка '\0'. Приклад:

```
char str[20]="Рядок символів";
int k;
k=strlen(str);
```

Змінна k прийме значення, рівне кількості символів рядка str, тобто k=14. При виконанні операції копіювання необхідно правильно визначити рядок-приймач символьних даних. Цей рядок можна визначити як буфер символів у статичній або динамічній пам'яті.

У статичній пам'яті рядок оголошується як масив символів, наприклад; char buf[20]. Динамічна пам'ять під рядок виділяється за допомогою функцій malloc() або calloc() :

```
char *buf;
buf=(char *)malloc(20);
```

В обох випадках необхідно передбачити виділення пам'яті під символ кінця рядка '\0'. Копіювання рядків здійснюється за допомогою бібліотечної функції strcpy(), прототип якої має вигляд:

char *strcpy(char *str1, char *str2);

Ця функція копіює рядок str2 у рядок str1. Обидва рядки задаються своїми вказівниками. Функція повертає вказівник на рядок-приймач str1. Довжина рядка str1 повинна бути достатньою для зберігання рядка str2, включаючи нуль-символ завершення рядка. Наприклад;

```
char str1 [15];
char str2[]='MS Visual Studio";strcpy (str1, str2);
```

Об'єднання двох рядків здійснює функція strcat():

char *strcat(char *str1, char *str2);

Рядок str2 буде дописаний в кінець рядка str1. Довжина рядка str1 повинна бути достатньою для розміщення результату об'єднання, включаючи символ завершення рядка '\0'. Приклад;

```
char str1 [20]="Visual ";
char *str2="C++";
```

strcat(str1,str2);

Результатом роботи функції strcat() буде рядок str1, який прийме значення символьної константи "C++"

Для порівняння рядків використовується функція strcmp(), яка має наступний прототип:

int strcmp(char *str1, char *str2);

Ця функція порівнює рядки str1 і str2 і повертає ціле число менше 0, якщо str1<str2; рівне 0, якщо str1=str2; більше 0, якщо str1>str2. Порівняння рядків здійснюється посимвольно зліва направо до першого неспівпадання кодів символів. При порівнянні необхідно пам'ятати, що в таблиці ASCII коди малих літер є більші від кодів великих літер. Приклад:

```
char str1[]="Turbo C" char str2[]="C++"
if(strcmp(str1,str2))
puts("Перший рядок більший від другого")
```

Інші функції обробки рядків символів приведені у таблиці 4.

Функції в мові C++

Функції - це будівельні блоки мови C++, самостійні одиниці програми, спроектовані для рішення конкретних задач, що звичайно повторюються декілька разів.

Основна форма опису (definition) функції має вигляд:

```
тип <ім'я функції>(список параметрів)
{
тіло функції
}
```

Тип функції визначає тип значення, що повертає функція за допомогою оператора return. Якщо тип не зазначений, то по умовчанню передбачається, що функція повертає ціле значення (типу int). Список параметрів складається з переліку типів і імен параметрів, розділених комами. Функція може не мати параметрів, але круглі дужки необхідні в будь-якому випадку.

У списку параметрів для кожного параметра повинний бути зазначений тип. Приклад правильного списку параметрів:

t(int x, int y, float z)

Приклад неправильного списку параметрів:

f(int x, y, float z)

Приведемо приклад функції, що реалізує зведення числа a в натуральний степінь b:

```
float step(float a, int b)
{
float i;
if(a<0) return (-1); /* основа негативна */
a=1;
for(i=b;i;i--) a*=a;
return a;
}
```

Ця функція повертає значення -1, якщо основа негативна, і a_n - якщо основа

невід'ємна. Оператор *return* має два використання.

По-перше, цей оператор викликає негайний вихід із поточної функції і повернення в програму, що викликає.

По-друге, цей оператор може використовуватися для повернення значення функції.

Відразу слід зазначити, що в тілі функції може бути декілька операторів return, але може і не бути жодного. У цьому випадку повернення в програму, відбувається після виконання останнього оператора тіла функції.

Інший приклад - функція для знаходження найбільшого з двох чисел:

```
max(int a, int b)
{
int  m; if(a>b) m=a;else m=b; return m;
}
```

Можливо також написати цю функцію без використання додаткової змінної:

```
max(int a, int b)
{
if(a>b) return a;else return b;
}
```

Можна ще коротше записати:

```
max(int a, int b)
{
if(a>b) return a;return b;
}
```

А можна і так:

```
max(int a, int b)
{
return (a>b)? a: b;
}
```

Якщо функція повинна повертати значення, але не виконує цього, компілятор видає попередження. Всі функції, що повертають значення, можуть використовуватися у виразах мови С, але вони не можуть використовуватися в лівій частині оператора присвоювання, за винятком тих випадків, що коли повертається значення покажчика.

Використання функцій, що повертають покажчики, має деякі особливості. Покажчики не є ні типом ціле (int), ні типом беззнакове ціле (unsigned int). Їхніми значеннями є адреси пам'яті даних визначеного типу. Відповідно повинна бути описана і функція. Розглянемо приклад опису функції, що повертає покажчик на тип char. Ця функція знаходить у рядку перший пробіл і повертає його адресу.

```
char* find(char* string)
{
int i=0;
while (siring[i] != ' ')&&(string[i] != '\0') i++;
if(s[i]) return &s[i]; /* повертає адреса першого пробілу */
else return NULL; /* повернення нульового покажчика */}
```

Коли функція не повертає ніякого значення, вона повинна бути описана як

функція типу void (порожня).

Ви не зобов'язані описувати функцію типу void, тоді вона по умовчанню буде мати тип int і не повертати ніякого значення. Це викликає попереджуюче повідомлення компілятора, але не буде перешкодою для компіляції. Однак оголошення типу що повертається значення функції є гарним правилом.

Прототипи функцій. Особливістю стандарту ANSI мови С є те, що для створення правильного машинного коду функції йому необхідно повідомити до першого виклику тип що повертається результату, а також кількість і типи аргументів. Для цієї цілі в С використовується поняття прототипу функції. Прототип функції задається в такий спосіб: тип <ім'я функції>(список параметрів);

Використання прототипу функції є оголошенням функції (declaration). Частіше усього прототип функції цілком збігається з заголовком в описі функції, хоча це і не завжди так. При оголошенні функції компілятору важливо знати ім'я функції, кількість і тип параметрів і тип значення що повертається. При цьому імена формальних параметрів функції не грають ніякої ролі й ігноруються компілятором. Тому прототип функції може виглядати або так:

```
int func(int a, float b, char* c);
```

або так:

```
int func(int, float, char*);
```

Два цих оголошення абсолютно рівноправні.

Розглянемо приклад.

```
# include <stdio. h>
float sqr(float a);  /* це прототип функції, оголошення функції */
int main()
 {
  float b;b=5.2;
  printf("Квадрат числа %f дорівнює %f", b, sqr(b));return 0;
 }
float sqr(float a)   /* Це опис функції */
 {
  return a*a;
 }
```

Наступні два приклади використання функцій викликають повідомлення про помилку при компіляції. У першому прикладі перешкодою для компіляції буде невідповідність значення що повертається оголошеному типу функції. Мови С і С++ автоматично перетворять дані до іншого типу, але тільки тоді, коли це можливо. Цілий тип не може бути автоматично перетвореним в покажчик на ціле. Перший приклад:

```
#include <stdio.h>    /* Приклад неправильний */
int *sqr(int *i)                  /* Прототип функції */
main( )
{
int i; sqr(&x);
}
int *sqr(int *i)                  /* Оголошення функції */
{
```

```
return *i=(*i)*(*i);
}
```

Другий приклад:

```
#include <stdio.h> /*Приклад неправильний */
int sqr(int *i) /* Прототип функції */
main()
{
int x=10: sqr(&x, 10); /* Невідповідність кількості аргументів */
}
int sqr(int *i)
{
*i=(*i)*(*i);
}
```

Звертаємо увагу на те, що якщо ми виправимо ці програми, то функція буде повертати квадрат числа і не через значення функції, а через параметр функції.

Якщо функція не має аргументів, то при оголошенні прототипу такої функції слід замість аргументів писати ключове слово void. Це повинно стосуватися і функції main(). Її оголошення повинно мати вигляд void main(void) або main(void).

```
#include <stdio.h>
void line_ (void); /* прототип функції */
main (void)
 {
    line_();
 }
void line_(void)
 {
int i:
for(i=0;i<80;i++) printf("-“);
 }
```

Заголовні файли мови С містять прототипи стандартних функцій, що відносяться до цього заголовного файла. Прикладами таких заголовних файлів є файли stdio.h, string.h, conio.h і ін.

Область дії й область видимості функцій. Область дії (scope rules) перемінної - це правила, що встановлюють, які дані доступні з даного місця програми. В мові С++ кожна функція - це окремий блок програми. Потрапити в тіло функції не можна інакше, як через виклик даної функції. Зокрема, не можна оператором локального переходу goto перейти в середину іншої функції.

З погляду області дії перемінних розрізняють три типи перемінних: глобальні, локальні і формальні параметри. Правила області дії визначають, де кожна з них може застосовуватися.

Локальні перемінні - це перемінні, оголошені усередині блока, зокрема усередині функції. Мова С++ підтримує просте правило: перемінна може бути оголошена усередині будь-якого блока програми. Локальна перемінна доступна усередині блока, у якому вона оголошена. Згадаємо що блок відчиняється фігурною дужкою і закривається фігурною дужкою. Область дії локальної перемінної - блок.

Локальна перемінна існує поки виконується блок, у котрому ця перемінна оголошена. При виході з блока ця перемінна (і її значення) втрачається.

```
#include < stdio.h>
void f(void);
main(void)
 {
  int i;i=1;f( );
  printf("В функції main значення і дорівнює %d\n", i);
 }
void f(void)
 {
  int i; i=10;
 printf("В функції f() значення і дорівнює %d\n", i);
 }
```

Приклад показує, що при виклику функції значення перемінної і, оголошеної в main( ), не змінилося.

Формальні параметри - це змінні, оголошені при описі функцій як її аргументи. Функції можуть мати деяку кількість параметрів, що використовуються при виклику функцій для передачі значень у тіло функції. Формальні параметри можуть використовуватися в тілі функції так само, як локальні перемінні, якими вони по суті діла і є. Область дії формальних параметрів - блок, що є тілом функції.

Глобальні змінні - це змінні, оголошені поза функціями. На відміну від локальних змінних глобальні змінні можуть бути використані в будь-якому місці програми, але перед їхнім першим використанням вони повинні бути оголошені. Область дії глобальної перемінної - уся програма.

Використання глобальних перемінних має свої недоліки:

- ✓ вони займають пам'ять протягом усього часу роботи програми;
- ✓ використання глобальних змінні робить функції менше загальними й утруднює їхнє використання в інших програмах;
- ✓ використання зовнішніх перемінних зумовлює появу помилок через побічні явища. Ці помилки, як правило, важко відшукати.

Перелік стандартних функцій по роботі з символами та рядками наведено в табл. 19-21

Таблиця 19. - Функції для роботи з рядками _#include <string.h>

Функція	*Прототип*	*Дія*
atof	double atof(char *str);	Перетворює рядок str в дійсне число подвійної точності. Перетворення здійснюється до першого недопустимого символа або до символа '\0'. Якщо неможе перетворити, то повертає 0
atoi	int atoi(char*str);	Перетворює рядок str в десяткове ціле. Якщо число перевищує діапазон int, то повертає 2 молодші байти. Якщо не може перетворити, то повертає 0
atol	long atol(char*str);	Перетворює рядок str в довге десяткове ціле

Функція	*Прототип*	*Дія*
ecvt	char *ecvt(double v, intdig, int *dec, int *sign);	Перетворює дійсне v у рядок: dig - кількість цифр числа, що будуть перетворені в рядок, dec - позиція десяткової крапки від початку рядка (якщо dec<=0, то позиція десяткової крапки знаходиться зліва від числа), signє {0,1} - знак числа. Символ '\0' додається. Повертає вказівник на рядок
fcvt	char *fcvt(double v, int dig, int*dec, int*sign);	Те ж саме що й ecvt, тільки dig - кількість цифр після крапки
gcvt	char *gcvt(double v, intdig, char *buf);	Перетворює дійсне v у рядок. На відмінувід ecvt() та fcvt() розміщає рядок у попередньо оголошений буфер buf. dig -це кількість символів рядка. Містить представлення числа з фіксованою або плаваючоюкрапкою в залежності від того, чи може число розміститися в dig позиціях
itoa	char *itoa(int v, char *str, int baz);	Перетворює ціле v в рядок str у системі числення baz (2<=baz<=36). Повертаєвказівник на рядок
ltoa	char *itoa(long v, char *str, int baz);	Перетворює довге ціле v в рядок символів str
strcat	char *strcat(char *sp, char *si);	Приписує рядок si до рядка sp
strch	char *strchr(char *str, char c);	Знаходить в рядку str перше входження символа c.
strcmp	int strcmp(char *str1,char *str2);	Порівнює рядки str1 і str2. Результат: <0, якщо str1<str2; =0, якщо str1=str2; >0,якщо str1>str2
strcmpi	int strcmpi(char *str1.char *str2);	Порівнює рядки str1 і str2 без врахуваннярегістру для буквенних символів. Повертає аналогічне значення що і й strcmp.
strcpy	char *strcpy(char *sp,char *si);	Копіює рядок si в рядок sp
strcpy	char *strcpy(char *sp, char *si);	Копіює рядок si в рядок sp
strcspn	int strcspn(char *str1,char *str2);	Визначає довжину першого сегментурядка str1, що містить символи, які не входять в множину символів рядка str2
strlen	unsigned strlen(char *str);	Обчислює довжину рядка str
striwr	char *strlwr(char *str);	Перетворює букви верхнього регістра в рядку в букви нижнього регістра
strncat	char *strncat(char *sp,char *si, int kol);	Приписує kol символів рядка si до рядкаsp
trncmp	int strncmp(char *str1,char *str2, int kol);	Порівнює kol перших символів рядків str1та str2. Результат аналогічний функції strcmp
strncmpi	int strncmpi(char *str1,char *str2, int kol);	Порівнює kol перших символів рядків str1та str2 без врахування регістру буквенних символів. Результат аналогічний функції strcmp

Функція	*Прототип*	*Дія*
strncpy	char *strncpy(char *sp, char *sp. int kol);	Копіює kol символів рядка si в рядок sp
strpbrk	char *strpbrk(char *str1,char *str2);	Знаходить в рядку str1 перше входження довільного символа із множини символів рядка str2
strrchr	char *strrchr(char *str, char c);	Знаходить в рядку str останнє входження символа с
strset	char*strset(char*str, intch):	Записує символ ch у всі позиції рядка str. Повертає вказівник на str
Strnset	char *strset(char *str, intch, zise t n);	Записує символ ch у перші n позиції рядкаstr. Повертає вказівник на str. Символ '\0' не затирається, якщо n > strien(str)
Strspn	int strspn(char *str1,char *str2);	Знаходить довжину першого сегменту рядка str1, що містить символи із множини символів, що входять в рядок str2
Strstr	char *strstr(char *str1,char *str2);	Повертає вказівник на елемент рядка str1,який є початком підрядка str2, і NULL, якщо str2 не входить в str1
Strupr	char *strupr(char *str);	Перетворює літери нижнього регістра рядка str у верхній
Ultoa	char *ultoa(unsignedlong v, char *ctr, int baz); n+-	Перетворює беззнакове довге ціле v в рядок символів

Таблиця 20. - Функції перевірки та перетворення символів #include <ctype.h>

Функція	*Прототип*	*Дія*
Isalnum	int isalnum(int c);	Дає відмінне від 0 значення, якщо с - буква (A-Z,a-z) або цифра (0-9), і 0 - віншому випадку
Isalpha	int isalpha(int c);	Дає відмінне від 0 значення, якщо с - буква (A-Z,a-z), і 0 - в інших випадках
Isascii	int isascii(int c);	Дає відмінне від 0 значення, якщо код символа с від 0 до 127, і 0 - в інших випадках
Iscntrl	int iscntrl(int c);	Дає відмінне від 0 значення, якщо с - управляючий символ (Ox7F або 0x00-0x1 F), і 0 - в інших випадках
Isdigit	int isdigit(int c);	Дає відмінне від 0 значення, якщо с - цифра (0-9), і 0 - в інших випадках
Isgraph	int isgraph(int c);	Дає відмінне від 0 значення, якщо с -символ, що має графічне позначення (0x21-Ox7E), і 0 - в інших випадках
Slower	int islower(int c);	Дає відмінне від 0 значення, якщо с - символ нижнього регістру, і 0 - в іншихвипадках
ispnnt	int isprint(int c);	Дає відмінне від 0 значення, якщо с - друкований символ (0x20-Ox7E). і 0 - вінших випадках
Ispunct	int ispunct(intc);	Дає відмінне від 0 значення, якщо с - управляючий символ
Toupper	int toupper(int c);	Перетворює букву с до верхнього регістру

Таблиця 21. - Функція для роботи з буферами (масивами символів) #include<mem.h> або #include<string.h>

Функція	*Прототип*	*Дія*
memcpy	void *memcpy (void *dest, void *src, size_t n);	Копіює блок n байт з src в dest. Буферине повинні перекриватися. Повертає вказівник dest.
memccpy	void *memccpy (void *dest, void *src, int c,size_t n);	Копіює блок n байт з src в dest. Буферине повинні перекриватися. Копіювання продовжується поки: 1. не зустрінеться символ c, який теж копіюється в dest. Повертає вказівник на наступний після символа c байт; 2. поки не скопіюється n байт. Повертає вказівник NULL
memmove	void *memmove(void *dest, void *src, size_t n);	Копіює блок n байт з src в dest. Буфери можуть перекриватися. Повертаєвказівник dest.
movmem	void *moymem (void *src, void *dest,unsigned n);	Копіює блок n байт з src в dest. Буфери можуть перекриватися. Повертаєвказівник dest.
movedata	void movedata (unsigned srcseg,unsigned srcoff, unsigned destseg,unsigned destoff, size_t n);	Копіює n байт з srcseg: srcoff в destseh: destoff
memcmp	int memcmp (void *s1, void *s2, size_tn);	Порівнює n перших байтів двох буферів s1 та s2 в лексикографічномупорядку. Повертає значення: <0, якщоs1<s2; ==0, якщо s1==s2; >0, якщо s1>s2
memicmp	int memicmp (void *s1, void *s2, size_tn);	Теж саме що й memcmp, але без урахування регістру символів.
memchr	void *memchr (void *s, int c, size_t n);	Шукає символ c в перших n байтах буфера s. Повертає вказівник на символc. Якщо символ не знайдений, то повертає NULL

Приклади алгоритмів на мові С++ з використанням символів та рядків

Приклад 1. Скласти програму, яка вводить речення, здійснює розбиття його на слова, підраховує кількість символів у кожному слові та виводить відповідну інформацію

Приклад програмної реалізації

```
#include <string.h>
#include <iostream>
#include <conio.h>
using namespace std;
int main ()
{char *tk, *spt=", . !";
char st[] = "Спеціальність 122 Штучний інтелект.";
cout << st<< endl;
```

```
int i=1;
tk = strtok (st, spt);
while (tk != NULL)
{
cout << i << " слово — " << tk << " — містить " << strlen(tk) << " символів" << endl;
tk = strtok(NULL, spt); i++;} }
```

Приклад 2. Скласти програму вилучення підрядка в **n** символів з **k**-ої позиції в рядку.

Приклад програмної реалізації

```
#include <iostream>
#include <string.h>
#include <conio.h>
using namespace std;
//--------------- функція видалення підрядка з рядка
void del (char *sp, int k, int n)
{ int i;
  for (i = k; i<strlen(sp); i++)
  sp[i] = sp[i+n];
  sp[i] = '\0';
}
int main()
{
  char st[50], pst[10];
  cout << "***** Введіть рядок\n";
  cin.getline(st, 50);
  cout << "***** Введіть підрядок\n";
  cin >> pst;
  cout << "Вихідний рядок: - "<< st << endl;
  del (st, strstr (st, pst)-st, strlen (pst));
  cout << "Новий рядок: - "<<st<<endl;
  getch(); }
```

Приклад 3.Скласти програму для підрахунку кількості входжень символа 'a' у зчитаний рядок s.

Приклад програмної реалізації

```
#include <iostream>
using namespace std;
int main()
{ string s;
  int j=0,k=0;
  bool b;
  getline(cin,s);
  do
  { k=s.find("a",k+1);
    b=((0<=k)&&(k<s.size()));
    if (b) j++;   }
  while(b);
  cout<<j;
  return 0; }
```

Приклад 4. Скласти програму для підрахунку кількості слів у введеному з клавіатури рядку. Словом вважати довільну послідовність символів,

відокремлену символами пробілів, яких немає на початку і в кінці рядка двома способами.

Приклад програмної реалізації. Спосіб 1.

```
#include <iostream>
using namespace std;
int main()
{ string s;
  int j=1,k=0;
  bool b;
  getline(cin,s);
  do
  { k=s.find(" ",k+1);
    b=((0<=k)&&(k<s.size()));
    if (b) j++;
  }
  while(b);
  cout<<j;
  return 0; }
```

Приклад програмної реалізації. Спосіб 2.

```
#include <iostream>
#include <vector>
#include <boost/algorithm/string.hpp>
using namespace std;
using namespace boost;
int main()
{ string s;
  vector <string> v;
  getline(cin,s);
  split(v, s, is_any_of(" "));
  cout << v.size() << endl;
  return 0;
}
```

Приклад 5. Вивести на друк лише заголовні латинські літери, що входять до заданого рядка.

Приклад програмної реалізації.

```
#include<iostream>
#include<string.h>
#include<stdio.h>
#define N 255
using namespace std;
int main()
{
 int i;
 char s[N];
 cout << "Enter string: ";
 gets(s);
 for (i = 0; i<=strlen(s) - 1; i++)
 if (s[i] >= 'A' && s[i] <= 'Z')
 cout << s[i];
 cout << '\n'; }
```

Приклад 6. У заданому рядку всі символи '0' замінити на '1', а символи '1' – на '0' відповідно.

Приклад програмної реалізації.

```
#include<iostream>
#include<string.h>
#include<stdio.h>
#define N 255
using namespace std;
int main()
{
 int i;
 char s[N];
 cout << "Введіть рядок: ";
 gets(s);
 for (i = 0; i <= strlen(s) - 1; i++){
 if (s[i] == '0')
 s[i] = '1';
 else if (s[i] == '1')
 s[i] = '0';
 }
 cout <<"Змінений рядок: " << s << '\n'; }
```

Приклад 7. Створити програму перетворення рядка, замінивши в ньому кожну крапку трьома крапками.

Приклад програмної реалізації.

```
#include<iostream>
#include<string.h>
#include<stdio.h>
#define N 255
using namespace std;
int main()
{
 int i;
 char s[N], s1[N];
 char a[2];
 strcpy(s1, "");
 a[1] = '\0';
 cout << "Введіть рядок: ";
 gets(s);
 for (i = 0; i <= strlen(s) - 1; i++){
 a[0] = s[i];
 strcat(s1, a);
 if (s[i] == '.')
 strcat(s1, "..");  }
 cout << s1 << '\n'; }
```

Приклад 8. Ввести до пам’яті комп’ютера список прізвищ, які розташовані в будь-якому порядку, та відсортувати їх за алфавітом двома способами.

Приклад програмної реалізації. Спосіб 1.

```
#include <iostream>
#include <string.h>
#include <conio.h>
```

```
using namespace std;
int main()
{
  const int n=5;
  char sp [n] [15], r [15];
  int i, k;
//------------------------------ введення прізвищ та ініціалів
cout<< "***** Введіть " << n << " прізвищ \n";
  for (i = 0; i < n; i++)
 { cout<<"Введіть "<<(i+1)<<" прізвище та ініціали\n";
     cin.getline (sp[i], sizeof(sp[i]) - 1);
   }
//------------------------------ сортування списку прізвищ
  for (k = 1; k < n; k ++)
    for (i = 0; i < n-k; i++)
    if (strcmp (sp[i], sp[i+1])>0)
    {strcpy(r, sp[i]);
      strcpy (sp[i], sp[i+1]);
      strcpy (sp[i+1], r);}
  cout<<"\n Відсортований масив прізвищ \n";
  for (i = 0; i < n; i++)
  cout << sp [i] << endl;
  getch(); }
```

Приклад програмної реалізації. <u>Спосіб 2.</u>

```
#include <iostream>
#include <string.h>
#include <conio.h>
using namespace std;
int main()
{
 /*сортування списку прізвищ в алфавітному порядку з використанням покажчиків */
  const int n=5;
  char sp [n] [15];
  int i, k;
  char *ps[n], *ptr; //ps[n] - масив покажчиків
// Введення прізвищ та ініціалізація масиву покажчиків
 cout << "***** Введіть прізвища \n";
  for (i = 0; i < n; i++)
  {
    gets(sp[i]);
    ps[i] = sp[i];
  }
//--------------------------- виведення вихідної інформації
  cout << "\n***** Вихідний список\n";
  for (i = 0; i < n; i++)
  puts (ps [i]);
//---------------------------------- сортування масиву
  for (k = 1; k < n; k ++)
    for (i = 0; i<n-k; i++)
    if (strcmp (ps[i], ps[i+1]) > 0)
    {ptr = ps[i];
      ps[i] = ps[i+1];
```

```
      ps[i+1] = ptr; }
//---------------------- виведення відсортованого масиву
  cout << "\n\n*****Відсортований список \n";
  for (i = 0; i < n; i++)
  puts (ps [i]);
  getch(); }
```

Приклад 9. Ввести рядок і видалити в ньому зайві пропуски.

Приклад програмної реалізації.

```
#include <iostream>
#include <string.h>
#include <conio.h>
using namespace std;
int main()
{
 char st[] = "Штучний    інтелект -   технологія    майбутнього";
  int i, n = 0; //n - для підрахунку кількості пропусків
 for (i = 0; i< strlen(st); i++)
  { if (st[i] != ' ')
    {cout<<st[i];
      n=0; }
    else n++;
    if (n == 1)
    cout << st[i];}
  getch(); }
```

Приклад 10. Із введеного списку прізвищ (без ініціалів) вилучити такі, що починаються на задану літеру і мають задане закінчення, та вивести повідомлення про прізвище з найменшою кількістю літер.

Приклад програмної реалізації.

```
#include <iostream>
#include <string.h>
#include <conio.h>
using namespace std;
int main()
{
  const int n = 6;
  char spis [n] [15], pok[5], p;
  int i, minfam, k = 0;
//----------------------------- введення списку прізвищ без ініціалів
cout << "***** Введіть "<<n<<" прізвищ\n";
 for (i = 0; i < n; i++)
  cin >> spis[i];
  cout << "***** Введіть першу букву\n";
  cin >> p;
  cout << "**** Введіть закінчення\n";
  cin >> pok;
// Визначення прізвища з заданою літерою і заданим закінченням
 cout << "***** Шукані прізвища\n";
  for (i = 0; i < n; i++)
  if (spis [i][0]==p && strcmp(strrchr(spis[i], pok[0]), pok)==0)
  cout <<spis [i] <<endl;
```

```
//-------------- пошук прізвища з найменшою кількістю літер
  minfam-strlen (spis [0]);
  for (i = 1; i < n; i++)
  if(strlen(spis[i])<minfam)
  { minfam = strlen (spis[i]);
    k=i; }
  cout<<"Прізвище з найменшою кількістю літер - "<<spis[k]<<" \n";
  cout << "Довжина шуканого прізвища= " << strlen(spis[k]) << " символів\n";
  getch(); }
```

Використання функцій в Python

Основна перевага використання функцій – це можливість повторного застосування програмного коду, тобто, їх можна викликати багато разів не тільки в тій програмі, де її було визначено, але, можливо, і в інших програмах, іншими користувачами та для інших цілей.

Функція – засіб, якийдозволяє групувати набори інструкцій які в програмі можуть використовуватись багаторазово. ***Функції*** – це:

1) програмні структури, які забезпечують багаторазове використання програмного коду і зменшують його надлишковість;

2) засіб проектування, який дозволяє розбити складну систему на прості і легко керовані частини;

3) засіб структурування програми;

4) можуть обчислювати деякий результат і дозволяють вказувати вхідні параметри, які різняться за своїми значеннями від виклику до виклику;

5) забезпечують можливість розбити складну систему на частини, кожна з яких грає визначену роль.

В табл. 22 наводяться основні інструменти, які мають відношення до функцій. В процесі розробки функції дозволяють мінімізувати надлишковість програмного коду.

Таблиця 22. - Інструкції та вирази по роботі з функціями

Інструкція	*Приклад*
Виклик	myfunc(‘spam‘, ‘eggs‘, meat=ham)
def, return	def adder(a, b=1, *c): return a+b+c[0]
global	def changer():
	global x; x = ‘new‘
nonlocal	def changer(): nonlocal x; x = ‘new‘
yield	def squares(x): for i in range(x): yield i ** 2
lambda	funcs = [lambda x: x**2, lambda x: x*3]

Інструкція ***def*** створює об‘єкт функції та зв‘язує його зім‘ям. У загальному вигляді інструкція має такий формат:

def <name>(arg1, arg2,... , argN):
<statements>

Інструкція ***def*** складається з рядка заголовка і наступного за ним блоку інструкцій, зазвичай з відступами (або простої інструкції слідом за двокрапкою).

Блок інструкцій утворює тіло функції, тобто програмний код, який виконується інтерпретатором щоразу, коли здійснюється виклик функції. У рядку заголовка інструкції ***def*** визначаються ім'я функції, з яким буде пов'язаний об'єкт функції, і список з нуля або більше аргументів (іноді їх називають параметрами) в круглих дужках. Імена аргументів в рядку заголовка будуть пов'язані з об'єктами, переданими в функцію, в точці виклику.

Тіло функції часто містить інструкцію ***return***:

```
def <name>(arg1, arg2,... argN):
    ...
    return <value>
```

Інструкцію *return* можна розташувати в будь-якому місці в тілі функції – вона завершує роботу функції і передає результат програмі, яка її викликає. Інструкція *return* містить об'єктний вираз, який надає результат функції. Інструкція *return* є необов'язковою – якщо вона відсутня, робота функції завершується, коли потік управління досягає кінця тіла функції. Функція без інструкції *return* повертає об'єкт None, проте цезначення зазвичай ігнорується.

Визначення функції відбувається під час виконання, тому віменах функцій немає нічого особливого. Важливим є тількиоб‘єкт, на який посилається ім‘я:

```
def func():
pass

othername = func  # Зв‘язування об‘єкта функції з ім‘ям
othername()  # Виклик функції
```

В цьому фрагменті функція зв‘язана з іншим ім‘ям і викликана з використанням нового імені. Функції – це звичайні об‘єкти; їх явно записують в пам‘ять під час виконання програми. Крім підтримки можливості виклику, функції дозволяють приєднати будь-які ***атрибути***, які можуть зберігати інформацію для наступного використання:

```
value = 'some value'
def func(): ... # Створюють об‘єкт функції
func() # Виклик об‘єкту
func.attr = value # Приєднання атрибуту до об‘єкту
print(func._dict_)
```

Параметри за замовчуванням. Згадаємо, наприклад, функцію **range()**. Її можна викликати в трьох різних формах:

- з одним параметром;
- із двома параметрами;
- з трьома параметрами.

Для організації такої поведінки своєї функції можна описати після звичайних (позиційних) ключові параметри зі значеннями за замовчуванням:

```
from pip._vendor.distlib.compat import raw_input
def ask_ok(prompt, retries=4, complaint='Yes/no, please!'):
"""функція ask_ok із параметрами зазамовчуванням"""
while 1:
ok = raw_input(prompt) #Введення значення
if ok in ('m', 'так', 'yes'): return 1
```

```
if ok in ('н', 'ні', 'no', 'nope'): return 0
retries = retries - 1
if retries < 0: raise IOError('Помилка')
print(complaint)
```

Виклик функції **ask_ok ("*Прошу ввести*:", 2)** встановить перший параметр – рядок, як запрошення, а другий параметр змінить кількість неправильних спроб із чотирьох на дві. Викликаючи функцію, ключові параметри можна ставити після позиційних параметрів у довільній послідовності, якщо явно вказано імена ключових параметрів.

ask_ok('Старт!', complaint='Так/ні англійською, будь ласка', retries=2)

Розглянемо наступний приклад функції:

```
i = 5
def f(arg=i):
print(arg)
i = 6
f()  # виведе не 6, а 5; f(10) виведе 10
```

Механізм параметрів за замовчуванням діє так: якщо змінну проініціалізовано до виклику функції, то у функцію передається саме це значення, в іншому випадку у функцію передається значення за замовчуванням.

Зауваження. Тіло функції не виконується при її визначенні, а тільки компілюється. І навпаки, значення за замовчуванням обчислюються при визначенні функції та зберігаються в об'єкті-функції.

Тобто, значення за замовчуванням установлюється лише один раз. Це відіграє роль при встановленні значення за замовчуванням спискам, наприклад:

```
def f(a, L=[]):
    L.append(a)
    return L

print(f(1))
print(f(2))
print(f(3))
```

В результаті виконання програми буде виведено елементи, які накопичуються в списку:

[1]
[1, 2]
[1, 2, 3]

Для передачі параметрів за замовчуванням без накопичення необхідно використовувати наступну форму:

```
def f(a, L=None): # None – порожній об'єкт, не вказанезначення
 if L is None:  # якщо параметр L не вказано
 L = []
 L.append(a)
 return L

print(f(1))
print(f(2))
```

```
print(f(3))
```

В результаті виконання програми отримаємо виведення даних:

[1]
[2]
[3]

Документування функцій. Вдалим стилем є документування кожної функції. Для цього в наступному рядку відразу після заголовка необхідно помістити короткий опис функції, укладений у потрійні ''' апострофи або """ лапки. Вміст всередині потрійних лапок виводиться наступною інструкцією:

print (ім'я_функції._doc_)

Такий спосіб дозволяє легко зрозуміти призначення функції, якщо прочитати початковий текст або скористатись спеціальним сервером документації ***Python***.

Передавання у функцію змінної кількості аргументів. Часто використовуваним прийомом у програмуванні є передавання у функцію змінного числа аргументів. Для цього в *Python* можна скористатись символом * перед списком аргументів змінної довжини.

Попереду списку аргументів може бути (не обов'язково) один або декілька обов'язкових аргументів:

```
def fprintf(message, *args):
    print(message, '{}'.format(args))

fprintf('Аргументи', 1)
fprintf('Аргументи', 1, 'sun')
fprintf('Аргументи', 1, [1, 2, 3])
```

Функції lambda. Lambda-вирази. В Python існує можливість створювати об'єкти функцій у формі виразів. Через схожість з аналогічною можливістю в мові LISP вона отримала назву ***lambda***. Ця назва походить з мови програмування LISP, в якій ця назва була запозичена з лямбда–числення.

Однак в Python це – ключове слово, яке вводить вираз синтаксично. Подібно інструкції ***def*** цей вираз створює функцію, яку викликають пізніше (але на відміну від інструкції ***def***, вираз повертає функцію, а не зв'язує її з ім'ям).

Саме тому ***lambda***-вирази іноді називають ***анонімними*** (***безіменними***) функціями. Їх часто використовують, як спосіб отримання вбудованої функції або відкласти виконання фрагмента програмного коду.

В загальному вигляді ***lambda***-вираз складається з ключового слова ***lambda***, за яким слідують один або більше аргументів (як список аргументів у круглих дужках у заголовку інструкції *def*) і далі, слідом за двокрапкою, розташовано вираз:

lambda *argument1, argument2,... argumentN*:
вираз, який використовує аргументи

Результатом *lambda*–виразу є такі ж об'єкти функцій, які утворюють інструкції *def*, але тут є кілька відмінностей:

1) *lambda* – це вираз, а не інструкція;
2) тіло *lambda* – це не блок інструкцій, а один вираз.

Наприклад, розглянемо функцію:

```
def func(x, y, z):
    return x + y + z
func(2, 3, 4)   # 9
```

Можна описати за допомогою *lambda*–виразу, присвоївши результат імені, яке пізніше буде використано для виклику функції:

```
f = lambda x, y, z: x + y + z
f(2, 3, 4) # 9
```

Тут імені *f* присвоєно об'єкт функції, створений *lambda*-виразом, – інструкція *def* працює так само, але присвоювання виконують автоматично.

Lambda–вирази використовують для створення не великих функцій і дозволяють вбудовувати визначення функцій в код, який їх використовує. Вони не є предметом першої необхідності (можна замість них використовувати інструкції *def*).

Lambda–вирази дозволяють спростити сценарії, де потрібно впроваджувати невеликі фрагменти програмного коду. Вони корисні в якості скороченого варіанту інструкції *def*, коли необхідно вставити маленькі фрагменти виконуваного коду туди, де використання інструкцій неприпустимо.

Наступний фрагмент коду створює список з трьох функцій, *lambda*–вирази вставлені в літерал списку. Інструкція *def* не може бути вставлена в літерал, тому що це – інструкція, а не вираз.

```
L = [lambda x: x ** 2,   # Вбудовані визначення функцій
lambda x: x ** 3,
lambda x: x ** 4]   # Список з трьох функцій
for f in L:
print(f(2))   # Виведе 4, 8, 16
print(L[0](3))   # Виведе 9
```

Для реалізації еквівалентної таблиці переходів із застосуванням інструкцій *def* необхідно створити іменовані функції поза контекстом їх використання:

```
# визначення іменованих функцій
def f1(x): return x ** 2
def f2(x): return x ** 3
def f3(x): return x ** 4
L = [f1, f2, f3]   # Посилання по імені
for f in L:
    print(f(2))   # Виведе 4, 8, 16
    print(L[0](3))   # Виведе 9
```

Приклад 11. На основі введеного значення n створити функцію Fn(), яка обчислює тригонометричний вираз за наступним правилом:

- sin(x)*cos(x), якщо n==1;
- $\sin(x^2)+\cos(x)$, якщо n==2;
- 1-sin(x), в противному випадку.

Для розрахунку кожного виразу створити в програмі відповідні функції.

Текст програмного коду створеного алгоритму:

```
# Імпортувати математичну бібліотеку
```

```
import math
# Ввести значення n
n = int(input('n = '))
x = float(input('x = '))
# Визначити код функції Fn()
if (n==1):
   def Fn(x):
      print("n=1. res = sin(x)*cos(x).")
      return math.sin(x)*math.cos(x)
else:
   if (n==2):
      def Fn(x):
         print("n=2. res = sin(x*x)+cos(x)")
         return math.sin(x*x)+math.cos(x)
   else:
      def Fn(x):
         print("n=3. res = 1-sin(x)")
         return 1-math.sin(x)
# Виклик функції Fn()
res = Fn(x)
print("res = ", res)
```

Виконання створеного алгоритму розрахунку заданої функції наведено на рис. 71

```
n = 3
x = 1.2
n=3. res = 1-sin(x)
res =  0.06796091403277371
```

Рисунок 71. - Результати роботи алгоритму поставленої задачі.

Приклад 12. Описати функцію, яка обчислює множину значень $y = \sqrt[n]{x}$, (x>0, n>0), з використанням рекурентної формули Ньютона:

$$y_{k+1} = \frac{k-1}{k} y_k + \frac{x}{k \cdot y_k^{k-1}}, \qquad k = 0, 1, \dots, y_0 = \frac{x+n-1}{2}$$

Необхідну точність оцінюють співвідношенням $|y_n+1 - y_n| \le \varepsilon$, прийняти значення для y0>0.

Текст програмного коду створеного алгоритму:

```
import itertools
import math

def my_sqrt(a,n,Epsilon):
  term1=(a+n-1)/2
  term2=(2/3)*term1 +(a/(3*term1**2))
  k=1
  while (abs(term2- term1) > Epsilon):
    term1= term2;
    term2=((k-1)/k)* term1 + a/(k*(term1**(k-1)))
    k+=1
    return term2
```

```
print(" x ", ": "," y ", " : "," yt", " : "," error " )
print(" ")
Epsilon=0.0001
a=0.1;b=1
for i in itertools.count(start=a, step=0.2):
  if i>b: break
  n=3
  y= my_sqrt(i,n,Epsilon)
  y1=i**(1/n) # точне значення функції
  error=abs(y-y1)
  print('%.2f' %i, ":", '%.4f' %y, " : ", '%.4f' %y1, " : ", '%.4f' %error)
```

Виконання створеного алгоритму розрахунку заданої функції наведено на рис. 72

```
   x  :   y   :   yt  :   error

0.10 : 0.1000  :  0.4642  :  0.3642
0.30 : 0.3000  :  0.6694  :  0.3694
0.50 : 0.5000  :  0.7937  :  0.2937
0.70 : 0.7000  :  0.8879  :  0.1879
0.90 : 0.9000  :  0.9655  :  0.0655
```

Рисунок 72. - Результати роботи алгоритму поставленої задачі.

Контрольні запитання

1. Яким чином оголосити змінну символьного типу?
2. Який специфікатор формату використовується для введення та виведення символів?
3. Що таке масиви символів? Яким чином їх оголосити?
4. Як звернутися до певного елемента у масиві символів?
5. Що таке рядок символів? Яким чином його оголосити?
6. Що таке масив символьних рядків? Як його оголосити?
7. Які ви знаєте функції без форматного вводу-виводу рядків .символів?

Практичне завдання 7.
Розробка програм мовою С++ з використанням структур

Мета: набуття практичних навичок у використанні структур, організації структурних даних та їх реалізації в середовищі програмування ***Microsoft Visual C++.***

Завдання до виконання. Мовою програмування С++ створити структуру відповідно до наданого варіанту (***табл. 23***); а також структуру з використанням вказівників; структуру, що містить функції обробки полів.

Таблиця 23. – Варіанти завдань до організації структур даних

№ вар.	Індивідуальне завдання
1	***Описати структуру з ім'ям STUDENT, яка містить наступні поля:*** ➢ Name – Прізвище та ініціали; ➢ Year – рік народження: ➢ Bal – оцінки з 4 предметів (масив з 4 елементів) ***Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних Group, що складається з N змінних типу STUDENT; ➢ Впорядковує записи за зростанням поля Year ➢ Виводить на екран прізвища і рік народження студентів середній бал яких > 4.0;
2	***1. Описати структуру з ім'ям SKLAD, яка містить наступні поля:*** ➢ Name – Назва товару; ➢ Type – одиниця вимірювання: ➢ Quantity – кількість одиниць товару; ➢ Cost – ціна одиниці товару. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних SHOP, що складається з N змінних типу SKLAD; ➢ Впорядковує записи по спаданню поля Name; ➢ Виводить на екран ціну та кількість товару, назва якого вводиться з клавіатури або виводить повідомлення про його відсутність.
3	***1. Описати структуру з ім'ям TRAIN, яка містить наступні поля:*** ➢ Nazv – Назва ➢ Numer – номер поїзда; ➢ Date – дата відправлення (структура: day; month, year - день, місяць, рік); ➢ Time – ціна одиниці товару. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних AVTOPARK що складається з N змінних типу TRAIN; ➢ Впорядковує записи по спаданню поля Numer; ➢ Виводить на екран рейси, яких час відправлення буде після 15-00 за заданою датою.
4	***1. Описати структуру з ім'ям ABONENT, яка містить наступні поля:*** ➢ Name – прізвище абонента; ➢ Init – ініціали абонента;

№ вар.	*Індивідуальне завдання*
	➢ Nomer – номер телефону; ➢ Adress –домашня адреса. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних TELEFON, що складається з N змінних типу ABONENT; ➢ Впорядковує записи по зростанню поля Name; ➢ Виводить на екран прізвище, ініціали та домашню адресу за введеним номером телефону, або виводить повідомлення про його відсутність.
5	***1. Описати структуру з ім'ям AEROFLOT, яка містить наступні поля:*** ➢ Nazv –назва пункту призначення; ➢ Numer – номер рейсу; ➢ Type –тип літака; ➢ Time –час відправлення. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних ROZKLAD, що складається з N змінних типу AEROFLOT; ➢ Впорядковує записи по зростанню поля Type; ➢ Виводить на екран всі номери рейсів, та час відправлення літаків які відправляються в введений з клавіатури пункт призначення або повідомляє про відсутність таких рейсів.
6	***1. Описати структуру з ім'ям DETAL, яка містить наступні поля:*** ➢ Name – назва деталі; ➢ Sort – сорт виробу; ➢ Date –дата виготовлення (структура: day; month, year - день, місяць, рік); ➢ Quant – кількість; ➢ Cost - ціна деталі. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних ZAKAZ, що складається з N змінних типу DETAL; ➢ Впорядковує записи по спаданню поля Name; ➢ Виводить на екран всі деталі I сорту які виготовлені пізніше заданої дати, яка введена з клавіатури.
7	***1. Описати структуру з ім'ям BOOK, яка містить наступні поля:*** ➢ Name – назва книги; ➢ Avtor – автори книги; ➢ Data –дата друку (структура: month, year - місяць, рік); ➢ Cost - ціна книги. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних SHOP, що складається з N змінних типу BOOK; ➢ Впорядковує записи по зростанню поля Avtor; ➢ Виводить на екран всі книги які були надруковані в заданому році.
8	***1. Описати структуру з ім'ям TOVAR, яка містить наступні поля:*** ➢ Name – назва товару; ➢ Cost_Z – ціна закупки товару; ➢ Cost_P - ціна продажу товари.

№ вар.	*Індивідуальне завдання*
	➢ Quantity –кількість одиниць товару; ➢ Pributok – прибуток. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних SHOP, що складається з N змінних типу TOVAR і обчислює прибуток по кожному товару; ➢ Впорядковує записи по зростанню поля Pributok; ➢ Виводить на екран всі товари, які мають найбільший прибуток.
9	***1. Описати структуру з ім'ям VISTAVA, яка містить наступні поля:*** ➢ Nazva – назва вистави; ➢ Date – дата вистави (структура: day; month, year - день, місяць, рік); ➢ Cost - ціна квитка. ➢ Day_week – день неділі; ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних THEATRE , що складається з N змінних типу VISTAVA; ➢ Впорядковує записи по зростанню поля Day_week; ➢ Виводить на екран всі вистави і дати їх проходження, які відбудуться в заданий день тижня.
10	***1. Описати структуру з ім'ям WORKER, яка містить наступні поля:*** ➢ Name – Ім'я працівника; ➢ Surname – Прізвище працівника ; ➢ Date – дата народження (структура: day; month, year - день, місяць, рік); ➢ Pos - посада працівника. ➢ Zarplata – заробітна плата працівника ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних OFFICE, що складається з N змінних типу WORKER; ➢ Впорядковує записи по віку працівників по спаданню; ➢ Виводить на екран всіх працівників старших 30 років, які мають заробітну плату меншу введеної з клавіатури.
11	***1. Описати структуру з ім'ям ZARPLATA, яка містить наступні поля:*** ➢ Name – прізвище ім'я та по батькові працівника; ➢ Date – дата народження (структура: day; month, year - день, місяць, рік); ➢ Work_day – кількість відпрацьованих днів; ➢ Stavka – ставка з урахуванням на 24 робочих дні; ➢ Naraxovano – нараховано заробітної плати з урахуванням ставки і кількості відпрацьованих днів; ➢ Do_viplati – сума виплати заробітної плати працівнику з вирахуванням 20% податку. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних BUHGALTER, що складається з N змінних типу ZARPLATA і підраховує поля нараховано та до виплати; ➢ Впорядковує записи по зростанню поля Name; ➢ Виводить на екран всіх працівників з їхньою заробітною платою, які відпрацювали менше 15 робочих днів.

№ вар.	*Індивідуальне завдання*
12	***1. Описати структуру з ім‘ям INSTRUMENT, яка містить наступні поля:*** ➢ Name – назва музикального інструменту; ➢ Type – тип музикального інструменту; ➢ Year – рік виготовлення інструменту; ➢ Vlasnik – власник інструменту; ➢ Vartist – вартість музикального інструменту; ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних ORKESTR, що складається з N змінних типу INSTRUMENT; ➢ Впорядковує записи по зростанню поля Vartist; ➢ Виводить на екран всі інструменти, вартість яких більша за 1000 грн.
13	***1. Описати структуру з ім‘ям COMPUTER, яка містить наступні поля:*** ➢ Procesor – процесор комп‘ютера; ➢ Ram – обсяг оперативної пам‘яті; ➢ HDD – структура що містить поля (Namt- виробник, V_Ram - обсяг, V- швидкість обертання диску); ➢ Monitor – ставка з урахуванням на 24 робочих дні; ➢ Keyboard – нараховано заробітної плати з урахуванням ставки і кількості відпрацьованих днів; ➢ Mouse – сума виплати заробітної плати працівнику з вирахуванням 20% податку. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних CLASS, що складається з N змінних типу COMPUTER; ➢ Впорядковує записи по зростанню поля V_ram; ➢ Виводить на екран всіх комп‘ютери за введеним процесором.
14	***1. Описати структуру з ім‘ям CUPURA, яка містить наступні поля:*** ➢ Name – назва валюти; ➢ Nominal – номінал; ➢ Year – рік випуску куп‘юри; ➢ Kurs – курс валюти відносно української гривні; ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних BANK, що складається з N змінних типу CUPURA; ➢ Впорядковує записи по зростанню поля Name; ➢ Виводить на екран всіх номінали вказаного року.

Зміст звіту

1. Титульна сторінка, тема та мета практичного заняття.
2. Завдання відповідно до номеру варіанту
3. Опис виконаних завдань, лістинги реалізованих програм, результати виконання програм.
4. Висновки, які відображають результати виконаних завдань та їх аналіз.

Теоретичні відомості

Теоретичні основи об'єктно-орієнтованого програмування мовою С++

Організація структур в С++. Структура – це спеціальний тип даних створений програмістом, щоскладається з декількох відомих типів даних, що називаються полями.

```
struct <назва структури>
{
<тип> <назва поля>;
...
<тип> <назва поля>;
}<назва змінних і вказівників>;
```

Опис структури закінчується символом «;», наприклад:

```
struct Student
{
char Name[20]; char SurName[20];int Year;
};      // крапка з комою обов'язкова.
```

Розмір структури дорівнює сумі розмірів її полів. В даному прикладі 20+20+4=24 байти. Для визначення обсягу структури можна використати функцію ***sizeof(****<назва типу структури або змінна>****).***

Оголошення змінної або вказівника структурованого типу даних аналогічне до оголошення звичайної змінної.

```
Student a, gr, *t1;
```

Оголосити змінні і структуру можна так:

```
struct Student
{
char Name[20]; char SurName[20];int Year;
} a,b,*t; // оголошення змінних
```

В мові С++ можна оголосити змінні не присвоюючи імені структурі:

```
struct                  //відсутня назва
{
char Name[20]; char SurName[20];int Year;
} a,b,*t;
```

Структура може містити поля типу структура:

```
struct Student
{
char Name[20]; char SurName[20];struct
{
int year,month, day;
}birthday;
} a,b,*t;
```

або

```
struct BIRTHDAY
{
int year,month, day;
};
```

```
struct Student
{
char Name[20]; char SurName[20];
BIRTHDAY birthday;
} a,b,*t;
```

Доступ до поля змінної типу структури відбувається за допомогою викликаної через крапку властивості: ***Strcpy(a.Name, "Ivan");***

Доступ до поля вказівника типу структури відбувається задопомогою «->»: ***t->birthday.year=1991;***

Бітові поля в C++. Під час оголошення структури можна задавати обсяг пам'яті, який займають змінні певного поля. Для цього після назви поля ставиться знак двокрапка та зазначається ціле число, або константа – кількість бітів. Такі поля називаються бітовими.

Приклад демонструє використання бітових полів.

```
#include <iostream>
using namespace std;

struct aa
{
short a:3; short b:6;
};

int main()
{
cout<<sizeof(aa); aa x;
x.a=0;
x.b=0;
cout<<endl<<x.a<<' '<<x.b<<endl;
   for(int i=1;i<=10;i++)
   {x.a++;
    x.b++;
   cout<<i<<' '<<x.a<<' '<<x.b<<endl;
   }
return 0; }
```

Об'єднання — це місце в пам'яті, яке використовується для зберігання змінних, різних типів. Об'єднання дає можливість інтерпретувати один і той же набір бітів не менше, чим двома різними способами. Оголошення об'єднання (розпочинається з ключового слова union) схоже на оголошення структури і в загальному вигляді має наступну форму:

```
union тег {
тип ім'я-члена;
тип ім'я-члена;
тип ім'я-члена;
…
} змінні-цього-об'єднання;
```

Наприклад:

```
union u_type {
 int i;
 char ch;
 };
```

Об'єднання часто використовуються тоді, коли треба виконати специфічне перетворення типів, тому що дані, що зберігаються в об'єднаннях, можна означати абсолютно різними способами. Наприклад, використовуючи об'єднання, можна маніпулювати байтами, що становлять значення типу double, і робити так, щоб міняти його точність або виконувати яке-небудь незвичайне округлення.

Бітові поля.

Мова С++ має можливість працювати з окремими бітами. Це дає можливість. Ця можливість корисна, для збереження інформації, якщо обмежена пам‘ять, деякі пристрої передають інформацію закодувавши її в один байт.

Метод використання бітових полів для доступу до бітів оснований на структурах. Бітове поле – це особий тип структури, що визначає яку довжину має кожен елемент.

Стандартний вигляд оголошення бітових полів є наступним:

```
struct ім’я структури
{
тип ім’я1: довжина;
тип ім’я2: довжина;
...
тип ім’яN: довжина;
}
```

Приклади реалізації структур мовою С++

Структури являють собою невеликі об'єкти з даними. Структури служать для опису невеликого об'єкта, який не має великої кількості змінних та функцій. Використання структур зручне, оскільки вони мають низку переваг:

- займають менше пам'яті, тому їх використання полегшує навантаження на процесор;
- мають простий синтаксис;
- мають лише змінні, функції та конструктори;
- на їх основі можна створювати об'єкти.

Виходить, що завдяки структурам можна легко створювати невеликі об'єкти з невеликою кількістю інформації. Такі об'єкти можуть описувати реальні об'єкти: об'єкт книги, об'єкт автомобіля, об'єкт дерева, тощо.

Для створення структури використовується синтаксис: ***struct NAME {}***. На основі однієї структури можна створювати необмежену кількість об'єктів, які мають однакові назви змінних та функцій, але при цьому значення у всіх об'єктів будуть різними.

Приклад 1. Реалізувати структуру, яка описує модель автомобіля

Програмний код реалізації структури

```
#include <iostream>
using namespace std;

struct Date {
  int year;
};

struct Auto {
  int wheels;
  float speed;
  char color;
  Date sozdanie;
};

int main()
{
  Auto shkoda;
  shkoda.color = 'r';
  shkoda.speed = 315.23;
  shkoda.wheels = 4;
  shkoda.sozdanie.year = 1999;
   Auto audi = {4, 300.23, 'b'};
  cout << "Audi speed = " << audi.speed << endl;
  cout << "Shkoda year " << shkoda.sozdanie.year << endl;
  return 0; }
```

Приклад 2. Створити структуру відповідного дерева, до якого необхідно додати наступні поля:

- висота;
- назва;
- вік;
- ширина.

На основі створеної структури створити два об'єкти та додати до них відповідні характеристики дерева. Вивести дані на екран.

Програмний код реалізації

```
#include <iostream>
#include <string>
using namespace std;
struct Tree { // Створення структури та перерахунок змінних
int height;
string name;
int years;
float width;
};

int main() {
Tree elka; // Створення об’єкта та ініціалізація всіх значень
elka.height = 180;
elka.name = "Elka";
elka.years = 9;
```

```
elka.width = 58.1;
Tree dub = {540, "Dub", 20, 182.6};
cout << "First object is " << elka.name << endl;
cout << "Second object is " << dub.name << endl;
cin.get();
return 0; }
```

Приклад 3. Створити структуру, яка визначає характеристики комп'ютера такі як:

- процесор;
- вага;
- час автономної роботи;
- кількість ядер;
- встановлена ОС.

На основі структури створити об'єкт і надати йому відповідні значення.

Програмний код реалізації

```
#include <iostream>
#include <string>
using namespace std;

struct Computer {
string proccesor;
float weight;
short int workingTime;
short int yadra;
string os;
};

int main() {
// Створення об’єкта
Computer myComp;
myComp.proccesor = "Core i7";
myComp.weight = 2.3;
myComp.workingTime = 7;
myComp.yadra = 4;
myComp.os = "Windows 10";
cout << myComp.proccesor;
cin.get();
return 0; }
```

Приклад 4. На веб-сайті відстежується, скільки грошей буде зароблено за день від розміщеної на ньому реклами. Необхідно оголосити структуру Advertising, яка відслідковує:

- скільки оголошень показано відвідувачам;
- скільки відсотків відвідувачів переглянуло оголошення;
- скільки в середньому зароблено за кожен клік оголошення.

Значення цих трьох полів задає користувач. Необхідно передати створену структуру Advertising у функцію, яка виведе кожне із наведених значень та підрахує суму зароблених грошей за день (перемножено 3 поля).

Програмний код реалізації

```
#include <iostream>
 // оголошуємо структуру Advertising
struct Advertising
{
   int adsShown;
   double clickThroughRatePercentage;
   double averageEarningsPerClick;
};

void printAdvertising(Advertising ad)
{
   using namespace std;
   cout << "Number of ads shown: " << ad.adsShown << endl;
   cout << "Click through rate: " << ad.clickThroughRatePercentage << endl;
   cout << "Average earnings per click: $" << ad.averageEarningsPerClick << endl;
    // ділимо ad.clickThroughRatePercentage на 100, тому що користувач вказує
відсотки, а не числове значення
   cout << "Total Earnings: $" << (ad.adsShown * ad.clickThroughRatePercentage / 100 *
ad.averageEarningsPerClick) << endl;
}

int main()
{
   using namespace std;
    // Оголошуємо змінну структури Advertising
   Advertising ad;
    cout << "How many ads were shown today? ";
   cin >> ad.adsShown;
   cout << "What percentage of users clicked on the ads? ";
   cin >> ad.clickThroughRatePercentage;
   cout << "What was the average earnings per click? ";
   cin >> ad.averageEarningsPerClick;
    printAdvertising(ad);
    return 0;
}
```

Приклад 5. Створити структуру для зберігання дробових чисел. Структура повинна мати 2 члени:

- цілочисельний чисельник;
- цілочисельний знаменник.

Оголосити дві змінні та отримати їх значення від користувача. Написати функцію multiply() (параметри містять значення двох відповідних змінних), яка перемножує задані значення та виводить результат десятковим числом.

Програмний код реалізації

```
#include <iostream>
 struct Drob
{
   int chislitel;
   int znamenatel;
};
```

```
void multiply(Drob d1, Drob d2)
{
   using namespace std;
    // використаємо static_cast, інакше компілятор виконає цілочисельне ділення!
   cout << static_cast<float>(d1.chislitel* d2.chislitel)/(d1.znamenatel* d2.znamenatel);
}

int main()
{
   using namespace std;
    // Визначаємо першу дробову змінну
   Drob d1;
   cout << "Input the first chislitel: ";
   cin >> d1.chislitel;
   cout << "Input the first znamenatel: ";
   cin >> d1.znamenatel;

   // Визначаємо другу дробову змінну
   Drob d2;
   cout << "Input the second chislitel: ";
   cin >> d2.chislitel;
   cout << "Input the second znamenatel: ";
   cin >> d2.znamenatel;

   multiply(d1, d2);
    return 0; }
```

Контрольні запитання

1. Який синтаксис опису структури?
2. Як розподіляється пам‘ять у структурі?
3. Як можна забезпечити доступ до елемента структури?
4. Як можна забезпечити доступ до елемента структури черезвказівник?
5. Як описати змінну структурованого типу?
6. Як описати в структурі змінну структурованого типу?
7. Що таке вкладені структури?
8. Як визначити розмір структури?
9. Що таке покажчики на структуру?
10. Що таке масив структур?
11. Що таке бітові поля?
12. Що таке об‘єднання?

Практичне завдання 8.
Робота з текстовими та двійковими файлами

Мета: ознайомити студентів з основними поняттями розробки та описання програм мовою програмування Visual C++ з використанням текстових та двійкових файлів.

Завдання 1. Мовою програмування С++ створити структуру відповідно до наданого варіанту з використанням текстових та бінарних файлів:

1. Зчитати масив даних структурованого типу з текстового файлу.
2. Відсортований масив вивести у текстовий файл з використанням бібліотеки ***cstdio***
3. Вивести масив структури у бінарний файл.
4. Зчитати з бінарного файлу всі дані у масив структур.
5. Із зчитаного масиву виконати відбір відповідно варіанту і результат вивести у інший текстовий файл з використанням бібліотеки ***fstream***

Таблиця 24. – варіанти завдань по роботі з структурами та файлами

№ вар.	Завдання до виконання
1	***1. Описати структуру з ім'ям ZARPLATA, яка містить наступні поля:*** ➢ Name – прізвище ім'я та по батькові працівника; ➢ Date – дата народження (структура: day; month, year - день, місяць, рік); ➢ Work_day – кількість відпрацьованих днів; ➢ Stavka – ставка з урахуванням на 24 робочих дні; ➢ Naraxovano – нараховано заробітної плати з урахуванням ставки і кількості відпрацьованих днів; ➢ Do_viplati – сума виплати заробітної плати працівнику з вирахуванням 20% податку. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних BUHGALTER, що складається з N змінних типу ZARPLATA і підраховує поля нараховано та до виплати; ➢ Впорядковує записи по зростанню поля Name; ➢ Виводить на екран всіх працівників з їхньою заробітною платою, які відпрацювали менше 15 робочих днів.
2	***1. Описати структуру з ім'ям INSTRUMENT, яка містить наступні поля:*** ➢ Name – назва музикального інструменту; ➢ Type – тип музикального інструменту; ➢ Year – рік виготовлення інструменту; ➢ Vlasnik – власник інструменту; ➢ Vartist – вартість музикального інструменту; ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних ORKESTR, що складається з N змінних типу INSTRUMENT; ➢ Впорядковує записи по зростанню поля Vartist; ➢ Виводить на екран всі інструменти, вартість яких більша за 1000 грн.

№ вар.	Завдання до виконання
3	***1. Описати структуру з ім'ям COMPUTER, яка містить наступні поля:*** ➢ Procesor – процесор комп'ютера; ➢ Ram – обсяг оперативної пам'яті; ➢ HDD – структура що містить поля (Namt- виробник, V_Ram - обсяг, V- швидкість обертання диску); ➢ Monitor – ставка з урахуванням на 24 робочих дні; ➢ Keyboard – нараховано заробітної плати з урахуванням ставки і кількості відпрацьованих днів; ➢ Mouse – сума виплати заробітної плати працівнику з вирахуванням 20% податку. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних CLASS, що складається з N змінних типу COMPUTER; ➢ Впорядковує записи по зростанню поля V_ram; ➢ Виводить на екран всіх комп'ютери за введеним процесором.
4	***1. Описати структуру з ім'ям CUPURA, яка містить наступні поля:*** ➢ Name – назва валюти; ➢ Nominal – номінал; ➢ Year – рік випуску куп'юри; ➢ Kurs – курс валюти відносно української гривні; ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних BANK, що складається з N змінних типу CUPURA; ➢ Впорядковує записи по зростанню поля Name; ➢ Виводить на екран всіх номінали вказаного року.
5	***1. Описати структуру з ім'ям STUDENT, яка містить наступні поля:*** ➢ Name – Прізвище та ініціали; ➢ Year – рік народження: ➢ Bal – оцінки з 4 предметів (масив з 4 елементів) ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних Group, що складається з N змінних типу STUDENT; ➢ Впорядковує записи за зростанням поля Year ➢ Виводить на екран прізвища і рік народження студентів середній бал яких більше 4.0.
6	***1. Описати структуру з ім'ям SKLAD, яка містить наступні поля:*** ➢ Name – Назва товару; ➢ Type – одиниця вимірювання: ➢ Quantity – кількість одиниць товару; ➢ Cost – ціна одиниці товару. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних SHOP, що складається з N змінних типу SKLAD; ➢ Впорядковує записи по спаданню поля Name; ➢ Виводить на екран ціну та кількість товару, назва якого вводиться з клавіатури або виводить повідомлення про його відсутність.

№ вар.	Завдання до виконання
7	***1. Описати структуру з ім'ям TRAIN, яка містить наступні поля:*** ➢ Nazv – Назва ➢ Numer – номер поїзда; ➢ Date – дата відправлення (структура: day; month, year - день, місяць, рік); ➢ Time – ціна одиниці товару. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних AVTOPARK що складається з N змінних типу TRAIN; ➢ Впорядковує записи по спаданню поля Numer; ➢ Виводить на екран всі рейси, які час відправлення яких після 15.00 по введеній даті.
8	***1. Описати структуру з ім'ям ABONENT, яка містить наступні поля:*** ➢ Name – прізвище абонента; ➢ Init – ініціали абонента; ➢ Nomer – номер телефону; ➢ Adress –домашня адреса. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних TELEFON, що складається з N змінних типу ABONENT; ➢ Впорядковує записи по зростанню поля Name; ➢ Виводить на екран прізвище, ініціали та домашню адресу за введеним номером телефону, або виводить повідомлення про його відсутність.
9	***1. Описати структуру з ім'ям AEROFLOT, яка містить наступні поля:*** ➢ Nazv –назва пункту призначення; ➢ Numer – номер рейсу; ➢ Type –тип літака; ➢ Time –час відправлення. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних ROZKLAD, що складається з N змінних типу AEROFLOT; ➢ Впорядковує записи по зростанню поля Type; ➢ Виводить на екран всі номери рейсів, та час відправлення літаків які відправляються в введений з клавіатури пункт призначення або повідомляє про відсутність таких рейсів.
10	***1. Описати структуру з ім'ям DETAL, яка містить наступні поля:*** ➢ Name – назва деталі; ➢ Sort – сорт виробу; ➢ Date –дата виготовлення (структура: day; month, year - день, місяць, рік); ➢ Quant – кількість; ➢ Cost - ціна деталі. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних ZAKAZ, що складається з N змінних типу DETAL; ➢ Впорядковує записи по спаданню поля Name; ➢ Виводить на екран всі деталі I сорту які виготовлені пізніше заданої дати, яка введена з клавіатури.

№ вар.	Завдання до виконання
11	***1. Описати структуру з ім'ям BOOK, яка містить наступні поля:*** ➢ Name – назва книги; ➢ Avtor – автори книги; ➢ Data –дата друку (структура: month, year - місяць, рік); ➢ Cost - ціна книги. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних SHOP, що складається з N змінних типу BOOK; ➢ Впорядковує записи по зростанню поля Avtor; ➢ Виводить на екран всі книги які були надруковані в заданому році.
12	***1. Описати структуру з ім'ям TOVAR, яка містить наступні поля:*** ➢ Name – назва товару; ➢ Cost_Z – ціна закупки товару; ➢ Cost_P - ціна продажу товари. ➢ Quantity –кількість одиниць товару; ➢ Pributok – прибуток. ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних SHOP, що складається з N змінних типу TOVAR і обчислює прибуток по кожному товару; ➢ Впорядковує записи по зростанню поля Pributok; ➢ Виводить на екран всі товари, які мають найбільший прибуток.
13	***1. Описати структуру з ім'ям VISTAVA, яка містить наступні поля:*** ➢ Nazva – назва вистави; ➢ Date – дата вистави (структура: day; month, year - день, місяць, рік); ➢ Cost - ціна квитка. ➢ Day_week – день неділі; ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних THEATRE , що складається з N змінних типу VISTAVA; ➢ Впорядковує записи по зростанню поля Day_week; ➢ Виводить на екран всі вистави і дати їх проходження, які відбудуться в заданий день тижня.
14	***1. Описати структуру з ім'ям WORKER, яка містить наступні поля:*** ➢ Name – Ім'я працівника; ➢ Surname – Прізвище працівника ; ➢ Date – дата народження (структура: day; month, year - день, місяць, рік); ➢ Pos - посада працівника. ➢ Zarplata – заробітна плата працівника ***2. Написати програму, що використовує дану структуру і виконує наступні дії:*** ➢ вводить з клавіатури масив даних OFFICE, що складається з N змінних типу WORKER; ➢ Впорядковує записи по віку працівників по спаданню; ➢ Виводить на екран всіх працівників старших 30 років, які мають заробітну плату меншу введеної з клавіатури.

Зміст звіту

1. Титульна сторінка, тема та мета практичного заняття.
2. Завдання відповідно до номеру варіанту
3. Опис виконаних завдань, лістинги реалізованих програм, результати виконання програм.
4. Висновки, які відображають результати виконаних завдань та їх аналіз.

Теоретичні відомості

Теоретичні основи об'єктно-орієнтованого програмування мовою с++

Робота з текстовими файлами. Багато мов програмування надають класи для роботи з файлами та директоріями проекту. Мова С++ має безліч класів для запису та читання даних із файлів. Працювати з файлами можна за допомогою стандартної бібліотеки С++ ***fstream***, яка дозволяє:

- записувати дані у файл (***ofstream***);
- зчитувати дані з файлу (***ifstream***).

При роботі з файлами завжди потрібно пам'ятати про виконання двох основних процесів:

1. Перед початком роботи файл/файли необхідно відкрити;
2. Після завершення роботи файл/файли необхідно закрити.

Якщо файл не відкритий або неправильно відкритий, то не можливе виконання процесів запису/читання.

Якщо не закрито файл, то програма буде працювати правильно, проте чим більше буде відкритих файлів, тим більше програма буде перевантажена і, як результат, приведе до зависання/вимкнення пристрою.

Створення файлу. Запис даних у файл. При записі даних у файл спочатку необхідно відкрити його, скориставшись стандартним класом ofstream.

При відкритті файлу можна вказати режим відкриття. Існує кілька режимів і всі вони представлені в табл. 25.

Таблиця 25. - Режими відкриття текстових файлів

Константа	*Опис*
ios_base:: app	відкриває файл в режимі запису даних
ios_base:: ate	перехід в кінець файлу перед читанням/записом
ios_base:: binary	відкриває текстовий файл в бінарному режимі
ios_base:: in	відкриває файл в режимі читання (за замовчуванням ifstream)
ios_base:: out	відкриває файл в режимі запису (за замовчуванням ofstream)
ios_base:: trunc	видаляє існуючий файл

Приклад 1. Створити текстовий файл та записати в нього відповідні дані.

Програмний код реалізації

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
// Створення/відкриття файлу "examples.txt"
```

```
    ofstream proger_file("examples.txt");
    // Можна було використовувати метод open
    // ofstream proger_file;
    // proger_file.open("examples.txt");
    // перший запис коротший, тому варто використовувати його

// Відкриваємо файл для запису до нього тексту
ofstream file("examples.txt", ios_base::out);
if (file.is_open()) { // Перевіряємо чи відкрити сам файл
file << "Simple world"; // Записуємо текст на початок файлу
file.close(); // Закриваємо файл
}else // Якщо файл не вдалося відкрити, тоді видаємо помилку
cout << "Error!" << endl;

cin.get();
return 0;}
```

Читання даних із файлу. Для читання даних, файл необхідно спочатку відкрити та прочитати вміст файлу, використовуючи клас ifstream. Після виконання програми файл необхідно закрити.

Приклад 2. Зчитати із текстового файлу дані та передати їх у тимчасову змінну в програмі.

Програмний код реалізації

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
// тут зберігатиметься проміжний текст,
// який ми зчитуватимемо з файлу
char temp [100];
ifstream some_file("examples.txt"); // відкриваємо файл для читання
// зчитуємо лише перше слово і поміщаємо у змінну temp
some_file >> temp;
cout << temp << endl; // виводимо цю змінну
// Зчитуємо певну кількість символів – 100
// і записуємо їх у нашу змінну
some_file.getline(temp, 100);
some_file.close(); // Закриваємо файл
cout << temp << endl; // Відображаємо змінну temp

cin.get();
return 0; }
```

Приклад 3. До попередньо створеного текстового файлу, занести в кінці файлу відповідну інформацію.

Програмний код реалізації.

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
```

```
 char text[50];
  ofstream textFile("examples.txt", ios_base::app);
if (textFile.is_open()) {
  textFile << "Hi! Everything works great!";
  textFile.close();
} else
  cout << "Error!" << endl;

ifstream file("examples.txt");
if (!file.is_open())
  cout << "Error! File is not found!" << endl;
else {
  file.getline(text, 50);
  cout << text << endl;
  file.close();
}
  cin.get();
return 0; }
```

Робота з текстовими та двійковими файлами. Створення файлів за допомогою стандартної бібліотеки <cstdio>. До бібліотеки ***<stdio.h>*** відносяться***:***

- Оголошення – ***FILE *file***;
- Відкриття – ***FILE *fopen(char *name, char *mode)***;
- Перевірка досягнення кінця файлу – ***int feof(FILE *file)***;
- Закриття файлу – ***int fclose(FILE *file)***.

Спеціальна структура, оголошена у файлі <stdio.h>, яка використовує час роботи з файлами. Для роботи з файлом потрібно оголосити змінну FILE *<ім’я>. Функція fopen використовується для відкриття файлу. Перший параметр задає файлу. Другий параметр mode задає тип доступу до файлу. Режими доступу до текстових та двійкових файлів наведено в табл. 26

Таблиця 26. – Режими запису / читання файлів

Mode	***Дія***
"r"	Відкриття для читання. Якщо файл не існує або не знайдено,функція fopen повертає признак про помилку (NULL).
"w"	Відкриття для запису. Якщо файл існує його зміст видаляється. Якщо файл не існує, він створюється.
"a"	Відкриття для добавлення. Якщо файл не існує то вінстворюється.
"r+"	Відкривається файл для читання та запису. Файл повиненіснувати.
"w+"	Відкриття пустого файлу для читання та запису. Якщо файліснує його зміст видаляється.
"a+"	Відкриття файлу для читання та добавлення. Якщо файл неіснує файл створюється.
"rb"	Відкриття двійковий файлу без дозволу на модифікацію, файлвідкривається лише для читання.
"wb"	Створення нового двійкового файлу тільки для запису, якщофайл із вказаним ім'ям вже існує, то він перезапишеться.

Текстовий режим. Під час введення/виведення в текстовому режимі відбувається перетворення зовнішнім представленням значення та внутрішнім (машинним) представленням цього значення, які наведено в табл. 27.

Таблиця 27. - Зчитування даних в текстовому режимі

Значення	***Функція***
Введення одного символу	int getc (FILE *file);
Виведення одного символу	int putc (int c, FILE *file);
Введення	int fscanf (FILE *file, char *format, ...);
Виведення	int fprintf(FILE *file, char *format, ...);
Введення рядка	char* fgets (char *line, intmaxline, FILE *file);
Виведення рядка	int fputs (char *line, FILE *file);

Приклад 4. Створити текстовий файл з використанням стандартної бібліотеки **<*cstdio*>** та записати до нього відповідну інформацію.

Програмний код реалізації

```
#include <cstdio>
int main ()
{
   FILE *fp;
   char name[50];
   int age;

   fp = fopen("example1.txt","w");
   fprintf(fp, "%s %d", "Tim", 31);
   fclose(fp);

   fp = fopen("example1.txt","r");
   fscanf(fp, "%s %d", name, &age);
   fclose(fp);

   printf("Hello %s, You are %d years old\n", name, age);
   return 0;}
```

Приклад 5. Створити двійковий файл та зчитати/записати до нього відповідну інформацію по-елементно.

Програмний код реалізації

```
#include<cstdio>
#include<cstring>
#include<iostream>
using namespace std;
struct user
{
char Name[20]; int year;
};
int main()
{
FILE *fb;
fb = fopen("binfileStruct.bin","wb");
```

```
    user T;
    T.year=2000;
    strcpy(T.Name,"Ivan");
    fwrite(&T,sizeof(user), 1,fb);
    fclose(fb);
    user B;
    T.year=0;
    fb = fopen("binfileStruct.bin","rb");
    fread(&B,sizeof(user), 1,fb);
    printf ("%s %d \n", B.Name, B.year);
    fclose(fb);
    return 0; }
```

Приклад 6. Створити двійковий файл та зчитати/записати до нього масив елементів.

```
#include <cstdio> // підключаємо функції для роботи з файлами
#include <conio.h> // Підключення getch
 struct int_double // у файл записуватимемо пакети
{
   int i;
   double d;
};

int main(int argc, char* argv[])
{
   int_double pack; // пакет
   FILE *f; // змінна до роботи з файлом
   int i; // лічильник

   f=fopen("alc.dat", "ab+"); // відкриваємо бінарний файл для запису та читання в
    режимі додавання, тобто, якщо файлу немає, то він створиться, а якщо файл є,
    то вміст файлу не буде знищено, з файлу можна буде читати і у файл можна буде
    записувати
   pack.i=0; // ініціалізація пакету
   pack.d=0.0;
   for (i=0; i<10; i++) // запишемо у файл у циклі 10 пакетів
   {
pack.i=pack.i+1;
pack.d=pack.d+11;
fwrite(&pack, sizeof(int_double), 1, f); // записуємо у файл f рівно 1 пакет pack розміру
    int_double
   }
   fseek(f, 4*sizeof(int_double), SEEK_SET); // переміщаємося від початку (SEEK_SET)
    файлу f на 4 довжини пакета int_double, тобто на початок 5-го пакету
   fread(&pack, sizeof(int_double), 1, f); // зчитуємо з файлу f рівно 1 пакет pack
    розміру int_double
   printf("%d %f",pack.i,pack.d); // виводимо 5-й пакет (555.0) на екран
   getch(); // очікуємо натискання будь-якої клавіші
   fclose(f); // закриваємо файл
   return 0; }
```

Приклади реалізації програм по роботі з файлами мовою С++

Існують три основні класи файлового вводу/виводу в мові С++:

- **ifstream** (є дочірнім класу **istream**);
- **ofstream** (є дочірнім класу **ostream**);
- **fstream** (є дочірнім класу **iostream**).

За допомогою цих класів можна виконувати однонаправлений файловий ввід, однонаправлений файловий вивід і двонаправлений файловий ввід/вивід. Для їх використання потрібно всього лише підключити **заголовок fstream**.

На відміну від потоків cout, cin, cerr і clog, які можна використовувати, файлові потоки повинні бути явно встановлені програмістом. Тобто, щоб відкрити файл для читання і/або запису, потрібно створити об'єкт відповідного класу файлового вводу/виводу, вказавши ім'я файлу в якості параметра. Потім, за допомогою оператора вставки (<<) або оператора вилучення (>>), можна записувати дані в файл або зчитувати вміст файлу. Після виконання даних дій потрібно закрити файл — явно викликати **метод close()** або просто дозволити файловій змінній вводу/виводу вийти з області видимості (**деструктор** файлового класу вводу/виводу закриє цей файл автоматично).

Приклад 7. Створити текстовий файл, використовуючи клас ***ofstream*** та записати до нього відповідну інформацію в два рядки.

Програмний код реалізації

```
#include <iostream>
#include <fstream>
#include <cstdlib> // для використання функції exit()

int main()
{
  using namespace std;
  // Клас ofstream використовується для запису даних в файл.
  // Створюємо файл Text.txt
  ofstream outf("Text.txt");
  // Якщо ми не можемо відкрити цей файл для запису даних,
  if (!outf)
  {
        // то виводимо повідомлення про помилку і виконуємо функцію exit()
        cerr << "Uh oh, Text.txt could not be opened for writing!" << endl;
        exit(1);
  }
  // Записуємо в файл наступні два рядки
  outf << "line 1!" << endl;
  outf << "line 2!" << endl;
  return 0;
  // Коли outf вийде з області видимості, то деструктор класу ofstream автоматично
  закриє файл }
```

Після виконання програми в каталозі проекту (ПКМ по вкладці з назвою .cpp-файлу в Visual Studio > "Открыть содержащую папку"), буде створено файл з іменем Text.txt.

Приклад 8. Прочитати вміст попередньо створеного файлу Text.txt, використовуючи ***метод ifstream***.

Програмний код реалізації

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib> // для використання функції exit()
 int main()
{
  using namespace std;
  // ifstream використовується для читання вмісту файла.
  // зчитування вмісту файлу Text.txt
  ifstream inf("Text.txt");
  // Якщо ми не можемо відкрити цей файл для читання його вмісту,
  if (!inf)
  {
     // то виводимо наступне повідомлення про помилку і виконуємо функцію exit()
        cerr << "Uh oh, Text.txt could not be opened for reading!" << endl;
        exit(1);
  }
  // Поки є дані у файлі, проходить процес зчитування
  while (inf)
  {
        // переміщуємо ці дані в рядок, який виводиться на екран
        string strInput;
        inf >> strInput;
        cout << strInput << endl;
  }
  return 0;
  // Коли inf вийде з області видимості, то деструктор класу ifstream автоматично
  закриє файл }
```

Метод ifstream поверне 0, якщо досягнено кінця файлу. Даний метод зручно використовувати для визначення «довжини» вмісту файлу.

Приклад 9. Прочитати вміст попередньо створеного файлу Text.txt, використовуючи ***метод getline()***

Програмний код реалізації

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib> // для використання функції exit()
 int main()
{
  using namespace std;
  // ifstream використовується для читання вмісту файлів.
  // зчитування вмісту файлу Text.txt
  ifstream inf("Text.txt");
  // Якщо не можливо відкрити файл для читання його вмісту,
  if (!inf)
  {
    // то виводиться повідомлення про помилку і виконується функцію exit()
```

```
        cerr << "Uh oh, Text.txt could not be opened for reading!" << endl;
        exit(1);
    }
    // Поки є текст у файлі
    while (inf)
    {
        // то переміщуємо вміст файлу в рядок та сформований рядок на екран
        string strInput;
        getline(inf, strInput);
        cout << strInput << endl;
    }
    return 0;
    // коли inf вийде з області видимості, то деструктор класу ifstream автоматично
    закриє файл }
```

Режими відкриття файлів. Повторний запуск вищенаведеної програми (найперша) показує, що вихідний файл повністю перезаписується при повторному запуску програми. Виявляється, **конструктори** файлового потоку приймають необов'язковий другий параметр, який дозволяє вказати програмісту спосіб відкриття файлу. В якості цього параметра можна передавати **наступні флаги** (які знаходяться в класі ios):

- **app** — відкриває файл в режимі додавання;
- **ate** — переходить в кінець файлу перед читанням/записом;
- **binary** — відкриває файл в бінарному режимі (замість текстового режиму);
- **in** — відкриває файл в режимі читання (за замовчуванням для ***ifstream***);
- **out** — відкриває файл в режимі запису (за замовчуванням для ***ofstream***);
- **trunc** — видаляє файл, якщо він вже існує.

Можна вказати відразу кілька флагів шляхом використання **побітового АБО (|)**.

- **ifstream** за замовчуванням працює в режимі ios::in;
- **ofstream** за замовчуванням працює в режимі ios::out;
- **fstream** за замовчуванням працює в режимі ios::in АБО ios::out, що означає, що ви можете виконувати як читання вмісту файлу, так і запис даних в файл.

Приклад 10. Написати програму, яка додає два рядки в раніше створений файл ***Text.txt***.

Програмний код реалізації

```
#include <iostream>
#include <cstdlib> // для використання функції exit()
#include <fstream>
 int main()
{
  using namespace std;
```

```
// Передаємо флаг ios:app, щоб повідомити fstream, про запис даних до існуючих даних файлу.
// оскільки файл не перезаписується не потрібно передавати флаг ios::out, так як ofstream за замовчуванням працює в режимі ios::out
ofstream outf("Text.txt", ios::app);
// Якщо не можливо відкрити файл для запису даних,
if (!outf)
{
    // то виводимо повідомлення про помилку і виконується exit()
    cerr << "Uh oh, Text.txt could not be opened for writing!" << endl;
    exit(1);
}
outf << "line 3!" << endl;
outf << "line 4!" << endl;
return 0;
// коли outf вийде з області видимості, то деструктор класу ofstream автоматично закриє файл }
```

Відкриття файлів за допомогою функції open(). Як і при закритті файлу за допомогою ***методу close()***, можна відкривати файл за допомогою **функції open()**. Функція ***open()*** працює аналогічно конструкторам класу файлового введення/виведення: приймає ім'я файлу і режим (необов'язково), в якому потрібно відкрити файл, в якості параметрів.

Приклад 11. Написати програму, використовуючи метод ***open(),*** записавши до файлу два рядки з інформацією. Додати в кінці файлу додаткову інформацію.

Програмний код реалізації

```
#include <fstream>
 int main()
{
   using namespace std;
   ofstream outf("Text1.txt");
   outf << "line #5!" << endl;
   outf << "line #6!" << endl;
   outf.close(); // закриваємо файл
   // дописати інформацію у файл
   outf.open("Text1.txt", ios::app);
   outf << "line #7!" << endl;
   outf.close();
   return 0;
   // Коли outf вийде з області видимості, то деструктор класу ofstream автоматично закриє файл }
```

Приклад 12. Реалізувати програмно структуру та передати її вміст у текстовий файл

Програмний код реалізації

```
#include <iostream>
#include <fstream>
#include <vector>
 struct Operation
```

```
{
    int sum;        // куплена сума валюти
    double rate;    // по курсу
    Operation(double s, double r) : sum(s), rate(r)
    {}
};
int main()
{
    std::vector<Operation> operations = {
        Operation(120, 57.7),
        Operation(1030, 57.4),
        Operation(980, 58.5),
        Operation(560, 57.2)
    };
     std::ofstream out("D:\\operations.txt");
      if (out.is_open())
    {
        for (int i = 0; i < operations.size(); i++)
        {
            out << operations[i].sum << " " << operations[i].rate << std::endl;
        }
    }
    out.close();

    std::vector<Operation> new_operations;
    double rate;
    int sum;
    std::ifstream in("D:\\operations.txt"); // відкрити файл для читання
    if (in.is_open())
    {
        while (in >> sum >> rate)
        {
            new_operations.push_back(Operation(sum, rate));
        }
    }
    in.close();

    for (int i = 0; i < new_operations.size(); i++)
    {
        std::cout << new_operations[i].sum << " - " << new_operations[i].rate << std::endl;
    }
    return 0;}
```

Приклад 13. Реалізувати структуру, яка описує модель автомобілів двома способами:

- Вивести масив структури у текстовий файл
- Вивести масив структури у бінарний файл

Програмний код реалізації

```
#include <fstream>
#include <iomanip>

int main() {
```

```
const int model_capacity = 20; // максимальний розмір рядка

struct Device {
  char model [model_capacity]; // Модель
  double price; // ціна
  int quantity; // кількість
};

int n_devices = 2;
Device *devices = new Device[n_devices]{
  {"MacBook Pro", 100500.0, 256 + 2},
  {"iMac", 100500.0, 256 + 2}
};

const int precision = 2; // знаків після коми у полі ціни
const int model_width = 19; // ширина поля моделі у текстовому файлі
const int price_width = 9; // ширина поля ціни у текстовому файлі
const int quantity_width = 4; // ширина поля кількості текстовому файлі

const int buffer_capacity = 80; // максимальний розмір буфера
char buffer [buffer_capacity]; //допоміжний буфер

//Виведення масиву структур у текстовий файл 1 способом
std::fstream file_txt;
file_txt.open("devices.txt", std::ios::out);
file_txt << n_devices << std::endl;
file_txt.setf(std::ios::left | std::ios::fixed);
file_txt.precision(precision);
for (int i = 0; i < n_devices; ++i) {
  file_txt << std::setw(model_width) << devices[i].model << ' ' <<
           std::setw(price_width) << devices[i].price << ' ' <<
           std::setw(quantity_width) << devices[i].quantity << std::endl;
}
file_txt.close();
delete [] devices;

// Введення масиву структур із текстового файлу 2 способом
file_txt.open("devices.txt", std::ios::in);
file_txt >> n_devices;
file_txt.getline(buffer, buffer_capacity); // NB
devices = new Device[n_devices]{};
for (int i = 0; i < n_devices; ++i) {
  file_txt.get(devices[i].model, model_capacity); // NB
  file_txt >> devices[i].price >> devices[i].quantity;
  file_txt.getline(buffer, buffer_capacity); // NB
}
file_txt.close();

// Виведення масиву структур у бінарний файл 1 способом
std::fstream file_bin;
file_bin.open("devices.bin", std::ios::out | std::ios::binary);
file_bin.write((char*) &n_devices, sizeof(n_devices)); // записати набір байтів
```

```
file_bin.write((char*) devices, n_devices * sizeof(Device)); // записати набір байтів
file_bin.close();
delete [] devices;

// Введення масиву структур із бінарного файлу 2 способом
file_bin.open("devices.bin", std::ios::in | std::ios::binary);
file_bin.read((char*) &n_devices, sizeof(n_devices)); // рахувати набір байтів
devices = new Device[n_devices]{};
file_bin.read((char*) devices, n_devices * sizeof(Device)); // рахувати набір байтів
file_bin.close();
delete [] devices;
return 0;}
```

Контрольні запитання

1. Що таке текстовий (бінарний) файл?
2. Як оголосити файлову змінну (stdio.h, fstream)?
3. Як відкрити файл для читання, запису, дозапису (stdio.h, fstream)?
4. Як записати дані у текстовий файл (stdio.h, fstream)?
5. Як зчитати дані з текстового файлу (stdio.h, fstream)?
6. Як записати дані у двійковий файл ?
7. Як зчитати дані з двійкового файлу ?
8. Як встановити позицію запису, зчитування даних з файлу (у файл)?
9. Як визначити позицію зчитування(запису)?
10. Як закрити текстовий, (бінарний) файл?
11. Як визначити кінець файлу?

Практичне завдання 9.
Розробка програм з використанням класів

Мета: набути уміння та навички розробки та описання програм з використанням методів об’єктно-орієнтованого програмування: об’єктів та класів в Visual C++; порівняти об'єктно-орієнтований та функціональний підходи у програмуванні; ознайомитись на практиці з класами, об'єктами та головними елементами об'єктного підходу; навчитись створювати і використовувати об’єкти типу клас; навчитись на практиці застосовувати успадкування класів, створювати ієрархію класів.

Завдання 1. Мовою програмування С++ створити консольний проект відповідно до заданого варіанті (***табл. 28***) з використанням об’єктів та класів відповідно до поданих варіантів. Проект створити з наступними файлами: файл з функціями класу *.cpp, головна функція main.cpp. Доступ до полів класу виконати через відповідні методи класу.

Таблиця 28. – Завдання до реалізації класів

№ вар.	Індивідуальне завдання
1	***1. Написати клас TOVAR, яка містить наступні поля:*** ➢ Name – назва товару; ➢ Cost_Z – ціна закупки товару; ➢ Cost_P - ціна продажу товари. ➢ Quantity –кількість одиниць товару; ➢ Pributok – прибуток. ***2. Написати програму, що використовує даний клас і виконує наступні дії:*** ➢ вводить з клавіатури масив даних SHOP, що складається з N змінних типу TOVAR і обчислює прибуток по кожному товару; ➢ Виводить на екран всі товари, які мають найбільший прибуток.
2	***1. Написати клас VISTAVA, яка містить наступні поля:*** ➢ Nazva – назва вистави; ➢ Date – дата вистави; ➢ Cost - ціна квитка. ➢ Day_week – день неділі; ***2. Написати програму, що використовує даний клас і виконує наступні дії:*** ➢ вводить з клавіатури масив даних THEATRE , що складається з N змінних типу VISTAVA; ➢ Виводить на екран всі вистави і дати їх проходження, які відбудуться в заданий день тижня.
3	***1. Написати клас WORKER, яка містить наступні поля:*** ➢ Name – Ім‘я працівника; ➢ Surname – Прізвище працівника ; ➢ Date – дата народження; ➢ Pos - посада працівника. ➢ Zarplata – заробітна плата працівника ***2. Написати програму, що використовує даний клас і виконує наступні дії:*** ➢ вводить з клавіатури масив даних OFFICE, що складається з N змінних типу

№ вар.	*Індивідуальне завдання*
	WORKER; ➢ Виводить на екран всіх працівників старших 45 років, які мають заробітну плату меншу введеної з клавіатури.
4	***1. Написати клас ZARPLATA, який містить наступні поля:*** ➢ Name – прізвище ім‘я та по батькові працівника; ➢ Date – дата народження; ➢ Work_day – кількість відпрацьованих днів; ➢ Stavka – ставка з урахуванням на 24 робочих дні; ➢ Naraxovano – нараховано заробітної плати з урахуванням ставки і кількості відпрацьованих днів; ➢ Do_viplati – сума виплати заробітної плати працівнику з 20% податком. ***2. Написати програму, що використовує даний клас і виконує наступні дії::*** ➢ вводить з клавіатури масив даних BUHGALTER, що складається з N змінних типу ZARPLATA і підраховує поля нараховано та до виплати ; ➢ Виводить на екран всіх працівників з їхньою заробітною платою, які відпрацювали менше 15 робочих днів.
5	***1. Написати клас COIN, яка містить наступні поля:*** ➢ Country – держава виготовлення; ➢ Name – назва монети; ➢ Year – рік виготовлення; ➢ Nominal – номінал; ➢ Price – ціна; ***2. Написати програму, що використовує даний клас і виконує наступні дії:*** ➢ вводить з клавіатури масив даних COIN_COLECTION, що складається з N змінних типу COIN; ➢ Виводить на екран список монет 2000 року.
6	***1. Написати клас CD, який містить наступні поля:*** ➢ Name – назва диска; ➢ Date – дата запису диска(структура: month, year - місяць, рік); ➢ Theme – тема; ➢ SIZE – розмір; ***2. Написати програму, що використовує даний клас і виконує наступні дії:*** ➢ вводить з клавіатури масив даних CD_COLECTION, що складається з N змінних типу CD; ➢ Виводить на екран список дисків які були записані у 2023 році.
7	***1. Написати клас VIDEO, який містить наступні поля:*** ➢ Name – назва відео фільму; ➢ Year – рік зйомки фільму; ➢ Genre – жанр фільму; ➢ Rezhiser – режисер; ***2. Написати програму, що використовує даний клас і виконує наступні дії:*** ➢ вводить з клавіатури масив даних VIDEO_COLECTION, що складається з N змінних типу VIDEO; ➢ Виводить на екран список фільмів, по введеному з клавіатури режисеру.
8	***1. Створити клас STUDENT, яка містить наступні поля:*** ➢ Name – Прізвище та ініціали;

№ вар.	*Індивідуальне завдання*
	➢ Year – рік народження: ➢ Bal – оцінки з 4 предметів (масив з 4 елементів) ***2. Написати програму, що використовує даний клас і виконує наступні дії:*** ➢ вводить з клавіатури масив даних Group, що складається з N змінних типу STUDENT; ➢ Виводить на екран прізвища і рік народження студентів середній бал яких більше 90.
9	***1. Створити клас SKLAD, яка містить наступні поля:*** ➢ Name – Назва товару; ➢ Type – одиниця вимірювання: ➢ Quantity – кількість одиниць товару; ➢ Cost – ціна одиниці товару. ***2. Написати програму, що використовує даний клас і виконує наступні дії:*** ➢ вводить з клавіатури масив даних SHOP, що складається з N змінних типу SKLAD; ➢ Виводить на екран ціну та кількість товару, назва якого вводиться з клавіатури або виводить повідомлення про його відсутність.
10	***1. Створити клас TRAIN, яка містить наступні поля:*** ➢ Nazv – Назва ➢ Numer – номер поїзда; ➢ Date – дата відправлення ➢ Time – час відправлення поїзда. ***2. Написати програму, що використовує даний клас і виконує наступні дії:*** ➢ вводить з клавіатури масив даних AVTOPARK що складається з N змінних типу TRAIN; ➢ Виводить на екран всі рейси, які час відправлення яких після 20-00 по введеній даті.
11	***1. Створити клас ABONENT, яка містить наступні поля:*** ➢ Name – прізвище абонента; ➢ Init – ініціали абонента; ➢ Nomer – номер телефону; ➢ Adress –домашня адреса. ***2. Написати програму, що використовує даний клас і виконує наступні дії:*** ➢ вводить з клавіатури масив даних TELEFON, що складається з N змінних типу ABONENT; ➢ Виводить на екран прізвище, ініціали та домашню адресу за введеним номером телефону, або виводить повідомлення про його відсутність.
12	***1. Написати клас AEROFLOT, яка містить наступні поля:*** ➢ Nazv –назва пункту призначення; ➢ Numer – номер рейсу; ➢ Type –тип літака; ➢ Time –час відправлення. ***2. Написати програму, що використовує даний клас і виконує наступні дії:*** ➢ вводить з клавіатури масив даних ROZKLAD, що складається з N змінних типу AEROFLOT; ➢ Виводить на екран всі номери рейсів, та час відправлення літаків, які

№ вар.	Індивідуальне завдання
	відправляються в введений з клавіатури пункт призначення або повідомляє про відсутність таких рейсів.
13	***1. Написати клас DETAL, яка містить наступні поля:*** ➢ Name – назва деталі; ➢ Sort – сорт виробу; ➢ Date –дата виготовлення ➢ Quant – кількість; ➢ Cost - ціна деталі. ***2. Написати програму, що використовує даний клас і виконує наступні дії:*** ➢ вводить з клавіатури масив даних ZAKAZ, що складається з N змінних типу DETAL; ➢ Виводить на екран всі деталі I сорту які виготовлені пізніше заданої дати, яка введена з клавіатури.
14	***1. Написати клас BOOK, яка містить наступні поля:*** ➢ Name – назва книги; ➢ Avtor – автори книги; ➢ Data –дата друку; ➢ Cost - ціна книги. ***2. Написати програму, що використовує даний клас і виконує наступні дії:*** ➢ вводить з клавіатури масив даних SHOP, що складається з N змінних типу BOOK; ➢ Виводить на екран всі книги які були надруковані в заданому році.

Завдання 2. Мовою програмування С++ створити консольний проект з використанням об'єктів та класів відповідно до поданих варіантів (***табл. 29***). В завданні повинні бути реалізовані наступні методи:

- метод ініціалізації Init,
- введення з клавіатури Read,
- виведення на екран Display,
- перетворення в рядок toString.

При реалізації, всі поля повинні бути закритими.

Таблиця 29. – Завдання по організації класів

№ вар.	Завдання до виконання
1	Створити клас ***Vector3d***, що задається трійкою координат. Реалізувати методи: додавання і віднімання векторів, скалярний добуток векторів, множення на скаляр, порівняння векторів, обчислення довжини вектора, порівняння довжини вектора.
2	Створити клас ***Money*** для роботи з грошовими сумами. Число представлено двома полями: гривні і копійки. Реалізувати методи: додавання і віднімання, ділення сум, розподіл суми на дробове число, множення суми на дробове число, операції порівняння.
3	Створити клас ***Triangle*** для подання трикутника. Поля: сторони і кути. Реалізувати операції: отримання і зміни полів даних, обчислення площі, обчислення периметра, обчислення висот, визначення виду трикутника.

№ вар.	Завдання до виконання
4	Створити клас ***Angle*** для роботи з кутами на площині. Поля: градуси і хвилини. Реалізувати методи: переклад в радіани, приведення до діапазону 0–360, збільшення і зменшення кута на задану величину, 8 отримання синуса, порівняння кутів.
5	Створити клас ***Data*** для роботи з датами. Поля: рік, місяць, день. Реалізувати операції: віднімання, порівняння дат, визначення високосного року, переклад в кількість днів.
6	Створити клас ***Time*** для роботи з часом. Поля: години, хвилини, секунда. Реалізувати операції: віднімання, порівняння часу, переклад в хвилини і секунди.
7	Створити клас ***Account***, який представляє собою банківський рахунок. Поля: прізвище власника, номер рахунку, відсоток нарахування, сума в гривнях. Реалізувати операції: зміна власника рахунку, зняття деякої суми з рахунку, покласти гроші на рахунок, нарахувати відсоток.
8	Створити клас ***Payment*** (Зарплата). Поля: ПІБ, оклад, рік надходження на роботу, прибутковий податок, відсоток надбавки, кількість відпрацьованих днів на місяць, нарахована і утримана суми. Реалізувати методи: обчислення нарахованої суми, обчислення утриманої суми, сума, що видається, обчислення стажу.
9	Створити клас ***Fraction***, для роботи з дробовими числами. Поля: ціла частина і дрібна. Реалізувати операції: додавання, віднімання, множення, операції порівняння.
10	Створити клас ***Goods*** (Товар). Поля: найменування товару, дата оформлення, ціна товару, кількість одиниць товару, номер накладної. Реалізувати методи: зміни ціни товару, зміни кількості товару, обчислення вартості товару.

Зміст звіту

1. Титульна сторінка, тема та мета практичного заняття.
2. Підготувати звіт по виконаній роботі, у якому представити наступні матеріали:
   - ✓ зміст завдання і варіант;
   - ✓ лістинг програми.
   - ✓ схему ієрархії класів програмного результати виконання – вигляд екрану при виконанні програми;
   - ✓ результати роботи розробленого програмного засобу;
3. Висновки, які відображають результати виконаних завдань та їх аналіз.

Теоретичні відомості

Модульне і об'єктно-орієнтоване програмування

Об'єктно-орієнтований підхід до програмування є пріоритетним при створюванні переважної більшості програмних проектів.

В остаточному вигляді кожна програма є набором інструкцій для процесора. І чим вище є рівень мови, тим у більш простій формі записуються одні й ті ж самі дії. З нарощуванням обсягу програм стає потрібним структурувати інформацію, виокремлювати в ній головне та відкидати несуттєве. Цей процес називається підвищенням ступеня *абстракції* програми.

Першим кроком до підвищення абстракції є ***використання функцій***, що дозволяє після написання та налагодження функції дистанціюватись від деталей її реалізації, оскільки для виклику функції треба знати лише її інтерфейс.

Наступний крок – ***оголошення власних типів даних***, які дозволяють

структурувати та групувати інформацію, подаючи її в більш природному вигляді. Оскільки для роботи з власними типами даних потрібні спеціальні функції, тому вважається за природне згруповувати їх разом з оголошенням цих типів в одному місці програми і у певний спосіб відокремити від решти її частин.

Отже, ***об'єднання в модулі*** оголошень типів даних та функцій, призначених для роботи з цими типами, разом із приховуванням від користувача модуля несуттєвих деталей, є подальшим розвитком структуризації програми.

Всі три згаданих вище методи підвищування абстракції ставлять за мету спростити структуру програми, тобто подати її у вигляді невеликої кількості більш потужних блоків та мінімізувати зв'язки поміж ними. Це дозволяє керувати великим обсягом інформації, а отже, успішно налагоджувати великі програми.

Введення поняття ***класу*** є розвитком ідей модульності. У класі поєднуються структури даних і функції їхнього опрацювання. Клас використовується лише через його інтерфейс – деталі реалізації для користувача класу є несуттєві.

Розглянемо існуючі стандартні типи даних, а саме:

- внутрішнє зображення даних у пам'яті комп'ютера, тобто який розмір пам'яті потрібний для розміщення певного об'єкта даних, і різновид значень, яких можуть набувати дані зазначеного типу;
- які операції та функції може бути застосовано до певних даних.

Для вбудованих типів таку інформацію закладено у компіляторі. А для типів, які визначає користувач, існує поняття класу.

Клас є типом даних, який визначає користувач. У класі задаються властивості та характер певного предмета чи процесу у вигляді полів даних (аналогічно до структури) і функцій для роботи з ними.

Суттєвою властивістю класу є те, що деталі його реалізації приховано від користувачів класу за інтерфейсом. Це захищає їх від випадкових змін. Інтерфейсом класу є заголовки його методів.

Ідея класів відображає будову об'єктів реального світу. Адже ж кожен об'єкт чи процес має набір певних характеристик чи відмінностей, інакше кажучи, певні властивості й поведінку. А користуватись об'єктами можна, не знаючи їхнього внутрішнього зображення. Наприклад, керувати автомобілем можна, не маючи уявлення про будову його двигуна чи будь-яких інших його частин.

Отже, клас як модель об'єкта реального світу є чорною скринькою, замкненою відносно зовнішнього світу. Ідея класів є підґрунтям об'єктно-орієнтованого програмування (ООП).

Основними поняттями, на яких базується ООП, є:

- ***інкапсуляція;***
- ***успадкування;***
- ***поліморфізм.***

Інкапсуляцією називається поєднання даних з функціями їхнього опрацювання разом із приховуванням зайвої для користування цими даними інформації. Інкапсуляція підвищує рівень абстракції програми, дозволяє

змінювати реалізацію класу без модифікації основної частини програми та використовувати клас в іншому оточенні.

Успадкування – це можливість створювання ієрархії класів, коли класи-нащадки успадковують властивості своїх базових класів (предків), можуть змінювати ці властивості й набувати нових. Властивості при успадкуванні не описуються, що скорочує обсяг програми.

Поліморфізм – це можливість використовувати у різних класах ієрархії одне ім'я для позначення близьких за змістом дій та гнучко обирати відповідні дії у перебігу виконування програми.

Отже, об'єктно-орієнтоване програмування у жодний спосіб не пов'язане з процесом виконування програми, а є лише новим способом її організації, тобто новою ***парадигмою*** програмування (парадигма – набір теорій та методів, які визначають спосіб організації знань).

Класи. Клас є абстрактним типом даних, визначуваним користувачем, і зображує модель реального світу у вигляді даних та функцій їхнього опрацювання. Дані класу називають ***полями*** (чи даними-членами), а функції класу – ***методами*** (чи функціями-членами). Поля та методи називаються ***елементами класу***.

Здебільшого специфікація класу складається з двох частин:

- ***оголошення класу*** – в ньому прописано всі поля й методи (елементи даних) класу на рівні інтерфейсу в термінах функцій-елементів;
- ***визначення методів класу***, тобто реалізації конкретних функцій-елементів даного класу.

Оголошення класу має наступний формат:

```
class <ім'я>
{ [ private: ]
    < Оголошення прихованих елементів >
    public:
    < Оголошення загальнодоступних елементів >
};     //Оголошення завершується крапкою з комою
```

Специфікатори доступу private та public керують видимістю елементів класу. Елементи, визначені після ключового слова **private**, є доступними лише у цьому класі. Цей різновид доступу прийнято у класі за замовчуванням. Поля і методи, визначені ключовим словом **public**, є доступні поза класом, тобто до них можна звертатися безпосередньо з програми, використовуючи об'єкти класу. Вміст загальнодоступного розділу public становить абстрактну частину конструкції, тобто загальнодоступний інтерфейс, який містить прототипи функцій-елементів чи повне визначення (для невеликих функцій).

Як приклад побудуємо клас Smallobj:

```
class Smallobj
  { private:
      int somedata;
      public:
        void setdata (int d)
```

```
    { somedata = d;}    //Метод, який змінює значення поля
    int getdata()
   { return  somedata;}  //Метод, який повертає значення поля
};
```

У цьому прикладі клас Smallobj містить лише одне поле даних somedata, яке має тип int, причому поле є доступним лише у класі, оскільки воно оголошено з ключовим словом private. Звертання до цього поля з програми неможливе і спричинить помилку. Однак доступ до даних можна організувати за допомогою методів, які ще називають методами доступу. Такими у класі Smallobj є два методи: setdata(), який присвоює полю значення, та getdata(), який повертає значення поля. Ці методи є доступні за межами класу, оскільки їх оголошено ключовим словом public. Тіла обох методів (коди функцій) містяться безпосередньо в оголошенні класу. Тут реалізація функцій не означає, що код функції розміщується у пам'яті. Це відбудеться лише при створенні об'єкта класу. Методи класу, визначені у подібний спосіб, за замовчуванням є вбудованими функціями. Але функції всередині класу найчастіше не визначаються, а лише оголошуються (тобто розміщуються їхні прототипи), а саме визначення функцій відбувається в іншому місці. Функція, визначена поза класом, за замовчуванням вже не є вбудованою.

Об'єкти. Тепер, коли клас визначено, можна створювати конкретні змінні типу "клас" – екземпляри класу, чи *об'єкти*. Час їхнього життя та видимість об'єктів залежить від форми та місця їхнього оголошення і підпорядковуються загальним правилам C++. При створюванні кожного об'єкта виділяється ділянка пам'яті, достатня для зберігання всіх його полів, і автоматично викликається конструктор, який виконує їхню ініціалізацію. Методи класу не тиражуються. При виході за межі області дії об'єкт знищується, при цьому автоматично викликається деструктор.

Приклад 1. Створити програму з використанням класу Smallobj у консольному режимі та організувати поля відповідних об'єктів.

Програмний код реалізації класу

```
#include <iostream>
using namespace std;

class Smallobj
{ private:
int somedata;
public:
void setdata(int d)
{ somedata = d; }
  int getdata()
{return somedata;}
};

int main ()
{ Smallobj s1, s2;
s1.setdata(2000);
s2.setdata(2022);
```

```
cout << "Значення поля є " << s1.getdata()<<endl;
cout << "Значення поля є " << s2.getdata()<<endl;
return 0;
}
```

Перший оператор функції main() визначає два об'єкти – s1 та s2 – класу Smallobj. Наступна пара операторів здійснює виклики методу setdata(). Тут доступ до елементів об'єкта (у даному разі до методів) є аналогічним до доступу до полів структури. Для цього після імені об'єкта ставиться крапка ".", а після неї зазначається ім'я методу. У такий спосіб задається метод і об'єкт застосування даного методу, тобто крапка пов'язує метод з ім'ям об'єкта. Але дужки після імені методу є обов'язковими, навіть якщо немає параметрів усередині. Дужки "говорять" про те, що здійснюється виклик функції, а не використовується значення змінної.

Операція крапки називається *операцією доступу до члена класу*. Отже, оператор s1.setdata(2000) викликає метод setdata() об'єкта s1, а метод присвоює полю somedata об'єкта s1 значення 2000. Наступний оператор присвоює полю somedata об'єкта s2 значення 2022.

Наступні два оператори викликають метод getdata() для об'єктів s1 та s2і виводять на екран значення полів відповідних об'єктів, повернутих цим методом.

Отже, результат виконання програми має вигляд:

Значення поля є 2000

Значення поля є 2022

Примітка. Якщо відбувається звертання до елемента не через ім'я об'єкта, а через покажчик, використовується операція доступу ->(замість крапки).

Звернутися за допомогою крапки чи -> можна лише до елементів зі специфікатором ***public***. Здобути доступ чи змінити значення елементів зі специфікатором ***private*** можна лише через звертання до відповідних методів.

Розглянемо приклад, який показує застосування об'єктів C++ в якості змінних типу, визначеного користувачем. Об'єкти будуть обчислювати і виводити проміжки часу в годинах та хвилинах. Далі цей приклад будемо вдосконалювати в міру набуття нових знань.

Приклад 2. Написати програму intervalobj1.cpp для консольного додатка переведення часу у форматі години/хвилини.

Програмний код реалізації класу

```
#include <iostream>
using namespace std;

class Interval
{ private:
int hour, minute;
 public:
void setinterval (int hr, int mn) // Встановлення значень полів
{ minute = mn % 60;
hour = hr + mn / 60;
}
void putinterval() // Введення значень полів з клавіатури
```

```
{ cout << "Введіть кількість годин: ";
cin >> hour;
cout << "Введіть кількість хвилин: ";
cin >> minute;
hour += minute / 60;
minute %= 60;
}
void showinterval() // Виведення часу на екран
{ cout << hour << " год. " << minute << " хв. ";}
};
int main()
{ Interval t1, t2;
t1.setinterval (1, 30);
t2.putinterval ();
// Виведення інтервалів на екран
cout << "\nt1 = ";
t1.showinterval ();
cout << "\nt2 = ";
t2.showinterval ();
cout << endl;
return 0; }
```

У цій програмі клас Interval містить два поля – hour та minute – і три методи:

- setinterval(), передбачений для задавання значень полів об'єкта через аргументи, які йому передаються;
- метод putinterval(), який забезпечує введення значень з клавіатури,
- метод showinterval() для виведення на екран проміжку часу в годинах та хвилинах.

Отже, значення полів об'єкта класу Interval можуть задаватися двома способами. У функції main() ми визначили дві змінні: t1 та t2. Значення поля для першої з них задається за допомогою функції setinterval(), викликаної з аргументами 1та 30, а значення поля змінної t2 вводиться користувачем.

Обидва методи здатні коригувати кількість хвилин, заданих користувачем: якщо значення поля minute перевищує 59, воно переводиться до значення з проміжку від 0 до 59, при цьому на відповідну кількість годин збільшується поле hour. Результат роботи програми має вигляд:

Введіть кількість годин: 3
Введіть кількість хвилин: 75
t1 = 1 год. 30 хв.
t2 = 4 год. 15 хв.

Приклади реалізації класів та об'єктів мовою С++

В класах методи можуть відрізнятись, кожен клас дозволяє створювати будь-яку кількість різних об'єктів, усі мають власні характеристики.

Створення класів. Для створення класу необхідно прописати ключове слово ***class*** та вказати назву класу. Прийнято починати назви класів із літери у верхньому регістрі, інакше виникне помилка.

У будь-якому класі можна створювати ***поля*** (змінні), ***методи*** (функції) та ***конструктори***.

Створивши новий клас і помістивши в нього будь-яку інформацію, можна створювати на його основі нові об'єкти. Об'єкти мають доступ до всіх характеристик класу, які позначені модифікатором ***public***.

Існують три модифікатори доступу:

• ***public*** - дані будуть видно всюди, як у класі, так і поза ним;

• ***protected*** - дані будуть видні лише у класі, де вони були створені, а також у класах спадкоємців;

• ***private*** – дані будуть видні лише у класі, де вони були створені.

Наведемо приклад простого класу:

```
class Book {
public:
    int pages;
    char name;
    float weight;

    void getInfoBook() {
    cout << "У книзі " << name << " наведено " << pages << " сторінок. " << endl;
    cout << "При цьому вона важить" << weight << endl;}};
```

На основі такого класу можна створити велику кількість об'єктів. Кожен об'єкт у разі представлятиме собою конкретну книжку. Для кожного об'єкта вказуються унікальні дані: кількість сторінок, назву книги та її вагу.

Щоб створити об'єкт, напишемо наступний код:

```
Book sherlock_holms; // Створення об'єкта
sherlock_holms.getInfoBook(); // Виклик методу класу
```

Щоб використати дані з класу через об'єкт, необхідно поставити крапку і вказати ім'я змінної або функції, яку використовуємо.

Приклад 3. Написати програму для консольного додатка, в якій необхідно реалізувати клас Auto з відповідними об’єктами.

Програмний код реалізації класу

```
#include <iostream>
using namespace std;
class Auto {
private:
  int year, month, day;
public:
  void message() {
    cout << "Class is working!" << endl;
  }

  void set(int date_year, int date_month, int date_day) {
    year = date_year;
    month = date_month;
    day = date_day;
  }

  void get() {
    cout << "Year of this auto is - " << year << ", month is - " << month << ", day is - "
<< day << endl;
  }
```

```
};

int main() {
  Auto shkoda;
  shkoda.message();
  shkoda.set(2005, 11, 23);
  shkoda.get();
  Auto bmw;
  bmw.set(2012, 1, 11);
  bmw.get();
  cin.get();
  return 0;
}
```

Приклад 4. Створити клас ***Tiles*** (кахель) в який помістити поля: ***brand, size_h, size_w, price***; метод ***getData***() для виведення інформації. Створити 2 об'єкти на основі класу та внести до них відповідні дані. Відобразити дані через метод ***getData***().

Програмний код реалізації класу

```
#include <iostream>
using namespace std;
class Tiles {
public:
char brand [32];
int size_h; // Розмір висоти
int size_w; // Розмір ширини
double price;
void getData() {
cout << brand << ": " << endl;
cout << size_h << "x" << endl;
cout << size_w << " - " << price << "$";
}
};

int main() {
Tiles obj;
cin >> obj.brand; // - поміщення рядка в масив char
obj.size_h = 10;
obj.size_w = 10;
obj.price = 30;

Tiles obj_2;
cin >> obj_2.brand;
obj_2.size_h = 20;
obj_2.size_w = 30;
obj_2.price = 25;
  cout << endl << endl; // пропустити 2 рядки
obj.getData();
cout << endl << endl; // пропустити 2 рядки
obj_2.getData();
return 0;
}
```

Приклад 5. Створити клас ***Room*** для визначення площі та об'єму житлової площі кімнат, використовуючи метод класу ***public***.

Програмний код реалізації класу

```
#include <iostream>
using namespace std;
// create a class
class Room {
  public:
   double length;
   double breadth;
   double height;
   double calculateArea() {
      return length * breadth;
   }
   double calculateVolume() {
      return length * breadth * height;
   }
};
int main() {
   // create object of Room class
   Room room1;
   // assign values to data members
   room1.length = 42.5;
   room1.breadth = 30.8;
   room1.height = 19.2;
   // calculate and display the area and volume of the room
   cout << "Площа кімнати =  " << room1.calculateArea() << endl;
   cout << "Об'єм кімнати =  " << room1.calculateVolume() << endl;
   return 0;
}
```

Приклад 6. Створити клас ***Room*** для визначення площі та об'єму житлової площі кімнат, використовуючи методи класу ***public*** та ***private***.

Програмний код реалізації класу

```
#include <iostream>
using namespace std;
class Room {
  private:
   double length;
   double breadth;
   double height;
  public:
   // function to initialize private variables
   void initData(double len, double brth, double hgt) {
      length = len;
      breadth = brth;
      height = hgt;
   }
   double calculateArea() {
      return length * breadth;
   }
```

```
    double calculateVolume() {
        return length * breadth * height;
    }
};
int main() {
    // create object of Room class
    Room room1;
    // pass the values of private variables as arguments
    room1.initData(42.5, 30.8, 19.2);
    cout << "Площа кімнати  =  " << room1.calculateArea() << endl;
    cout << "Об'єм кімнати  =  " << room1.calculateVolume() << endl;
    return 0; }
```

Контрольні запитання

1. Який синтаксис опису класу?
2. Що таке поля, методи класу?
3. Які є специфікатори доступу класу, їхнє призначення?
4. Як можна забезпечити доступ до елементів класу?
5. Що таке конструктор?
6. Які правила створення та роботи конструктора ?
7. Що таке конструктор по замовчанні, конструктор копіювання?
8. Що таке деструктор? Які правила створення та роботи деструктора ?
9. Які існують ініціалізації елементів у конструкторах?
10. Який порядок виклику конструкторів та деструкторів?

Практичне завдання 10.
Застосування конструкторів та деструкторів в мові С++

Мета: отримання практичних навичок в написанні програм з використанням конструкторів та деструкторів організованих класів в мові програмування С++; роботи з конструкторами та деструкторами; практично навчитись застосовувати успадкування класів, створювати їх ієрархію.

Завдання 1. Створити клас відповідно до варіанту (***табл. 30***) для обробки записів бази даних у відповідності з вказаною предметною областю. Розробити програму, дотримуючись таких вимог:

1. Розмістити інтерфейс класу у заголовочному файлі, а визначення функцій та головну функцію програми – у двох окремих файлах.
2. Передбачити можливість роботи з довільним числом записів, а також реалізувати окремими функціями класу:
   - конструктори без параметрів та з параметрами;
   - додавання об’єктів;
   - знищення об’єктів;
   - виведення інформації на екран;
   - пошук потрібної інформації за конкретною ознакою;
   - редагування записів;
   - сортування за різними полями.
3. При розробці програми слід здійснити захищення даних (опис з модифікатором *private*) для ізоляції елементів-даних класу від підпрограм, в яких цей клас використовується.
4. Програма повинна містити меню для перевірки всіх методів класу. Бажано для реалізації меню розробити окрему функцію, яка повертає номер вибраного пункту меню.
5. Утворити похідний клас, залучивши до нього як мінімум два додаткових поля. Для похідного класу використати конструктор, щоб він містив усі аргументи, необхідні для ініціалізації об'єктів. Створити додаткові функції, які дозволяють перевірити роботу похідних класів.

Таблиця 30. - Варіанти до виконання завдання 1

№	Предметна область	Реквізити об’єкту	Параметр сортування	Параметр пошуку
1	Бібліотека	інвентарний номер, автор, назва, кількість сторінок, рік видання	Рік видання	Автор
2	Телефонний довідник	Прізвище, ім'я, по батькові, домашня адреса, телефон.	Телефон	Прізвище
3	Розклад руху літаків	Номер рейсу, тип літака, напрямок руху, періодичність вильоту.	Номер рейсу	Тип літака
4	Колекція компакт-дисків	Інвентарний номер, назва альбому, об'єм диску, тип, дата запису.	Дата запису	Назва альбому

№	Предметна область	Реквізити об’єкту	Параметр сортування	Параметр пошуку
5	Записна книжка	Прізвище, ім'я, домашня адреса, телефон, електронна пошта.	Прізвище	Електронна пошта
6	Предметний покажчик	Слово; номера сторінок, де це слово зустрічається; кількість цих слів на даній сторінці	Номер сторінки	Слово
7	Розклад пар	Номер пари, предмет, прізвище викладача, форма заняття.	Предмет	Номер пари
8	Список файлів	ім'я файла, розширення, розмір, дата створення, атрибути.	Розширення	Дата створення
9	Архів програм	Назва програми, операційна система, розмір програми, дата запису	Назва програми	Операційна система
10	Рахунки банку	Прізвище, ім'я, дата останньої операції, сума вкладу	Сума вкладу	Дата операції
11	Користувачі локальної мережі	Прізвище, група, обліковий запис, тип облікового запису.	Тип облікового запису	Прізвище
12	Камера схову	Прізвище, дата здачі, термін зберігання, інвентарний номер та назва предмета	Інвентарний номер	Дата здачі
13	Склад товарів	інвентарний номер, назва товару, вага, ціна, кількість	Вага	Назва товару
14	Каса продажу квитків	Назва пункту, час відправлення, дата відправлення, час прибуття, дата прибуття, ціна квитка	Час від-правлення	Назва пункту
15	Успішність студентів	Прізвище, номер групи, оцінки з трьох предметів	Прізвище	Номер групи

Завдання 2. Мовою програмування С++ створити консольний проект з використанням конструкторів/деструкторів класів відповідно до поданих в табл. 31 варіантів.

Таблиця 31. - Варіанти до виконання завдання 2

№ вар.	Завдання до виконання
1	Створити клас ДАТИ з полями у закритій частині: день (1-31), місяць (1-12), рік (ціле число). Клас має конструктор, методи встановлення дня, місяця і року, методи отримання значень дня місяця року, а також методи виведення за шаблонами ”12 лютого 2023” і 12.02.2023.
2	Створити клас МАТРИЦІ, який у закритій частині містить вказівник на int, кількість рядків, стовпців та змінну стану. Визначити конструктор без параметрів, конструктор з одним параметром, та конструктор з двома параметрами, деструктор. Визначити метод для повернення значення елемента за індексами [i][j]. Визначити функцію виведення матриці. Визначити функції додавання, віднімання та множення матриць. Визначити функції множення матриці на число. Перевірити роботу цього класу. У випадку виходу за межі масиву встановлювати код помилки у змінній стану.

№ вар.	Завдання до виконання
3	Створити клас ЧАСУ з полями у закритій частині: година (0-23), хвилини (0-59), секунди (0-59). Клас має конструктор, методи встановлення часу, методи отримання годин, хвилин, і секунд, а також два методи виведення за шаблонами “16 годин 18 хвилин 3 секунди” і “4 p.m. 18 хвилин 3 секунди”. Методи встановлення полів класу повинні перевіряти коректність параметрів, що задаються.
4	Створити клас ОДНОЗВЯЗНИЙ СПИСОК. Методи класу додають елемент до списку, вилучають елементи зі списку, сортують список, відображають елементи списку від початку до кінця. Вилучити заданий елемент списку.
5	Створити клас ПРЯМОКУТНИК. У закритій частині визначити поля - висоту і ширину. Метод класу обчислюють площу, периметр, встановлюють поля даних і повертають їхні значення. У методах встановлення значень полів класу необхідно перевіряти коректність заданих параметрів. Визначити функцію виведення елементів класу.
6	Створити клас ДВОЗВЯЗНИЙ СПИСОК. Методи класу додають елемент до списку, 26 вилучають елементи зі списку, сортують список, відображають елементи списку від початку до кінця. Знайти у списку заданий елемент.
7	Створити клас КАЛЕНДАР. У закритій частині визначити дані - день, місяць, рік. Визначити необхідні конструктори, деструктори та методи. Методи класу встановлюють та зчитують значення полів даних, визначають назву тижня за заданою датою, виводять результат на екран. Ввести дату та визначити назву дня свого дня народження.
8	Створити клас ДАТА з полями у закритій частині: день (1-31), місяць (-12), рік (ціле число). Клас має конструктор, методи встановлення дня, місяця і року, методи читання дня, місяці і року. У методах встановлення полів класу необхідно перевіряти коректність заданих параметрів. Визначити метод, який збільшує значення дати на 1 день.
9	Створити клас ЧАС з полями у закритій частині: година(0-23), хвилини (0-59), секунди (0-59). Клас має конструктор, методи встановлення часу, методи читання години, хвилини і секунди. У методах встановлення полів класу необхідно перевіряти коректність параметрів, що задаються. Розробити метод для збільшення значення часу на 1 секунду, 1 хвилину, 1 годину.
10	Створити клас КВАДРАТ з полями у закритій частині: координати головної діагоналі. Методи класу обчислюють довжину сторони квадрату, площу, периметр, встановлюють значення полів і повертають їх значення.
11	Створити клас СТЕК. Розробити методи класу для введення елемента у стек та вилучення елемента зі стека. Розробити функцію, яка визначає кількість елементів у стеку. Методи полів класу повинні перевіряти коректність параметрів, що задаються.
12	Розробити клас квадратна матриця. У закритій частині визначити дані: порядок матриці та вказівник на її початок в області динамічної пам’яті. Клас має конструктор та деструктор. Розробити методи класу, які виконують такі операції: встановлення та виведення значень елементів матриці, визначення сліду матриці (суми елементів головної діагоналі), суми елементів вище та нижче головної діагоналі.
13	Розробити клас МНОЖИНА ЦІЛИХ ЧИСЕЛ. У закритій частині визначити вказівник на цілий тип (на елементи множини). Визначити конструктори, деструктор та методи створення множини, виведення вмісту множини, об’єднання, різниці та перетину множин.
14	Розробити клас РЯДОК СИМВОЛІВ. У закритій частині визначити вказівник на символьний тип (на початок рядка). Визначити конструктори, деструктор та методи введення/виведення рядка, конкатенації, порівняння рядків, входження підрядка.

№ вар.	Завдання до виконання
15	Розробити клас СТУДЕНТ. У закритій частині визначити дані - прізвище, номер залікової книжки, оцінки з предметів. Визначити конструктори, деструктор та методи встановлення і читання значень полів даних, знаходження середнього балу, визначення кількості незадовільних оцінок.
16	Розробити клас КНИГА. У закритій частині визначити дані - автор, назва, видавництво, рік видання. Визначити конструктори, деструктор та методи встановлення і читання значень полів даних, визначення відповідності книги пошуковим критеріям.
17	Цифровий лічильник - це змінна з обмеженим діапазоном, яка скидається у початкове значення, коли її цілочисельне значення досягає визначного максимуму. Опишіть клас такого ЛІЧИЛЬНИКА. Забезпечте можливість встановлення максимального і мінімального значень, збільшення значень лічильника на 1, повернення поточного значення.
18	Створити клас для виконання арифметичних операцій над ЦІЛИМИ ЧИСЛАМИ у двійковій системі числення. Числа задаються як стрічка символів в області динамічної пам'яті. Визначити конструктори, деструктор, методи виконання операцій, введення чисел та виведення результату.
19	Створити клас для виконання арифметичних операцій над ЦІЛИМИ ЧИСЛАМИ у шістнадцятковій системі числення. Числа задаються як стрічка символів в області динамічної пам'яті. Визначити конструктори, деструктор, методи виконання операцій, введення чисел та виведення результату.
20	Розробити клас, який виконує статистичну обробку ТЕКСТОВОГО ФАЙЛУ - підрахунок кількості символів, слів, речень. Визначити необхідні конструктори, деструктор та методи роботи з файлом.
21	Розробити клас для роботи зі СЛОВНИКОМ, який складається з масиву слів та їх перекладу іншою мовою. Визначити конструктори, деструктор та методи для додавання нових слів, пошуку слів, злиття двох словників без повторень елементів.
22	Розробити клас для роботи з КАТОТЕКОЮ КНИГ. Клас містить дані про назви книги: автор, назва книги, видавництво, рік видання. Реалізувати методи додавання даних про книгу до картотеки, витирання з картотеки. Пошук книг за прізвищем автора, назвою книги, роком видання.
23	Розробити клас для роботи з ПРАЙС-АРКУШЕМ комп'ютерної фірми, у якому вказано дані про марку комп'ютера, тип процесора, частоту процесора, об'єм оперативної та дискової пам'яті, характеристики відеокарти, ціну комп'ютера.
24	Створити клас для створення ДІЛОВОГО ЗАПИСНИКА з планом роботи на тиждень. Визначити конструктори, деструктор, методи роботи із записником - додавання нових записів, корегування та витирання записів.
25	Створити клас для роботи з ЧЕРГОЮ мешканців міста, які потребують покращення житлових умов. У закритій частині класу визначити поля даних - номер черги, прізвище та ініціали, склад сім'ї, дата поставлення у чергу. У відкритій частині класу визначити методи для поставлення та вилучення з черги, пошуку за номером черги, прізвищу, дані.
26	Розробити клас для роботи з ЧЕРГОЮ на біржі праці. У закритій частині класу визначити необхідні дані: прізвище, паспортні дані, освіта, професія, дата поставлення на облік, бажаний вид зайнятості, бажана зарплата. Визначити методи для роботи з цими даними - додавання, вилучення, корегування записів, виведення статистики.

Зміст звіту

1. Титульна сторінка, тема та мета практичного заняття.
2. Підготувати звіт по виконаній роботі, у якому представити наступні матеріали:
   - ✓ зміст завдання і варіант;
   - ✓ лістинг програми.
   - ✓ схему ієрархії класів програмного результати виконання – вигляд екрану при виконанні програми;
   - ✓ результати роботи розробленого програмного засобу;
3. Висновки, які відображають результати виконаних завдань та їх аналіз.

Теоретичні відомості

Конструктори. Конструктор – це метод, який виконується автоматично в момент створення об'єкта. Він може не лише ініціалізовувати змінні класу, але й виконувати будь-які інші завдання ініціалізації, потрібні для підготовки об'єкта до використання. Для створення екземпляра класу потрібно, щоб конструктор був методом типу ***public***.

Властивості конструкторів. Розглянемо *основні властивості* конструктора.

- Ім'я конструктора абсолютно збігається з іменем класу. Отже, компілятор відрізняє конструктор від інших методів класу.
- Конструктор не повертає жодного значення, навіть типу ***void***. Не можна здобути і покажчик на конструктор. Це пояснюється тим, що конструктор автоматично викликається системою, а отже, не існує програми чи функції, яка його викликає, і якій конструктор міг би повернути значення. Отже, задавати значення, яке повертається, для конструктора немає сенсу.
- Клас може мати кілька конструкторів з різними параметрами для різних видів ініціалізації, при цьому використовується механізм перевантаження.
- Конструктор може прийняти яку завгодно кількість аргументів (включаючи нульову).
- Параметри конструктора можуть бути якого завгодно типу, окрім цього класу. Можна задавати значення параметрів за замовчуванням. Їх може містити лише один з конструкторів.
- Конструктори не успадковуються.
- Конструктори не можна оголошувати з модифікаторами const, virtual, static.

Розглянемо різні способи визначення конструкторів.

Конструктор з параметрами. Конструктор може ініціалізувати поля класу, використовуючи вираз присвоювання, наприклад:

```
someClass (int mm1,  int mm2,  int mm3)
{ m1 = mm1;  m2 = mm2;  m3 = mm3;}
```

У цьому разі створення об'єкта може мати вигляд ***someClass obj (5, 20, 13);***

Тут викликається конструктор, до якого передаються відповідні значення параметрів. Окрім того, при створюванні екземпляра класу значення параметрів можна передавати конструктору, використовуючи оператор new, наприклад:

someClass *Pobj = new someClass (5, 20, 13);

Конструктор зі списком ініціалізації. Вказаний конструктор не може бути використаним при ініціалізації полів-констант чи полів-посилань, оскільки їм не можуть бути присвоєні значення. Для цього передбачено спеціальну властивість конструктора, називану списком ініціалізації, який дозволяє ініціалізувати одну чи більше змінних і не надавати їм значення. Список ініціалізації розташовується поміж прототипом методу та його тілом і після двокрапки. Ініціалізуюче значення розміщується у дужках після імені поля. Значення у списку розділяються комами, наприклад:

someClass(): m1(0), m2(10), m3(15) {}

При ініціалізації полів-об'єктів поля можна ініціалізувати за допомогою списку ініціалізаторів, у якому значення можуть бути виразами, наприклад:

someClass(int mm1,int mm2,int mm3):m1(mm1),m2(mm2),m3(mm3){}

В першому випадку конструктор викликається при створюванні об'єкта, наприклад, при визначенні:

someClass obj1, obj2;

У другому випадку при створюванні об'єкта obj буде викликано конструктор з відповідними, зазначеними у дужках параметрами, наприклад:

someClass obj(5,20,13);

Конструктор за замовчуванням. Конструктор без параметрів називають *конструктором за замовчуванням*. Такий конструктор зазвичай ініціалізує поля класу константними значеннями. Якщо конструктор для будь-якого класу не визначено, то компілятор сам генерує конструктор за замовчуванням. Такі конструктори не присвоюють значення полям класу. Тому, якщо треба однозначно ініціалізувати поля, треба визначити власний конструктор (ним може бути і конструктор за замовчуванням). Тоді для класу, який має конструктор за замовчуванням, можна визначити об'єкт класу без передавання параметрів, наприклад: ***someClass obj1, obj2;***

До визначення об'єкта класу вже не треба включати порожні круглі дужки. У класі може бути визначено не один конструктор з одним і тим самим іменем. Тоді говорять, що конструктор є перевантаженим. Який з них буде виконуватись при створюванні об'єктів, залежить від того, скільки аргументів використовується у виклику. Якщо у класі визначено будь-який конструктор, то компілятор не створить конструктора за замовчуванням. І якщо у класі не буде конструктора за замовчуванням, у певних ситуаціях можуть виникати помилки.

У такому разі слід визначити свій власний конструктор за замовчуванням, наприклад: ***someClass (): m1(0), m2(0), m3(0) {}***

Конструктор копіювання. Вище розглянуто два види конструкторів – конструктор без аргументів, який ініціалізує поля об'єкта константними значеннями, та конструктор, який має хоча б один аргумент, котрий ініціалізує

поля значеннями, переданими йому в якості аргументів. Тепер розглянемо ще один спосіб ініціалізації об'єкта, який використовує значення полів вже існуючого об'єкта.

Такий конструктор не треба самим створювати, він надається компілятором для кожного створюваного класу і називається *конструктором копіювання за замовчуванням*. Він має єдиний аргумент, який є об'єктом того ж самого класу.

У вищерозглянутій програмі intervalobj1.cpp видозмінимо функцію main(), де створимо три об'єкти t1, t2, t3 у різні способи

```
int main ()
{ Interval t1(2,30);
   Interval t2(t1);
   Interval t3 = t1;
   cout << "\nt1 = ";
   t1.showinterval();
   cout << "\nt2 = ";
   t2.showinterval();
   cout << "\nt3 = ";
   t3.showinterval();
   cout << endl;
}
```

Тут ініціалізовано об'єкт t1 значенням 2 год. 30 хв. за допомогою конструктора з двома аргументами. Потім визначено ще два об'єкти класу з іменами t2 та t3, які ініціалізуються значеннями об'єкта t1. Для копіювання значень полів об'єкта t1у відповідні поля об'єктів t2та t3 двічі викликається конструктор копіювання. Результат роботи програми має вигляд:

t1 = 2 год. 30 хв.
t2 = 2 год. 30 хв.
t3 = 2 год. 30 хв.

Якщо клас містить покажчики чи посилання, виклик конструктора копіювання призведе до того, що й копія, й оригінал вказуватимуть на одну й ту саму ділянку пам'яті. В такому разі конструктор копіювання має бути створений програмістом і мати вигляд

T :: T(const T&)

де ***T***– ім'я класу.

Наприклад: ***someClass :: someClass (const someClass &D) { . . . }***

Визначення методів класу поза класом. У кожному з прикладів класів, уже розглянутих у попередніх підрозділах, функції-члени повністю визначались усередині опису класу. Однак можна всередині опису класу розмістити лише прототип функції (оголосити функцію), а визначити її поза класом.

При зовнішньому визначенні функції-члена перед її ім'ям слід зазначити тип результату, ім'я класу й оператор розширення області видимості "::". Така форма запису встановлює взаємозв'язок функції та класу, до якого належить ця функція, наприклад:

void someClass :: func(someClass obj1, someClass obj2)

Приклад 1. Написати програму для консольного додатка переведення часу у форматі години/хвилини, застосовуючи конструктори класу **Interval.**

Програмний код реалізації класу

```
#include <iostream>
using namespace std;
class Interval
{ private:
int hour, minute;
public:
// Конструктор за замовчуванням
Interval() : hour(0), minute(0){}
// Конструктор із двома аргументами – лише оголошення
Interval (int h, int m);
// Метод, який дозволяє змінювати поля
void setinterv (int h, int m)
{ minute = m % 60;
hour = h + m/60;
}
void putinterval() // Введення часу з клавіатури
{ cout << "Введіть кількість годин: ";
cin >> hour;
cout << "Введіть кількість хвилин: ";
cin >> minute; hour += minute / 60; minute %= 60;
}
void showinterval() // Виведення часу на екран
{ cout << hour << " год. " << minute << " хв. ";
}
Interval add_interv(Interval); // Прототип.
};
// Визначення конструктора з двома аргументами
Interval :: Interval (int h, int m) : hour(h), minute(m)
{ hour += minute / 60; minute %= 60;
}
// Додавання даного об’єкта до d2, повертання суми
Interval Interval:: add_interv(Interval d2)
{ Interval temp;
temp.minute = minute + d2.minute;
temp.hour = hour + d2.hour;
if(temp.minute >59)
{ temp.minute -= 60; temp.hour++;}
return temp;
}
int main()
{ Interval t1(2, 30);
Interval t2, t3;
t2.putinterval();
t3 = t1.add_interv(t2);
// Виведення інтервалів на екран
cout << "\nt1 = ";
t1.showinterval(); cout << "\nt2 = ";
```

```
t2.showinterval(); cout << "\nt3 = ";
t3.showinterval(); cout << endl; }
```

Внесені вдосконалення до програми є наступними:

По-перше, наявність двох конструкторів: один – за замовчуванням, який присвоює нульові значення полям hour та minute, другий – такий, що ініціалізує ці поля тими значеннями, які передаються конструкторові як аргументи. Конструктором за замовчуванням ми користувались при створенні об'єктів t2 та t3, а другим конструктором – при створенні об'єкта t1.

По-друге, долучили функцію add_interv(), яка додає два проміжки часу. Функція є методом класу Interval, але її визначення розміщено поза описом класу. У програмі метод add_interv() викликається для об'єкта t1 і пов'язаний з ним операцією ".". Об'єкт t1 додається до об'єкта t2, який пере- дається до функції add_interv() як аргумент. Результат додавання зберігається в об'єкті temp класу Interval, повертається у функцію main() і об'єкту t3 присвоюється результат. Синтаксис передавання об'єктів до функції є такий самий, як і для змінних стандартних типів. Оскільки метод add_interv() є методом класу Interval, він має доступ до кожного поля кожного об'єкта класу Interval, використовуючи операцію ".", наприклад d2.hour. Коли всередині функції здійснюється звертання до полів hour і minute, це означає, що звертання відбувається до полів об'єкта t1, оскільки метод є викликаний об'єктом t1.

Отже, кожен виклик методу класу є неодмінно пов'язаний з конкретним об'єктом цього класу (винятком є виклик статичної функції). Метод може безпосередньо звертатись до будь-яких, відкритих і закритих, членів цього об'єкта і має непрямий (через операцію крапки) доступ до членів інших об'єктів свого класу; останні виступають в якості аргументів методу.

Деструктори. *Деструктор* – це особлива форма методу, який застосовується для звільнення пам'яті, зайнятої об'єктом. Деструктор за суттю є антиподом конструктора. Він викликається автоматично, коли об'єкт виходить з області видимості.

Ім'я деструктора розпочинається з тільди (~), безпосередньо за якою йде ім'я класу. Деструктор має такий формат:

~<ім'я класу>() {}

Деструктор не має аргументів і значення, яке повертається. Але він може виконувати деякі дії, наприклад, виведення остаточних значень елементів даних класу, що буває зручно при налагодженні програми. Якщо деструктор явно не визначено, компілятор автоматично створює порожній деструктор.

Деструктори працюють у зворотному напрямку відносно конструкторів. Конструктор викликається під час створення об'єкта, а деструктор - під час видалення. Шаблон створення деструктора:

```
class Some {
public:
        ~Some () {
        cout << "Робота з об'єктом завершена.";} };
```

Розміщувати деструктор у класі явно треба у разі, якщо об’єкт містить покажчики на пам’ять, яка виділяється динамічно, – інакше при знищенні об’єкта пам’ять, на яку посилались його поля-покажчики, не буде позначено як звільнену. Покажчик на деструктор визначити не можна. Деструктор не успадковується.

Важливо пам'ятати:

1. конструктор і деструктор завжди оголошуємо в розділі *public*;
2. при оголошенні конструктора тип повернення не вказується, в тому числі – void;
3. у деструктора так само немає типу повернення, деструктору не можна передавати ніяких параметрів;
4. ім'я класу і конструктора повинні бути ідентичними;
5. ім'я деструктора ідентично імені конструктора, але з приставкою ~;
6. у класі допустимо створювати декілька конструкторів, якщо це необхідно. Імена будуть однаковими, а компілятор буде їх розрізняти по переданим параметрами (перевантаження функцій). Якщо ми не передаємо в конструктор параметри, він вважається конструктором за замовчуванням;
7. у класі може бути оголошений лише один деструктор.

Успадкування класів. Упадкування класів дозволяє створювати похідні класи (класи спадкоємці), взявши за основу всі методи й елементи базового класу (класу батька). Таким чином, економиться маса часу на написання і налагодження коду нової програми. Об'єкти похідного класу вільно можуть використовувати все, що створено і налагоджено в базовому класі. При цьому ми можемо в похідний клас дописати необхідний код для удосконалення програми: додати нові поля, методи і т. д. Базовий клас залишиться недоторканим.

Приклад 2. Написати програму, в якій створити два класи: базовий – FirstClass і похідний від нього SecondClass.

Програмний код реалізації класу

```
#include <iostream>
using namespace std;
class FirstClass    // базовий клас
{
protected:          // специфікатор доступу до елементу value
   int value;
public:
   FirstClass()      {  value = 0;   }
   FirstClass(int x)  {  value = x;   }
   void show_value()  {  cout << value << endl;  }
};
class SecondClass : public FirstClass   // похідний клас
{
public:
        // конструктор класу SecondClass викликає конструктори класу
        // FirstClass
   SecondClass() : FirstClass() {}
```

```
    SecondClass(int inputS) : FirstClass(inputS) {}
    void ValueSqr() // Без специфікатора  protected не можна змінити value
    {
       value *= value;
    }
};
int main()
{   setlocale(LC_ALL, "rus");
    FirstClass F_object(3);     // об’єкт базового класу
    cout << "\n\tvalue F_object = ";
    F_object.show_value();
    SecondClass S_object(4);    // об’єкт похідного класу
    cout << "\tvalue S_object = ";
    S_object.show_value();      // виклик методу базового класу
    S_object.ValueSqr();        // підносимо value до квадрату
    cout << "\tквадрат value S_object = ";
    S_object.show_value();
    // F_object.ValueSqr();     // ПОМИЛКА: немає доступу
    cout << endl;
    return 0; }
```

Основна інформація про успадкування класів:

1. Успадкування – це визначення похідного класу, який може звертатися до всіх елементів і методів базового класу, за винятком тих, що перебувають у розділі *private*.
2. Похідний клас ще називають нащадком або підкласом, а базовий – батьківським, або надкласом, або суперкласом.
3. Синтаксис визначення похідного класу:
   class Імя_Похідн_Класу: специфікатор доступу Імя_Баз_Класу { . . . };
4. Похідний клас має доступ до всіх полів і методів базового класу, а базовий клас може використовувати тільки свої власні поля і методи.
5. У похідному класі необхідно явно визначати свої конструктори, деструктори і перевантажені оператори присвоювання через те, що вони не успадковуються від базового класу. Але їх можна викликати явним чином при визначенні конструктора, деструктора або перевантаження оператора присвоєння похідного класу, наприклад, таким чином (для конструктора): **Конструктор_Похідн_Класу (...): Конструктор_Баз_Класу (...) {...}.**

Приклади реалізації класів та об’єктів мовою С++

Конструктори та деструктори підвищують роботу з класами у мові програмування С++. Завдяки ним можна створювати об'єкти на основі класів та ініціалізувати змінні для них.

Конструктор класу зручний і за своїм виглядом схожий по структурі із користувацькими функціями. За допомогою конструктора можна встановити значення для об'єкта відразу під час його створення.

В одному класі може бути одразу декілька конструкторів. Створивши кілька конструкторів, ми можемо передавати різну кількість параметрів при

створенні об'єкта. Компілятор автоматично визначає який конструктор необхідно використовувати в залежності від переданих параметрів та їх типів даних. Конструктор повинен мати однакову назву із класом.

Приклад 3. Написати програму, в якій створити клас ***Book*** з використанням конструкторів. Додатково створити два об'єкти, які будуть виводити всю необхідну інформацію

Програмний код реалізації класу

```
#include <iostream>
using namespace std;
class Book {
private:
int pages;
float weight;
public:
char name;
void getInfoBook() {
cout << "У книзі " << name << " перебувати " << pages << " сторінок. " << endl;
cout << "При цьому вона важить" << weight << endl;
}
Book (int _pages, float _weight) {
pages = _pages;
weight = _weight;
}
Book (int _pages, float _weight, char _name) {
pages = _pages;
weight = _weight;
name = _name;
}
};
int main() {
Book sherlock_holms (460, 1.7f); // Використовуємо 1 конструктор
sherlock_holms.name = 'S'; // Встановлюємо вручну значення для nate
sherlock_holms.getInfoBook();
 Book green_mile(750, 2.8f, 'G'); // Використовуємо другий конструктор
green_mile.getInfoBook();
return 0;}
```

У даній програмі змінні використовуються разом із модифікатором доступу ***private***. Оскільки всі змінні додаються до методу ***private***, або ***protected***, то доступ до змінних повинен здійснюватись за рахунок методів та конструкторів. Доступ до полів повинен бути закритий.

Оскільки змінна ***name*** містить ***public*** модифікатор доступу, то можна посилатись безпосередньо до цієї змінної. Хоча, рекомендується встановити модифікатор private до змінної ***name***. Конструктор може приймати декілька параметрів, то під час створення об'єкта не потрібно додатково передавати властивості.

Приклад 4. Написати програму, в якій створити клас *Auto* з використанням конструкторів. Створити два об'єкти, які будуть виводити всю необхідну інформацію. Організувати деструктор до даного класу.

Програмний код реалізації класу

```
#include <iostream>
using namespace std;
class Auto {
private:
    int year, month, day;
public:
    Auto () {
        year = 2022;
        month = 12;
        day = 31;
       get();
        cout << endl;
    }
   Auto (int date_year, int date_month, int date_day) {
        year = date_year;
        month = date_month;
        day = date_day;

        get();
        cout << endl;
    }
    void message() {
        cout << "Class is working!" << endl;
    }
    void set (int date_year, int date_month, int date_day) {
        year = date_year;
        month = date_month;
        day = date_day;
    }
    void get() {
    cout << "Year of this auto is - " << year << ", month is - " << month << ", day is - " <<
        day << endl;
    }
   ~Auto () {
        cout << "Class is not working" << endl;
    }
};
int main() {
    Auto shkoda (2015, 2, 13);
    shkoda.message();
    shkoda.set (2005, 11, 23);
    shkoda.get();
    Auto bmw;
    bmw.set(2012, 1, 11);
    bmw.get();
    cin.get();
    return 0;
}
```

Приклад 5. Написати програму, в якій створити клас ***Tiles.*** У класі ***Tiles*** створити три конструктори:

- Конструктор, який встановлює лише висоту (size_h) та ширину (size_w);
- Конструктор, який встановлює всі змінні, крім змінної brand;
- Конструктор, який не встановлює нічого, але виводить повідомлення: «Empty constructor».

Створити три об'єкти на основі класу та використати всі організовані конструктори.

Програмний код реалізації класу

```
#include <iostream>
using namespace std;
class Tiles {
public:
char brand[32] = "Best Tile";
int size_h; // Розмір у висоту
int size_w; // Розмір завширшки
double price;
 Tiles(int _size_h, int _size_w) { // Встановлено лише два значення
 size_h = _size_h;
 size_w = _size_w;
 }
 // Встановлено три значення
 Tiles(int _size_h, int _size_w, double _price) {
 size_h = _size_h;
 size_w = _size_w;
 price = _price;
 }
  Tiles() { // Порожній конструктор
 cout << "Empty constructor!" << endl;
 }
void getData() {
cout << brand << ": " << endl;
cout << size_h << "x" << endl;
cout << size_w << " - " << price << "$";
}
};
 int main() {
  Tiles obj(4, 38); // Перший конструктор
  Tiles obj_2 (2, 59, 99.99); // Другий конструктор
  Tiles obj_3; // Третій конструктор
  obj.getData();
  cout << endl << endl;
  obj_2.getData();
  cout << endl << endl;
  obj_2.getData();
   return 0;
}
```

Приклад 6. За завданням попереднього прикладу створити клас ***Tiles*** з використанням конструкторів. Додати деструктор всередині класу. Деструктор

повинен виводити на консоль загальну кількість створених об'єктів на основі даного класу.

Програмний код реалізації класу

```
#include <iostream>
using namespace std;
// Створено змінну, яка не щоразу обнуляться, оскільки не належить ні до класу, ні до
  об'єктів
short int countObjects = 0;
class Tiles {
public:
char brand[32] = "Best Tile";
int size_h; // Розмір у висоту
int size_w; // Розмір завширшки
double price;
Tiles() { // Порожній конструктор
}
// Створено деструктор, який збільшує задану змінну і виводить результат
~Tiles() {
countObjects++;
cout << "Here is " << countObjects << " objects!" << endl;
}
};
int main() {
Tiles obj;
Tiles obj_2;
 return 0; }
```

Контрольні запитання

1. Дати визначення поняття конструктора та описати його призначення.
2. Описати процес ініціалізації даних-членів класу за допомогою конструкторів.
3. Охарактеризувати призначення конструкторів копіювання.
4. Описати порядок виклику конструкторів і деструкторів базових та похідних класів.
5. Дати визначення поняття деструктора та описати його призначення.
6. Що таке конструктори і деструктори класу? Для чого їх використовують?
7. Які особливості конструкторів у класах?
8. Які особливості деструкторів у класах?
9. Наведіть синтаксис опису успадкованого класу. Поясніть на прикладах.
10. Що Ви можете сказати про модифікатори доступу в успадкованих класах?
11. Як успадковуються конструктори у похідних класах?
12. Дати визначення поняття конструктора та описати його призначення.
13. Описати процес ініціалізації даних-членів класу за допомогою конструкторів.
14. Охарактеризувати призначення конструктори копіювання.
15. Описати порядок виклику конструкторів і деструкторів базових та похідних класів.
16. Дати визначення поняття деструктора та описати його призначення.

Практичне завдання 11.
Застосування дружніх функцій, дружніх класів, дружніх методів в мові програмування С++

Мета: отримання практичних навичок в особливостях використання функцій мовою програмування С++, навчитися передавати аргументи функції за значеннями, вказівниками, посиланнями та використовувати дружні функції, дружні класи, дружні методи.

Завдання 1. Мовою програмування С++ розробити програму для розрахунку прямокутної матриці $A=\{a_{ij}\}$ відповідно до варіанту (***табл. 32***). При реалізації програми необхідно дотримуватись наступних вимог:

- елементи матриці задаються користувачем безпосередньо з клавіатури;
- відсортувати матрицю одним із алгоритмів;
- для відсортованої матриці знайти значення функції $F(f_i(a_{ij}))$;
- алгоритм сортування матриці, обчислення $f_i(a_{ij})$, введення і виведення матриці оформити у вигляді функцій/дружніх функцій;
- програма повинна вивести на екран відсортовану матрицю, всі значення $f_i(a_{ij})$ та значення функції $F(f_i(a_{ij}))$:
- застосувати клас одномірного масиву, функції-члени, дружні функції, дружні класи, дружні методи

Таблиця 32. - Варіанти до виконання завдання 1

№ п/п	Алгоритм впорядкування матриці	Алгоритм для розрахунку $f_i(a_{ij})$ та $F(f_i(a_{ij}))$	Матриця
1	Впорядкувати елементи рядків матриці за спаданням їх значень методом вибору	$f_i(a_{ij})$-середнє геометричне значення елементів у кожному стовпці матриці; $F(f_i(a_{ij}))$-сума $f_i(a_{ij})$	-3 -5 –45 -71 -5 0 1 3 2 7 11 9 45 0 4 9 19 55 44 90 -3 -4 –1 -5 0
2	Впорядкувати елементи стовпців матриці за зростанням їх значень методом вставки	$f_i(a_{ij})$-середнє арифметичне значення елементів у кожному рядку матриці; $F(f_i(a_{ij}))$-добуток $f_i(a_{ij})$	40 72 6 92 98 18 -33 -48 81 26 1 -4 6 -2 0 36 9 0 4 1 -55 2 66 70 -3
3	Впорядкувати елементи рядків матриці за спаданням їх значень методом обміну	$f_i(a_{ij})$-добуток елементів у кожному стовпці матриці; $F(f_i(a_{ij}))$-середнє арифме-тичне значення $f_i(a_{ij})$	90 7 89 -2 17 1 -4 8 56 32 -4 -6 –99 19 39 2 4 -7 0 75 11 41 22 80 -5
4	Впорядкувати елементи стовпців матриці за зростанням їх значень методом вибору	$f_i(a_{ij})$-сума елементів у кожному рядку матриці; $F(f_i(a_{ij}))$-середнє геометричне значення $f_i(a_{ij})$	2 0 33 -1 -21 78 7 -4 -3 11 -2 -7 -1 -9 0 13 61 60 42 -10 1 0 4 0 16
5	Впорядкувати елементи рядків матриці за зростанням їх значень методом вставки	$f_i(a_{ij})$-мінімальний елемент у кожному стовпці матриці; $F(f_i(a_{ij}))$-добуток $f_i(a_{ij})$	34 -8 27 7 12 -5 23 45 67 -2 13 -12 34 -3 25 17 56 -6 17 21 0 15 4 9 -14

№ п/п	Алгоритм впорядкування матриці	Алгоритм для розрахунку $f_i(a_{ij})$ та $F(f_i(a_{ij}))$	Матриця
6	Впорядкувати елементи стовпців матриці за спаданням їх значень методом вставки	$f_i(a_{ij})$-максимальний елемент у кожному рядку матриці; $F(f_i(a_{ij}))$-сума $f_i(a_{ij})$.	-12 7 23 13 4 67 15 34 -5 9 2 5 17 -23 45 26 -6 23 -5 -9 18 37 -8 26 12
7	Впорядкувати елементи стовпців матриці за спаданням їх значень методом обміну	$f_i(a_{ij})$-середнє геометричне значення елементів в кожному рядку над головною діагоналлю матриці; $F(f_i(a_{ij}))$-сума $f_i(a_{ij})$	0 2 -2 89 21 -1 -4 36 41 71 56 93 51 -2 -51 1 3 -8 0 9 23 41 5 8 -2
8	Впорядкувати елементи рядків матриці за зростанням їх значень методом вибору	$f_i(a_{ij})$-середнє арифметичне значення елементів у кожному стовпці під голов-ною діагоналлю матриці; $F(f_i(a_{ij}))$-добуток $f_i(a_{ij})$	1 16 21 11 6 2 17 22 12 7 3 18 23 13 8 4 19 24 14 9 5 20 25 15 10
9	Впорядкувати елементи стовпців матриці за спаданням їх значень методом вставки	$f_i(a_{ij})$-добуток елементів у кожному рядку під допоміжною діагоналлю матриці; $F(f_i(a_{ij}))$-середнє арифметичне значення $f_i(a_{ij})$	-1 -5 -47 -8 -1 -4 -98 –90 -45 -78 -3 -2 -5 -9 -4 -8 -67 –33 -91 -40 -2 -58 –11 -65 -77
10	Впорядкувати елементи рядків матриці за зростанням їх значень методом вибору	$f_i(a_{ij})$-сума елементів у кожному стовпці над допоміжною діагоналлю матриці; $F(f_i(a_{ij}))$-середнє геометричне значення $f_i(a_{ij})$	44 -2 -5 38 –91 2 0 6 3 22 13 1 -4 90 11 -3 -6 -98 -23 -24 10 34 32 31 69
11	Впорядкувати елементи рядків матриці за спаданням їх значень методом обміну	$f_i(a_{ij})$-сума елементів у кожному стовпці над головною діагоналлю матриці; $F(f_i(a_{ij}))$-добуток $f_i(a_{ij})$	9 67 –65 45 1 12 61 48 -5 -1 0 39 0 41 2 36 95 -8 -5 0 11 22 71 3 63
12	Впорядкувати елементи стовпців матриці за зростанням їх значень методом вибору	$f_i(a_{ij})$-добуток елементів у кожному рядку під головною діагоналлю матриці; $F(f_i(a_{ij}))$-сума $f_i(a_{ij})$	6 34 12 70 -1 -7 97 80 99 -99 1 6 -3 2 -8 3 33 -1 0 -78 -3 -5 -8 -56 -23
13	Впорядкувати елементи рядків матриці за спаданням їх значень методом вибору	$f_i(a_{ij})$-середнє арифметичне значення елементів у кожному стовпці над допоміжною діагоналлю; $F(f_i(a_{ij}))$- добуток $f_i(a_{ij})$	33 -5 -9 -20 -11 0 -42 86 83 71 -6 -9 33 13 22 52 -5 -7 53 19 -3 98 72 68 0
14	Впорядкувати елементи стовпців матриці за зростанням їх значень методом вибору	$f_i(a_{ij})$-середнє геометричне значення елементів у кожному рядку матриці; $F(f_i(a_{ij}))$-середнє арифметичне значення $f_i(a_{ij})$	66 21 -3 -1 90 1 74 -2 80 -1 10 30 20 -50 91 2 4 5 81 0 33 69 -5 51 24
15	Впорядкувати елементи рядків матриці за спаданням їх значень методом вставки	$f_i(a_{ij})$-сума елементів у кожному стовпці під головною діагоналлю матриці; $F(f_i(a_{ij}))$-середнє геометричне значення $f_i(a_{ij})$	3 5 9 24 2 -23 0 37 29 10 0 1 4 -2 -5 -5 -83 –74 82 -1 11 88 -5 81 -39
16	Впорядкувати елементи стовпців матриці за зростанням їх значень методом обміну	$f_i(a_{ij})$-добуток елементів у кожному рядку над головною діагоналлю; $F(f_i(a_{ij}))$-середнє арифметичне значення $f_i(a_{ij})$	50 98 -4 85 -8 40 73 -2 -9 -19 1 6 73 21 0 0 25 2 -5 -3

№ п/п	Алгоритм впорядкування матриці	Алгоритм для розрахунку $f_i(a_{ij})$ та $F(f_i(a_{ij}))$	Матриця
17	Впорядкувати елементи рядків матриці за зростанням їх значень методом вибору	$f_i(a_{ij})$-сума елементів у кожному стовпці під допоміжною діагоналлю матриці; $F(f_i(a_{ij}))$ - середнє геометричне значення $f_i(a_{ij})$	87 98 57 29 95 -8 59 -2 9 -11 6 10 20 59 -23 12 13 51 46 -7 -2 87 69 90 -3
18	Впорядкувати елементи рядків матриці за спаданням їх значень методом вставки	$f_i(a_{ij})$-середнє арифметичне значення елементів у кожному стовпці над допоміжною діагоналлю матриці; $F(f_i(a_{ij}))$-добуток $f_i(a_{ij})$	12 46 –23 72 -5 59 7 -8 0 67 7 -8 -4 -97 -55 77 -1 -5 34 -8 0 22 27 24 24
19	Впорядкувати елементи стовпців матриці за спаданням їх значень методом обміну	$f_i(a_{ij})$-середнє геометричне значення елементів у кожному рядку під головною діагоналлю матриці; $F(f_i(a_{ij}))$-сума $f_i(a_{ij})$	9 24 -2 86 -3 40 49 -4 -3 0 27 -76 77 -1 69 71 -89 -94 -51 50 2 96 42 36 -1
20	Впорядкувати елементи стовпців матриці за зростанням їх значень методом вибору	$f_i(a_{ij})$-сума елементів у кожному рядку над головною діагоналлю матриці; $F(f_i(a_{ij}))$-середнє геометричне значення $f_i(a_{ij})$	31 65 –83 -2 -85 9 -2 11 -4 70 52 73 -8 -1 60 57 83 -1 82 50 1 -3 -2 78 -9
21	Впорядкувати елементи рядків матриці за спаданням їх значень методом вставки	$f_i(a_{ij})$-середнє арифметичне значення елементів у кожному стовпці матриці; $F(f_i(a_{ij}))$-середнє геометричне значення $f_i(a_{ij})$	30 31 36 63 -2 2 24 -3 -7 -1 45 28 -98 2 -8 0 -1 -2 -3 93 11 10 72 85 66
22	Впорядкувати елементи стовпців матриці за зростанням їх значень методом вибору	$f_i(a_{ij})$-добуток елементів у кожному рядку над допоміжною діагоналлю матриці; $F(f_i(a_{ij}))$-сума $f_i(a_{ij})$	22 41 45 -45 -49 5 1 3 -2 0 34 97 48 72 -1 -3 -7 5 92 20 0 -3 -57 9 1
23	Впорядкувати елементи рядків матриці за спаданням їх значень методом обміну	$f_i(a_{ij})$-добуток елементів у кожному стовпці під головною діагоналлю матриці; $F(f_i(a_{ij}))$-середнє арифметичне значення $f_i(a_{ij})$	19 62 –45 -1 84 23 54 -4 -2 68 36 39 96 94 97 -3 -8 -4 -6 -22 98 -5 -3 0 11
24	Впорядкувати елементи стовпців матриці за зростанням їх значень методом вибору	$f_i(a_{ij})$-середнє геометричне значення елементів у кожному рядку над головною діагоналлю; $F(f_i(a_{ij}))$-сума $f_i(a_{ij})$	34 45 65 23 98 1 -4 67 -3 -18 23 -5 -1 94 -25 2 24 –4 79 -63 10 29 25 30 -6
25	Впорядкувати елементи рядків матриці за спаданням їх значень методом вибору	$f_i(a_{ij})$-добуток елементів у кожному стовпці під допоміжною діагоналлю матриці; $F(f_i(a_{ij}))$-середнє арифметичне значення $f_i(a_{ij})$	10 32 1 -8 -1 2 4 91 -82 96 33 62 -1 -8 0 5 -5 6 -6 7 -19 0 3 -22 -3

Завдання 2. Створити клас довільного джерела даних відповідно до варіанту (***табл. 33***) та утворити від нього два похідних класи – сервера та клієнта (робочої станції).

У ***базовому класі*** описати дві захищені властивості: унікальний номер джерела даних і тип джерела (сервер чи клієнт). У конструкторі цього класу передбачити отримання унікального номеру для джерела даних, в залежності від поточного значення деякої глобальної змінної, та збільшення на одиницю

значення цієї змінної. Також потрібно створити у класі порожній віртуальний деструктор, абстрактну віртуальну функцію надання команди іншому джерелу та функцію отримання значення властивості "тип джерела". Формальний параметр функції надання команди – посилання на об'єкт цього класу.

Клас клієнта має містити наступні закриті властивості: номер блоку даних, які потрібно отримати від сервера; поле для збереження даних, отриманих від сервера (лінійний масив символів); вказівник на сервер, запит до якого буде надсилатися клієнтом. Крім цього, у такому класі мають бути створені наступні функції:

1) Закрита функція введення номеру блоку даних, які потрібно отримати від сервера. Передбачити аналіз ситуації, при якій вказаний користувачем номер не відповідає діапазону номерів блоків даних сервера.

2) Конструктор. Здійснити ініціалізацію усіх властивостей даного класу та властивості "тип джерела" базового класу надати відповідне значення.

3) Деструктор. Виведення на екран повідомлення про те, що клієнт з відповідним унікальним номером завершив роботу.

4) Перевантажена функція відсилання команди. Зберегти вказівник на об'єкт сервера, якому відсилається команда. Виконати аналіз того, чи дійсно отриманий вказівник є вказівником на сервер і, якщо так, то викликати за ним перевантажену функцію відсилання команди. Вивести повідомлення про результат отримання даних від сервера.

5) Виведення даних, отриманих клієнтом. У випадку, коли поле даних клієнта порожнє, вивести відповідне повідомлення.

Функції 2) – 5) – загальнодоступні.

У ***класі сервера*** описати такі закриті властивості: поточна кількість рядків у базі сервера; база сервера (двомірний масив символів, кожен рядок якого – це блок даних, що може бути переданий клієнтові); вказівник на клієнта, запит якого виконує сервер. У цьому класі мають бути створені такі функції:

1) Закрита функція завантаження бази сервера з текстового файлу. Як параметр цієї функції, використовується ім'я файлу. Кожен рядок файлу – це окремий блок даних. Передбачити випадок, коли ім'я файлу вказане не коректно.

2) Конструктор. Як параметр, конструктору передається ім'я файлу з базою. Вивести повідомлення про результат завантаження даних з бази, ініціалізувати властивості класу сервера та встановити відповідне значення для властивості "тип джерела" базового класу.

3) Деструктор. Виведення на екран повідомлення про те, що сервер з відповідним унікальним номером завершив роботу.

4) Перевантажена функція відсилання команди. Зберегти вказівник на об'єкт клієнта, команда якого виконується. Якщо отриманий вказівник дійсно є вказівником на об'єкт клієнта, то за ним викликати функцію отримання номеру блоку даних сервера та передати ці дані клієнтові. Відповідним текстовим повідомленням інформувати користувача про результат виконання команди.

Функції 2) – 4) – загальнодоступні.

Описати глобальну функцію, котрій, як параметри, передаються: кількість елементів лінійного масиву вказівників на об’єкти базового класу джерела даних та цей масив; властивість логічного типу – ознака об’єктів (істина – сервер, похибка – клієнт). Результат роботи функції – виведення на екран номерів тих об’єктів, які відповідають вказаному значенню властивості логічного типу.

У головній програмі запропонувати користувачу створити динамічний масив вказівників на об’єкти базового класу та за цими вказівниками, у залежності від вибору користувача, створити об’єкти серверів та клієнтів.

Налаштувати інтерфейс програми таким чином, щоб користувач міг обрати тип дії: отримання даних від сервера або виведення на екран даних, отриманих клієнтом. Якщо користувач обирає тип дії "отримання даних від сервера", то вивести на екран номери усіх серверів та клієнтів і запропонувати обрати з них певного клієнта і сервер. Потім здійснити аналіз коректності вказаних користувачем номерів та виконати команду відсилання даних від обраного сервера до обраного клієнта. Якщо ж обрано тип дії "виведення на екран даних, отриманих клієнтом", то вивести на екран номери клієнтів, запропонувати обрати одного з них та, у випадку коректного номера, вивести дані, що містить клієнт. У подальшому пропонувати користувачу або завершити роботу програми, або знову запропонувати обрати тип дії.

Таблиця 33. - Варіанти до виконання завдання 2

№	Предметна область	Реквізити об’єкту	Параметр сортування	Параметр пошуку
1.	Архів програм	Назва програми, операційна система, розмір програми, дата запису	Назва програми	Операційна система
2.	Бібліотека	інвентарний номер, автор, назва, кількість сторінок, рік видання	Рік видання	Автор
3.	Записна книжка	Прізвище, ім'я, домашня адреса, телефон, електронна пошта.	Прізвище	Електронна пошта
4.	Камера схову	Прізвище, дата здачі, термін зберігання, інвентарний номер та назва предмета	Інвентарний номер	Дата здачі
5.	Каса продажу квитків	Назва пункту, час відправлення, дата відправлення, час прибуття, дата прибуття, ціна квитка	Час від-правлення	Назва пункту
6.	Колекція компакт-дисків	Інвентарний номер, назва альбому, об'єм диску, тип, дата запису.	Дата запису	Назва альбому
7.	Користувачі локальної мережі	Прізвище, група, обліковий запис, тип облікового запису.	Тип облікового запису	Прізвище
8.	Предметний покажчик	Слово; номера сторінок, де це слово зустрічається; кількість цих слів на даній сторінці	Номер сторінки	Слово

№	Предметна область	Реквізити об’єкту	Параметр сортування	Параметр пошуку
9.	Рахунки банку	Прізвище, ім'я, дата останньої операції, сума вкладу	Сума вкладу	Дата операції
10.	Розклад пар	Номер пари, предмет, прізвище викладача, форма заняття.	Предмет	Номер пари
11.	Розклад руху літаків	Номер рейсу, тип літака, напрямок руху, періодичність вильоту.	Номер рейсу	Тип літака
12.	Склад товарів	інвентарний номер, назва товару, вага, ціна, кількість	Вага	Назва товару
13.	Список файлів	ім'я файла, розширення, розмір, дата створення, атрибути.	Розширення	Дата створення
14.	Телефонний довідник	Прізвище, ім'я, по батькові, домашня адреса, телефон.	Телефон	Прізвище
15.	Успішність студентів	Прізвище, номер групи, оцінки з трьох предметів	Прізвище	Номер групи

Зміст звіту

1. Титульна сторінка, тема та мета практичного заняття.
2. Підготувати звіт по виконаній роботі, у якому представити наступні матеріали:
   - ✓ зміст завдання і варіант;
   - ✓ лістинг програми.
   - ✓ схему ієрархії класів програмного результати виконання – вигляд екрану при виконанні програми;
   - ✓ результати роботи розробленого програмного засобу;
3. Висновки, які відображають результати виконаних завдань та їх аналіз.

Теоретичні відомості

Відомо, що дані вашого класу повинні бути **private**. Однак може виникнути ситуація, коли є **клас** і функція, яка працює з цим класом, але не знаходиться у його тілі. Наприклад, є клас, в якому зберігаються дані, і функція (або інший клас), яка виводить ці дані на екран. Хоча код класу і код функції виводу розділені (для спрощення підтримки коду), код функції виводу тісно пов’язаний з даними класу. Отже, зробивши члени класу private, бажаного ефекту не отримано. В таких ситуаціях є два варіанти:

1. Зробити відкритими методи класу і через них функція взаємодіятиме з класом. Однак тут є кілька нюансів.
   - ➢ По-перше, ці відкриті методи необхідно буде визначити, на що знадобиться виділити час, і вони будуть захаращувати інтерфейс класу.
   - ➢ По-друге, в класі потрібно буде відкрити методи, які не завжди повинні бути відкритими і надавати доступ зовні.

2. Використати дружні класи і дружні функції, за допомогою яких можна буде надати функції виводу доступ до закритих даних класу.

Це дозволить функції виводу безпосередньо звертатися до всіх закритих змінних-членів і методів класу, зберігаючи при цьому закритий доступ до даних класу для всіх інших функцій поза тілом класу.

Дружні функції. Мова С++ має такі поняття як дружні функції, а також дружні класи. ***Дружня функція*** - це функція, що має доступ до полів та методів класу, при цьому сама функція може бути створена поза класом.

Для створення дружньої функції необхідно прописати звичайну функцію поза класами і далі для класів, що працюватимуть з цією функцією оголосити її, вказавши ім'я, а також поставивши оператор ***friend*** перед типом даних функції.

Після створення такої функції можна через неї керувати всіма даними у всіх класах, які вказані в дружній функції. Доступ буде навіть до тих полів, у яких модифікатор доступу стоїть як ***private*** чи ***protected***.

Приклад 1. Створити клас для визначення суми двох елементів. В створеному класі застосувати дружню функцію.

Програмний код реалізації прикладу

```
#include <iostream>
using namespace std;
class b;
class a {
  friend int sum(a, b);
private:
  int i;
public:
  a() {
    cout << "Enter i: ";
    cin >> i;
  }
};
class b {
  friend int sum(a, b);
private:
  int y;
public:
  b() {
    cout << "Enter y: ";
    cin >> y;
  }
};
int sum(a first, b second) {
  return (first.i + second.y);
}
int main() {
    a first;
    b second;
    cout << sum(first, second) << endl;
    cin.get();
    return 0;}
```

Приклад 2. Створити клас ***Auto.*** В створеному класі застосувати дружню функцію. Додати до класу:

- ***поля***: name, price;
- ***конструктор***, який встановлює всі значення;
- ***дружні методи*** drive та setPrice.

Метод ***drive*** приймає об'єкт як посилання та виводить його назву в консоль. Метод ***setPrice*** приймає об'єкт як посилання та ціну, після чого встановлювати ціну для об'єкта, якщо ціна більша за 0. У головній функції ***main*** створити об'єкт та застосувати реалізовані дружні функції.

Програмний код реалізації прикладу

```
#include <iostream>
#include <string>
using namespace std;
class Auto {
friend void drive(Auto &);
friend void setPrice(Auto &, int price);
private:
string name; // назва авто
int price; // ціна авто
public:
Auto(string autoName, int autoPrice) {
        name = autoName;
        price = autoPrice;
}
 string getName() { return name; }
int getPrice() { return price; }
};
void drive(Auto &a) {
cout << a.name << " is driven" << endl;
}
void setPrice(Auto &a, int price) {
if (price > 0)
        a.price = price;
}
int main() {
Auto tesla("Tesla", 5000);
drive(tesla);
cout << tesla.getName() << " : " << tesla.getPrice() << endl;
setPrice(tesla, 8000);
cout << tesla.getName() << " : " << tesla.getPrice() << endl;
 return 0; }
```

Тут визначено клас ***Auto***, який репрезентує автомобіль. Цей клас визначає приватні закриті змінні ***name*** (назву автомобіля) і ***price*** (ціна автомобіля). Також у класі оголошено дві дружні функції: ***drive*** (функція водіння автомобіля) та ***setPrice*** (функція призначення ціни). Обидві ці функції приймають як параметр посилання на об'єкт ***Auto***.

Коли оголошені дружні функції, то фактично компілятор визначає, що це друзі класу і вони мають доступ до всіх членів цього класу, зокрема

закритих. При цьому для дружніх функцій не важливо, чи визначаються вони під специфікатором ***public*** або ***private***. Для них це не має значення.

Визначення цих функцій виконується поза класом. І оскільки ці функції є дружніми, то всередині цих функцій ми можемо через передане посилання ***Auto*** звернутися до всіх закритих змінних.

Дружня функція — це функція, яка має доступ до закритих членів класу, наче вона сама є членом цього класу. У всіх інших аспектах дружня функція є звичайною функцією. Нею може бути, як звичайна функція, так і метод іншого класу. Для оголошення дружньої функції використовується ***ключове слово friend*** перед ***прототипом функції***, яку ви хочете зробити дружньою класу. Неважливо, чи оголошується в ***public***- чи в ***private***-зоні класу.

Приклад 3. Створити клас ***Anything*** з використанням дружніх функцій для визначення температурного режиму, вологості**.**

Програмний код реалізації прикладу

```
#include <iostream>
using namespace std;
class Anything
{
private:
   int m_value;
public:
   Anything() { m_value = 0; }
   void add(int value) { m_value += value; }
    // створюємо функцію reset() яка буде дружньою до класу Anything
   friend void reset(Anything &anything);
};
// Функція reset() тепер є дружньою з класом Anything
void reset(Anything &anything)
{
   // відкрито доступ до закритих членів об'єктів класу Anything
   anything.m_value = 0;
}
int main()
{
   Anything one;
   one.add(4); // додаємо 4 до m_value
   reset(one); // обнуляємо значення m_value в 0
   return 0;}
```

В даному прикладі оголошена функцію ***reset***(), яка приймає об’єкт класу ***Anything*** і встановлює ***m_value*** значення 0. Оскільки ***reset***() не є членом класу Anything, то в звичайній ситуації функція reset() не мала б доступу до закритих членів Anything. Однак, оскільки ця функція є дружньою класу Anything, вона має доступ до закритих членів Anything.

Варто звернути увагу, що повинен передаватись об’єкт ***Anything*** в функцію ***reset***() в якості параметра. Це пов’язано з тим, що функція reset() не

є методом класу. Вона не має ***вказівника *this*** і, вона не зможе взаємодіяти з класом, а тільки передавати об’єкт. Наприклад:

Програмний код реалізації прикладу

```
#include <iostream>
using namespace std;
class Something
{
private:
    int m_value;
public:
    Something(int value) { m_value = value; }
    friend bool isEqual(const Something &value1, const Something &value2);
};
 bool isEqual(const Something &value1, const Something &value2)
{
    return (value1.m_value == value2.m_value);
}
 int main()
{
    return 0;}
```

Тут оголошена функція ***isEqual***() дружньою класу ***Something***. Функція ***isEqual***() приймає в якості параметрів два об’єкти класу ***Something***. Оскільки isEqual() є другом класу ***Something***, то функція має доступ до всіх закритих членів об’єктів класу ***Something***. Функція ***isEqual***() порівнює значення змінних-членів двох об’єктів і повертає ***true***, якщо вони рівні.

Визначення дружніх функцій у класі. Дружні функції можуть визначатися в іншому класі. Наприклад, визначимо клас ***Person***, який використовує об'єкт ***Auto***:

Програмний код реалізації прикладу

```
#include <iostream>
#include <string>
using namespace std;

class Auto;
 class Person
{
public:
    Person(string n)
    {
        name = n;
    }
    void drive(Auto &a);
    void setPrice(Auto &a, int price);
private:
    string name;
};
class Auto
{
    friend void Person::drive(Auto &);
```

```
    friend void Person::setPrice(Auto &, int price);
public:
    Auto(string autoName, int autoPrice)
    {
        name = autoName;
        price = autoPrice;
    }
    string getName() { return name; }
    int getPrice() { return price; }
private:
    string name;   // назва автомобіля
    int price;  // ціна автомобіля
};
void Person::drive(Auto &a)
{
    cout << name << " drives " << a.name << endl;
}
void Person::setPrice(Auto &a, int price)
{
    if (price > 0)
        a.price = price;
}
int main()
{
    Auto tesla("Tesla", 5000);
    Person tom("Tom");
    tom.drive(tesla);
    tom.setPrice(tesla, 8000);
    cout << tesla.getName() << " : " << tesla.getPrice() << endl;
     return 0;}
```

Спочатку визначено клас ***Person***, який представляє людину. Однак оскільки клас ***Person*** використовує клас ***Auto***, перед класом ***Person*** оголошується клас ***Auto***.

Дві функції класу ***Person*** приймають посилання на об'єкт ***Auto***:

void drive(Auto &a);

void setPrice(Auto &a, int price);

Тобто, фігурально користувач водить автомобіль і призначає йому ціну за допомогою цих функцій.

Клас Auto визначає дружні функції з тією самою сигнатурою:

friend void Person::drive(Auto &);

friend void Person::setPrice(Auto &, int price);

Причому оскільки ці функції будуть визначені у класі Person, то назви цих функцій передує префіксу ***Person::***. Оскільки в цих функціях передбачається використовувати об'єкт ***Auto***, то на час визначення цих функцій всі члени об'єкта ***Auto*** повинні бути відомі, тому визначення функцій знаходяться не в самому класі ***Person***, а після ***Auto*** класу. І оскільки ці функції визначені у класі Auto як дружні, ми можемо звернутися до цих функцій до закритих членів класу Auto.

Дружні функції і кілька класів. Функція може бути другом відразу для кількох класів. Розглянемо наступний приклад.

Програмний код реалізації прикладу

```
#include <iostream>
using namespace std;
class Humidity;
 class Temperature
{
private:
    int m_temp;
public:
    Temperature(int temp=0) { m_temp = temp; }
     friend void outWeather(const Temperature &temperature, const Humidity
&humidity);
};
 class Humidity
{
private:
    int m_humidity;
public:
    Humidity(int humidity=0) { m_humidity = humidity; }
     friend void outWeather(const Temperature &temperature, const Humidity
&humidity);
};

void outWeather(const Temperature &temperature, const Humidity &humidity)
{
    cout << "Температура " << temperature.m_temp <<
       " Вологість " << humidity.m_humidity << '\n';
}
int main()
{
    Temperature temp(15);
    Humidity hum(11);
     outWeather(temp, hum);
     return 0;}
```

Тут є дві речі, на які слід звернути увагу. *По-перше*, оскільки функція outWeather() є другом для обох класів, то вона має доступ до закритих членів обох класів. *По-друге*, зверніть увагу на наступний рядок у вищенаведеному прикладі: ***class Humidity;***

Це прототип класу, який повідомляє компілятору, що визначається клас ***Humidity***. Без цього рядка компілятор видав би помилку, незнаючи, що таке ***Humidity*** при аналізі прототипу дружньої функції outWeather() всередині класу Temperature. Прототипи класів виконують ту ж роль, що і прототипи функцій: вони повідомляють компілятору про об’єкти, які пізніше будуть визначені, але які зараз потрібно використовувати.

Однак, на відміну від функцій, класи не мають типу повернення або параметрів, тому їх прототипи лаконічні: ***class ім'я класу***; (наприклад, class Anything).

Дружні класи. Один клас може бути дружнім іншому класу. Це відкриє всім членам першого класу доступ до закритих членів другого класу, наприклад:

Програмний код реалізації прикладу

```
#include <iostream>
using namespace std;
class Values
{
private:
    int m_intValue;
    double m_dValue;
public:
    Values(int intValue, double dValue)
    {
        m_intValue = intValue;
        m_dValue = dValue;
    }
     // зробимо клас Display другом класу Values
    friend class Display;
};
class Display
{
private:
    bool m_displayIntFirst;
public:
    Display(bool displayIntFirst) { m_displayIntFirst = displayIntFirst; }
     void displayItem(Values &value)
    {
        if (m_displayIntFirst)
            std::cout << value.m_intValue << " " << value.m_dValue << '\n';
        else // або спочатку виводимо double
            std::cout << value.m_dValue << " " << value.m_intValue << '\n';
    }
};
int main()
{
    Values value(7, 8.4);
    Display display(false);
     display.displayItem(value);
     return 0; }
```

Оскільки клас Display є другом класу Values, то будь-який з членів Display має доступ до private-членів Values. В результаті виконання програми маємо: ***8.4 7***

***Примітки про дружні класи*:**

- ***По-перше***, навіть незважаючи на те, що Display є другом Values, Display не має прямого доступу до вказівника *this об'єктів Values.
- ***По-друге***, навіть якщо Display є другом Values, це не означає, що Values також є другом Display. Якщо ви хочете зробити обидва класи дружніми, то кожен з них повинен вказати в якості друга протилежний клас. Нарешті, якщо клас A є другом B, а B є другом

С, то це не означає, що А є другом С.

Будьте уважні при використанні дружніх функцій і класів, оскільки це може порушувати принципи ***інкапсуляції***. Якщо деталі одного класу зміняться, то деталі класу-друга також будуть змушені змінитися. Тому обмежуйте кількість і використання дружніх функцій і класів.

Розглянувши роботу з дружніми функціями, перейдемо до дружніх класів. Навчимося створювати дружні класи (***friend classes***). Дружні класи зі своєї роботи схожі з дружніми функціями. Єдина відмінність полягає в об'єднанні не просто функцій, а цілих класів.

Завдяки дружній властивості ми можемо вказати кілька класів, що пов'язані між собою. Це дозволить брати дані через об'єкт одразу з кількох класів. За рахунок цього ми можемо точно описати логіку для різних персонажів у грі або для вікон у додатку. Всі вони можуть бути записані в різних класах, але при цьому кожен матиме доступ один до одного.

Приклад створення дружнього класу:

```
class Auto;
class Motorcycle {
friend class Auto;
private:
bool isStillWorking = true;
};
class Auto {
public:
void CrashDTP(Auto &moto) {
        moto.isStillWorking = false;
        cout << "Motorcycle is currently not working!"; } };
```

З прикладу видно, що з дружніх класів використовується така структура що з дружніх функцій. Перед оголошенням класу ***Motorcycle*** вказується існування класу Auto: ***class Auto***. Далі зробимо клас ***Auto*** дружнім та для цього використовуємо ключове слово ***friend***.

В даному прикладі, клас ***Person*** використовує тільки дві функції з класу ***Auto***. Але згодом виникла необхідність додати в клас Auto ще ряд дружніх функцій, які будуть визначені в класі Person. Або ми можемо припускати, що клас Person активно використовуватиме об'єкти Auto. І в цьому випадку доцільно визначати не окремі дружні функції, а визначити дружнім весь клас. Наприклад:

Програмний код реалізації прикладу

```
#include <iostream>
#include <string>
using namespace std;
class Auto;
class Person
{
public:
   Person(string n)
   {
```

```
        name = n;
    }
    void drive(Auto &a);
    void setPrice(Auto &a, int price);
 private:
    string name;
};
class Auto
{
    friend class Person;
public:
    Auto(string autoName, int autoPrice)
    {
        name = autoName;
        price = autoPrice;
    }
    string getName() { return name; }
    int getPrice() { return price; }
private:
    string name;   // назва авто
    int price;  // ціна авто
};
void Person::drive(Auto &a)
{
    cout << name << " drives " << a.name << std::endl;
}
void Person::setPrice(Auto &a, int price)
{
    if (price > 0)
        a.price = price;
}
int main()
{
    Auto tesla("Tesla", 5000);
    Person tom("Tom");
    tom.drive(tesla);
    tom.setPrice(tesla, 8000);
    cout << tesla.getName() << " : " << tesla.getPrice() << endl;
     return 0; }
```

Єдине, що в даному випадку змінилося порівняно з попереднім прикладом те, що в класі Auto визначення дружніх функцій було замінено визначенням дружнього класу: ***friend class Person;***

Тобто тим самим ми знову ж таки говоримо, що клас ***Person*** - це друг класу ***Auto***, тому об'єкти Person можуть звертатися до приватних змінних класу Auto. Після цього в класі Person можна звертатися до закритих членів Auto з будь-яких функцій.

Приклад 4. Створити клас для ігрового персонажу ***Dog*** та реалізуйте дружній клас для гравців ***Person***.

Програмний код реалізації прикладу

```
#include <iostream>
```

```
using namespace std;
class Person;
class Dog {
  friend class Person;
private:
  int health = 100;
};
class Person {
public:
  void Damage(Dog &d) {
    d.health -= 20;
    cout << "Health of the animal is " << d.health << endl;
  }
    void Kill(Dog &d) {
    d.health = 0;
    cout << "Health of the animal is " << d.health << endl;
  }
  void Heal(Dog &d) {
    d.health += 30;
    cout << "Health of the animal is " << d.health << endl;
  }
};
int main() {
    Dog skuby;
    Person Volodya;
    Volodya.Damage(skuby);
    Volodya.Kill(skuby);
    Volodya.Damage(skuby);
    Volodya.Heal(skuby);
    cout << endl;
    Dog haski;
    Volodya.Damage(haski);
    cin.get();
    return 0; }
```

Приклад 5. Створіть два класи ***Enemy*** та ***Player***. Додайте до кожного з них змінну ***health***, що відповідає за рівень життя.

Створіть метод, що приймає об'єкт як параметр і забирає об'єкт іншого класу -10 життів.

У головній функції створіть два об'єкти та викличте метод, що забирає життя для обох об'єктів.

Програмний код реалізації прикладу

```
#include <iostream>
using namespace std;
class Player; // Ідентифікатор класу
class Enemy { // Клас ворогів
friend class Player; // Вказуємо дружній клас
private:
int health = 100; // Зміна рівня життя
public:
// Створюємо лише скелет методу, оскільки клас
// Player ще не створено, то ми не можемо працювати з його
```

```
// Елементами, тому ми реалізуємо цю функцію після
// Реалізації самого класу Player
void Uron (Player & pl);
};
class Player { // Клас гравця
friend class Enemy; // Дружній клас Ворог
private:
int health = 100; // Змінне життя
public:
// Оскільки клас Enemy вже реалізований, ми можемо
// Реалзувати метод у цьому класі відразу ж
void Uron (Enemy &e) {
e.health -= 10; // Забираємо життя та виводимо інформацію
cout << "Enemy now has " << e.health << " lifes." << endl;
}
};
// Реалізовуємо функцію Uron для класу Enemy
void Enemy::Uron(Player &pl) {
pl.health -= 10;
cout << "Player now has " << pl.health << " lifes." << endl;
}
int main() {
// Створюємо класи та викликаємо методи
Enemy monstr;
Player main_player;
monstr.Uron(main_player);
main_player.Uron(monstr);
cin.get();
return 0; }
```

Контрольні запитання

1. Що таке функція? Види функцій.
2. Параметри та аргументи функції.
3. Що таке прототип?
4. Способи оголошення функцій.
5. Що таке визначення функції?
6. Яка різниця між локальними та глобальними змінними?
7. Де вказується тип значення, яке повертає функція?
8. Що таке перевантаження функції?
9. Що таке вбудовані функції?
10. Яка різниця між вказівником та посиланням?
11. Яка різниця між передаванням функції аргументів за значеннями, вказівниками і посиланнями?
12. Що таке дружні функції?

Практичне завдання 12.
Шаблони функцій і класів в мові програмування С++.
Параметризовані контейнерні класи бібліотеки STL

Мета: ознайомитися із базовими механізмами використання шаблонів функцій; навчитись створювати та використовувати шаблони класів; засвоїти правила створення та використання параметризованих контейнерних класів; дослідити основні класи з бібліотеки STL.

Завдання 1. В мові програмування С++ створити клас відповідно до варіанту (***табл. 34***), який описує та забезпечує дії над даними параметризованого масиву, розмірність якого визначається під час роботи програми.

Таблиця 34. - Варіанти до виконання завдання 1

№ вар.	*Завдання до виконання*
1	В масиві обчислити: ➢ номер елемента масиву, найближчого до середньоарифметичного його значень; ➢ різниця елементів масиву, що розташовані між першим від'ємним та другим додатним елементами. Перетворити масив так, щоб у його першій половині розташовувались елементи, що стоять в парних позиціях, а в другій – елементи, що стоять в непарних позиціях.
2	Дана прямокутна матриця. Визначити: – кількість від'ємних елементів в тих рядках, які містять хоча б один нульовий елемент; – суму модулів елементів, які розташовані після першого додатного елемента Впорядкувати елементи матриці за спаданням модулів елементів
3	У довільній матриці обчислити: – кількість елементів масиву, рівних нулю; – суму елементів масиву, які лежать в діапазоні від A до B. Впорядкувати елементи масиву за спаданням модулів елементів
4	У довільній матриці обчислити: – кількість елементів масиву, рівних нулю; – суму елементів масиву, які лежать в діапазоні від A до B. Впорядкувати елементи масиву за спаданням модулів елементів
5	В одновимірному масиві елементів, обчислити: – номер максимального за модулем елемента; – суму модулів елементів, які розташовані після першого додатного елемента. Перетворити масив таким чином, щоб спочатку розташовувались всі елементи, ціла частина яких лежить в інтервалі [a,b], а потім – всі інші
6	В масиві обчислити: – мінімальний за модулем елемент масиву; – суму модулів елементів, які розташовані після першого від'ємного елемента. Ущільнити масив, видаливши з нього елементи, величина яких знаходиться на інтервалі [a,b]. Місце, яке звільниться в кінці масиву заповнити символом чи числом з клавіатури.

№ вар.	*Завдання до виконання*
7	В масиві обчислити: – мінімальний за модулем елемент масиву; – суму модулів елементів масиву, розташованих після першого нільового елемента. Перетворити масив так, щоб в першій його половині розташовувались елементи, що стоять на парних позиціях, а в другій – елементи, що стоять в непарних позиціях.
8	У матриці обчислити: – максимальний за модулем елемент масиву; – суму елементів масиву, що розташовані між першим і другим додатними елементами. Перетворити матрицю так, щоб всі елементи, рівні нулю, розташовувались в кінці.
9	Дана прямокутна матриця. Визначити: – кількість рядків, які не містять жодного нульового елемента; – максимальне із чисел, що зустрічається в заданій матриці більше одного разу Перетворити матрицю так, щоб всі нульові елементи розташовувались на початку.
10	Дана прямокутна матриця. Визначити: – кількість стовпців, які не містять жодного нульового елемента. – кількість елементів, менших за A , але більших за B. Переставляючи рядки заданої матриці, розташувати їх у відповідності із зростанням суми значень у стовпцях.
11	В одномірному масиві обчислити: – добуток елементів масиву з парними номерами; – суму елементів масиву, які розташовані між першим і останнім нульовими елементами. Впорядкувати масив таким чином, щоб спочатку розташовувались всі додатні елементи, а потім – всі від'ємні (елементи, рівні 0 вважати додатними).
12	Дана прямокутна матриця. Визначити : – кількість стовпців, які містять хоча б один нульовий елемент; – номер рядка, в якому знаходиться найдовша серія з однакових елементів. Впорядкувати масив таким чином, щоб спочатку розташовувались всі серії з однакових елементів, а потім – всі решта елементів.
13	В одномірному масиві, що складається з N дійсних елементів, обчислити: – суму елементів масиву з непарними елементами; – суму елементів масиву, які розташовані між першим і останнім від'ємними елементами. Перетворити масив, видаливши з нього всі елементи, модуль яких не перевищує число, що вводиться з клавіатури. Елементи, які звільняться в кінці масиву заповнити нулями.
14	Дана прямокутна матриця. Визначити : – добуток елементів в тих рядках, які не містять від'ємних елементів; – максимум серед сум елементів діагоналей, паралельних головній діагоналі матриці. Перетворити матрицю, видаливши з неї всі елементи, модуль яких не перевищує число, що вводиться з клавіатури. Елементи, які звільняться в кінці масиву, заповнити нулями.

№ вар.	*Завдання до виконання*
15	В одномірному масиві, що складається з N дійсних елементів, обчислити: – максимальний елемент масиву; – суму елементів масиву, що розташовані до останнього додатного елемента. Перетворити масив, видаливши з нього всі елементи, модуль яких знаходиться в інтервалі [a,b]. Елементи, які звільняються в кінці масиву заповнити нулями.

Зміст звіту

1. Титульна сторінка, тема та мета практичного заняття.
2. Підготувати звіт по виконаній роботі, у якому представити наступні матеріали:
   - ✓ зміст завдання і варіант;
   - ✓ лістинг програми.
   - ✓ схему ієрархії класів програмного результати виконання – вигляд екрану при виконанні програми;
   - ✓ результати роботи розробленого програмного засобу;
3. Висновки, які відображають результати виконаних завдань та їх аналіз.

Теоретичні відомості

Шаблони функцій (template)

Шаблони функцій – це потужний інструмент у C++, який суттєво спрощує роботу програміста. Наприклад, нам потрібно запрограмувати функцію, яка виводила б на екран елементи масиву. При цьому ми хочемо, щоб функція виводила масиви типу *int, double, float* і *char*. Тобто, нам потрібно запрограмувати 4 функції, які виконують одні й ті самі дії, але для різних типів даних. Скористаємося перевантаженням функцій.

Шаблони значно скорочують код, а також є найважливішими конструкціями мови C++. Шаблони функцій, і навіть шаблони класів у мові C++ це потужні технології, що дозволяють створювати шаблонні конструкції і далі передачі даних як значення, і типи даних.

Для створення шаблонної функції необхідно використовувати ключове слово template, а також у кутових дужках необхідно вказувати тип даних, що приймається. Можна приймати один або кілька типів даних.

Шаблони функцій *– це інструкції, згідно з якими створюються локальні версії функції для певного набору параметрів і типів даних.*

Синтаксис:

template <class T>
template <typename T>
template <typename T1, typename T2>

Всі шаблони функцій починаються зі слова *template*, після якого йдуть кутові дужки, в яких перераховується список параметрів. Кожному параметру має передувати зарезервоване слово *class* або *typename*.

Ключове слово *typename* говорить про те, що у шаблоні буде використовуватися вбудований тип даних, такий як: *int*, double, float, *char* і т. д.

А ключове слово *class* повідомляє компілятору, що в шаблоні функції як параметр будуть використовуватися типи даних користувача, тобто класи.

Ми створюємо один шаблон, в якому описуємо всі типи даних. Таким чином код не буде захаращуватися.

Приклад 1. Використовуючи шаблон функції, вивести на екран елементи масивів за різними типами даних.

Програмний код реалізації прикладу

```
#include <iostream>
using namespace std;
   // шаблон функції printArray
template <typename T>
void printArray(const T * array, int count)
{   for (int ix = 0; ix < count; ix++)
       cout << array[ix] << "   ";
   cout << endl;
}
int main()
{   const int iSize = 10, dSize = 7, fSize = 10, cSize = 5;
       // масиви різних типів даних
   int    iArray[iSize]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
   double dArray[dSize]={1.2345,2.234,3.57,4.67876,5.346,6.1545,7.7682};
   float  fArray[fSize]={1.34,2.37,3.23,4.8,5.879,6.345,73.434,8.82,9.33,10.4};
   char   cArray[cSize] = {"MARS"};
   cout << "\n\t Використання шаблонів функцій:\n";
   cout << "\n\tМасив типу int:\n\t";  printArray(iArray, iSize);
   cout << "\n\tМасив типу double:\n\t"; printArray(dArray, dSize);
   cout << "\n\tМасив типу float:\n\t"; printArray(fArray, fSize);
   cout << "\n\tМасив типу char:\n\t"; printArray(cArray, cSize);
   system("pause");
   return 0;
}
```

Приклад 2. Створити шаблонну функцію, яка виводить на екран масиви різних типів даних та відповідну їх кількість елементі.

Програмний код реалізації прикладу

```
#include <iostream>
using namespace std;
template <typename T1, typename T2>
T2 printArr (const T1 * array, int i) {
    int count = 0;
    for (int j = 0; j < i; j++) {
       cout << array[j] << " ";
       count++;
    }
    cout << endl;
    return count;
}
int main() {
       const int iSize = 3, fSize = 2, dSize = 3, cSize = 5;
       int i_arr[iSize] = {23, 45, 78};
    float f_arr[fSize] = {12.2, 67.5};
```

```
    double d_arr[dSize] = {3.345, 7.567, 9.023};
    char c_arr[cSize] = {"HI!!"};
    cout << "Масив з типом даних int:";
    cout << "Кількість елементів: " << printArr<int, int>(i_arr, iSize) << endl;
    cout << "Масив з типом даних float:";
    cout << "Кількість елементів: " << printArr<float, int>(f_arr, fSize) << endl;
    cout << "Масив з типом даних double:";
    cout << "Кількість елементів: " << printArr<double, int>(d_arr, dSize) << endl;
    cout << "Масив з типом даних char:";
    cout << "Кількість елементів: " << printArr<char, int>(c_arr, cSize) << endl;
      cin.get();
    return 0;
}
```

Приклад 3. Створити шаблон функції, яка приймає масив з різними типами даних та повертає мінімальний елемент у масиві.

Програмний код реалізації прикладу

```
#include <iostream>
#include <clocale>
using namespace std;
template <typename Type>
Type minimal (Type *arr, int coutElements) {
Type min = arr [0]; // мінімальне значення масиві
for (int i = 0; i < coutElements; i++)
if (min > arr[i])
min = arr [i];
return min;
}
int main() {
// Установка підтримки кирилиці
setlocale(LC_CTYPE, "rus");
 const int iSize = 3, fSize = 2, dSize = 3;
int i_arr[iSize] = {23, 45, 78};
float f_arr[fSize] = {12.2, 67.5};
double d_arr[dSize] = {3.345, 7.567, 9.023};
 cout << "Мінімальний елемент (int): " << minimal<int>(i_arr, iSize) << endl;
cout << "Мінімальний елемент (float):" << minimal<float>(f_arr, fSize) << endl;
cout << "Мінімальний елемент (double):" << minimal<double>(d_arr, dSize) << endl;
 cin.get();
return 0;
}
```

Приклад 4. Створити шаблон функції, яка приймає 3 параметри та перемножує їх між собою.

Програмний код реалізації прикладу

```
#include <iostream>
#include <clocale>
using namespace std;
template <typename Type>
Type multiple (Type a, Type b, Type c) {
Type res = a * b * c;
return res;
```

```
}
int main(){
// Установка підтримки кирилиці
setlocale(LC_CTYPE, "rus");
cout << "Множення (int): " << multiple<int>(3, 3, 2) << endl;
cout << "Множення (float): " << multiple<float>(3.3, 9.1, 8.2) << endl;
cout << "Множення (double): " << multiple<double>(5.233, 8.9999, 0.356) << endl;
cin.get();
return 0; }
```

Шаблони класів C++. Шаблони класів працюють за тим самим принципом як і шаблони функцій. Різниця полягає в тому, що шаблони класів описують шаблонну структуру класу, а не функції. Щоб створити шаблон класу, використовуйте ключове слово template.

Шаблон класу *визначає тіпонезалежний клас, який надалі служить для створення об'єктів необхідних типів.*

Якщо компілятор C++ зустрічає оголошення об'єкта, засноване на шаблоні класу, то для побудови класу необхідного типу він буде використовувати типи, зазначені при оголошенні. Дозволяючи швидко створювати класи, що відрізняються тільки типом, шаблони класів скорочують обсяг програмування, що, в свою чергу, заощаджує час.

Пояснимо використання шаблонів класів на простому прикладі. Нехай необхідно створити простий клас "Масив", який буде виконувати прості дії: Додавання і Відображення елементів.

Приклад 5. Створити клас з іменем ***Massiv***, який приймає масив за різними типами даних.

Програмний код реалізації прикладу

```
#include <iostream>
using namespace std;
       /*КЛАС*/
 class
 {
   int Array[10];    //Масив цілочислених значень з 10 елементів
   int count;        //Лічильник елементів масиву
  public:
   void Add(int );   //Метод для додавання елементів в масив
   void Show();      //Метод для відображення масиву на екрані
 };
 void Massiv::Add(int x)
 {
  static int pos=0;
  Array[pos]=x;
  pos++;
  count=pos;
 }
 void Massiv::Show()
 {
  for (int i=0;i<count;i++)
     cout<<Array[i]<<"\t";
```

```
 cout<<endl;
 }
int main()
{
 Massiv Arr;
 Arr.Add(100.555);
 Arr.Add(200);
 Arr.Add(300);
 Arr.Show();
 system("pause");
 return 0;
}
```

Це приклад створення звичайного класу. Але іноді виникає необхідність створення такого ж класу, в якому відрізняється тільки тип даних. Наприклад, може знадобитися створення класу, в якому потрібні масиви, які будуть зберігати і обробляти не цілочисельні змінні, а рядкові. Як варіант, можна дописати класи для кожного з типів змінних, але це не раціонально. Тоді програмний код вийде занадто громіздким. Ось тут і приходять на допомогу шаблони класів.

Приклад 6. Створити клас з іменем ***Massiv***, який приймає масив за різними типами даних. В програмі застосувати шаблон класу.

Програмний код реалізації прикладу

```
#include <iostream>
using namespace std;
int pos=0; //Позиція в масиві
template <class T> //Шаблон с класу з параметром T
 class Massiv
 {
   T Array[10];      //Масив цілочислених значень з 10 елементів
   int count;        //Лічильник елементів масиву
  public:
   void Add(T ); //Метод для додавання елементів в масив
   void Show();      //Метод для відображення масиву на екрані
 };
 template <class T> void Massiv<T>::Add(T x)
 { static int pos=0;
  Array[pos]=x;
  pos++;
  count=pos;
 }
 template <class T> void Massiv<T>::Show()
 {  cout<<"\t"<<endl;
   for (int i=0;i<count;i++)
       cout<<"\t"<<Array[i];
   cout<<endl;
 }
int main()
{
 setlocale(0,"");
 Massiv<int> Arr;
 Arr.Add(100.555);
```

```
 Arr.Add(200);
 Arr.Add(300);
 Arr.Show();
 Massiv< char *> Arr2;
 Arr2.Add("Рядок");
 Arr2.Add("Шаблон класу");
 Arr2.Add("Копія класу");
 Arr2.Show();
 cout<<endl;
 system("pause");
 return 0;
}
```

Приклад 7. Реалізувати шаблон класу, який буде обробляти об’єкти за різними типами даних.

Програмний код реалізації прикладу

```
#include <iostream>
using namespace std;
const int arr_length = 3;
template <class Type>
class Arr {
private:
  Type arr [arr_length];
public:
  Arr(Type* a) {
    for (int i = 0; i < arr_length; i++)
      arr[i] = a[i];
  }
    Type getElement(int i) {
    return arr[i];
  }
};
int main() {
    int arr[] = {12, 23, 45, 6};
    Arr<int> obj(arr);
    cout << obj.getElement(2) << endl;
       char arr_1[] = {"Mars"};
    Arr<char> object(arr_1);
    cout << object.getElement(0) << endl;
       cin.get();
    return 0; }
```

Приклад 8. Використовуючи шаблон класу, необхідно створити ***Person*** клас до якого додається:

- поле для зберігання першої літери імені;
- поле ***Health*** для встановлення рівня здоров'я. Це поле може мати різні типи даних;
- конструктор для встановлення всіх даних;
- метод виведення поля ***Health***.

Програмний код реалізації прикладу

```
using namespace std;
```

```
template <class HealthType>
class Person {
    private:
    HealthType health;
    char firstLetterName;
    public:
    Person(HealthType health, char firstLetterName) {
    this->health = health;
    this->firstLetterName = firstLetterName;
}
 HealthType getHealht() {
return health;
}
};
int main() {
   Person<int> john(100, 'J');
  cout << john.getHealht() << endl;
   Person<float> george(98.05, 'G');
  cout << george.getHealht() << endl;
   cin.get();
return 0; }
```

Стандартна бібліотека шаблонів STL. Стандартна бібліотека шаблонів надає набір добре сконструйованих узагальнених компонентів C++. Бібліотека містить п'ять основних видів компонентів:

1. алгоритм (*algorithm*) – визначає обчислювальну процедуру;
2. контейнер (*container*) – управляє набором об'єктів в пам'яті;
3. ітератор (*iterator*) – забезпечує для алгоритмів засіб доступу до вмісту контейнера;
4. функціональний об'єкт (*function object*) – інкапсулює функцію в об'єкті для використання іншими компонентами;
5. адаптер (*adaptor*) – адаптує компонент для забезпечення різного інтерфейсу.

До найпопулярніших контейнерів відносять такі:

- vector – колекція елементів, збережених в масиві, що збільшується в міру необхідності. Заголовочний файл – <*vector*>.

- list – колекція елементів, збережених, як двонаправлений зв'язаний список. Заголовочний файл – <*list*>.

- map – це колекція, яка зберігає пари значень pair <*Key,T*> та призначена для швидкого пошуку значення за ключем Key. Як ключ може бути використано, наприклад, рядок або *int*, але при цьому необхідно пам'ятати, що головною особливістю ключа є можливість застосувати до нього операцію порівняння. Важливо: ключ повинен бути унікальним. Заголовочний файл – <*map*>.

- set – це колекція унікальних значень *Key* – кожне з яких є також і ключем. Тобто, це відсортована колекція, призначена для швидкого пошуку необхідного значення. До ключа пред'являються ті ж вимоги, що й у випадку ключа для *map*. Природно, використовувати її для цієї мети немає сенсу, якщо зберігати в ній прості типи даних, щонайменше необхідно визначити свій клас, який зберігає

пару ключ – значення і визначає операцію порівняння по ключу. Дуже зручно використовувати дану колекцію, якщо треба уникнути повторного збереження одного і того ж значення. Заголовочний файл – <*set*>.

- *multimap* – це модифікований *map*, в якому відсутня вимоги унікальності ключа Тобто, якщо здійснювати пошук по ключу, то повернеться не одне значення, а набір значень, збережених з даними ключем. Заголовочний файл – <*map*>.

- *multiset* – те ж саме відноситься і до цієї колекції, вимоги унікальності ключа в ній не існує, що призводить до можливості зберігання дублікатів значень. Тим не менш, існує можливість швидкого знаходження значень по ключу у випадку, якщо визначити свій клас. Оскільки всі значення в *map* і *set* зберігаються у відсортованому вигляді, то в цих колекціях можна швидко відшукати необхідне значення за ключем, але при цьому операція вставляння нового елемента буде складнішою, ніж, наприклад, в *vector*. Заголовочний файл – <*set*>.

Приклад 9. Реалізувати шаблон класу, який буде обробляти об’єкти за різними типами даних. В програмі використати контейнера *vector* з бібліотеки *STL* для обробки цілих чисел.

Програмний код реалізації прикладу

```
#include <vector>
#include <algorithm>
#include <iostream>
using namespace std;
template <class T> void pr(T& v)
{        //===== Шаблон функції виведення даних за допомогою ітератора
   T::iterator p;    //====== Ітератор для будь-якого контейнера
   for ( p=v.begin(),int i=0; p!=v.end(); p++, i++)
        cout << endl << i + 1 <<". "<< *p;
   cout << '\n';
}
int main ()
{   vector <int> v(10);     //======== Вектор цілочисельного типу даних
   cout << "\nInt Vector:\n";
   for (int i=0; i<v.size(); i++)
   {     v[i] = rand()%10 + 1;
       cout << v[i]<< "; ";
   }
   sort (v.begin (), v.end());   //======= Сортування за замовчуванням
   cout << "\n\nAfter default sort\n";
   for (int i=0; i<v.size(); i++) cout << v[i]<< "; ";
   cout << "\n\nUsing iterators\n\n";
   pr(v);
   v.erase(v.begin());     //======== Видалення елементів
   cout << "\n\nAfter first element erasure\n";
   for (int i=0; i<v.size(); i++) cout << v[i]<< "; ";
   v.erase (v.end()-2, v.end());
   cout << "\n\nAfter last 2 elements erasure\n";
   for (int i=0; i<v.size(); i++) cout << v[i]<< "; ";
   int size = 2;          //======== Зміна розмірів
```

```
v.resize(size);
cout << "\n\nAfter resize, the new size: " << v.size()<< endl;
for (uint i=0; i<v.size(); i++) cout << v[i]<< "; ";
v.resize(6,-1);
cout << "\n\nAfter resize, the new size: " << v.size()<< endl;
for (uint i=0; i<v.size(); i++) cout << v[i]<< "; ";
cout << "\n\nVector's maximum size: " << v.max_size()
     << "\nVector's capacity: " << v.capacity() << endl;
v.reserve (100);
cout << "\nAfter reserving storage for 100 elements:\n"<< "Size: "
    <<v.size()<<endl<<"Maximum size: "<<v.max_size()<< endl
    << "Capacity: " << v.capacity() << endl;
v.resize(2000);
cout << "\nAfter resizing storage to 2000 elements:\n"<<"Size: "<<v.size()
    <<endl<<"Maximum size: "<<v.max_size() << endl<<"Capacity: "
    <<v.capacity()<<endl<< "\n\n";
_getch(); }
```

Контрольні запитання

1. Чим відрізняється параметр шаблона від параметру функції?
2. Як створити шаблонний клас?
3. Що являє собою стандартна бібліотека шаблонів STL?
4. Які основні шаблонні класи колекцій ви знаєте?
5. Що представляють собою ітератори?

Список використаних джерел

Базова

1. Вінник В.Ю. Алгоритмічні мови та основи програмування: мова С++. – Житомир: ЖДТУ, 2007. – 328 с.
2. Матвієнко М.П. Алгоритми та структури даних : навч. посіб. / М. П. Матвієнко. – Київ : Видавництво "Ліра-К", 2014. – 340 с.
3. Федорченко В. М., Методичні рекомендації до виконання лабораторних робіт з на вчальної дисципліни "Алгоритмізація та програмування" для студентів напряму підготовки 6.050101 "Комп'ютерні науки" всіх форм навчання / В. М. Федорченко, О. В. Тарасов, А. В. Щербаков, Ю. Э. Парфенов. – Харків : Вид. ХНЕУ, 2012. –180 с
4. Трофименко О.Г., Основи програмування. Базові алгоритми : метод. вказівки для лаб. і практ. робіт / О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката. – Ч. 1. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2014. – 108 с.
5. Трофименко О.Г., С++. Алгоритмізація та програмування : підручник / О.Г. Трофименко, Ю.В. Прокоп, Н.І. Логінова, О.В. Задерейко. 2-ге вид. перероб. і доповн. – Одеса : Фенікс, 2019. – 477 с.
6. Трофименко О.Г., С++. Основи програмування. Теорія та практика: підручник / [О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката та ін.] ; за ред. О. Г. Трофименко. – Одеса : Фенікс, 2010. – 544 с.
7. Трофименко О.Г., С++. Теорія та практика: навч. посіб. з грифом МОНУ/ [О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката та ін.] ; за ред. О. Г. Трофименко. – Одеса : ВЦ ОНАЗ, 2011. – 587 с.
8. Шпак З.Я., Програмування мовою С: Навч. посібник. / З.Я. Шпак. − 2-е видання, доповнене. − Львів: «Львівська політехніка», 2011.− 436 с.

Додаткова

9. Ковалюк Т.В. Основи програмування: навчальний посібник. / Т.В. Ковалюк − Київ: ВНЗ, 2005. − 400 с.
10. Трофименко О.Г., Основи програмування. Програмне опрацювання файлів: метод. вказівки для лаб. і практ. робіт / О.Г. Трофименко, Ю. В. Прокоп, І.Г. Швайко, Л.М. Буката. – Ч. 3. – Одеса: ВЦ ОНАЗ ім. О.С. Попова, 2015. – 80с.
11. Трофименко О. Г. Створення багатомодульних програмних проектів для опрацювання даних у файлах засобами С++: метод. вказівки для виконання курсової роботи з дисципліни "Основи програмування" / Трофименко О. Г., Прокоп Ю. В. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2015. – 44 с.
12. Трофименко О.Г., Основи програмування. Опрацювання структурованих типів: метод. вказівки для лаб. і практ. робіт / О. Г. Трофименко, Ю. В. Прокоп , І. Г. Швайко, Л. М. Буката. – Ч. 2. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2014. – 130 с.
13. Співаковський О. В., Осипова Н. В., Львов М. С., Бакуменко К. В. Основи алгоритмізації та програмування. Обчислювальний експеримент. Розв’язання проблем ефективності в алгоритмах пошуку та сортування: Навчальний посібник. – Херсон: Айлант. – 2010. – 100 с.: іл.

14. Чичкарьов Є.А., Зінченко О.В., Єльченко С.В., Прикладне програмування на Python, навч. пос. // Київ, ДУІКТ, 2022 р., - с. 120

Інформаційні ресурси

15. Алгоритмізація та програмування: ДУТ [електронний ресурс]-Режим доступу: http://dn.dut.edu.ua/course/АП (дата звернення 1.12.2021). – Назва з екрана.
16. Algorithms and Data Structures [Электронный ресурс]. – Режим доступу: https://sites.google.com/site/indy256/algo_cpp.
17. Sorting Algorithm Animations [Электронный ресурс]. – Режим доступу: http://www.sorting-algorithms.com
18. Алгоритмізація та програмування: КНУ. [електронний ресурс] – Режим доступу: http://kiis.knu.ua/temi-algoritmizacija-ta-programuvannja/