**МІНІСТЕРСТВО ОСВІТИ І НАУКИ,
МОЛОДІ ТА СПОРТУ УКРАЇНИ**

**ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«НАЦІОНАЛЬНИЙ ГІРНИЧИЙ УНІВЕРСИТЕТ»**

**В.В. Ткачов, П.Ю. Огєєнко,
Р.В. Макітренко**

КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ ТА ПРОГРАМУВАННЯ

Навчальний посібник

Том 2

ДОДАТКОВІ ВІДОМОСТІ ТА ПРАКТИЧНІ ЗАВДАННЯ

Дніпропетровськ
НГУ
2012

УДК 004.42 (075.8)
ББК 32.973-018.1я73
Т-48

Рекомендовано Міністерством освіти і науки, молоді та спорту України як навчальний посібник для студентів вищих навчальних закладів напряму підготовки "Автоматизація та комп'ютерно-інтегровані технології" (лист № 1/11-6162 від 03.05.2012 р.).

Рецензенти:

А.І. Купін, д-р техн. наук, професор, завідувач кафедри "Комп'ютерні системи і мережі" (Криворізький технічний університет)

І.В. Жуковицький, д-р техн. наук, професор, завідувач кафедри "Електронні обчислювальні машини" (Дніпропетровський національний університет залізничного транспорту ім. академіка В. Лазаряна)

Ткачов, В.В.

Т-48 Комп'ютерні технології та програмування [Текст]. Т. 2. Додаткові відомості та практичні завдання: навч. посібник / В.В. Ткачов, П.Ю. Огєєнко, Р.В. Макітренко – Д.: Національний гірничий університет, 2011. – 179 с.

ISBN 978-966-350-361-5
978-966-350-363-9 (Т.2)

Розглянуто застосування систем числення, базисні алгоритми впорядковування послідовностей та бібліотеки для роботи з рядками і файлами. Подано лабораторні роботи та тестові завдання для поглиблення знань та отримання практичних навиків застосування мови С++ для розробки програмних проектів.

Посібник укладено відповідно до програми дисципліни "Комп'ютерні технології та програмування" для студентів, що навчаються за напрямом підготовки 050202 "Автоматизація та комп'ютерно-інтегровані технології", а також бути використано для студентів галузі "Автоматика та управління".

УДК 004.42 (075.8)
ББК 32.973-018.1я73

ISBN 978-966-350-361-5
978-966-350-363-9 (Т.2)

ЗМІСТ

Присвячується 50-річчю кафедри
автоматизації та
комп'ютерних систем
Державного вищого навчального закладу
"Національний гірничий університет"

Передмова

Навчальний посібник "Комп'ютерні технології та програмування" складається з двох томів. Перший том містить теоретичні відомості щодо застосування основ програмування мовою С++. Другий том присвячено розгляду спеціалізованих відомостей та практичним завданням. Він складається з трьох частин.

Перша частина містить додаткові до курсу теоретичні відомості, що є необхідними для поліпшення розуміння та засвоєння основ програмування. Вона розкриває наступні теми: системи числення, алгоритми методик сортування невпорядкованих послідовностей, бібліотечний клас роботи з рядками та бібліотечні функції файлового вводу/виводу.

Для перевірки засвоєння матеріалу теоретичних частин наприкінці кожен з розділів має перелік контрольних питань. Ключові слова мови С++ у тексті подано напівжирним шрифтом, а коментарі та пояснення програм – курсивом. Це полегшує сприйняття матеріалу навчального посібника та дозволяє привернути увагу читача до базових конструкцій мови програмування.

Друга частина є практичною і включає лабораторні роботи за дисципліною. Індивідуальні завдання за варіантами призначені для прикладного засвоєння та поглиблення отриманих вмінь для розробки програм на мові С++.

Третя частина присвячена завданням для самоперевірки засвоєння теоретичного та практичного матеріалу посібника. Вона містить тестові завдання відкритої та закритої форм, що відповідають змісту екзаменаційних білетів.

Додатки включають основні положення розробки схем алгоритмів процесів та програм, приклад оформлення лабораторної роботи та відповіді на тестові завдання п'ятої частини.

Навчальний посібник побудовано у відповідності до програми дисципліни "Комп'ютерні технології та програмування", але його матеріал може бути застосовано і для інших предметів, що базуються на вивченні основ програмування мовою С++.

Автори вважають, що теоретичні відомості та велика кількість практичних прикладів створення програм з детальним їх описом дозволять студентам використовувати посібник для самостійного навчання та полегшать ознайомлення з дисципліною "Комп'ютерні технології та програмування".

ЧАСТИНА І
ДОДАТКОВІ ВІДОМОСТІ ТА АЛГОРИТМИ

1. СИСТЕМИ ЧИСЛЕННЯ

Навчальною метою розділу є ознайомлення читача з основними системами числення, що застосовуються при програмуванні та роботі з персональним комп'ютером.

Внаслідок вивчення матеріалу даного розділу читач повинен вміти:

– формувати значення у двійковій, вісімковій, десятковій та шістнадцятковій системах числення;

– перетворювати числа між системами.

1.1. Системи числення

Важливим при проектуванні та написанні структурних та об'єктно-орієнтованих програм є розуміння найбільш поширених систем числення, згідно з якими реалізується частина операцій мови С++.

Такими системами є двійкова, вісімкова, десяткова та шістнадцяткова. Для звичайної людини найбільш знайомою є десяткова. Вона використовується у всьому розвиненому світі для обчислень. Це цифри від "0" до "9". Всі математичні операції людина звикла вирішувати за допомогою десяткової системи числення. На відміну від людини, логіка персонального комп'ютера є бінарною. В її основі є двійкова система числення.

Назва систем числення відповідає кількості символів, що можна використовувати для формування чисел (табл. 1.1).

Таблиця 1.1

Системи числення

Назва системи		**Основа системи**	**Ймовірні значення розряду числа**
двійкова	BIN	2	0,1
вісімкова	OCT	8	0,1,2,3,4,5,6,7
десяткова	DEC	10	0,1,2,3,4,5,6,7,8,9
шістнадцяткова	HEX	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Часто говорять, що в основі системи знаходиться число, яке відповідає її назві. Це означає, що при формуванні значень кожен з розрядів має множник, який є основою системи у степені. Збільшення чисел відтворюється справа наліво, як і степенів у множниках. Найкращим прикладом для розуміння такого підходу є розклад десяткового значення, що організується у голові людини інтуїтивно:

$$\ldots \; X_{10^4} \; X_{10^3} \; X_{10^2} \; X_{10^1} \; X_{10^0}$$

Степені мають значення, відповідні до номеру розряду числа, що формується. Числа у інших трьох системах числення створюються за тим самим правилом:

... X X X X X
2^4 2^3 2^2 2^1 2^0

... X X X X X
8^4 8^3 8^2 8^1 8^0

... X X X X X
16^4 16^3 16^2 16^1 16^0

Множник є також першим значенням, що може бути отримано завдяки відповідному розряду.

Задля того, щоб при записі було зрозуміло, яка система числення використовується для формування числа, слугує нижній індекс, що відповідає основі системи і записується за числом:

$5D21_{16}$; 453_{16}; $12AF_{16}$; 101_{16};
1010_{10}; 902_{10}; 399_{10}; 101_{10};
5742_8; 1026_8; 3577_8; 101_8;
10111_2; 11100_2; 10001_2; 101_2.

Такий підхід є необхідним, якщо застосовується більше однієї системи. Він дозволяє відрізняти числа різних систем, що мають однакову форму запису. У прикладі такі числа є заключними для кожного з рядків.

Перехід з десяткової системи у інші можна виконувати двома шляхами: розписувати за степенями чи ділити на основу.

Розписати за степенями можна кожне число, але найбільшу зручність цей підхід набуває при переведенні чисел у двійкову форму. Для вісімкової та шістнадцяткової системи розпис за степенями є достатньо складним, бо потребує багато обчислень.

Наприклад, якщо треба отримати значення десяткового числа 1243 у трьох інших формах, то по-перше визначаються скільки розрядів буде використано:

BIN – 11
OCT – 4
HEX – 3,

потім у кожний з розрядів записують відповідне до системи значення так, щоб при застосуванні множників у сумі було отримано число десяткової форми.

1	0	0	1	1	0	1	1	0	1	1
2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1

1*1024+1*128+1*64+1*16+1*8+1*2+1*1=1243

2	3	3	3
8^3	8^2	8^1	8^0
512	64	8	1

2*512+3*64+3*8+3*1=1243

4	D	B
16^2	16^1	16^0
256	16	1

4*256+13*16+11*1=1243

Таким чином, як і було зазначено раніше, спосіб розкладу за степенями є зручним лише для двійкової системи, бо максимальним значенням розряду є одиниця, тому обчислення значення зводиться до вміння складати.

Спосіб переводу чисел шляхом ділення на основу однаково зручний для всіх систем, але він є досить громіздким. Згідно з ним число, а потім і частку треба ділити покроково на основу системи числення до тих пір, поки не буде отримано значення, що є меншим за дільник. Потім всі залишки збираються у зворотному порядку і формують кінцеве значення.

Наприклад, для десяткового числа, що було розглянуто вище:

1243	2									
1242	621	2								
1	620	310	2							
	1	310	155	2						
		0	154	77	2					
			1	76	38	2				
				1	38	19	2			
					0	18	9	2		
						1	8	4	2	
							1	4	2	2
								0	2	1
									0	

$1243_{10}=10011011011_2$

1243	8		
1240	155	8	
3	152	19	8
	3	16	2
		3	

$1243_{10}=2333_8$

1243	16	
1232	77	16
11(B)	64	4
	13(D)	

$1243_{10}=4DB_{16}$

Зворотній переклад з двійкової, вісімкової та шістнадцяткової систем у десяткову виконується відповідно до першого з розглянутих шляхів прямого переведення:

$AC17_{16}$
$10*16^3+12*16^2+1*16^1+7*16^0=44055_{10}$

7320_8
$7*8^3+ 3*8^2+2*8^1=3792_{10}$

101110_2
$1*2^5+1*2^3+1*2^2+1*2^1=46_{10}$

На відміну від прямого такий шлях зворотного переведення є зручним, бо як значення розрядів, так і множники вже відомі.

Перетворення чисел між системами, кожна з яких не є десятковою, також зручно здійснювати через проміжний перехід у десяткову систему за схемою, що наведено на рис. 1.1.

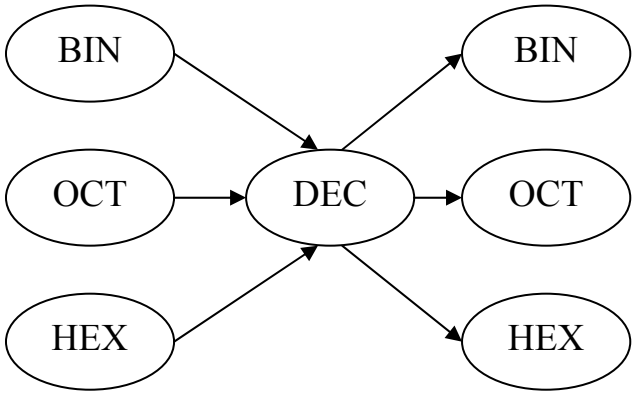
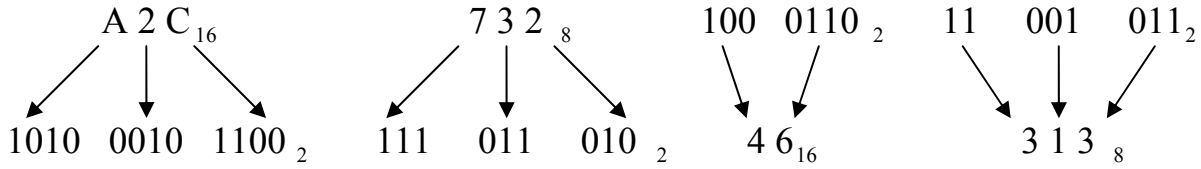


Рис. 1.1. Схема перетворення між системами числення

Для деяких переходів є зручним використання математичних знань про те, що $2^4 = 16$ та $2^3 = 8$. Згідно з цим, кожен розряд шістнадцяткової форми може бути подано, як чотири, а вісімкової, як три розряди двійкової форми, що значно спрощує перетворення між цими системами.



Висновки

У даному розділи було розглянуто наступні основні питання:

– двійкова система числення;

– вісімкова система числення;

– десяткова система числення;

– шістнадцяткова система числення.

Контрольні питання

1) Для чого застосовуються системи числення?

2) Які системи числення є найбільш поширеними?

3) Які значення може приймати розряд числа у шістнадцятковій формі?

4) Як формуються числа у різних системах числення?

5) Яке значення називають основою системи числення?

6) Що означає розкласти число за степенями?

7) Які є шляхи переходу з десяткової форми в інші?

8) Як шляхом ділення перетворити число з десяткової у вісімкову форму?

9) Як перетворюються числа з різних форм у десяткову?

10) Між якими формами є шляхи швидкого перетворення, на чому вони базуються?

2. АЛГОРИТМИ ВПОРЯДКУВАННЯ

Навчальною метою розділу є ознайомлення читача з основними алгоритмами, що застосовуються для впорядковування послідовностей даних.

Внаслідок вивчення матеріалу даного розділу читач повинен знати:

– метод бульбашкового сортування;
– метод сортування Шейкера;
– метод сортування вибіркою;
– метод сортування вставкою;
– метод сортування вибіркою-вставкою;
– метод сортування Шелла;
– метод швидкого сортування;
– перевагу, що надає впорядкованість при застосуванні алгоритмів пошуку.

2.1. Методи сортування

При роботі з масивами неврегульованих значень для програміста важливу роль грає знання методик сортування. Сортування дозволяє упорядковувати значення послідовностей і на основі цього зменшує час пошуку елементів в них.

Сортування – це процес поетапного впорядковування значень елементів масиву за убуванням або за зростанням.

Прохід впорядковування – це перебір елементів послідовності від одного краю до іншого.

Кроком сортування називають етап проходу впорядковування, під час якого опрацьовується один елемент послідовності.

Існує безліч методик сортування. Найпростішими (найбільший час роботи) є сортування методами обміну, вибірки і вставки.

2.1.1. Сортування обміном

Сортування обміном – це, коли впорядковування елементів послідовності виконується за допомогою обміну значень між елементами.

Принцип обміну двох елементів масиву значеннями засновано на схемі, наведеній на рис. 2.1.

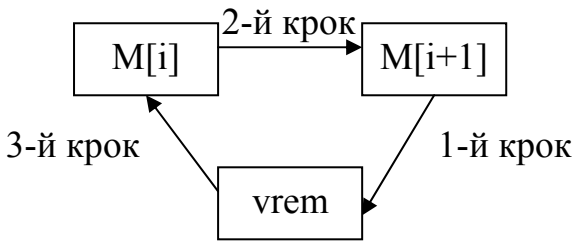


Рис. 2.1. Схема обміну елементів значеннями

Спочатку створюється допоміжна змінна. Значення одного з елементів переписується у неї (1-й крок). Потім у цей елемент перезаписується значення з

іншого елемента (2-й крок). І на останньому етапі значення з допоміжної змінної переприсвоюється у другий елемент (3-й крок).

Неоптимізований метод бульбашкового сортування здійснюється шляхом обміну. Його засновано на виконанні для неввпорядкованої послідовності значень N-1 проходів по N-1 кроків (N – число елементів послідовності). На кожному кроці виконується порівняння значення поточного елемента із значенням наступного і їх розташування в потрібному порядку шляхом обміну. За $(N-1)^2$ кроків можна виконати повне впорядковування будь-якої послідовності.

Схематичний приклад сортування за бульбашковим методом цілочисельного масиву від більшого до меншого наведено нижче:

масив:		-3	12	3	4	21
1-й прохід	1-й крок	12	-3	3	4	21
1-й прохід	2-й крок	12	3	-3	4	21
1-й прохід	3-й крок	12	3	4	-3	21
1-й прохід	4-й крок	12	3	4	21	-3
2-й прохід	1-й крок	12	3	4	21	-3
2-й прохід	2-й крок	12	4	3	21	-3
2-й прохід	3-й крок	12	4	21	3	-3
2-й прохід	4-й крок	12	4	21	3	-3
3-й прохід	1-й крок	12	4	21	3	-3
3-й прохід	2-й крок	12	21	4	3	-3
3-й прохід	3-й крок	12	21	4	3	-3
3-й прохід	4-й крок	12	21	4	3	-3
4-й прохід	1-й крок	21	12	4	3	-3
4-й прохід	2-й крок	21	12	4	3	-3
4-й прохід	3-й крок	21	12	4	3	-3

4-й прохід	4-й крок	21	12	4	3	-3

Оптимізований метод бульбашкового сортування відрізняється від неоптимізованого тим, що після кожного проходу число елементів, що піддаються подальшому порівнянню, зменшується на один. Оптимізація заснована на тому, що за один прохід з будь-якої позиції послідовності максимальне або мінімальне значення переміщується на крайню, таким чином, при наступному проході значення цього елементу обміну не підлягає.

Схематичний приклад сортування за оптимізованим бульбашковим методом цілочисельного масиву від більшого до меншого наведено нижче:

масив:		-3	12	3	4	21
1-й прохід	1-й крок	12	-3	3	4	21
1-й прохід	2-й крок	12	3	-3	4	21
1-й прохід	3-й крок	12	3	4	-3	21
1-й прохід	4-й крок	12	3	4	21	-3
2-й прохід	1-й крок	12	3	4	21	-3
2-й прохід	2-й крок	12	4	3	21	-3
2-й прохід	3-й крок	12	4	21	3	-3
3-й прохід	1-й крок	12	4	21	3	-3
3-й прохід	2-й крок	12	21	4	3	-3
4-й прохід	1-й крок	21	12	4	3	-3

Метод сортування Шейкера є вдосконаленням оптимізованого бульбашкового сортування і ґрунтується на тому, що кожен наступний прохід виконується із зміною напряму вичитування елементів послідовності. Завдяки цьому протягом двох проходів на відповідних позиціях виявляються як максимальне, так і мінімальне значення послідовності.

Схематичний приклад сортування за методом Шейкера цілочисельного масиву від більшого до меншого наведено нижче:

масив:		-3	12	3	4	21

1-й прохід	1-й крок	12	-3	3	4	21
1-й прохід	2-й крок	12	3	-3	4	21
1-й прохід	3-й крок	12	3	4	-3	21
1-й прохід	4-й крок	12	3	4	21	-3
2-й прохід	1-й крок	12	3	21	4	-3
2-й прохід	2-й крок	12	21	3	4	-3
2-й прохід	3-й крок	21	12	3	4	-3
3-й прохід	1-й крок	21	12	3	4	-3
3-й прохід	2-й крок	21	12	4	3	-3
4-й прохід	1-й крок	21	12	4	3	-3

Оптимізоване бульбашкове сортування і сортування Шейкера можуть бути покращені за допомогою перевірки кількості виконаних обмінів у кінці кожного проходу і зупинці сортування при їх нульовому числі. Підхід ґрунтується на тому, що під час сортування обміном з кроком в один елемент при нульовому числі обмінів за прохід, послідовність є повністю впорядкованою і подальші проходи лише збільшують час виконання сортування.

Схематичне відображення зупинення сортування при нульовому числі обмінів протягом проходу наведено у вигляді схеми алгоритму на рис. 2.2.

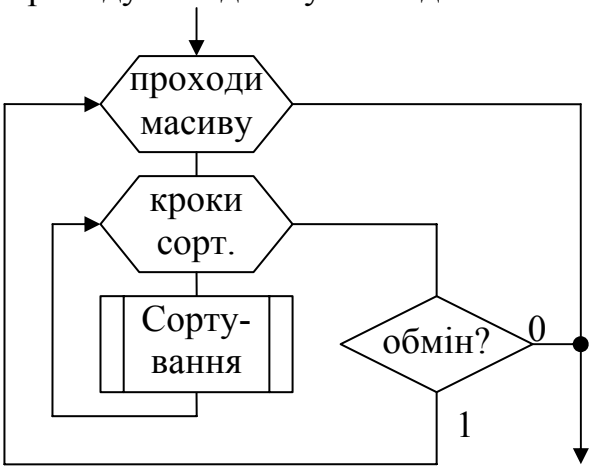


Рис. 2.2. Схема алгоритму рахування числа обмінів за прохід

Схематичний приклад сортування за оптимізованим бульбашковим методом цілочисельного масиву від більшого до меншого з рахуванням кількості обмінів за прохід наведено нижче:

масив:		-3	12	3	6	5
	k=0 → 1-й прохід					
1-й прохід	1-й крок	12	-3	3	6	5
						k=1
1-й прохід	2-й крок	12	3	-3	6	5
						k=2
1-й прохід	3-й крок	12	3	6	-3	5
						k=3
1-й прохід	4-й крок	12	3	6	5	-3
						k=4
	k≠0, k=0 → 2-й прохід					
2-й прохід	1-й крок	12	3	6	5	-3
						k=0
2-й прохід	2-й крок	12	6	3	5	-3
						k=1
2-й прохід	3-й крок	12	6	5	3	-3
						k=2
	k≠0, k=0 → 3-й прохід					
3-й прохід	1-й крок	12	6	5	3	-3
						k=0
3-й прохід	2-й крок	12	6	5	3	-3
						k=0
	k=0, зупинення					

Схематичний приклад сортування за методом Шейкера цілочисельного масиву від більшого до меншого з рахуванням кількості обмінів за прохід наведено нижче:

масив:		-3	12	4	3	21
	k=0 → 1-й прохід					
1-й прохід	1-й крок	12	-3	4	3	21
						k=1
1-й прохід	2-й крок	12	4	-3	3	21
						k=2
1-й прохід	3-й крок	12	4	3	-3	21
						k=3
1-й прохід	4-й крок	12	4	3	21	-3
						k=4

k≠0, k=0 → 2-й прохід

2-й прохід	1-й крок	12	4	21	3	-3
						k=1
2-й прохід	2-й крок	12	21	4	3	-3
						k=2
2-й прохід	3-й крок	21	12	4	3	-3
						k=3

k≠0, k=0 → 3-й прохід

3-й прохід	1-й крок	21	12	4	3	-3
						k=0
3-й прохід	2-й крок	21	12	4	3	-3
						k=0

k=0, зупинення

2.1.2. Сортування вибіркою

Сортування вибіркою – це, коли вся послідовність умовно розбивається на впорядковану і невпорядковану множини, причому з кожним кроком число впорядкованих елементів збільшується на одиницю, починаючи з нуля. На кожному кроці проводиться пошук максимального або мінімального, залежно від напряму впорядковування, значення серед невпорядкованої множини і обмін його значення з елементом на поточній позиції, який після обміну стає останнім значенням у впорядкованій множині.

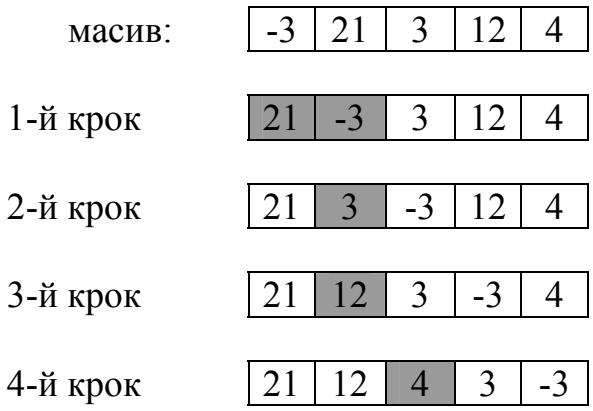
Схематичний приклад сортування методом вибірки від більшого до меншого наведено нижче:

масив:	-3	21	3	4	12
1-й крок	21	-3	3	4	12
2-й крок	21	12	3	4	-3
3-й крок	21	12	4	3	-3
4-й крок	21	12	4	3	-3

2.1.3. Сортування вставкою

Сортування вставкою засновано на послідовному переборі невпорядкованих елементів і виконанні вставки поточного елемента на визначену правилами сортування позицію серед вже впорядкованих значень. Всі елементи, які виявляються меншими (більшими) за поточне значення, переміщуються на одну позицію в напрямку проходу сортування до колишньої позиції поточного елемента.

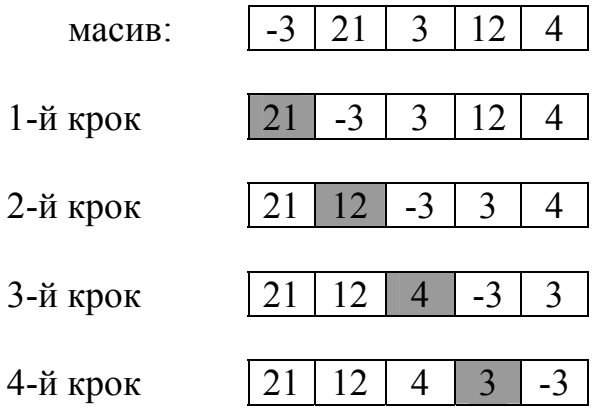
Схематичний приклад сортування методом вставки від більшого до меншого наведено нижче:



2.1.4. Сортування вибіркою-вставкою

Метод сортування вибіркою-вставкою є результатом синтезу двох простих сортувань – вибірки і вставки. Його засновано на аналогічній методиці вибірки: розбиття послідовності на дві множини, проте замість обміну значеннями між знайденим елементом і поточним виконується вставка його значення на відведену йому правилом сортування позицію у вже впорядкованій множині, при цьому виконується переміщення значень елементів масиву на 1 крок.

Схематичний приклад сортування методом вставки від більшого до меншого наведено нижче:



2.1.5. Поліпшені методи сортування

Серед поліпшених методів сортування виділяють метод Шелла і швидке сортування, як найчастіше вживані і такі, що відрізняються відносно малим часом впорядковування.

Під час сортування за методом Шелла для кожного проходу обирається свій унікальний крок (не перевищує значення N-1, де N – число елементів масиву). Поточний елемент послідовності порівнюється з елементом, який відрізняється від нього на крок поточного проходу і при невідповідності їх порядку виконується обмін. З математичної точки зору слід уникати кроків за степенями значення два, проте, обов'язково повинен бути присутнім крок із значенням один. Для більшої ефективності сортування значення кроків для

двох послідовних проходів (неостанніх) повинно відрізнятися більш, ніж на одиницю. Зазвичай максимальний крок обирається більшим чи таким, що дорівнює значенню N/2.

Схематичний приклад сортування методом Шелла від більшого до меншого наведено нижче:

		-3	12	3	4	21
масив:						
1-й прохід	1-й крок	4	12	3	-3	21
1-й прохід	2-й крок	4	21	3	-3	12
2-й прохід	1-й крок	4	21	3	-3	12
2-й прохід	2-й крок	4	21	3	-3	12
2-й прохід	3-й крок	4	21	12	-3	3
3-й прохід	1-й крок	21	4	12	-3	3
3-й прохід	2-й крок	21	12	4	-3	3
3-й прохід	3-й крок	21	12	4	-3	3
3-й прохід	4-й крок	21	12	4	3	-3

Швидке сортування – найбільш ефективна методика, яка доволі часто застосовується у системному програмному забезпеченні. В основі цього сортування лежить рекурсивний підхід. Сортування базується на виборі для послідовності неврегульованих значень елемента, значення якого виступатиме в ролі компаранда (зазвичай близьке до середнього арифметичного значення елементів масиву). Інші елементи розміщуються щодо компаранда наступним чином – великі з одного боку, а менші з іншого. Після першого проходу та сама операція виконується з послідовностями значень справа та зліва від компаранда. Таким чином на кожному проході число компарандів для всієї послідовності дорівнює $2^{(N-1)}$ (N – число елементів масиву). Операції виконуються до тих пір, поки хоча б одна послідовність містить більше ніж два елементи.

Схематичний приклад швидкого сортування від більшого до меншого наведено нижче:

	-3	12	3	4	21
масив:					

	компаранд = 4					
1-й прохід	1-й крок	12	3	4	21	-3
1-й прохід	2-й крок	12	4	21	-3	3
1-й прохід	3-й крок	21	12	4	-3	3
	компаранди = 12, 3					
2-й прохід	1-й крок	21	12	4	3	-3

2.2. Застосування алгоритмів впорядкування

Основним призначенням алгоритмів впорядкування є оптимізація пошуку елементів у послідовностях.

Для розуміння розглянемо два алгоритми пошуку, перший застосовується до невпорядкованих послідовностей і називається лінійним, а другий, для впорядкованих, двійковим.

Сутність лінійного пошуку полягає у послідовному переборі всіх елементів невпорядкованої послідовності до тих пір, поки значення не буде знайдено.

Схематичний приклад лінійного пошуку наведено нижче:

Пошук значення 5

	послідовність
Крок 1, N≠1	1
Крок 2, N≠7	7
Крок 3, N≠12	12
Крок 4, N≠8	8
Крок 5, N≠3	3
Крок 6, N≠9	9
Крок 7, N≠10	10
Крок 8, N≠34	34
Крок 9, N≠-2	-2
Крок 10, N=5, зупинення	5
	11

Двійковий пошук засновано на бінарних запитах до впорядкованої послідовності. Запит виконується до серединного елемента множини елементів послідовності. На запит у якості результату отримується наступні значення:

0 – значення менше за серединний елемент;

1 – значення більше за серединний елемент.

Кожен наступний запит зменшує область пошуку вдвічі.

Схематичний приклад двійкового пошуку наведено нижче:

Пошук значення 5

	послідовність
	-2
	1
Крок 2, N>3	3
Крок 4, N=5, зупинення	5
Крок 3, N<8	8
Крок 1, N<9	9
	10
	11
	12
	34

Таким чином, впорядкованість послідовності дозволяє застосовувати двійковий пошук, що у декілька разів зменшує час пошуку вказаного елемента.

Висновки

У даному розділи було розглянуто наступні основні питання:

– прості методики сортування;

– поліпшені методики сортування;

– застосування впорядкування для зменшення часу на пошук.

Контрольні питання

1) Дайте визначення поняттю сортування.

2) Які є основні види сортування?

3) Як реалізується неоптимізований метод бульбашкового сортування?

4) Скільки порівнянь виконується під час неоптимізованого методу бульбашкового сортування?

5) У чому полягає оптимізація методу бульбашкового сортування?

6) Чим метод сортування Шейкера відрізняється від методу бульбашкового сортування?

7) Для чого при сортуванні методами обміну може знадобитися рахування кількості обмінів за прохід?

8) Як виконується обмін значеннями між двома елементами масиву?

9) У чому полягає сутність сортування вибіркою?

10) Як виконується сортування за методом вставки?

11) Чим відрізняються між собою методи сортування вставкою та вибіркою-вставкою?

12) У чому полягає сутність сортування методом Шелла?

13) Яких правил треба дотримуватися при виборі ряду значень кроків при сортуванні за методом Шелла?

14) Який смисл несе компаранд при швидкому сортуванні?

15) Для чого потрібно впорядкування послідовностей?

3. КЛАС РОБОТИ З РЯДКАМИ

Навчальною метою розділу є ознайомлення читача з роботою із рядками на основі спеціального класу мови С++.

Внаслідок вивчення матеріалу даного розділу читач повинен вміти:

- застосовувати клас роботи з рядками в програмі;
- виконувати операції з об'єктами класу;
- застосовувати методи класу.

3.1. Клас string

У мові С++ для роботи з текстовою інформацією застосовується спеціальний клас, опис якого знаходиться у заголовному файлі "string.h". Окрім його підключення у програмі, треба також зазначити простір імен std. Методи класу будуть доступні крізь його об'єкти. Клас string дозволяє виконувати зі своїми об'єктами стандартні операції присвоєння, додавання та порівняння. Таким чином, якщо треба поєднати декілька рядків у один, достатньо скористатися оператором "+". Нижче наведено табл. 3.1 з операторами, які можна використовувати при роботі з об'єктами класу string.

Таблиця 3.1

Операції з об'єктами класу роботи з рядками

Оператор	Назва операції
=	присвоєння
+	додавання (конкатенація)
+=	присвоєння та додавання (конкатенація)
==	схожість за змістом
!=	несхожість за змістом
<	менше за кількістю символів
<=	менше чи стільки ж символів
>	більше за кількістю символів
>=	більше чи стільки ж символів
<<	вивід на консоль
>>	ввід з консолі

Розглянемо для прикладу програму, що виводить на консоль декілька рядків, що є значеннями об'єктів класу роботи з рядками:

```
#include <iostream>      //підключення бібліотеки потокового вводу/виводу
#include <string>                  //підключення бібліотеки "string.h"

using namespace std;               //застосування простору імен std

void main()
{
    string s, s1, s2;              //оголошення об'єктів класу string s, s1, s2
```

```
    string s3("!!!");                    //оголошення та визначення об'єкта s3
    s1="This year is a very ";           //визначення змінної s1
    s2="important for me";               //визначення змінної s2
    s=s1+s2+s3;                          //додавання рядків
    cout<<s<<"\r\n"<<endl;               //вивід результату на консоль
    s1+=s2+s3;                           //додавання рядків
    cout<<s1<<"\r\n"<<endl;              //вивід результату на консоль
//порівняння рядків за кількістю символів і вивід результату на консоль
    cout<<(s1>s2)<<"\r\n"<<endl;
//порівняння рядків за змістом символіввивід результату на консоль
    cout<<(s2= =s3)<<"\r\n"<<endl;
}
```

Після виконання наведеного коду на консоль буде виведено наступні рядки:

```
This year is a very important for me!!!
This year is a very important for me!!!
1                    (рядок s1 більше s2)
0                    (рядок s2 відрізняється від s3 по символам)
```

Основними методами для роботи з рядками, що поєднує у собі клас "string", є наведені у табл. 3.2.

Таблиця 3.2

Методи класу роботи з рядками

Метод класу	Призначення
append()	додавання до рядку символів та послідовностей символів
assign()	присвоєння одного рядка чи сукупності символів іншому
at()	отримання символу рядка
c_str()	перетворення рядка класу string у символьний масив
capacity()	повертає розмір зайнятої пам'яті
compare()	порівняння рядків чи послідовностей символів рядків
copy()	копіювання рядка в масив
empty()	очищення змісту рядка
erase()	видалення послідовності символів з рядка
find()	пошук символу чи послідовності символів у рядку
find_first_not_of()	пошук у рядку першого символа, який не входить до зазначеного переліку
find_first_of()	пошук у рядку першого символа, який входить до зазначеного переліку
find_last_not_of()	пошук у рядку останнього символа, який не входить до зазначеного переліку
find_last_of()	пошук у рядку останнього символа, який входить до зазначеного переліку

Закінчення табл. 3.2

Метод класу	Призначення
insert()	вставка послідовності символів на зазначену позицію рядка
length()	отримання довжини рядка у байтах
max_size()	отримання максимального значення для розміру рядка
replace()	заміна послідовності символів у рядку вказаною
resize()	зміна довжини рядка
rfind()	пошук символа, починаючи з кінця рядка
size()	отримання довжини рядка у символах
substr()	отримання послідовності символів вказаного рядка
swap()	обмін змістом між двома об'єктами рядків

Спрощені формати та приклади роботи з цими методами наведено нижче.

string append(string str, **int** start, **int** num)

Аргументи:
str – рядок, символи якого буде додано;
start – номер першого символа послідовності у рядку str;
num – число символів послідовності.
Значення, що повертається: отриманий рядок.

string append(**int** count, string str)

Аргументи:
count – кількість повторів для доданих символів;
str – послідовність символів для додавання.
Значення, що повертається: отриманий рядок.

Приклад застосування:

```
string s1,s2;          //оголошення двох об'єктів класу роботи з рядками
s1="It's a beautiful"; //визначення об'єкта константним рядком
s2="Only one day";     //визначення об'єкта константним рядком
s1.append(s2,8,4);     //доповнити перший об'єкт підрядком з другого
s1.append(3,'!');      //доповнити перший об'єкт символом '!'
cout<<s1<<endl;        //вивести результат на консоль
```

Результат роботи, що буде виведено на консоль:

It's a beautiful day!!!;

string assign(string str, **int** start, **int** num)

Аргументи:

str – рядок, послідовність символів якого буде присвоєно;

start – номер першого символа послідовності у рядку str;

num – кількість символів.

Значення, що повертається: отриманий рядок.

char at(**int** num)

Аргументи:

num – номер символа, що повертається.

Значення, що повертається: символ, що знаходиться у рядку на позиції num.

const char *c_str()

Аргументи: відсутні.

Значення, що повертається: покажчик на перший елемент масиву символів.

int capacity()

Аргументи: відсутні.

Значення, що повертається: розмір зайнятої об'єктом пам'яті.

int compare(**int** start, **int** num, string str)

Аргументи:

start – позиція першого символа послідовності для порівняння;

num – кількість символів для порівняння;

str – рядок, з яким буде відбуватися порівняння.

Значення, що повертається: -1 – якщо вибрана послідовність за довжиною коротше ніж str; 1 – якщо вибрана послідовність довша ніж str; 0 – якщо вони є такими, що дорівнюються.

Приклад застосування:

```
string s1,s2,s3,s4;          //оголошення чотирьох об'єктів класу роботи з рядками
s1="123456789";              //визначення об'єкта числовим рядком
s2="123456789";              //визначення об'єкта числовим рядком
s3="1234567890";             //визначення об'єкта числовим рядком
s4="12345678";               //визначення об'єкта числовим рядком
//вивід на консоль результату порівняння
cout<<s1.compare(0,9,s2)<<"\r\n"<<endl;
//вивід на консоль результату порівняння
cout<<s1.compare(0,9,s3)<<"\r\n"<<endl;
```

```
//вивід на консоль результату порівняння
cout<<s1.compare(0,9,s4)<<"\r\n"<<endl;
```

Результат роботи, що буде виведено на консоль:

```
0
-1
1
```

int copy(**char** *ch, **int** num, **int** start)

Аргументи:

ch – покажчик на перший елемент масиву для запису послідовності з рядка;

num – кількість символів, що буде скопійовано;

start – номер початкового символа послідовності.

Значення, що повертається: кількість символів, що насправді було скопійовано.

bool empty()

Аргументи: відсутні.

Значення, що повертається: true – якщо рядок було очищено; false – якщо рядок ще має символи.

string erase(**int** start=0, **int** num=-1)

Аргументи:

start – номер першого символа, що буде видалено;

num – кількість символів для видалення.

Значення, що повертається: отриманий рядок.

int find(string str, **int** start=0)

Аргументи:

str – символ, чи послідовність символів для пошуку;

start – позиція у рядку, з якої треба почати пошук.

Значення, що повертається: позиція першого символа послідовності пошуку у рядку.

int find_first_not_of(string str, **int** start=0)

Аргументи:

str – символ чи сукупність символів для пошуку;

start – позиція у рядку, з якої треба почати пошук.

Значення, що повертається: позиція першого символа у рядку, який не входить до заданої послідовності.

int find_first_of(string str, **int** start=0)

Аргументи:
str – символ чи сукупність символів для пошуку;
start – позиція у рядку, з якої треба почати пошук.
Значення, що повертається: позиція першого символа у рядку, який входить до заданої послідовності.

int find_last_not_of(string str, **int** start=0);

Аргументи:
str – символ чи сукупність символів для пошуку;
start – позиція у рядку, з якої треба почати пошук.
Значення, що повертається: позиція останнього символа у рядку, який не входить до заданої послідовності.

int find_last_of(string str, **int** start=0)

Аргументи:
str – символ чи сукупність символів для пошуку;
start – позиція у рядку, з якої треба почати пошук.
Значення, що повертається: позиція останнього символа у рядку, який входить до заданої послідовності.

Приклад застосування:

```
string s="See also synchronous verb completion.";
cout<<s.find('e')<<"\r\n"<<endl;
cout<<s.find_first_of("hak")<<"\r\n"<<endl;
cout<<s.find_first_not_of("esa")<<"\r\n"<<endl;
cout<<s.find_last_of("fop")<<"\r\n"<<endl;
cout<<s.find_last_not_of("fop")<<"\r\n"<<endl;
cout<<s.rfind('o')<<"\r\n"<<endl;
```

Результат роботи, що буде виведено на консоль:

1 (*символ 'e' знайдено на 1-й позиції*)
4 (*символ 'a', який є у сукупності "hak", знайдено на 4-й позиції*)
0 (*символ 'S', якого немає у сукупності "esa", знайдено на 0-й позиції*)
34 (*символ 'o', який є у сукупності "fop", знайдено на 34-й позиції*)
36 (*символ '.', якого немає у сукупності "fop", знайдено на 36-й позиції*)
34 (*символ 'o' знайдено на 34-й позиції*)

string insert(**int** pos, string str)

Аргументи:
pos – позиція, з якої почнеться вставка послідовності символів;
str – послідовність символів для вставки.
Значення, що повертається: отриманий рядок.

int length()

Аргументи: відсутні.
Значення, що повертається: довжина рядка у байтах.

Приклад застосування:

```
string s1;                          //оголошення об'єкта класу роботи з рядками
s1="Today We have a lot of news"; //визначення об'єкта константним рядком
for (int i=0; i<s1.length(); i++)          //цикл перебору всіх символів рядка
{
        if (s1.at(i)=='a')                  //якщо поточний символ є 'a'
                cout<<"pos = "<<i<<"\r\n"<<endl;    //вивести номер позиції
}
```

Результат роботи, що буде виведено на консоль:

```
pos = 3
pos = 10
pos = 14
```

int max_size()

Аргументи: відсутні.
Значення, що повертається: максимально можлива довжина рядка.

string replace(**int** start, **int** num, string str)

Аргументи:
start – номер початкового символа послідовності, що буде замінено;
num – кількість символів рядка, що буде замінено;
str – послідовність символів, яка буде розміщена у рядку замість замінених.
Значення, що повертається: новий рядок.

void resize(**int** size)

Аргументи:
size – новий розмір рядка.
Значення, що повертається: відсутнє.

int rfind(string str, **int** start=-1)

Аргументи:
str – символ, чи послідовність символів для пошуку з кінця рядка;
start – номер символа, з якого почнеться пошук послідовності.
Значення, що повертається: номер першого символа послідовності, яка шукається з кінця рядка.

int size()

Аргументи: відсутні.
Значення, що повертається: розмір рядка у символах.

string substr(**int** start, **int** num)

Аргументи:
start – номер початкового символа послідовності;
num – кількість символів послідовності.
Значення, що повертається: вказана послідовність з рядку.

void swap(string s)

Аргументи:
s – об'єкт, з яким буде виконано обмін рядками.
Значення, що повертається: відсутнє.

Застосування класу роботи з рядками є досить поширеним. Це пов'язано з тим, що більшість інформації, яка обробляється людиною, є текстовою. Окрім того, текстовий формат подання даних є найбільш універсальним.

Висновки

У даному розділи було розглянуто наступне питання: клас роботи з рядками та його методи.

Контрольні питання

1) Який заголовний файл дозволяє застосовувати клас роботи з рядками?

2) Які прості операції можна виконувати з об'єктами класу роботи з рядками?

3) Які методи класу роботи з рядками дозволяють виконувати пошук символів чи послідовностей символів у рядку?

4) Які методи класу роботи з рядками дозволяють отримати розмір рядка у байтах та символах?

5) Які методи класу роботи з рядками дозволяють працювати з підрядками?

6) Як виконати обмін значеннями між двома об'єктами класу роботи з рядками?

7) За допомогою якого методу можна перевірити чи порожній рядок?

8) За допомогою якого методу можна отримати константний покажчик на символьний масив, що вміщує рядок об'єкта класу роботи з рядками?

9) Для чого застосовується метод capacity()?

10) З чим пов'язано широке застосування роботи з рядками при програмуванні?

4. РОБОТА З ФАЙЛАМИ

Навчальною метою розділу є ознайомлення читача про принципи роботи з даними, що зберігаються у вигляді файлів.

Внаслідок вивчення матеріалу даного розділу читач повинен вміти:

– застосовувати в програмі функції роботи з файлами;

– виконувати відкриття файлів у різних режимах;

– читати та записувати дані;

– працювати з курсором файлу;

– закривати файл.

4.1. Файловий ввід/вивід за допомогою потокових функцій

Файл – це дані, що збережено на носії інформації. Для роботи з файлами у мові C++ найчастіше застосовується ввід/вивід на основі потокових функцій.

Всі необхідні функції, макроси та типи даних, що забезпечують повний доступ до файлів, входять до вже знайомої бібліотеки "stdio.h". Для оперування файловою інформацією достатньо знати наступні функції, та типи даних, наведені у таблицях 4.1 й 4.2.

Таблиця 4.1

Типи даних для роботи з файлами

Службовий тип даних	Призначення
FILE	тип даних змінних-покажчиків на структуру роботи з файлом
size_t	цілочисельний тип даних, який визначає всі змінні для роботи з розміром файлу
fpos_t	цілочисельний тип даних, який визначає всі змінні для роботи з позицією курсору у файлі

Таблиця 4.2

Функції роботи з файлами

Назва функції	Призначення
fopen()	відкриває файл
fclose()	закриває файл
fputc()	записує символ до файлу
fputs()	записує рядок до файлу
fgetc()	вичитує символ з файлу
fgets()	вичитує рядок з файлу
fseek()	встановлює курсор на задану позицію файлу
ftell()	повертає позицію курсору у файлі
fprintf()	файловий аналог функції printf()
fscanf()	файловий аналог функції scanf()
feof()	повертає одиницю, якщо курсор знаходиться у кінці файлу
ferror()	повертає одиницю, якщо виникла помилка
fflush()	очищує потік

Формати оголошень функцій, що показані у таблиці, та приклади їх застосування наведено нижче:

FILE *fopen(**const char** **file_name*, **const char** **mode*);

Аргументи:
file_name – шлях до файлу;
mode – режим відкриття файлу.
Значення, що повертається: адресу структури роботи з файлом, яка буде застосована для всіх функціях.

Можливо використовувати наступні режими:

r – відкрити текстовий файл для читання;
w – створити текстовий файл для запису;
a – додати у кінець текстового файлу;
rb – відкрити бінарний файл для читання;
wb – створити бінарний файл для запису;
ab – додати у кінець бінарного файлу;
r+ – відкрити текстовий файл для читання/запису;
w+ – створити текстовий файл для читання/запису;
a+ – додати у кінець текстового файлу чи створити текстовий файл для читання/запису;
r+b – відкрити бінарний файл для читання/запису;
w+b – створити бінарний файл для читання/запису;
a+b – додати у кінець бінарного файлу чи створити бінарний файл для читання/запису.

int fclose(FILE **fp*);

Аргументи:
fp – адреса структури роботи з файлом.
Значення, що повертається: 1 – якщо виникла помилка.

int fputc(**int** *c*, FILE **fp*);

Аргументи:
fp – адреса структури роботи з файлом;
c – символ для запису.
Значення, що повертається: позиція курсору.

int fputs(**const char** **string*, FILE **fp*);

Аргументи:
fp – адреса структури роботи з файлом;
string – константний рядок для запису.

Значення, що повертається: позиція курсору.

int fgetc(FILE **fp*);

Аргументи:
fp – адреса структури роботи з файлом.
Значення, що повертається: символ.

char *fgets(**char** **string*, **int** *n*, FILE **fp*);

Аргументи:
fp – адреса структури роботи з файлом;
string – адреса символьного рядка;
n – кількість символів.
Значення, що повертається: адреса символьного рядка.

int fseek(FILE **fp*, **long** *offset*, **int** *origin*);

Аргументи:
fp – адреса структури роботи з файлом;
offset – крок зсуву;
origin – макрос визначення початкової позиції для зсуву курсору.
Можливо використовувати наступні макроси:
SEEK_CUR – починати з позиції курсору;
SEEK_END – починати з кінця файлу;
SEEK_SET – починати з початку файлу.
Значення, що повертається: нуль, якщо була виповнена успішно.

long ftell(FILE * *fp*);

Аргументи:
fp – адреса структури роботи з файлом.
Значення, що повертається: позиція курсору у файлі.

int fprintf(FILE * *fp*, **const char** **format* [, *argument*]...);

Аргументи:
fp – адреса структури роботи з файлом;
format – рядок, що форматується;
argument – список змінних.
Значення, що повертається: кількість записаних байтів.

int fscanf(FILE * *fp*, **const char** **format* [, *argument*]...);

Аргументи:
fp – адреса структури роботи з файлом;
format – рядок, що форматується;
argument – список змінних.
Значення, що повертається: кількість зчитаних байтів.

int feof(FILE * *fp*);

Аргументи:
fp – адреса структури роботи з файлом.
Значення, що повертається: значення EOF, якщо курсор у кінці файлу.

int ferror(FILE * *fp*);

Аргументи:
fp – адреса структури роботи з файлом.
Значення, що повертається: номер помилки чи нуль, якщо помилки не було.

int fflush(FILE * *fp*);

Аргументи:
fp – адреса структури роботи з файлом.
Значення, що повертається: нуль, якщо виконана успішно, та EOF, якщо виникла помилка.

Приклад програми, що відкриває текстовий файл та виводить його зміст на консоль:

```
#include <stdio.h>                  //підключення бібліотеки вводу/виводу

void main()                         //оголошення та визначення головної функції
{
//оголошення змінної-покажчика на структуру роботи з файлом
        FILE *f;
        f=fopen("text.txt", "r");   //відкриття файлу text.txt для читання
        while (!feof(f))            //цикл до визначення кінця файлу
        {
                printf("%c", fgetc(f));   //посимвольний вивід тексту на консоль
        }
        printf("\r\n");             //перехід на наступний рядок
        fclose(f);                  //закриття файлу
}
```

Приклад програми, що записує рядок "This file was created by _" з введеним користувачем ім'ям у файл output.txt.

```
#include <stdio.h>                    //підключення бібліотеки вводу/виводу
#include <string>                     //підключення бібліотеки роботи з рядком
#include <iostream>        //підключення бібліотеки потокового вводу/виводу

using namespace std;                  //застосування простору імен

void main()                           //оголошення та визначення головної функції
{
//оголошення змінної-покажчика на структуру роботи з файлом
        FILE *f;
        string st;                    //оголошення об'єкта класу роботи з рядками
        f=fopen("output.txt", "w");   //створення файлу output.txt для запису
        cout<<"Your name:  ";         //вивід запиту до користувача
        cin>>st;                      //отримання значення від користувача
        fprintf(f, "This file was created by %s", st.c_str());//запис тексту у файл
        fclose(f);                    //закриття файлу
}
```

Приклад програми, що вичитує текст з файлу text.txt та формує з нього у файлі output.txt псевдографічний трикутник:

```
#include <stdio.h>                    //підключення бібліотеки вводу/виводу
#include <string>                     //підключення бібліотеки роботи з рядком
#include <iostream>        //підключення бібліотеки потокового вводу/виводу

using namespace std;                  //застосування простору імен

void main()                           //оголошення та визначення головної функції
{
//оголошення змінної-покажчика на структуру роботи з файлом
        FILE *f;
        string st, sn;                //оголошення об'єктів класу роботи з рядками
        int n=0;//оголошення та визначення змінної лічильника номера символа

        f=fopen("text.txt", "r");     //відкриття файлу text.txt для читання
        while (!feof(f))                      //цикл до визначення кінця рядка
        {
                st+=fgetc(f);                 //формуємо рядок
        }
        fclose(f);                            //закриття файлу

        for (int i=0; i<15; i++)              //цикл за рядками трикутника
```

```
    {
        for (int j=0; j<(15-i); j++)        //цикл для додавання відступів
            sn+='  ';                       //додати у рядок символ пробілу
        for (int k=0; k<2*i+1; k++)//цикл для додавання символів тексту
        {
            sn+=st.at(n);                   //додати один символ у рядок
            n++;                    //збільшення номера символа на одиницю
        }
        sn+="\r\n";                         //додати перехід на новий рядок
    }
    cout<<sn;                               //вивід результату на консоль
    f=fopen("output.txt", "w");             //створення файлу для запису
    fprintf(f, "%s", sn.c_str());           //запис рядка у файл
    fclose(f);                              //закриття файлу
}
```

Висновки

У даному розділі було розглянуто наступне питання: функції для потокової роботи з даними, що збережено у вигляді файлів.

Контрольні питання

1) Який заголовний файл підключає функції роботи з файлами?
2) Які спеціальні службові типи даних зазначено для роботи з файлами?
3) За допомогою якої функції виконується відкриття файлу?
4) У яких режимах може бути відкрито файл?
5) Яка функція дозволяє отримати позицію курсору файлу?
6) За допомогою яких функцій виконується читання тексту з файлу?
7) Як у програмному коді перевірити чи досягнуто при читанні кінець файлу?
8) Які значення містять змінні типу FILE?
9) Які функції виконують запис даних у файл?
10) Для чого слугує функція fflush()?

ЧАСТИНА IV
ЛАБОРАТОРНИЙ ПРАКТИКУМ

5. ЛАБОРАТОРНІ РОБОТИ

Лабораторний практикум дисципліни "Комп'ютерні технології та програмування" складається з 15 робіт, що надають студентові можливість отримати практичні навички програмування на мові С++. Лабораторні роботи охоплюють наступні теми:

– розробка схем алгоритмів систем та програм;
– типи даних та операції мови С++;
– застосування математичної бібліотеки для вирішення складних математичних виразів;
– розгалуження процесу виконання програми;
– циклічні дії у програмі;
– роботи з одновимірними та двовимірними масивами;
– застосування змінних-покажчиків;
– робота з динамічними масивами;
– алгоритми сортування;
– створення та застосування структур;
– розробка функцій для вирішення підзадач;
– інкапсуляція на основі класів;
– спадкування властивостей класу;
– робота з рядками;
– робота з файловими даними;
– багатофайлові проекти.

Приступаючи до виконання лабораторної роботи, студенти повинні ознайомитись з теоретичними відомостями та методичними вказівками, а після закінчення – оформити звіт, що включає схему алгоритму, програмний код з коментуванням та зроблені висновки. Для самопідготовки до захисту кожна лабораторна робота містить перелік питань до опрацьованих тем.

5.1. Лабораторна робота №1: Побудова схем алгоритмів для описання простих та складних процесів і програм

Мета роботи

Ознайомитись з символами даних та отримати практичні навички побудови схем алгоритмів.

Хід роботи

1) Ознайомитись з методичними вказівками до лабораторної роботи та "ГОСТ 19.701-90" побудови схем алгоритмів;

2) у відповідності до варіанта завдання розробити схему алгоритму роботи об'єкта, що описано у завданні;

3) виконати схему алгоритму на аркуші А4 згідно "ГОСТ 19.701-90";

4) описати схему алгоритму, посилаючись на номери блоків;

5) зробити висновки;

6) підготувати звіт про виконання лабораторної роботи, що включає наступні пункти: номер, тема, мета та хід лабораторної роботи, завдання, схема алгоритму роботи процесу, опис схеми алгоритму, висновки.

Теоретичні відомості та рекомендації до виконання

Алгоритм – це словесний, рукописний чи схемний опис етапів роботи (діяльності) будь-якого об'єкта з певною ступінню деталізації.

У житті кожної людини кожен день розплановано – час пробудження, час прийняття їжі, час роботи і відпочинку тощо. Залежність щоденних планів від дня тижня, пори року, подій, що виникають у житті, дозволяє здійснювати логічні розгалуження. Таким чином, кожен день життя людини може бути подано у вигляді алгоритму з вірогідними подіями.

При програмуванні на будь-якій з алгоритмічних мов, складання схем алгоритмів грає найбільш важливу роль – формування схематичного уявлення особливостей роботи програми, що розробляється. Завдяки схемам алгоритмів, програміст може передбачати ймовірну поведінку виконання своєї програми, ще не приступивши до самого процесу програмування.

Кожна мова програмування заснована на суворих правилах формування програм. Вони потрібні для того, щоб написаний код було правильно розпізнано комп'ютером, і програміст отримав саме той продукт, який був запланований. Без дотримання цих правил, написана програма втрачає цілісність своєї логічної структури. Саме завдяки такому підходу всі мови програмування є алгоритмічними, і всі програми формуються відповідно до вбудованих синтаксичних і логічних правил.

Таким чином, програма – це алгоритм виконання певних задач комп'ютером, поданий у формі переліку команд, а схема алгоритму програми – її схематичне відображення.

Використовуючи побудовану схему алгоритму за основу для написання програмного коду, можна у значній мірі полегшити як розуміння ходу роботи програми, так і сам процес програмування в цілому.

При виконанні завдання до лабораторної роботи студент повинен розібратися з особливостями побудови схем алгоритмів, що описують дію технологічних об'єктів чи виконання простих процесів. Це буде основою для розуміння принципів розробки схем алгоритмів програм лабораторного курсу дисциплін "Комп'ютерні технології та програмування" та "Алгоритмічні мови і програмування".

Виконання лабораторної роботи починається ознайомленням з теоретичними відомостями про символи і особливості побудови схем алгоритмів на основі "ГОСТ 19.701-90". Засвоївши їх, студент виконує розробку і побудову схеми алгоритму роботи об'єкта відповідно до завдання за варіантом. Для закріплення отриманих у ході виконання роботи знань, побудова виконується олівцем на аркуші формату А4.

Згідно з завданням до лабораторної роботи необхідно побудувати схему алгоритму роботи об'єкта.

Студент починає виконання роботи з формування схеми алгоритму за "ГОСТ 19.701-90" на аркуші в клітинку. Потім, коли вже виправлено всі помилки і схема отримала кінцевий вигляд, вона переноситься на аркуш А4.

Описання схеми виконується при розгляді формуючих її символів. Причому, для зв'язку з схемою в описанні вказуються посилання на номери відповідних блоків.

У висновках до звіту студент повинен розкрити призначення теми "Побудова схем алгоритмів для описання простих та складних процесів і програм". Висновок має бути поданий в формі пояснення сфери застосування вивченого матеріалу згідно із розумінням його студентом.

Завдання для самостійного виконання

Розробити схему алгоритму роботи об'єкта, опис якого наведено у табл. 5.1 за варіантом завдання. Вхідні дані з датчиків і сигнали зобразити за допомогою блоку ручного вводу, вихідні – блоку дисплей.

Таблиця 5.1

Варіанти завдань

№ Вар.	Опис об'єкта
1	Є три конвеєри: верхній А, верхній Б та нижній В. Для переміщення вантажу, що надходить з конвеєрів А і Б, на нижній конвеєр використовується робот-маніпулятор. Принцип його роботи наступний: при надходженні сигналу про те, що прийшов вантаж з конвеєра А, маніпулятор відкриває захват, робить підйом, поворот на 180 градусів, чекає сигналу "вантаж у захваті", закриває захват, робить поворот у зворотній бік, опускається і відкриває захват, при надходженні сигналу "вантажу немає", захват закривається. При появі вантажу з конвеєра Б дії виконуються у відповідності до конвеєра А, однак поворот здійснюється на 270 градусів. У разі неотримання сигналу "вантаж у захваті" або "вантажу немає" протягом 5 секунд вмикається лампа "ПОМИЛКА" і маніпулятор повертається у початкову позицію.
2	Для автотранспорту розміщений світлофор з чотирма станами: червоний, жовтий, зелений, поворот. Його принцип дії наступний: 30 секунд горить червоне, потім 10 – жовте, 30 – зелене, якщо є автотранспорт, що збирається повертати, то надходить сигнал "4-й стан" і після 10 секунд роботи зеленого включається поворот. Відключення зеленого і повороту відбувається одночасно.
3	Нагрівач води працює наступним чином: при включенні води також включаються газові пальники. Температура води контролюється на виході. Якщо вона нижча виставленого на нагрівачі значення нижньої межі, то пальники розпалюються сильніше, якщо вище верхньої межі – інтенсивність зменшується. На нагрівачі можна виставити 4 значення: відключений, 30-45, 45-50, 50-60.

Продовження табл. 5.1

№ Вар.	Опис об'єкта
4	Пральна машинка працює в одному з трьох режимів, що задаються за допомогою перемикача: звичайне прання (40 хв), лагідне прання (1 год), прання з прасуванням (1 год 20 хв). Для кожного режиму можна включити функцію швидке прання, яка зменшує час прання на 20 відсотків та збільшує обороти на 10 відсотків. Запуск виконується при натисканні кнопки "ПУСК". Процес складається з чотирьох етапів: попереднє полоскання (10% часу), прання (70% часу), полоскання (10% часу), віджим (10% часу). Перші три етапи починаються з закачування води, а четвертий зі злиття. Якщо води немає чи прання закінчилося, то на екрані з'являється відповідне повідомлення і машинка переходить у режим очікування. У третьому режимі обороти на другому етапі знижені на 50 відсотків.
5	Ігровий автомат працює відповідно до алгоритму: до опускання у віконце монети він перебуває в режимі очікування. Коли монету опущено, вона перевіряється і, якщо має відповідний номінал, розблокується важіль розкручування барабанів. При відсутності запуску протягом 2 хвилин, автомат блокує важіль і входить в режим очікування. Якщо важіль було опущено вниз, то проводиться почергове розкручування трьох барабанів, важіль повертається на місце і блокується. Після зупинки барабанів перевіряється збіг значення на них: збігаються – гравцеві видаються 10 монет і здійснюється перехід в режим очікування, не збігаються – перехід в режим очікування.
6	Для закриття дверей від небажаних відвідувачів використовується кодовий замок, що складається з 11 кнопок: 10 цифр і 1 команда. При натисканні на кнопку команди перевіряється значення, що було набрано за допомогою цифр. Для того, щоб двері відчинилися, необхідно набрати 3 цифри. При правильному наборі двері відкриваються на 10 секунд. Після їх закінчення двері знову закриваються. Якщо комбінацію введено невірно, видається сигнал помилки.
7	Електрочайник дозволяє вибрати один з трьох режимів: 1-й – "нагріти воду", 2-й – "закип'ятити воду", 3-й – "прокип'ятити воду". У всіх режимах при включенні спіраль розігрівається до 300 градусів. При виборі 1-го режиму, коли вода досягає температури в 70 градусів, чайник вимикається. При 2-му режимі після закипання води, спіраль остигає до 200 градусів, і вода кипить протягом 1 хвилини, потім чайник вимикається. Третій режим відповідає 2-му, але після закипання чайник працює ще 10 хвилин.
8	Дитяча машинка "Місяцехід" працює наступним чином: починається рух вперед, якщо надходить сигнал "далі дороги немає", машинка від'їжджає на 10 см назад, повертається направо і знову їде вперед. Якщо при русі назад надходить сигнал "далі дороги немає" машинка проїжджає 3 см вперед, повертає ліворуч і їде далі.

Продовження табл. 5.1

№ Вар.	Опис об'єкта
9	Два послідовних конвеєри виробляють доставку вантажу з точки А в точку Б. Система керування запускає перший конвеєр, якщо надійшов сигнал про наявність вантажу і зупиняє його через 1 хвилину після отримання сигналу про те, що вантажу більше немає. Другий конвеєр запускається через 50 секунд і зупиняється через 1 хвилину після першого. У разі незапланованої зупинки першого конвеєра система повідомляє про помилку, другий вимикається через 1 хвилину. Якщо не заплановано був зупинений другий конвеєр, перший зупиняється відразу.
10	Пристрій трансивер з'єднує дві ділянки лінії передачі А і Б. При отриманні повідомлення "ЗАПИТ" з будь-якої ділянки він відповідає повідомленням "ВІДПОВІДЬ". При отриманні повідомлення "ДАНІ" трансивер включає режим підсилення сигналу і передає їх з однієї ділянки в іншу і очікує 10 мс повідомлення "ПІДТВЕРДЖЕНО" коректну доставку. Якщо з якоїсь причини повідомлення не надходить, трансивер відсилає до ділянки, з якої було отримано дані, повідомлення "ПОМИЛКА ПЕРЕДАЧІ". Коли повідомлення "ЗАПИТ" чи "ДАНІ" однієї з ділянок не приходить протягом 30 мс, пристрій посилає до іншої ділянки повідомлення "ВТРАЧЕНО ЗВ'ЯЗОК".
11	Система виклику ліфта багатоповерхового будинку працює наступним чином: під час виклику ліфта перевіряється різниця між номером поточного поверху і номером поверху, з якого було зроблено виклик. Якщо різниця негативна, ліфт починає рухатися вгору, якщо позитивна, то кабіна рухається вниз. Коли різниця дорівнює нулю, дверцята ліфту відкриваються. Розгін кабінки проходить в три етапи – відповідно на малій, середній і великій швидкості. Контроль положення виконується після проходження кожного чергового поверху. Якщо різниця дає число більше або таке, що дорівнює трьом, то здійснюється повний розгін ліфта, якщо менше, то рух ліфта вповільнюється. Повний розгін/гальмування кабінки відбувається за 3 поверхи. Може бути натиснуто кнопку тільки на одному поверсі.
12	Навантажувальний кран працює наступним чином: До команди "ПУСК" він знаходиться в стані готовності. Коли команду було подано, перевіряється з якої платформи необхідно забирати вантаж і, відповідно з цим, відбувається переміщення по 1, 2 або 3 шляху. При досягненні платформи кран опускає магнітний захват і включає електромагніт. Потім захват піднімається, і кран повертається до початкової позиції, розвертається на 180 градусів, опускає захват і відключає електромагніт. Після цього захват знову піднімається, кран розвертається і переходить в режим очікування команди з однією з платформ. Для всіх платформ алгоритм однаковий.

Продовження табл. 5.1

№ Вар.	Опис об'єкта
13	Два пристрої А і Б з'єднано лінією зв'язку в мережу. Алгоритм ініціалізації пристрою ініціатора, виявлення обриву лінії зв'язку і передачі повідомлень працює наступним чином: після включення пристрої перебувають в режимі ініціалізації. Пристрій А з інтервалом в 1 секунду передає повідомлення зі своїм номером, в проміжках чекаючи такого самого повідомлення від пристрою Б. Коли повідомлення отримано, проводиться порівняння номерів, і пристрій, що володіє більшим номером, стає ініціатором опитування лінії. Проводиться перехід в режим опитування та передачі. Пристрій ініціатор відправляє повідомлення "ЗАПИТ" і протягом 10 мс очікує повідомлення "ВІДПОВІДЬ". Якщо воно отримано, то відправляється новий запит. Коли "ВІДПОВІДЬ" не отримано пристрій посилає "ЗАПИТ" ще двічі, і при тому самому результаті вмикає червоний світлодіод. Кожного разу перед посилкою "ЗАПИТУ" перевіряється, чи є дані для передачі: якщо є, – замість "ЗАПИТУ" передаються дані; якщо немає, – "ЗАПИТ". Передача даних не відрізняється від передачі "ЗАПИТУ" – відразу після 10 мс очікується "ВІДПОВІДЬ". Якщо пристрій не ініціатор, то він працює у пасивному режимі – на отриманий "ЗАПИТ" передається "ВІДПОВІДЬ", а коли для передачі є дані, то вони передаються замість повідомлення "ВІДПОВІДЬ".
14	Вугільна шахта обладнана вентиляторною установкою, робота якої полягає в наступному: в нормальному режимі працює тільки головний вентилятор на середніх оборотах. У випадку, якщо з одного із датчиків метану надходить сигнал, вентилятор включається на повну потужність. При виключенні сигналу протягом 5 хвилин потужність знову спадає до середньої. В іншому випадку включається додатковий вентилятор. Якщо сигнал не зникає після 10 хвилин роботи двох вентиляторів, то подається сигнал про можливу небезпеку вибуху і евакуації персоналу шахти. Коли основний вентилятор зупиняється у зв'язку з аварією, то в диспетчерську шахти передається аварійний сигнал і включається додатковий вентилятор. Після усунення причин аварії і включення основного вентилятора – додатковий відключається.
15	Прохідницький комбайн вугільної шахти працює у відповідності з наступним алгоритмом: він встановлюється в одному з двох паралельних тунелів (приймаємо, що в лівому), між якими знаходиться вугільний пласт, що розробляється. Після команди "ПУСК" комбайн включає бурову установку і починає рух вправо по прокладених рейках, що складаються з декількох десятків секцій. Коли секція залишається позаду, спеціальний механізм переміщує її на крок вперед. Після завершення проходу вправо (до другого тунелю), комбайн переміщається на крок вперед і починає рух вліво. Цикл триває до закінчення запланованого пласта (N кроків).

Продовження табл. 5.1

№ Вар.	Опис об'єкта
16	Система керування, що дозволяє іграшковій машинці об'їжджати перешкоди, працює таким чином: машинка їде вперед, постійно використовуючи вбудований ехолокатор. До тих пір, поки на відстані 3-х метрів попереду немає перешкод, відбувається рух вперед. Коли з'являється перешкода, машинка зупиняється і починає пошук шляху. Ехолокатор повертатися вправо з кроком в 10 градусів і посилає хвилі. Якщо до досягнення 90 градусів буде знайдено шлях, машинка повернеться і поїде у відповідному напрямку, а ехолокатор повернеться у початкове положення. Коли справа шляху немає, ехолокатор повертається у початкове положення і починає покроково повертатися вліво. Принцип той самий. Якщо шляху немає ні праворуч, ні ліворуч, машинка робить розвертання і починає дослідження шляху назад за тим самим алгоритмом.
17	Банкомат працює наступним чином: після сигналу "КАРТКА" виводиться вікно вибору мови (російська, англійська, українська). При виборі здійснюється підключення відповідної бази. Далі запитується ПІН-код, що складається з 4 цифр. Якщо код введено правильно, то з'являється вікно вибору дії. Якщо ні, – пропонується ввести код ще раз. Після 3-х неправильних вводів банкомат повідомляє про вилучення картки і переходить в режим очікування. Вікно вибору дій включає: друк чеку, вибір суми для зняття і вихід. При виборі друку чека звіряється баланс і друкується чек, потім пропонується продовжити роботу з банкоматом (ТАК – вікно вибору дій, НІ – викид картки і перехід в режим очікування). Вибір суми пропонує ввести суму, потім значення звіряється з балансом і, якщо воно є меншим, то видаються гроші, а, якщо ні, – пропонується ввести іншу суму або виконати іншу дію (перехід до вікна вибору). При виборі "вихід" – повертається картка і відбувається перехід в режим очікування.
18	Прес для стиснення металевого брухту працює за наступним алгоритмом: лом надходить в приймальний короб по конвеєрній стрічці, при цьому постійно фіксується його наявність. Коли прес готовий до роботи, включається стрічка та брухт починає завантажуватися в приймальний короб до тих пір поки не отримано сигнал від одного з двох датчиків: "короб заповнено" або "оптимальна вага". Після сигналу конвеєр зупиняється і починається процес стиснення. Коли стиснення закінчено, перевіряється сигнал "оптимальна вага" і, якщо він є, то пакет висувається назовні, а прес готується до наступної обробки. Якщо сигналу немає, то знову включається конвеєр і відповідні дії повторюються. Коли брухту більше немає, прес переходить в режим очікування. Це відбувається як при пустому коробі, так і при неоптимальній вазі пакету.

Продовження табл. 5.1

№ Вар.	Опис об'єкта
19	Система, яка контролює прохід в метро працює за наступним алгоритмом: до отримання жетона або квитка система знаходиться в режимі очікування – прохід закритий. Коли людина кидає жетон або вставляє багаторазовий квиток, система включається. Якщо було кинуто жетон метро, то прохід відкривається на 10 секунд. Монета або жетон іншого типу не повертаються, прохід залишається закритим. Коли був виявлений квиток, то перевіряється його значення (може бути розраховані на 5, 10, 25, 50 проїздів) і поточний стан. У випадку, якщо квиток не є дійсним для проїзду в метро – він вилучається, прохід залишається закритим. Якщо квиток відповідний, вираховується різниця і приймається одне з трьох рішень: результат "0" – квиток вилучається, прохід закритий; результат "1" – квиток вилучається, прохід відкривається на 10 секунд; результат більше "1" – ставиться штамп на задній стороні квитка, поточне значення зменшується на "1", прохід відкривається на 10 секунд.
20	Цифровий фотоапарат після включення перевіряє, який з трьох режимів зйомки було обрано: 1-й – авто, 2-й – далекі об'єкти, 3-й – вночі. Далі виконується перехід в режим очікування. При напівнатиснутій кнопці "КАДР" виконується підлаштовування фотоапарата відповідно до режиму: 1-й – фокусування на об'єкті(ах) в межах 10 метрів; 2-й – фокусування на об'єкті(ах) на відстані 50 - 100 метрів; третя – підвищення контрастності і яскравості, фокусування на об'єкті в межах 10 метрів, встановлення тривалої витримки. При відпусканні кнопки відбувається скидання виставлених значень. При повному натисканні кнопки фотоапарат робить кадр відповідно до виставлених параметрів. Якщо кадр було зроблено, фотоапарат дозволяє переглянути його після натискання на кнопку "ПЕРЕГЛЯД", друге натиснення повертає фотоапарат в стан готовності.
21	Мережний пристрій працює наступним чином: при включенні він посилає кадр зі своїм номером, щоб майстер мережі знав про підключення нового вузла. При помилці надсилається повідомлення "ПОМИЛКА". За умовчуванням пристрій знаходиться в режимі прослуховування лінії. Якщо з лінії приходять дані, то вони перевіряються на коректність і правильні – обробляються. При неправильних лічильник помилок збільшується на 1 і надсилається повідомлення "ПОМИЛКА ДАНИХ". Потім відновлюється режим прослуховування. Коли лічильник досягає значення 255, пристрій посилає повідомлення про те, що він входить в режим самоналаштовування і перестає слухати лінію. Проводиться самоналаштовування і потім знову підключення до лінії зв'язку в режимі прослуховування, лічильник помилок під час самоналаштовування обнуляється.

Закінчення табл. 5.1

№ Вар.	Опис об'єкта
22	Зрошувальна установка на прохідницькому комбайні включається у разі високої запиленості поруч з комбайном. Вона працює таким чином: перевіряється рівень запиленості і, якщо він вище норми, установка отримує команду "ПУСК". Далі проводиться перевірка, в якому напрямку рухається комбайн, і відповідно до напрямку руху включаються праві або ліві зрошувачі. Система змочує породу до її розробки. Коли комбайн зупиняється, зрошення не проводиться. Напрямок руху перевіряється на стадії включення системи і на стадії зміни руху комбайна. Якщо рівень запиленості знижується на 25% нижче нормального, система отримує команду "СТОП".
23	Електронний годинник з будильником працюють за наступним алгоритмом. Після включення він пропонує ввести поточний час. Коли введення виконано (натискання кнопки "ПУСК"), годинник починає змінювати своє значення з кроком в 1 секунду. Причому, коли число секунд дорівнює 60, воно скидається в 0, а до числа хвилин додається 1. Коли число хвилин дорівнює 60, воно скидається в 0 і до числа годин додається 1. Якщо число годин досягає 24, воно скидається в 0. При натисканні кнопки "БУД" включається режим виставлення будильника. Будильник вважається встановленим, якщо натиснута кнопка "ПУСК". Одного разу на хвилину значення будильника буде звірятися з поточним і, коли вони зрівняються, протягом хвилини лунатиме сигнал. Під час виставлення будильника годинник працює.
24	Робота принтера, підключеного до персонального комп'ютера, полягає в наступному: при включенні принтер розігрівається протягом 1 хвилини. Потім він очікує подачі команди з комп'ютера. Якщо приходять дані для друку, здійснюється перевірка наявності паперу. Якщо папір є, то друкується лист. Якщо ні – вмикається червона лампа. Папір може бути в приймальному лотку або в приймальному вікні. Якщо папір є у вікні, то він затискається валиком. Спочатку використовується лист з вікна, а вже потім перевіряється лоток. Якщо на стадії виконання з'являється помилка, вмикається червона лампа. Якщо друку не було, розігрів принтера відбувається кожні 10 хвилин.
25	Холодильна установка працює наступним чином: повітря охолоджується за допомогою двох морозильних установок: МУ1 і МУ2. Температура контролюється за допомогою трьох датчиків: верхнього рівня – ДВУ, середнього – ДСУ та нижнього – ДНУ. Індикація роботи проводиться за допомогою 3-х світлодіодів: червоного, зеленого та білого. Умови роботи: при температурі нижче ДНУ працюють МУ1, МУ2 і світлодіоди вимкнені; при температурі між ДНУ та ДСУ працює МУ2 і включений білий світлодіод; при температурі між ДСУ і ДВУ працює МУ1 і включений зелений світлодіод; при температурі вище ДВУ працюють МУ1 і МУ2 і включений червоний світлодіод.

Приклад виконання завдання

Розробити схему алгоритму роботи водонапірної башти. Вхідні дані з датчиків і сигнали зобразити за допомогою блоку ручного вводу, вихідні – блоку дисплей.

Принцип роботи водонапірної башти полягає у наступному: рідина подається двома насосами: Н1 і Н2. Її рівень контролюється за допомогою трьох датчиків: верхнього рівня – ДВР, середнього рівня – ДСР і нижнього рівня – ДНР. Індикація роботи насосів виконується за допомогою 3-х ламп: червоної, зеленої та білої. Умови роботи насосів і ламп: при рівні рідини нижче за ДНР працюють Н1, Н2 і всі лампи виключені; при рівні між ДНР та ДСР працює Н2 і ввімкнено зелену лампу; при рівні між ДСР і ДВР працює Н1 і ввімкнуто білу лампу; при рівні вище ДВР Н1 та Н2 виключені, ввімкнуто червону лампу.

Розробимо схему алгоритму роботи водонапірної башти (рис. 5.1.)

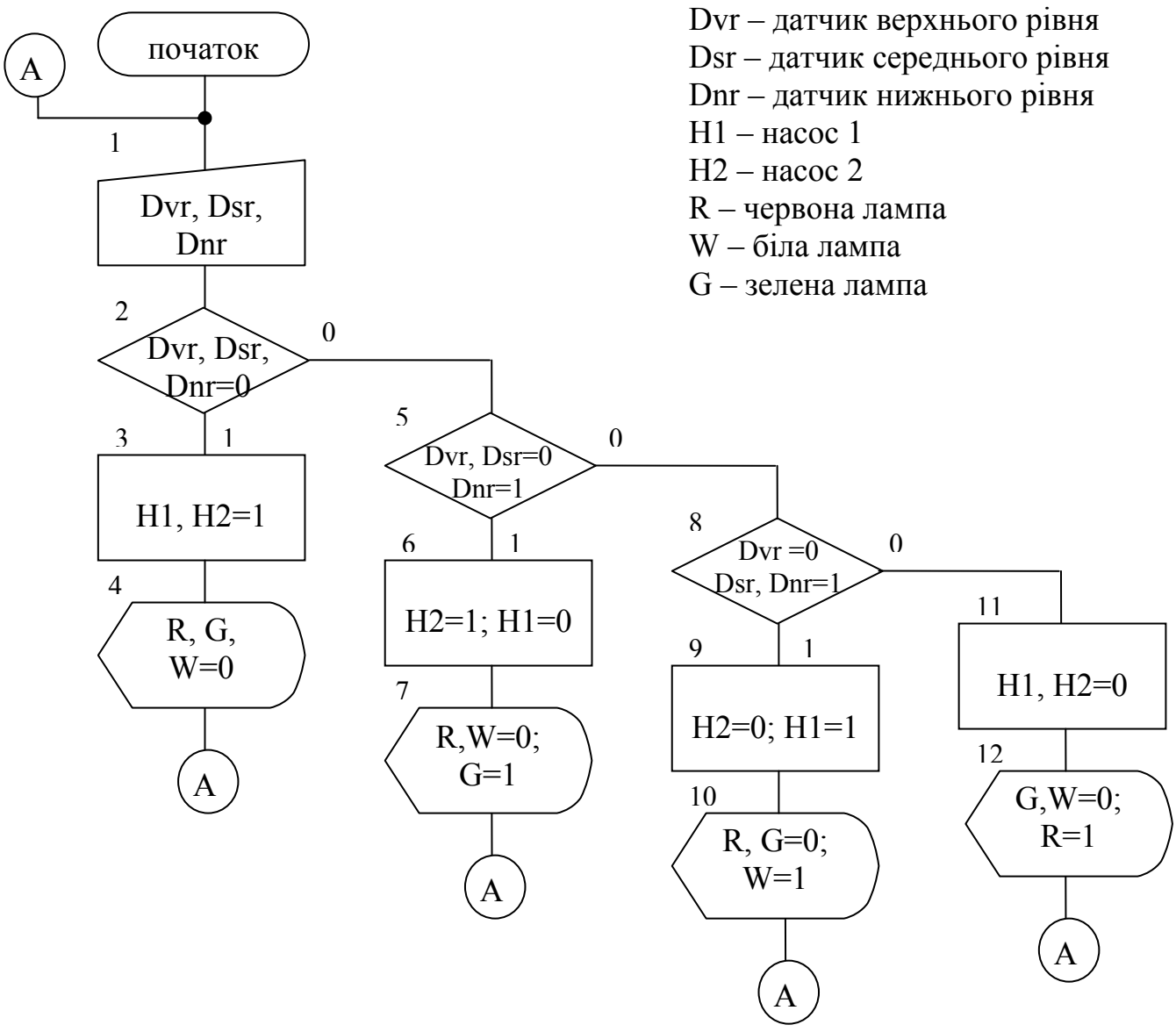


Рис. 5.1. Схема алгоритму роботи водонапірної башти

Опис схеми алгоритму роботи водонапірної башти:

Робота процесу починається з опитування датчиків рівня (блок 1). Потім виконується перевірка спрацьовування датчиків. Якщо не спрацював жоден з них (блок 2), то вмикаються обидва насоси (блок 3), лампи відключені (блок 4). Якщо спрацьовує датчик нижнього рівня (блок 5), то працює тільки другий насос (блок 6) і ввімкнено зелену лампу (блок 7). При спрацьовуванні датчиків нижнього і середнього рівнів (блок 8) працює тільки перший насос (блок 9) і ввімкнено білу лампу (блок 10). У випадку, коли спрацьовують всі датчики, обидва насоси вимикаються (блок 11) і включається червона лампа (блок 12). Процес виконується по колу (блок сполучувач А) до тих пір, поки систему не буде вимкнено.

Питання для підготовки до захисту лабораторної роботи

1) Для чого використовуються схеми алгоритмів?

2) Як відносяться одна до одної сторони блоків?

3) Що таке лінія поєднання?

4) Як можуть перетинатися лінії поєднання блоків?

5) Якими способами за "ГОСТ 19.701-90" виконується розривання лінії поєднання?

6) Для чого слугує блок зумовленого процесу?

7) В чому полягає сенс застосування блоку підготовки?

8) Для чого використовується блок рішення? Скільки виходів він може мати?

9) В яких випадках при використанні ліній поєднання є необхідним застосовувати стрілки?

10) Для чого слугує і як використовується блок сполучувач?

11) Як виконується додавання коментарів у схему алгоритму?

12) Що таке ідентифікатор символа? Для чого він слугує?

5.2. Лабораторна робота №2: Типи даних та прості алгоритмічні операції

Мета роботи

Отримати практичні навички розробки простих програм на мові С++.

Хід роботи

1) Ознайомитись з методичними вказівками до лабораторної роботи та темою "Структура програми на мові С++";

2) у відповідності до варіанта завдання розробити схему алгоритму роботи програми розрахунку математичної формули;

3) розрахувати задану формулу, підставивши відповідні до варіанта значення. Якщо змінні X, Y, Z задані типом даних вибрати довільні значення X, Y, Z;

4) за схемою алгоритму на мові С+ написати програму для розрахунку формули та отримати результати її роботи. При необхідності внести зміни до формули (у висновках зазначити причину змін);

5) порівняти значення, що були отримані при програмному та ручному розрахунках;

6) зробити висновки (у випадку, якщо відповідь отримана при ручному розрахунку, відрізняється від результату програми, зазначити обґрунтування);

7) підготувати звіт про виконання лабораторної роботи, що включає наступні пункти: номер, тема, мета та хід лабораторної роботи, завдання, розрахунок формули, схема алгоритму програми, лістинг програми з коментуванням кожної інструкції, результат роботи програми (скріншот), висновки.

Теоретичні відомості та рекомендації до виконання

Основою будь-якої мови програмування є дії, які можна зробити зі значеннями змінних. Більшість програм зводяться лише до виконання арифметичних, логічних або порозрядних операцій. Мова програмування дозволяє зробити розрахунок величин з можливістю введення початкових значень і, практично миттєвого, отримання результатів обчислень будь-якого ступеня складності у вигляді програми алгоритму. Це робить мови програмування дуже зручними для створення спеціалізованих математичних додатків.

Однак типи даних і змінні потрібні не тільки для обчислень, але й для збереження значень у пам'яті. Жодна програма не обходиться без використання змінних.

При виконанні завдання до лабораторної роботи студент повинен ознайомитись з операціями, які можна виконувати при програмуванні на мові С++ на прикладі рішення математичного виразу.

Виконання лабораторної роботи починається ознайомленням з теоретичними відомостями про зазначені в мові С++ типи даних, службові слова, що впливають на типи даних, та оператори для виконання простих

арифметичних, логічних і порозрядних операцій. Засвоївши основи застосування змінних і типів даних, студент приступає до розрахунку формули за завданням та розробки схеми алгоритму програми для машинного розрахунку. За схемою алгоритму виконується написання програми на мові С++.

Отримавши результат програмного розрахунку, студент повинен порівняти його з розрахованим вручну, а потім зробити висновок про відповідність значень. Якщо значення не збігаються, то необхідно за допомогою поетапних обчислень формули знайти у чому причина розбіжності.

У висновках до звіту студент повинен описати результат порівняння отриманих значень і розкрити призначення теми "Типи даних та прості алгоритмічні операції". Висновок має бути поданий в формі пояснення сфери застосування вивченого матеріалу згідно із розумінням його студентом.

Завдання для самостійного виконання

Розробити схему алгоритму програми для розрахунку математичної формули за варіантом (табл. 5.2). За схемою алгоритму написати програму і отримати результат обчислення. Порівняти значення, що отримані шляхами ручного та програмного розрахунків зазначеної формули.

Таблиця 5.2

Варіанти завдань

№ вар.	Змінні			Вираз
	X	Y	Z	
1	**float**	**int**	**float**	$\frac{x-y\%2+z/y}{x-z}+1.22x+y/z*2$
2	3.14	-27	-23.2	$1.2x-1.3x/y+\frac{1.1\%y+1.2z/x-12}{0.8}$
3	**int**	**int**	**int**	$\frac{(x+y)*z-23+1.2z}{1.45z+12y}+7x$
4	$1.2*10^2$	3.2	7.8	$xyz+\frac{32/x-17x*2z+3y/4}{(z-1.67y)*y}*2$
5	**int**	**char**	**float**	$\frac{1.3z-2(y+1.1x)\%2}{x-y+z}-10y+z$
6	2.81	12	13.1	$x/y-\frac{2.1x-102y+45\%z-x/2y}{x+y/2z-1y}$
7	**int**	**float**	**float**	$\frac{21x-12y+435\%z-x/y}{x+y/z-12y}$
8	7	3.2	$-1.2*10^{-3}$	$\frac{13x-y/2+z/101y}{y-17z}+2z+y/3z$
9	**int**	**int**	**float**	$12x-x/y+\frac{1.1\%y+z/x-12}{11}$
10	$11*10^{-2}$	46	3.5	$\frac{(x+y)*z-23+1.2z}{1.45z+12y}+7x$

Продовження табл. 5.2

№ вар.	Змінні			Вираз
	X	Y	Z	
11	**char**	**char**	**float**	$\frac{32/x-17y*2z+3z/4}{(z-1.67y)*z}\%2$
12	-7.3	$15*10^{-2}$	36	$\frac{12z/y+11x-y\%3*100*(x+y)}{4x/y+2z/x}$
13	**int**	**float**	**int**	$\frac{13z-y*y*z+1.1z/2}{1.47y+18z}\%3$
14	2.3	2^3	12	$\frac{3.1429x/3z+y*18z-(2/y+2)}{y-5.6y+5z/y}$
15	**float**	**float**	**int**	$\frac{z/y+11-y\%3*100*(x-y)}{z/y+z/x}$
16	8^{-3}	1.3	0.05	$\frac{16y+1.7x-x/7.89z+z/1.3y}{1.13z*y-x/y}$
17	**char**	**int**	**float**	$\frac{5.17x/3+1.78y*z-(z/y+2)}{z-56y+x/y}$
18	742	-84.60	110	$\frac{1.3z-(y+y)3z*z+1.1z/2}{17y+1.8z}*3$
19	**float**	**float**	**char**	$\frac{x-1.6y*1.4z/x-5}{x-y+1.6z}+12x$
20	75.3	$57*10^{-2}$	2	$-\frac{yx+1.2x/z-y/7x-3}{15z+1.2}$
21	**int**	**char**	**int**	$\frac{z+17x-7.89z+z/1.3y}{1.13z-x/y}$
22	45	-54	$45*10^{-1}$	$\frac{19y-1.5y+(13z-y)/x*y-13}{2.54z+12y/(z+1)}$
23	**char**	**char**	**float**	$z\%x-1.5\frac{x+34y-4.3z/x}{12z}$
24	4^2	5.3	57	$\frac{(1-z)/x-23y/z+(-13z+4.76z*y)}{z/3.14+2y-1.7z}$
25	**float**	**int**	**int**	$\frac{z/x-23y\%z+(-13z+4.76x)}{z/34+y-1.7z}$
26	85.3	32.7	5	$\frac{x+z-4z/x}{12z}-\frac{2x+34y-4z/x}{1.2z}$
27	**float**	**float**	**float**	$\frac{13z-1.5x\%4+13z/x*y-13}{2.54z+12y/z}$
28	43	57.1	2^8	$\frac{z/x-23y\%z+(-13z+4.76x)}{z/34+y-1.7z}$

Закінчення табл. 5.2

№ вар.	Змінні			Вираз
	X	Y	Z	
29	**char**	**float**	**char**	$-\frac{1.134x+12/z-z/1.4x-10}{153.1z-12}$
30	-2	2.3	56	$\frac{yx+1.2x/z}{15z}-\frac{y+2x/z}{15z+1.2}$

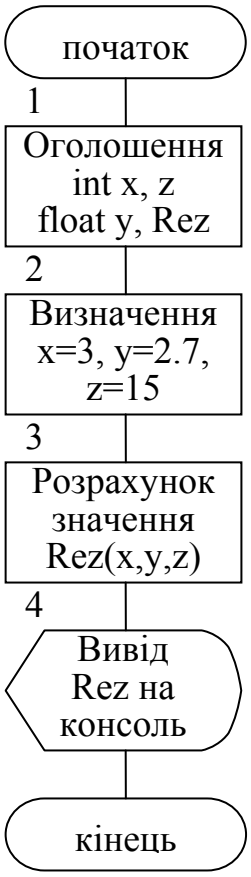
Приклад виконання завдання

№ вар.	Змінні			Вираз
	X	Y	Z	
N	3	2.7	15	$\frac{yx-3x/z}{2z}-\frac{y+2x/z}{15z/x+1.3}$

Підставимо значення у формулу та розрахуємо результат:

$$\frac{2.7\cdot 3-3\cdot 3/15}{2\cdot 15}-\frac{2.7+2\cdot 3/15}{15\cdot 15/3+1.3}=0.2094$$

Розробимо схему алгоритму роботи програми для машинного розрахунку формули (рис. 5.2):



$$\text{Rez(x,y,z)}=\frac{yx-3x/z}{2z}-\frac{y+2x/z}{15z/x+1.3}$$

Рис. 5.2. Схема алгоритму роботи програми розрахунку формули

Згідно зі схемою алгоритму напишемо програму на мові С++:

```
#include<stdio.h>                    //підключення бібліотеки вводу/виводу

void main()                          //оголошення та визначення головної функції
{
        int x, z;                    //оголошення цілочисельних змінних x, z
        float y, Rez;                //оголошення дійсних змінних y, Rez
        x=3;                         //визначення x
        y=2.7;                       //визначення y
        z=15;                        //визначення z
        Rez=(y*x-3*x/z)/(2*z)-(y+2*x/z)/(15*z/x+1.3); //розрахунок виразу
        printf("Rez=%f", Rez);       //вивід результату на консоль
}
```

Результат, що буде виведено програмою:

```
Rez=0.234613
```

Результат не збігається з отриманим при розрахунках вручну, оскільки при діленні цілочисельного значення на цілочисельне за допомогою оператора "/" буде отримано ціле значення, дробову частину буде відкинуто. Для отримання правильного результату необхідно помножити цілі чисельники на дійсне значення 1.0 і лише потім виконати ділення.

```
Rez=(y*x-3*x*1.0/z)/(2*z)-(y+2*x*1.0/z)/(15*z*1.0/x+1.3); //розрахунок виразу
```

Результат, що виведе виправлена програма:

```
Rez=0.209371
```

Отриманий результат збігається до четвертого знаку за комою. Це є задовільним. Невелика розбіжність пов'язана з округленням при ручних розрахунках.

Питання для підготовки до захисту лабораторної роботи

1) Які стандартні типи даних застосовуються в мові С++?
2) Чим відрізняються типи даних **char** та **int**?
3) Які додаткові ключові слова можуть використовуватися для визначення типу даних змінної?
4) Що таке таблиця ASCII кодів?
5) Для чого застосовується тип даних **bool**?
6) Які арифметичні операції ви знаєте?
7) Назвіть логічні операції мови С++.
8) У чому особливість використання операції ділення за модулем?

5.3. Лабораторна робота №3: Бібліотека математичних функцій

Мета роботи

Ознайомитись з функціями математичної бібліотеки та отримати практичні навички розробки програм для розрахунку складних математичних виразів.

Хід роботи

1) Ознайомитись з методичними вказівками до лабораторної роботи;

2) відповідно до завдання за варіантом розробити схему алгоритму програми розрахунку значення за вказаною формулою;

3) за схемою алгоритму написати програму на мові C++ та отримати результати її роботи;

4) зробити висновки;

5) підготувати звіт про виконання лабораторної роботи, що включає наступні пункти: номер, тема, мета та хід лабораторної роботи, завдання, схема алгоритму програми, лістинг програми з коментуванням кожної інструкції, результат роботи програми (скріншот), висновки.

Теоретичні відомості та рекомендації до виконання

Для розрахунку складних математичних виразів простих операторів мови C++ недостатньо. Такі операції виконуються за допомогою функцій математичної бібліотеки, яка підключається через заголовний файл "math.h". Можливості, що надаються цією бібліотекою, є важливими при обчисленні результатів складних арифметичних, алгебраїчних, логарифмічних та тригонометричних виразів.

При виконанні завдання до лабораторної роботи студент на прикладі рішення складного математичного виразу повинен ознайомитись з функціями математичної бібліотеки, які можна застосовувати при програмуванні на мові C++.

Виконання лабораторної роботи починається з ознайомлення із теоретичними відомостями. Потім студент приступає до розробки схеми алгоритму програми відповідно до завдання за варіантом. За схемою алгоритму виконується написання програми на мові C++.

Перелік основних функцій математичної бібліотеки та їх опис наведено нижче:

double abs(**double** x) – модуль x;
double exp(**double** x) – експонент від x (e в степені x);
double sqrt(**double** x) – корінь від x;
double pow(**double** x, **double** y) – піднесення x у степінь y;
double log(**double** x) – натуральний логарифм від x;
double log10(**double** x) – десятковий логарифм від x;
double ceil(**double** x) – приведення десяткового x до найближчого більшого цілого;

double floor (**double** x) – приведення десяткового x до найближчого меншого цілого;
double atof(**char** *s) – перетворення рядка чисельних символів в число;
double sin(**double** x) – синус від x;
double cos(**double** x) – косинус від x;
double tan(**double** x) – тангенс від x
double atan(**double** x) – арктангенс від x;
double asin(**double** x) – арксинус x;
double acos(**double** x) – арккосинус x.

Значення, що передається до функцій, необов'язково повинно мати тип даних **double**. Може бути використана будь-яка змінна чи значення цілочисельного або дійсного типу даних. Однак для того, щоб результат розрахунків не було спотворено чи втрачено, значення, що повертається, повинно бути записано у змінну типу даних **double**.

У висновках до звіту студент повинен розкрити призначення теми "Бібліотека математичних функцій". Висновок має бути подано в формі пояснення сфери застосування вивченого матеріалу згідно із розумінням його студентом.

Завдання для самостійного виконання

Розробити схему алгоритму та написати програму для розрахунку значення y за формулою згідно з варіантом (табл. 5.3). Результат обчислень округлити за завданням до найближчого більшого або меншого цілого.

Результат виконання програми повинен бути зображений в наступному вигляді:

Laboratory work #3
Group <назва групи>
<Ф.І.Б.>

Task: <Формула>
Calculation result for x=<значення>: <значення>

Таблиця 5.3

Варіанти завдань

№ вар.	**Формула розрахунку *y***	**Округлити *y* до найближчого…**
1	$\frac{\cos^2(x) + ctg^3(x) - \lvert x \rvert}{x^{1/7} + 5}$	більшого
2	$\frac{\lg x - \sin(x) + tg(1/x)}{12 + \sqrt[4]{x+1}}$	меншого
3	$\frac{\sin^3 x + \cos^2 x - \sqrt[3]{\lvert x \rvert}}{3 + x^{7/8}}$	більшого

Продовження табл. 5.3

№ вар.	Формула розрахунку *y*	Округлити *y* до найближчого…
4	$\frac{tg(x^3)+17x-\cos^3 x}{\sin^2 x-\lvert x\rvert}$	меншого
5	$\frac{1}{\cos^3 x}-\frac{1}{\sin^3 x}+\lg(x^2)$	більшого
6	$\frac{\sin x}{tg^2 x}-\sqrt[3]{\frac{\sin x}{\lvert x\rvert}}+\lg\sqrt{x}$	меншого
7	$tg\left(\frac{\sin x}{\lvert x\rvert}\right)+ctg\left(\frac{\cos x}{\sqrt{x}}\right)$	більшого
8	$\frac{12x^3+\sin^3 x-tg^5 x}{17\lvert x\rvert+x^{1/3}}$	меншого
9	$\frac{tg(\ln x)+ctg(\sqrt{x})}{\sin(x^{1/2})+3x}$	більшого
10	$\frac{\cos(\lvert x\rvert)+\sin(x^{1/5})}{x^2-\sqrt[4]{x+1}}$	меншого
11	$\frac{\sin(\sqrt{\lvert x\rvert})}{\cos(x+x^2)}-\sqrt{\frac{\sin x}{\cos(\lg x)}}$	більшого
12	$\frac{tg\left(\frac{\lg x}{\lvert 1/2x\rvert}\right)}{ctg\left(\frac{\cos x}{\sqrt{x}}\right)}+1/x$	меншого
13	$\frac{\sin(\sqrt{\lvert x\rvert})}{\cos(x^2)}-\sqrt[3/4]{\frac{\cos(\lg x)+\sin x}{\cos(\lg x)}}$	більшого
14	$\frac{\cos^4(\lvert x\rvert)^{1/2}+\sin(x^{1/5})}{\sqrt[4]{x+1}}$	меншого
15	$\frac{\cos^2(1/8)+ctg^3(x)-\lvert 5^{-2}\rvert}{x^{1/x}+5x}$	більшого
16	$\frac{\log_2^4 x-\sin^{1/3}(x)}{5^{1/x}+\sqrt[4]{x+1}}$	меншого
17	$\frac{tg^x(x^3)-\frac{\cos^3 x}{17x}}{\sin^x(2)-\lvert x\rvert}$	більшого
18	$\frac{ctg(\cos(x^3))+17/x-\cos^x x}{\sin^2 x-\lvert x\rvert}$	меншого

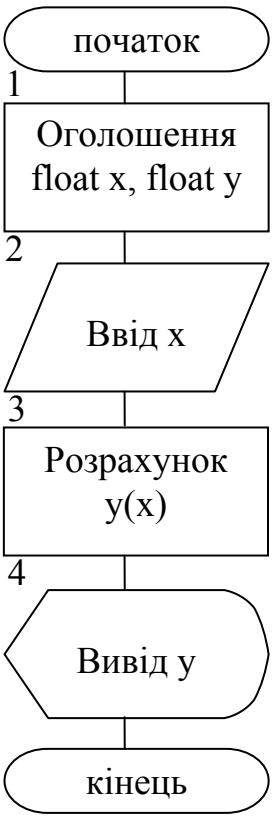
Закінчення табл. 5.3

№ вар.	Формула розрахунку y	Округлити y до найближчого…
19	$\frac{1-\sin^3 x}{\cos^3 x}-\frac{1}{x}\lg^2(x^2)$	більшого
20	$\frac{\cos(12+x)}{tg^2(x-2)}-\sqrt[1/3]{\frac{\sin x}{\mid x \mid}}+\ln\sqrt{x}$	меншого
21	$tg\left(ctg(\mid \cos x \mid)\right)+ctg\left(\sqrt{x}\cos x\right)$	більшого
22	$\frac{34x^{1/3}+\sin\ (x-tg^5 x)}{28 \mid x \mid +x^3}$	меншого
23	$\frac{tg(\ln x)+ctg(\sqrt{x})}{\sin(x^{1/2})+3x}$	більшого
24	$\frac{ctg(x)+\sin(x^{1/5})}{\sin(x^{1/4})-\sqrt[2/4]{x+1}}$	меншого
25	$\frac{\sin(\sqrt{\mid x/2 \mid}+1)}{\cos(x+x^2)}-\sqrt{\sin^2 x}$	більшого
26	$\sqrt{\cos(tg^2 x)}\cos(x-3)-\frac{ctg(\sqrt[3]{\sin x})}{\lvert\cos(x-10)\rvert}$	меншого
27	$\lg(\ln(x+10))+\frac{\sqrt[4]{\ln(\sqrt{x})}}{\cos(\lg x)}$	більшого
28	$\sqrt[3]{\frac{\cos(x/5)\sin(x/5)+tgxctg(x/4)}{x-\ln x}}$	меншого
29	$\frac{\sqrt[2/3]{\sin x-tg(x-3)+\cos(x/4)}}{\ln x-\lg x}$	більшого
30	$\left(\cos(\frac{\lg x}{\sin(x-2)})+\sin(\frac{\ln x}{\cos(x-2)})\right)^{3/4}$	меншого

Приклад виконання завдання

№ вар.	Формула розрахунку y	Округлити y до найближчого…
N	$\frac{\sqrt[3]{tg(1/(\sin x\cos x))+\ln(x+3)}}{\ln x+\lg x}$	меншого

Розробимо схему алгоритму програми для розрахунку заданої формули (рис. 5.3).



$$y(x) = \frac{\sqrt[3]{tg(1/(\sin x \cos x)) + \ln(x+3)}}{\ln x + \lg x}$$

Рис. 5.3. Схема алгоритму програми розрахунку математичного виразу

За схемою алгоритму напишемо програму на мові С++:

```
#include<stdio.h>                    //підключення бібліотеки вводу/виводу
#include<math.h>          //підключення бібліотеки математичних функцій

void main()                          //оголошення та визначення головної функції
{
        float x;                              //оголошення дійсної змінної
        int y;                                //оголошення цілої змінної
        printf("Enter x value: ");            //вивід запиту до користувача
        scanf("%f", &x);              //отримання значення від користувача
//розрахунок виразу за завданням
        y=floor(pow((tan(1.0/(sin(x)*cos(x)))+log(x+3)),1.0/3)/(log(x)+log10(x)));
        printf("\r\nLaboratory work #3\r\n");        //вивід константного рядка
        printf("Group OE-00\r\n");                   //вивід константного рядка
        printf("Ivanov Igor Olegovich\r\n\r\n");     //вивід константного рядка
//вивід константного рядка
        printf("Task: <pow((tan(1.0/(sin(x)*cos(x)))");
//вивід константного рядка
        printf("+log(x+3)),1.0/3)/(log(x)+log10(x))");
        printf("\r\n");                              //перехід на новий рядок
        printf("Calculation result for x=%f: %i\r\n", x, y); //вивід результату
}
```

Результат роботи наведеної програми буде наступним:

```
Enter x value: 1.25

Laboratory work #3
Group OE-00
Ivanov Igor Olegovich

Task: (pow((tan(1.0/(sin(x)*cos(x)))+log(x+3)),1.0/3)/(log(x)+log10(x))
Calculation result for x=1.250000: 3
```

Питання для підготовки до захисту лабораторної роботи

1) Яка бібліотека містить функції для виконання складних математичних операцій?

2) Яка функція використовується для округлення числа до найближчого більшого?

3) Яка функція використовується для отримання значення тангенса аргументу?

4) Які логарифми можна брати за допомогою функцій математичної бібліотеки?

5) Як за допомогою математичної бібліотеки виконати знаходження котангенса?

6) Яка функція слугує для перетворення символьного рядка в число?

5.4. Лабораторна робота №4: Розгалуження програм на основі операторів умови

Мета роботи

Ознайомитись з операторами умови мови С++ та отримати практичні навички реалізації розгалуження програм за вказаною умовою.

Хід роботи

1) Ознайомитись з методичними вказівками до лабораторної роботи та темою "Розгалуження";

2) відповідно до завдання за варіантом розробити схему алгоритму програми, що опрацьовує значення змінних x, y та z;

3) за схемою алгоритму написати програму на мові С++ та отримати результати її роботи;

4) зробити висновки;

5) підготувати звіт про виконання лабораторної роботи, що включає наступні пункти: номер, тема, мета та хід лабораторної роботи, завдання, схема алгоритму програми, лістинг програми з коментуванням кожної інструкції, результат роботи програми (скріншот), висновки.

Теоретичні відомості та рекомендації до виконання

Як і в житті, при виконанні програми може виникнути ситуація, коли залежно від якої-небудь події доводиться робити вибір і від цього вибору залежить подальший шлях, за яким підуть обчислення. У програмуванні такі ситуації називаються розгалуженням. Розгалуження – одна з найважливіших особливостей програмування, яка надає можливість в залежності від поставленої умови отримати різні результати.

При виконанні завдання до лабораторної роботи студент повинен навчитися застосовувати три способи розгалуження: два оператори умови та умовну операцію.

Виконання починається ознайомленням із теоретичними відомостями про оператори розгалуження програм та особливості їх застосування. Потім студент приступає до розробки схеми алгоритму програми відповідно до завдання за варіантом. За схемою алгоритму виконується написання програми на мові С++.

У висновках до звіту студент повинен розкрити призначення теми "Розгалуження програм на основі операторів умови". Висновок має бути поданий в формі пояснення сфери застосування вивченого матеріалу згідно із розумінням його студентом.

Завдання для самостійного виконання

Розробити схему алгоритму та написати програму, що надає користувачу можливість вибору шляху для визначення цілочисельних змінних x, y та z (програмний чи ввід з консолі) і виконує над їх значеннями дії у відповідності до завдання за варіантом (табл. 5.4).

Таблиця 5.4

Варіанти завдань

№ вар.	Завдання
1	вивести на консоль значення змінних у порядку зростання
2	знайти та вивести на консоль найбільше серед трьох значень
3	визначити та вивести на консоль пари значень, сума яких є парною
4	знайти та вивести на консоль пару значень, різниця між якими є максимальною
5	знайти та вивести на консоль пару значень, сума яких ділиться на третє без залишку
6	знайти та вивести на консоль пару значень, що у сумі дорівнюють третьому
7	знайти та вивести на консоль таке співвідношення між x, y, z та a, b, c, щоб виконувалася умова: a·b-c є парним
8	знайти та вивести на консоль значення, що дорівнює добутку двох інших
9	знайти та вивести на консоль таке співвідношення між x, y, z та a, b, c, щоб виконувалася умова: a-b+c=3
10	знайти та вивести на консоль пару значень, частка від ділення яких одне на одне є максимальною
11	знайти та вивести на консоль таке співвідношення між x, y, z та a, b, c, щоб виконувалася умова: $(a+b)^2= c$
12	знайти та вивести на консоль пари значень, що є у сумі більшими за третє
13	знайти та вивести на консоль таке співвідношення між x, y, z та a, b, c, щоб виконувалася умова: a·(b+c) є кратним трьом
14	вивести на консоль значення змінних у порядку спадання
15	знайти та вивести на консоль найменше серед трьох значень
16	визначити та вивести на консоль пари значень, сума яких є не парною
17	знайти та вивести на консоль те значення, яке ділиться на інші два без залишку
18	знайти та вивести на консоль таке співвідношення між x, y, z та a, b, c, щоб виконувалася умова: a+b·c=11
19	знайти та вивести на консоль значення, що дорівнює сумі інших двох
20	знайти та вивести на консоль таке співвідношення між x, y, z та a, b, c, щоб виконувалася умова: a/b+c є непарним
21	знайти та вивести на консоль пару значень, частка від яких дорівнює третьому
22	знайти та вивести на консоль значення, що ділиться на різницю інших двох без залишку
23	знайти та вивести на консоль таке співвідношення між x, y, z та a, b, c, щоб виконувалася умова: $(b-c)^2=a$
24	знайти та вивести на консоль пари значень, різниця яких є меншою за третє та позитивною

Закінчення табл. 5.4

№ вар.	Завдання
25	знайти та вивести на консоль таке співвідношення між x, y, z та a, b, c, щоб виконувалася умова: c·(a+b) є кратним п'яти
26	знайти та вивести на консоль таке співвідношення між x, y, z та a, b, c, щоб виконувалася умова: a+b/c=5
27	знайти та вивести на консоль те значення, на яке інші два діляться без залишку
28	знайти та вивести на консоль середнє зі значень
29	знайти та вивести на консоль пару значень, різниця між якими є мінімальною
30	знайти та вивести на консоль пару значень, частка від ділення яких одне на одне є мінімальною

Приклад виконання завдання

Знайти та вивести на консоль таке співвідношення між x, y, z та a, b, c, щоб виконувалася умова: a+b-c/b є непарним.

Розробимо схему алгоритму програми, що опрацьовує значення змінних x, y та z (рис. 5.4):

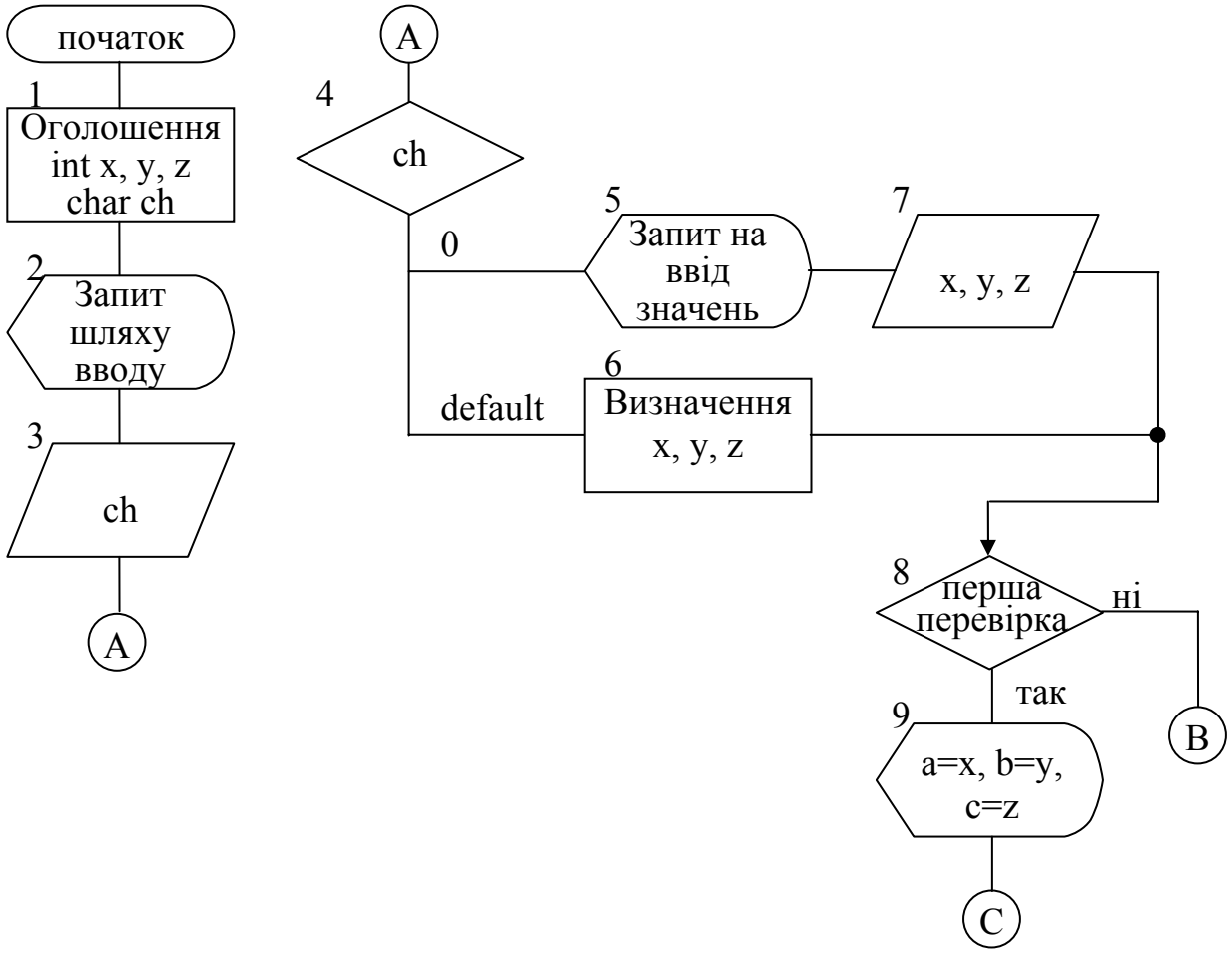


Рис. 5.4. Початок схеми алгоритму програми роботи зі змінними

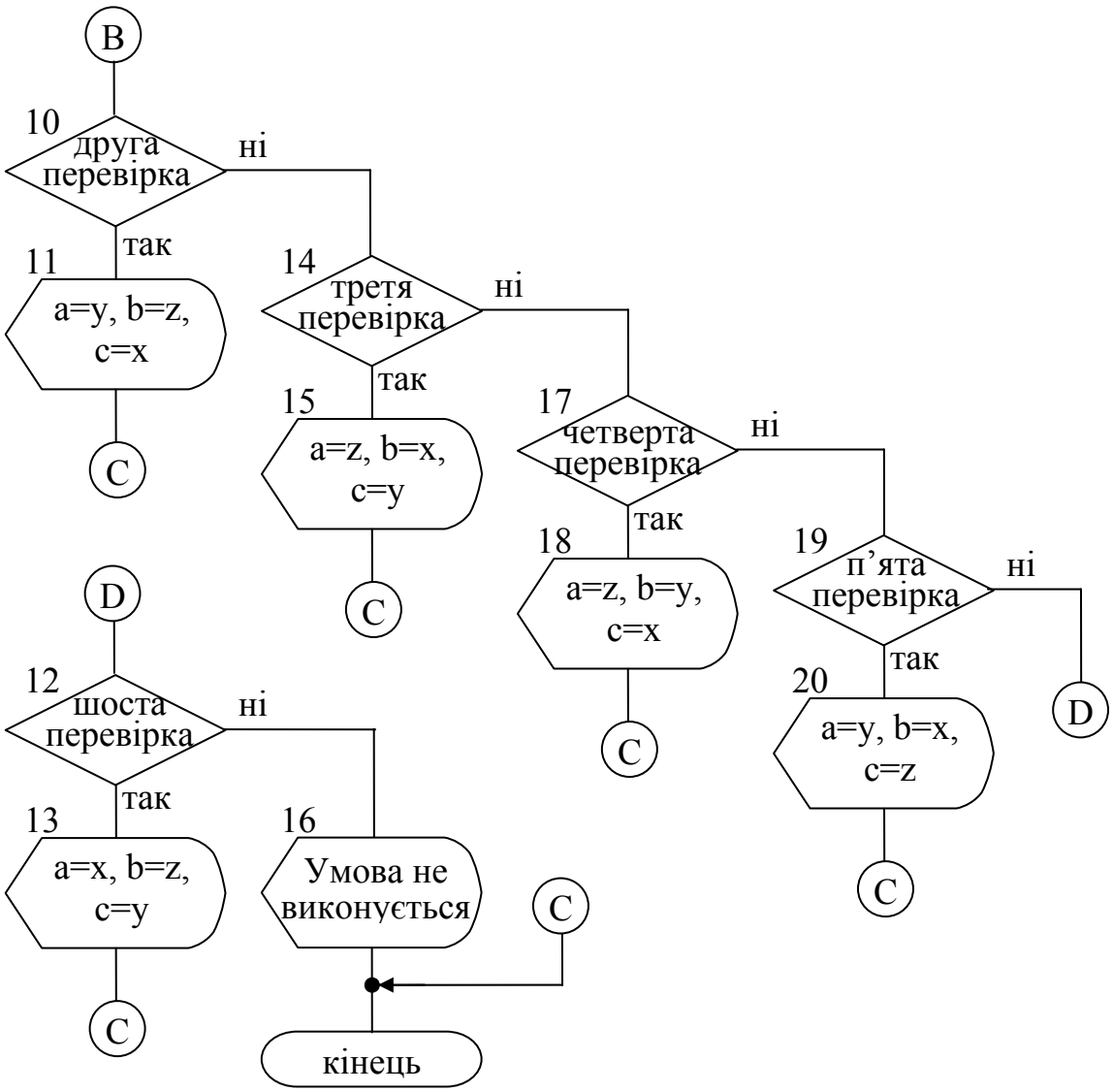


Рис. 5.4. Кінець схеми алгоритму програми роботи зі змінними

За схемою алгоритму напишемо програму на мові С++:

```
#include <stdio.h>                          //підключення бібліотеки вводу/виводу

void main()                          //підключення бібліотеки математичних функцій
{
        int x, y, z;                                //оголошення цілочисельних змінних
        char ch;                                    //оголошення символьної змінної
//вивід запиту до користувача
        printf("Please, choose initialization way: 0 – Manual, 1 – Auto\r\n");
        scanf("%c", &ch);                           //отримання значення від користувача
        switch(ch)                          //перевірка значення символьної змінної
        {
                case '0':                                   //якщо значення є символом '0'
                        printf("Manual mode/\r\n");     //вивід константного рядка
                        printf("Enter value for x: ");  //вивід запиту до користувача
```

```
//отримання значення від користувача
                    scanf("%i", &x);
                    printf("Enter value for y: ");     //вивід запиту до користувача
//отримання значення від користувача
                    scanf("%i", &y);
                    printf("Enter value for z: ");     //вивід запиту до користувача
//отримання значення від користувача
                    scanf("%i", &z);
                    break;                             //вихід з оператора
                default:                               //інший символ
                    printf("Auto mode/\r\n");          //вивід константного рядка
                    x=15;
                    y=7;
                    z=3;
        }
        printf("For conditions: a+b-c/b is negative\r\n"); //вивід константного рядка
//якщо ділення не має залишку та відповідає співвідношенню
        if(z%y==0 && (x+y-z/y)%2==1)
        {
            printf("a=x; b=y; c=z");                   //вивід константного рядка
        }
//інакше, якщо ділення не має залишку та відповідає співвідношенню
        else if(x%z==0 && (y+z-x/z)%2==1)
        {
            printf("a=y; b=z; c=x");                   //вивід константного рядка
        }
//інакше, якщо ділення не має залишку та відповідає співвідношенню
        else if(y%x==0 && (z+x-y/x)%2==1)
        {
            printf("a=z; b=x; c=y");                   //вивід константного рядка
        }
//інакше, якщо ділення не має залишку та відповідає співвідношенню
        else if(x%y==0 && (z+y-x/y)%2==1)
        {
            printf("a=z; b=y; c=x");                   //вивід константного рядка
        }
//інакше, якщо ділення не має залишку та відповідає співвідношенню
        else if(z%x==0 && (y+x-z/x)%2==1)
        {
            printf("a=y; b=x; c=z");                   //вивід константного рядка
        }
//інакше, якщо ділення не має залишку та відповідає співвідношенню
        else if(y%z==0 && (x+z-y/z)%2==1)
        {
            printf("a=x; b=z; c=y");                   //вивід константного рядка
```

```
        }
//інакше, якщо співвідношення не відповідає
        else
        {
//вивід константного рядка
                printf("Values do not match to condition!");
        }
}
```

Результат роботи програми наступний:

```
Please, choose initialization way: 0 - Manual, 1 - Auto
0
Manual mode/
Enter value for x: 25
Enter value for y: 5
Enter value for z: 7
For conditions: a+b-c/b is negative
a=z; b=y; c=x
```

Питання для підготовки до захисту лабораторної роботи

1) Які оператори умови використовуються в мові С++?

2) Які операції використовуються для порівняння двох чисел?

3) Чим оператор **switch** відрізняється від оператора **if-else**?

4) Який тип даних повинен бути у змінної для перевірки її в операторі **switch**?

5) Для чого слугує оператор **break**?

6) Чим є і для чого слугує **else if**?

7) Наведіть приклад формату запису оператора **switch**.

8) Для чого слугує операція "?:"? Чим вона відрізняється від оператора **if-else**?

5.5. Лабораторна робота №5: Циклічні дії у програмах на основі операторів циклу

Мета роботи

Ознайомитись з операторами циклів мови С++ та отримати практичні навички застосування циклічних програмних кодів.

Хід роботи

1) Ознайомитись з методичними вказівками до лабораторної роботи та темою "Цикли";

2) відповідно до завдання розробити схему алгоритму програми, яка за допомогою різних операторів циклу виконує: розрахунок суми елементів послідовності, що задано формулою; пошук всіх простих чисел у вказаному діапазоні; пошук корнів рівняння;

3) за схемою алгоритму написати програму на мові С++;

4) зробити висновки;

5) підготувати звіт про виконання лабораторної роботи, що включає наступні пункти: номер, тема, мета та хід лабораторної роботи, завдання, схема алгоритму програми, лістинг програми з коментуванням кожної інструкції, результат роботи програми (скріншот), висновки.

Теоретичні відомості та рекомендації до виконання

Цикли існують як у житті, так і в програмах. До них відносяться дії або події, які повторюються з певною регулярністю. Оператори циклів мови С++ організовують такі дії, зображуючи їх у вигляді правила затримки та повторюваного коду. Циклічні коди дозволяють виконувати математичні розрахунки, пов'язані з послідовністю значень (середнє арифметичне, сума, добуток, пошук факторіала тощо), працювати з масивами (пошук мінімального і максимального значень, упорядкування тощо), використовувати програму протягом будь-якого проміжку часу (зациклення тіла головної функції) і багато інших операцій.

При виконанні завдання до лабораторної роботи студент повинен ознайомитись з операторами циклів і навчитися їх використовувати при створенні проектів на мові С++.

Виконання лабораторної роботи починається ознайомленням з теоретичними відомостями про оператори циклів з перед- та постумовою. Потім студент приступає до розробки схеми алгоритму програми, що виконує три різні операції, відповідно до завдання за варіантом. За схемою алгоритму виконується написання програми мовою С++.

У висновках до звіту студент повинен розкрити призначення теми "Циклічні дії у програмах на основі операторів циклу". Висновок має бути поданий в формі пояснення сфери застосування вивченого матеріалу згідно із розумінням його студентом.

Завдання для самостійного виконання

Розробити схему алгоритму та написати програму, що на основі трьох різних операторів циклу виконує (табл. 5.5):

1) формування послідовності, що задано правилом та діапазоном за варіантом, та розрахунок суми її елементів. Результат виконання вивести на консоль у формі:

$$A_1+A_2...A_{max-1}+A_{max}=\sum_{1}^{max} A_n$$

2) пошук всіх простих чисел у вказаному за варіантом діапазоні і вивід отриманої послідовності на консоль;

3) пошук та вивід на консоль перших ненульових коренів a, b та c рівняння, що задано за варіантом.

Таблиця 5.5

Варіанти завдань

№ вар.	Правило формування послідовності	Діапазон значень (m)	Рівняння	Оператор циклу відповідно до номеру підзадачі		
	Номер підзадачі			for	while	do-while
	1	2	3			
1	непарні A_n, кратні 3	0..100	$3a^2-2b+5c=0$	1	2	3
2	$A_n=m/2-3$, кратні 5	0..200	$b^2-(a+c)^2=0$	2	3	1
3	парні A_n, кратні 13	0..300	$a*b-c^3-9=0$	3	1	2
4	$A_n=m+4$, кратні 3	0..400	$(a+5)^2-(2b-c)^2=0$	1	3	2
5	непарні A_n, кратні 7	0..500	$a+3c-7b^2=0$	3	2	1
6	$A_n=m-3$, кратні 4	0..600	$a^2+3b-4c=0$	2	1	3
7	парні A_n, кратні 17	0..700	$(a-b)^2-c^2=0$	1	2	3
8	$A_n=2*m-7$, кратні 9	100..200	$a^3-b^2+c=0$	2	3	1
9	непарні A_n, кратні 5	100..300	$(a-3)^2-(b+2c)^2=0$	3	1	2
10	$A_n=3*m+5$, кратні 4	100..400	$2a+5c^2-11b=0$	1	3	2
11	парні A_n, кратні 15	100..500	$2a^2-7b-3c=0$	3	2	1
12	$A_n=2*m+3$, кратні 5	100..600	$(a-c)^2-b^2=0$	2	1	3
13	непарні A_n, кратні 11	100..700	$a^3-b*c+11=0$	1	2	3
14	$A_n=m+13$, кратні 11	100..800	$(a+b)^2-(c-10)^2=0$	2	3	1
15	парні A_n, кратні 7	200..300	$7a^2-9c+7b^2=0$	3	1	2
16	$A_n=m+7$, кратні 3	200..400	$a^2-5b+8c=0$	1	3	2
17	непарні A_n, кратні 17	200..500	$(b+c)^2-a^2=0$	3	2	1
18	$A_n=m/3-5$, кратні 11	200..600	$a^2-3b^2-2c=0$	2	1	3
19	парні A_n, кратні 5	200..700	$(3a+2)^2-(b-c)^2=0$	1	2	3
20	$A_n=m-11$, кратні 7	200..800	$4a+c^3-3b^2=0$	2	3	1
21	непарні A_n, кратні 13	200..900	$4a^2+7b-11c=0$	3	1	2
22	$A_n=m+15$, кратні 3	300..400	$(a+b)^2-c^2=0$	1	3	2
23	парні A_n, кратні 11	300..500	$a^3-(b+3)^2-c=0$	3	2	1
24	$A_n=2*m-5$, кратні 11	300..600	$(a-3)^2-(b+2c)^2=0$	2	1	3
25	непарні A_n, кратні 19	300..700	$9a-2c-3b^2=0$	1	2	3

Приклад виконання завдання

Розробити схему алгоритму та написати програму, що трьома шляхами на основі операторів циклу формує послідовність, задану правилом та діапазоном, і розраховує суму її елементів. Результат виконання зобразити за формою:

$$A_1+A_2...A_{max-1}+A_{max}=\sum_{1}^{max} A_n$$

Вар. №	Діапазон значень (m)	Правило формування послідовності
N	0..100	всі непарні

Розробимо схему алгоритму програми (рис. 5.5):

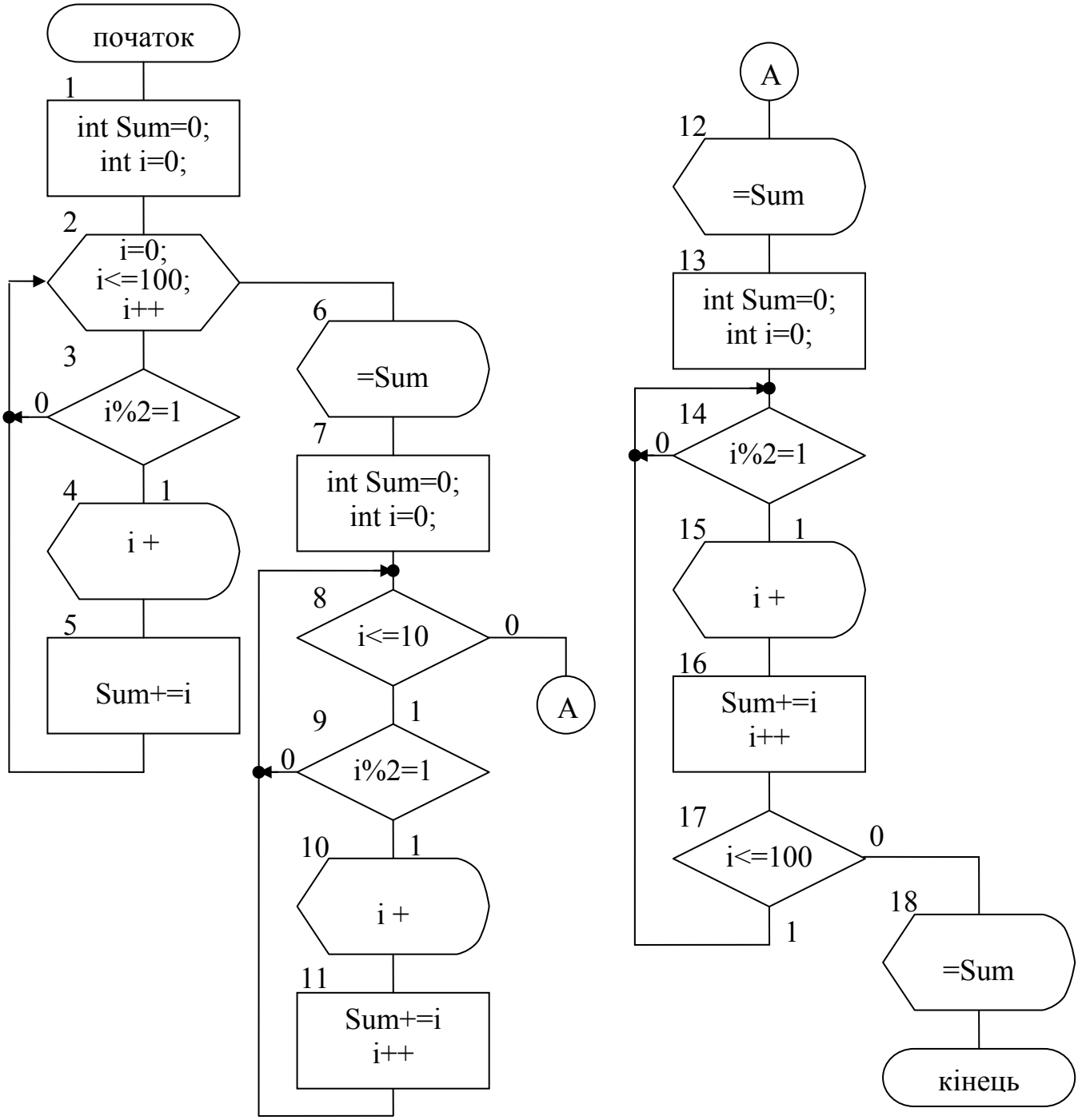


Рис. 5.5. Схема алгоритму програми, що трьома шляхами на основі операторів циклу формує послідовність та рахує суму її елементів

За схемою алгоритму напишемо програмний код:

```
#include <stdio.h>                          //підключення бібліотеки вводу/виводу

void main()                         //підключення бібліотеки математичних функцій
{
        int Sum=0;          //оголошення та визначення змінної рахування суми
        int i=0;                    //оголошення та визначення змінної лічильника
        for(i=0; i<=100; i++)               //цикл перебору значень від 0 до 100
        {
                if(i%2==1)                                  //якщо число непарне
                {
//якщо поточний елемент останній в послідовності
                        if(i+2>100)
                                printf("%i", i);            //вивід значення на консоль
                        else                    //якщо елемент неостанній в послідовності
                                printf("%i+", i);           //вивід значення зі знаком "+"
                        Sum+=i;                             //додати значення до суми
                }
        }
        printf("=%i\r\n\r\n", Sum);       //вивід знаку "=" та суми значень

        Sum=0;                              //перевизначення змінної рахування суми
        i=0;                                //перевизначення змінної лічильника
        while(i<=100)    //цикл, доки значення лічильника менше або дорівнює 100
        {
                if(i%2==1)                                  //якщо число непарне
                {
//якщо поточний елемент останній в послідовності
                        if(i+2>100)
                                printf("%i", i);            //вивід значення на консоль
                        else                    //якщо елемент неостанній в послідовності
                                printf("%i+", i);           //вивід значення зі знаком "+"
                        Sum+=i;                             //додати значення до суми
                }
                i++;                                    //збільшення значення лічильника на 1
        }
        printf("=%i\r\n\r\n", Sum);               //вивід знаку "=" та суми значень

        Sum=0;                              //перевизначення змінної рахування суми
        i=0;                                //перевизначення змінної лічильника
        do                                  //початок циклу з постумовою
        {
                if(i%2==1)                                  //якщо число непарне
                {
```

```
//якщо поточний елемент останній в послідовності
                    if(i+2>100)
                        printf("%i", i);          //вивід значення на консоль
                    else                  //якщо елемент неостанній в послідовності
                        printf("%i+", i);         //вивід значення зі знаком "+"
                    Sum+=i;                       //додати значення до суми
                }
                i++;                              //збільшення значення лічильника на 1
        }while(i<=100);    //цикл доки значення лічильника менше або дорівнює 100
        printf("=%i\r\n\r\n", Sum);               //вивід знаку "=" та суми значень
}
```

Результат виконання програми буде наступним:

```
1+3+5+7+9+11+13+15+17+19+21+23+25+27+29+31+33+35+37+39+41+43+45+47+49+51+53+55+5
7+59+61+63+65+67+69+71+73+75+77+79+81+83+85+87+89+91+93+95+97+99=2500

1+3+5+7+9+11+13+15+17+19+21+23+25+27+29+31+33+35+37+39+41+43+45+47+49+51+53+55+5
7+59+61+63+65+67+69+71+73+75+77+79+81+83+85+87+89+91+93+95+97+99=2500

1+3+5+7+9+11+13+15+17+19+21+23+25+27+29+31+33+35+37+39+41+43+45+47+49+51+53+55+5
7+59+61+63+65+67+69+71+73+75+77+79+81+83+85+87+89+91+93+95+97+99=2500
```

Питання для підготовки до захисту лабораторної роботи

1) Які оператори циклів використовуються для виконання повторюваних операцій в мові С++?

2) Чим відрізняються цикли з перед- і постумовами?

3) Який формат запису оператора **for**?

4) Що таке вкладені цикли?

5) Як можна задати вічний цикл?

5.6. Лабораторна робота №6: Застосування операторів циклів для формування псевдографічних рисунків

Мета роботи

Ознайомитись з принципами побудови псевдографічних зображень та поглибити отримані знання використання складного оператора циклу.

Хід роботи

1) Ознайомитись з методичними вказівками до лабораторної роботи щодо формування псевдографічних зображень засобами мов програмування та повторити тему "Цикли";

2) розкласти задану фігуру за рівнями і заповнити таблицю залежностей між кількістю символів та номером поточного рядка;

3) розробити схему алгоритму програми, яка формує і виводить на консоль псевдографічне зображення згідно із варіантом завдання;

4) за схемою алгоритму написати програму на мові С++;

5) зробити висновки;

6) підготувати звіт про виконання лабораторної роботи, що включає наступні пункти: номер, тема, мета та хід лабораторної роботи, завдання, псевдографічне зображення з розділенням на рівні, таблиця з рівняннями для формування рисунка, схема алгоритму програми, лістинг програми з коментуванням кожної інструкції, результат роботи програми (скріншот), висновки.

Теоретичні відомості та рекомендації до виконання

Текстові редактори не мають можливостей малювання за допомогою стандартних графічних об'єктів, тому програмісти, які базують свої процедурні програми для роботи з текстовою інформацією, змушені користуватись методами псевдографічної побудови фігур. Псевдографіка – це спосіб зображення рисунка за допомогою символів тексту. Нині псевдографічні зображення мають мале поширення – в основному це логотипи розробника в програмах чи файлах. Однак побудова псевдографічних зображень дозволяє поглибити розуміння особливостей використання операторів циклів. Складні графічні об'єкти легше намалювати, написавши просту програму, ніж проводити декілька годин у кропіткому підрахунку символів кожного рядка зображення. Причому використання операторів циклів не тільки спрощує малювання псевдографічних фігур, а й при правильному використанні вносить можливість їх масштабування.

При виконанні завдання до лабораторної роботи студент повинен поглибити і закріпити розуміння особливостей застосування операторів циклів на основі формування псевдографічного зображення.

Студент починає виконання роботи з малювання заданого зображення в пропорціях на аркуші в клітинку. Потім складна фігура розбивається за рівнями, що складаються лише з простих фігур, і рахуються символи в основі

кожного рівня. Після формуються залежності між кількістю символів та номером рядка поточного рівня і заповнюється відповідна таблиця.

Орієнтуючись на таблицю, розробляється схема алгоритму роботи програми, що будує псевдографічне зображення. За схемою алгоритму виконується написання тексту програми на мові С++.

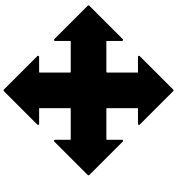
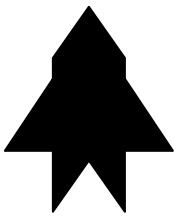
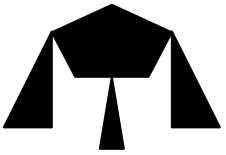
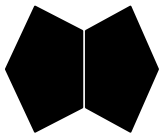
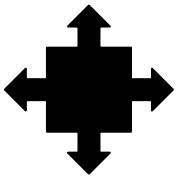
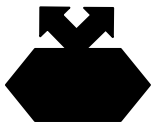
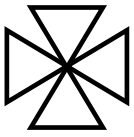
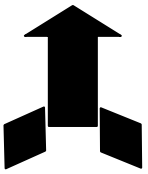
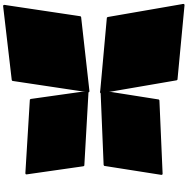
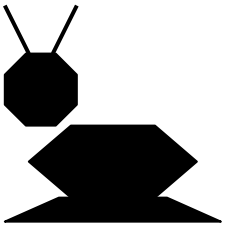
У висновках до звіту студент повинен розкрити призначення теми "Застосування операторів циклів для формування псевдографічних рисунків". Висновок має бути поданий в формі пояснення сфери застосування вивченого матеріалу згідно із розумінням його студентом.

Завдання для самостійного виконання

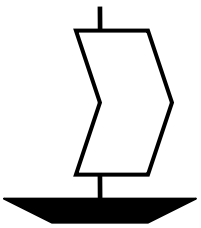
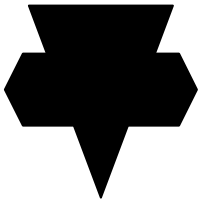
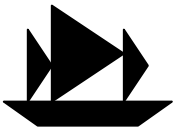
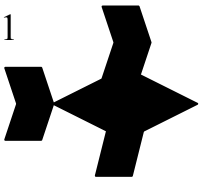
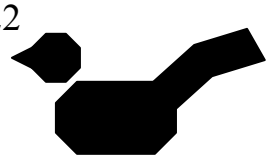
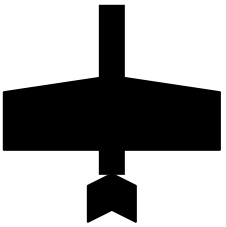
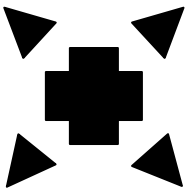
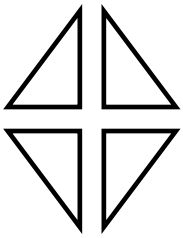
Розробити схему алгоритму та написати програму, яка виводить на консоль псевдографічне зображення згідно із варіантом (табл. 5.6). Для малювання фігури застосувати будь-який обраний символ.

Таблиця 5.6

Варіанти завдань

1 	2 	3 	4 
5 	6 	7 	8 
9 	10 	11 	12 
13 	14 	15 	16 

Закінчення табл. 5.6

17 	18 	19 	20 
21 	22 	23 	24 
25 	26 	27 	28 

Приклад виконання завдання

Псевдографічне зображення будь-якої складності може бути виконано за допомогою ряду рівнів з простих геометричних фігур.

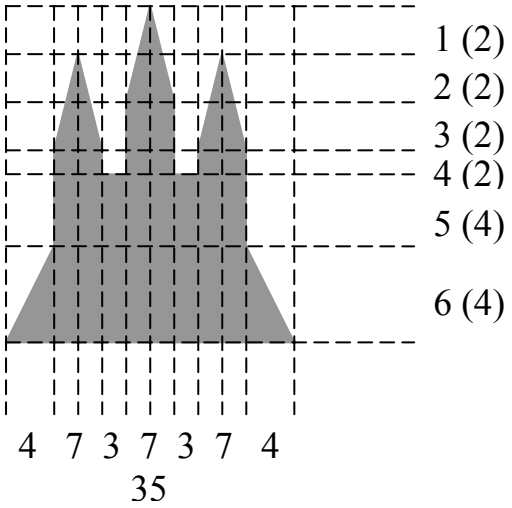


Рис. 5.6. Псевдографічне зображення замку, розкладене за рівнями

На рис. 5.6 за допомогою горизонтального розбивання складної фігури виконується визначення кількості простих рівнів для її малювання (кількість циклів) і число рядків, які складають кожний рівень. Вертикальне розбивання дозволяє визначити кількість символів основи фігури і її елементів. Звичайно перед виконанням написання програми для малювання псевдографічної фігури

необхідно виконати її побудову в пропорціях на аркуші у клітинку. Це полегшує уявлення про кінцевий вигляд зображення.

Для малювання фігури на консолі необхідно враховувати факт, що незайняті використовуваним символом позиції також повинні бути заповнені. Для цього звичайно використовується символ "ПРОБІЛ" (далі – пропуск), який при виводі відображається як вільна позиція. Таким чином, все зображення являє собою матрицю, в якій малюнок виконується обраним символом, а всі інші ланки заповнюються пропусками.

Щоб використовувати оператори циклів, необхідно знайти залежність кількості кожного з символів від номера рядка поточного рівня.

Пряму залежність, коли номер рядка і число символів збільшуються, знайти досить просто. Складність полягає у знаходженні зворотної залежності, коли при збільшенні номера рядка число символів зменшується. В цьому випадку необхідно згадати, що робота ведеться з матрицею чи підматрицею, з чого випливає, що формула зворотної залежності може бути виведена через формулу прямої.

Так, для наведеної нижче матриці

OOO**X**OOO
OO**XXX**OO
O**XXXXX**O
XXXXXXX

формула прямої залежності кількості символів "X" від номера рядка буде мати наступний вигляд:

X=2*i-1, де i-номер рядка поточного рівня.

Звідси формула залежності числа символів "O" в рядку зліва від трикутника буде наступною:

O=N-(2*i-1)/2, де N – число символів, які складають основу фігури.

Всю матрицю можна відобразити шляхом послідовного виводу символів "O" і "X", які складають зображення за допомогою циклів. Причому всі цикли формування послідовності з поточного символа будуть об'єднані одним зовнішнім, який виконує перебір рядків.

Так, для наведеної матриці програмний код псевдографічного зображення буде наступним:

```
for (i=1; i<=4; i++)              //цикл за рівнями малюнка (4 – кількість рядків)
{
        for(j=1; j<=((7-(2*i-1))/2); j++)        //цикл виводу відступів
            printf("O");                         //вивід символа "O"
        for(j=1; j<=(2*i-1); j++)                //цикл виводу символа "X"
```

```
            printf("X");                          //вивід символа "X"
        for(j=1; j<=((7-(2*i-1))/2); j++)         //цикл виводу відступів
            printf("O");                          //вивід символа "O"
        printf("\r\n");                           //перехід на новий рядок
}
```

Таким самим чином можна формувати і більш складні фігури, що складаються з декількох простих шляхом послідовного використання формул.

```
OOOXOOOOOOOOXOOO
OOXXXOOOOOOOXXXOO
OXXXXXOOOOOXXXXXO
XXXXXXXOOXXXXXXX
```

O=N-(2*i-1)/2;
X=2*i-1;
O=N-(2*i-1)+2 – кількість "O" повторюється двічі і в кожний рядок додається по два символи;
X=2*i-1;
O=N-(2*i-1)/2.

Програмний код псевдографічного зображення буде наступним:

```
for (i=1; i<=4; i++)        //цикл за рівнями малюнка (4 – кількість рядків)
{
        for(j=1; j<=((7-(2*i-1))/2); j++)       //цикл виводу відступів
            printf("O");                        //вивід символа"O"
        for(j=1; j<=(2*i-1); j++)               //цикл виводу символа "X"
            printf("X");                        //вивід символа "X"
        for(j=1; j<=((7-(2*i-1))+2); j++)       //цикл виводу відступів
            printf("O");                        //вивід символа"O"
        for(j=1; j<=(2*i-1); j++)               //цикл виводу символа "X"
            printf("X");                        //вивід символа "X"
        for(j=1; j<=((7-(2*i-1))/2); j++)       //цикл виводу відступів
            printf("O");                        //вивід символа "O"
        printf("\r\n");                         //перехід на новий рядок
}
```

Для більш складних фігур досить зручним є формування таблиці, згідно з якою будуть записуватись формули залежності числа символів від номера рядка за рівнем.

Для наведеного прикладу зображення замку можна сформувати таблицю (табл. 5.7).

Таблиця 5.7

Залежності кількості символів від номеру рядку

P.	**" "**	**"X"**	**" "**	**"X"**	**" "**	**"X"**
1	14+(7-(2*i-1))/2	2*i-1				
2	4+(7-(2*i-1))/2	2*i-1	3+(7-(2*i-1))/2+(7-(2*(i+2)-1))/2	2*(i+2)-1	3+(7-(2*i-1))/2+(7-(2*(i+2)-1))/2	2*i-1
3	4+(7-(2*(i+2)-1))/2	2*(i+2)-1	3+(7-(2*(i+2)-1))/2	7	3+(7-(2*(i+2)-1))/2	2*(i+2)-1
4	4	7	3	7	3	7
5	4	27				
6	(35-(2*(i+13)-1))/2	2*(i+13)-1				

Згідно із таблицею програма на мові С++ для формування псевдографічного зображення на консолі буде мати наступний вигляд:

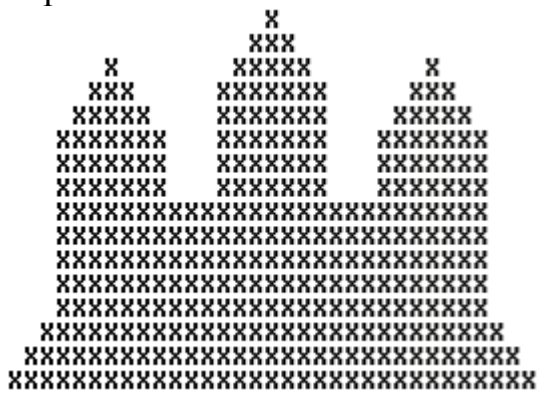
```
#include <stdio.h>                    //підключення бібліотеки вводу/виводу

void main()                           //оголошення та визначення головної функції
{
        int i, j;                     //оголошення змінних лічильників цілого типу
        for (i=1; i<=2; i++)          //цикл за першим рівнем малюнка
        {
                for(j=1; j<=(14+(7-(2*i-1))/2); j++)  //цикл виводу відступів
                        printf(" ");                  //вивід символа "ПРОБІЛ"
                for(j=1; j<=(2*i-1); j++)             //цикл виводу символа "X"
                        printf("X");                  //вивід символа "X"
                printf("\r\n");                       //перехід на новий рядок
        }
        for (i=1; i<=2; i++)          //цикл за другим рівнем малюнка
        {
                for(j=1; j<=(4+(7-(2*i-1))/2); j++)   //цикл виводу відступів
                        printf(" ");                  //вивід символа "ПРОБІЛ"
                for(j=1; j<=(2*i-1); j++)             //цикл виводу символа "X"
                        printf("X");                  //вивід символа "X"
//цикл виводу відступів
                for(j=1; j<=(3+(7-(2*i-1))/2+(7-(2*(i+2)-1))/2); j++)
                        printf(" ");                  //вивід символа "ПРОБІЛ"
                for(j=1; j<=(2*(i+2)-1); j++)         //цикл виводу символа "X"
                        printf("X");                  //вивід символа "X"
//цикл виводу відступів
```

```
        for(j=1; j<=(3+(7-(2*i-1))/2+(7-(2*(i+2)-1))/2); j++)
                printf(" ");                    //вивід символа "ПРОБІЛ"
        for(j=1; j<=(2*i-1); j++)               //цикл виводу символа "X"
                printf("X");                    //вивід символа "X"
        printf("\r\n");                         //перехід на новий рядок
}
for (i=1; i<=2; i++)                    //цикл за третім рівнем малюнка
{
        for(j=1; j<=(4+(7-(2*(i+2)-1))/2); j++)     //цикл виводу відступів
                printf(" ");                    //вивід символа "ПРОБІЛ"
        for(j=1; j<=(2*(i+2)-1); j++)           //цикл виводу символа "X"
                printf("X");                    //вивід символа "X"
        for(j=1; j<=(3+(7-(2*(i+2)-1))/2); j++)     //цикл виводу відступів
                printf(" ");                    //вивід символа "ПРОБІЛ"
        for(j=1; j<=7; j++)                     //цикл виводу символа "X"
                printf("X");                    //вивід символа "X"
        for(j=1; j<=(3+(7-(2*(i+2)-1))/2); j++)     //цикл виводу відступів
                printf(" ");                    //вивід символа "ПРОБІЛ"
        for(j=1; j<=(2*(i+2)-1); j++)           //цикл виводу символа "X"
                printf("X");                    //вивід символа "X"
        printf("\r\n");                         //перехід на новий рядок
}
for (i=1;i<=2;i++)                      //цикл за четвертим рівнем малюнка
{
        for(j=1; j<=4; j++)                     //цикл виводу відступів
                printf(" ");                    //вивід символа "ПРОБІЛ"
        for(j=1; j<=7; j++)                     //цикл виводу символа "X"
                printf("X");                    //вивід символа "X"
        for(j=1; j<=3; j++)                     //цикл виводу відступів
                printf(" ");                    //вивід символа "ПРОБІЛ"
        for(j=1; j<=7; j++)                     //цикл виводу символа "X"
                printf("X");                    //вивід символа "X"
        for(j=1; j<=3; j++)                     //цикл виводу відступів
                printf(" ");                    //вивід символа "ПРОБІЛ"
        for(j=1; j<=7; j++)                     //цикл виводу символа "X"
                printf("X");                    //вивід символа "X"
        printf("\r\n");                         //перехід на новий рядок
}
for (i=1; i<=4; i++)                    //цикл за п'ятим рівнем малюнка
{
        for(j=1; j<=4; j++)                     //цикл виводу відступів
                printf(" ");                    //вивід символа "ПРОБІЛ"
        for(j=1; j<=27; j++)                    //цикл виводу символа "X"
                printf("X");                    //вивід символа "X"
        printf("\r\n");                         //перехід на новий рядок
```

```
        }
        for (i=1; i<=4; i++)                    //цикл за шостим рівнем малюнка
        {
                for(j=1; j<=((35-(2*(i+13)-1))/2); j++)       //цикл виводу відступів
                        printf(" ");                          //вивід символа "ПРОБІЛ"
                for(j=1; j<=(2*(i+13)-1); j++)                //цикл виводу символа "X"
                        printf("X");                          //вивід символа "X"
                printf("\r\n");                               //перехід на новий рядок
        }
}
```

Результатом виконання програми буде наступне псевдографічне зображення на консолі:



Для того, щоб була можливість масштабування малюнка по вертикалі в кожний цикл по рядках можна додати вкладений цикл повторення рядків N кількість разів.

Так, для розглянутої матриці з символів "X" та "O" програмний код за масштабуванням по вертикалі буде мати наступний вигляд:

```
int N = 2;
for (i=1; i<=4; i++)        //цикл за рівнем малюнка (4 – кількість рядків)
{
        for (int k=1; k<=N; k++)                //цикл дублювання рядків N разів
        {
                for(j=1; j<=((7-(2*i-1))/2); j++)       //цикл виводу відступів
                        printf("O");                    //вивід символа "O"
                for(j=1; j<=(2*i-1); j++)               //цикл виводу символа "X"
                        printf("X");                    //вивід символа "X"
                for(j=1; j<=((7-(2*i-1))/2); j++)       //цикл виводу відступів
                        printf("O");                    //вивід символа "O"
                printf("\r\n");                         //перехід на новий рядок
        }
}
```

Результатом виконання цього коду буде матриця:

OOO**X**OOO
OOO**X**OOO
OO**XXX**OO
OO**XXX**OO
O**XXXXX**O
O**XXXXX**O
XXXXXXX
XXXXXXX

Для масштабування псевдографічного малюнка по горизонталі необхідно побудувати прямо пропорційну залежність усіх формул від кількості рядків.

Питання для підготовки до захисту лабораторної роботи

1) Що таке псевдографіка?

2) В чому полягає спрощення формування псевдографічного зображення з використанням операторів циклів?

3) Як виконується формування псевдографічного зображення за допомогою операторів циклів?

4) Як можна знайти зворотню залежність між числом символів та номером рядка?

5) Як виконується масштабування псевдографічних зображень?

5.7. Лабораторна робота №7: Принципи роботи з одновимірними масивами

Мета роботи

Ознайомитись з принципами формування масивів у мові С++ та отримати практичні навички роботи з їх елементами.

Хід роботи

1) Ознайомитись з методичними вказівками до лабораторної роботи та темою "Масиви";

3) відповідно до завдання розробити схему алгоритму програми, яка формує масив з 15 елементів, за вибором користувача ініціалізує його вручну або автоматично та виконує над ним вказану дію;

4) за схемою алгоритму написати програму на мові С++;

5) зробити висновки.

6) підготувати звіт про виконання лабораторної роботи, що включає наступні пункти: номер, тема, мета та хід лабораторної роботи, завдання, схема алгоритму програми, лістинг програми з коментуванням кожної інструкції, результат роботи програми (скріншот), висновки.

Теоретичні відомості та рекомендації до виконання

Для зберігання великої кількості значень одного типу даних в мові С++ використовуються масиви. Масиви дозволяють не тільки зберігати значення, але і надають можливість зручної організації операцій і дій з будь-яким з проміжних або всіма значеннями в цілому за допомогою операторів циклів.

При виконанні завдання до лабораторної роботи студент повинен поглибити і закріпити розуміння особливостей організації і застосування масивів у мові С++.

Виконання лабораторної роботи починається ознайомленням з теоретичними відомостями про організацію масивів. Потім студент приступає до розробки схеми алгоритму програми згідно із завданням за варіантом. За схемою алгоритму виконується написання програми на мові С++.

Згідно із завданням до лабораторної роботи необхідно написати програму, що виконує з масивом вказану за варіантом дію. Програма повинна дозволяти заповнювати масив як вручну через консоль, так і автоматично.

У висновках до звіту студент повинен розкрити призначення теми "Принципи роботи з одновимірними масивами". Висновок має бути поданий в формі пояснення сфери застосування вивченого матеріалу згідно із розумінням його студентом.

Завдання для самостійного виконання

Розробити схему алгоритму та написати програму, яка дозволяє заповнити масив Arr[15] за вибором користувача (вручну з консолі або програмно) та виконує з ним дію відповідно до варіанта завдання (табл. 5.8).

Таблиця 5.8

Варіанти завдань

№ вар.	Завдання
1	розрахунок середнього арифметичного значення для всіх елементів массиву, що є кратними 3-м, і вивід результату на консоль
2	пошук максимального серед всіх негативних значень елементів масиву і вивід результату на консоль
3	підрахунок і вивід на консоль кількості парних значень елементів масиву
4	вивід на консоль спочатку парних значень елементів масиву, а потім – непарних
5	розрахунок суми значень елементів масиву, що є кратними 7-ми, і вивід результату на консоль
6	розрахунок добутку всіх позитивних значень елементів масиву та вивід результату на консоль
7	розрахунок добутку всіх непарних значень елементів масиву та вивід результату на консоль
8	вивід на консоль всіх результатів обчислень середнього арифметичного значення для розташованих поруч елементів масиву
9	вивід на консоль всіх результатів розрахунку суми для пар елементів масиву, симетричних щодо центру
10	розрахунок середнього арифметичного значення для всіх негативних значень елементів масиву та вивід результату на консоль
11	пошук мінімального з усіх непарних значень елементів масиву і вивід його на консоль
12	пошук найменшого елемента массиву, що є кратним 3-м, і вивід його на консоль
13	розрахунок добутку всіх парних значень елементів масиву, не включаючи 0, і вивід результату на консоль
14	вивід на консоль сум для всіх розташованих поруч пар елементів масиву
15	пошук максимального непарного значення елементів масиву та вивід результату на консоль
16	підрахунок кількості негативних значень елементів масиву та вивід отриманого значення на консоль
17	пошук 3-х максимальних значень елементів масиву (max1> max2> max3) і вивід їх на консоль
18	вивід на консоль спочатку позитивних значень елементів масиву, а потім негативних
19	пошук максимального з усіх парних значень елементів масиву і вивід його на консоль
20	пошук мінімального з усіх позитивних значень елементів масиву і вивід його на консоль

Закінчення табл. 5.8

№ вар.	Завдання
21	розрахунок добутку всіх негативних значень елементів масиву та вивід результату на консоль
22	пошук найбільшого серед значень елементів масиву, що є кратними 5-ти, і вивід його на консоль
23	розрахунок суми всіх негативних значень елементів масиву та вивід результату на консоль
24	підрахунок кількості позитивних значень елементів масиву та вивід результату на консоль
25	вивід на консоль спочатку негативних значень елементів масиву, а потім позитивних
26	пошук значення в масиві, яке є найбільш близьким до середнього арифметичного всіх елементів, і вивід його на консоль
27	розрахунок суми всіх парних значень елементів масиву та вивід результату на консоль
28	вивід на консоль спочатку непарних значень елементів масиву, а потім парних
29	підрахунок кількості непарних значень елементів масиву та вивід результату на консоль
30	розрахунок добутку всіх значень елементів масиву, що є кратними 3-м, та вивід результату на консоль

Приклад виконання завдання

Розробити схему алгоритму та написати програму, яка дозволяє заповнити масив Arr[15] за вибором користувача (вручну з консолі або автоматично) та виконує розрахунок суми значень добутку всіх пар елементів массиву, симетричних щодо центру.

Розробимо схему алгоритму роботи програми (рис. 5.8).

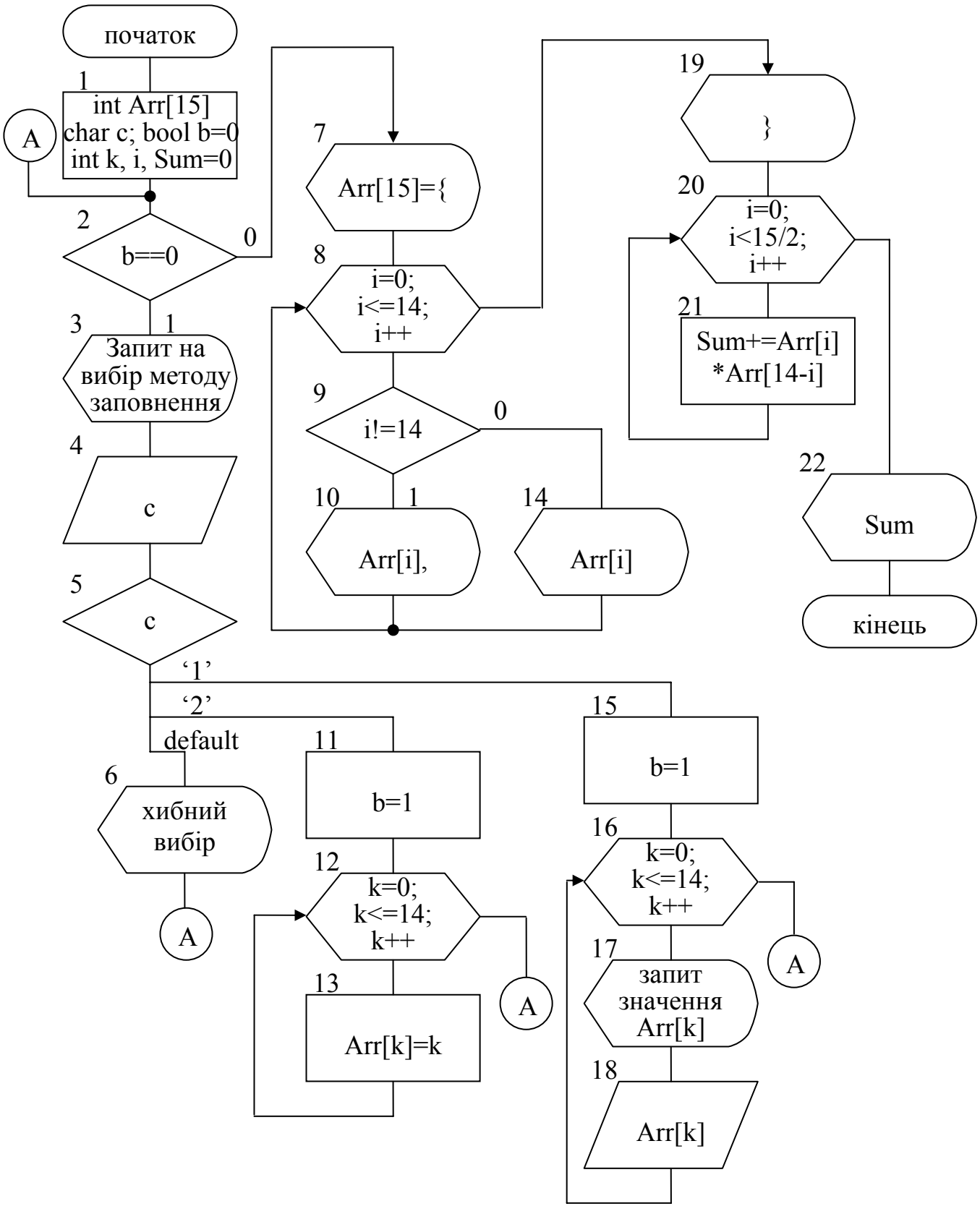


Рис. 5.8. Схема алгоритму програми, що виконує у масиві з 15 елементів обчислення добутку всіх значень, що симетричні відносно центра

Згідно зі схемою алгоритму напишемо програмний код:

```
#include <stdio.h>                  //підключення бібліотеки вводу/виводу

void main()                         //оголошення та визначення головної функції
{
        int Arr[15];                //оголошення цілого масиву з 15 елементів
        char c;                     //оголошення змінної  вибору методу заповнення
        bool b=0;    //оголошення та визначення прапорця правильного вибору
        int k,i;                    //оголошення змінних лічильників індексів масиву
        int Sum=0;  //оголошення та визначення змінної для розрахунку суми

        while(b==0)       //цикл, доки не буде обрано правильний метод заповнення
        {
                printf("Method for the arrey filling: "); //вивід константного рядка
                printf("1 – manual; 2 – auto\r\n");       //вивід константного рядка
                scanf("%c", &c);                  //отримання символа від користувача
                switch(c)                               //перевірка введеного символа
                {
                        case '1':                       //якщо  було  введено символ '1'
                                b=1;                    //встановити прапорець
                                printf("\r\n");         //перехід на новий рядок
//цикл заповнення масиву значеннями, що вводяться з консолі
                                for(k=0; k<=14; k++)
                                {
//запит значення поточного елементу
                                        printf("Arr[%i]=", k);
//отримання введеного значення
                                        scanf("%i", &Arr[k]);
                                }
                                break;
                        case '2':                       //якщо  було  введено символ '2'
                                b=1;                    //встановити прапор
//цикл заповнення масиву значеннями від 0 до 14
                                for(k=0; k<=14; k++)
                                {
                                        Arr[k]=k;       //визначення елемента масиву
                                }
                                break;                  //вихід з оператора
                        default:                  //якщо  було  введено інший символ
//вивід повідомлення користувачеві
                                printf("You enter wrong symbol!\r\n\r\n");
                }
        }
        printf("\r\n");                                 //перехід на новий рядок
```

```
        printf("Massiv Arr[15]={");                    //вивід константного рядка
        for(i=0; i<=14; i++)                //цикл виводу значень заповненого масиву
                if(i!=14)                           //якщо елемент неостанній
                        printf("%i,", Arr[i]); //вивід значення неостаннього елемента
                else                                //якщо елемент останній
                        printf("%i", Arr[i]); //вивід значення останнього елемента
        printf("}\r\n");                            //вивід константного рядка
//цикл перебору половини елементів масиву для розрахунку суми добутків
        for(i=0; i<15/2; i++)
        {
//додавання до змінної Sum добутку  симетричних відносно центра значень
                Sum+=Arr[i]*Arr[14-i];
        }
        printf("\r\nCalculation result: %i\r\n\r\n", Sum);          //вивід результату
}
```

Результат роботи програми при ручному заповненні масиву буде наступним:

```
Method for the arrey filling: 1 - manual; 2 - auto
1

Arr[0]=1
Arr[1]=7
Arr[2]=3
Arr[3]=4
Arr[4]=9
Arr[5]=34
Arr[6]=10
Arr[7]=-5
Arr[8]=0
Arr[9]=12
Arr[10]=16
Arr[11]=-2
Arr[12]=6
Arr[13]=13
Arr[14]=2

Massiv Arr[15]={1,7,3,4,9,34,10,-5,0,12,16,-2,6,13,}

Calculation result: 655
```

Питання для підготовки до захисту лабораторної роботи

1) Дайте визначення поняттям масив, елемент масиву та індекс елемента масиву.

2) Як виконується ініціалізація масиву при оголошенні?

3) Як отримати доступ до значення елементу масиву?

4) Для чого слугують масиви?

5) Яких видів бувають масиви?

6) Як інакше називають одновимірні і двовимірні масиви?

7) Що називають автоматичним визначенням масиву?

5.8. Лабораторна робота №8: Принципи роботи з двовимірними масивами

Мета роботи

Ознайомитись з двовимірними масивами та отримати практичні навички роботи з їх елементами

Хід роботи

1) Ознайомитись з методичними вказівками до лабораторної роботи та повторити тему "Масиви";

2) відповідно до завдання розробити схему алгоритму програми, що виконує зчитування чи заповнення масиву за вказаною схемою;

3) за схемою алгоритму написати програму на мові С++;

4) зробити висновки;

5) підготувати звіт про виконання лабораторної роботи, що включає наступні пункти: номер, тема, мета та хід лабораторної роботи, завдання, схема алгоритму програми, лістинг програми з коментуванням кожної інструкції, результат роботи програми (скріншот), висновки.

Теоретичні відомості та рекомендації до виконання

Людина завжди прагне до спрощення сприйняття навколишнього світу. Одним з проявів є сортування об'єктів і об'єднання їх у групи за будь-якою ознакою. Таким самим чином роблять і в програмуванні, об'єднуючи змінні з однаковим типом даних під одним ім'ям і користуючись ними як елементами групи. Це спрощує задачу сприйняття та обробки безлічі змінних одного типу даних. Такі групи називають масивами. Масиви можуть слугувати як для зберігання N-ї кількості значень, так і для виконання з ними операцій. Причому використання масивів не обмежується роботою з послідовністю значень. Існують також поняття багатовимірних масивів, які являють собою послідовності з послідовностей значень, що досить часто може бути зручно. В основному використовуються одновимірні, двовимірні та тривимірні масиви.

При виконанні завдання до лабораторної роботи студент повинен ознайомитися з особливостями оголошення та ініціалізації двовимірних масивів в мові С++, а також навчитися виконувати операції над ними.

Виконання лабораторної роботи починається ознайомленням з теоретичними відомостями про масиви. Потім студент приступає до розробки схеми алгоритму програми. Залежно від обраного викладачем завдання, виконується заповнення двовимірного масиву розмірністю 5х5 значеннями "0" та "1" згідно з табл. 5.9 або читання/запис значень елементів масиву за алгоритмом, що зазначено в табл. 5.10. За схемою алгоритму здійснюється написання програми мовою С++.

Всі схеми, наведені в таблицях, формуються на основі формул залежностей між індексами за розмірностями. Так, для кожної з діагоналей двовимірного масиву можна вказати свою залежність:

для правих діагоналей

	i+j=0	i+j=1	i+j=2	i+j=3		
i/j	**0**	**1**	**2**	**3**	**4**	
0	0,0	0,1	0,2	0,3	0,4	
1	1,0	1,1	1,2	1,3	1,4	i+j=5
2	2,0	2,1	2,2	2,3	2,4	i+j=6
3	3,0	3,1	3,2	3,3	3,4	i+j=7
4	4,0	4,1	4,2	4,3	4,4	i+j=8,

для лівих діагоналей

i/j	**0**	**1**	**2**	**3**	**4**	
0	0,0	0,1	0,2	0,3	0,4	i-j=-4
1	1,0	1,1	1,2	1,3	1,4	i-j=-3
2	2,0	2,1	2,2	2,3	2,4	i-j=-2
3	3,0	3,1	3,2	3,3	3,4	i-j=-1
4	4,0	4,1	4,2	4,3	4,4	

i-j=4	i-j=3	i-j=2	i-j=1		i-j=0.

При правильному виборі алгоритму заповнення масиву програма дозволить виконувати заповнення масиву будь-якої розмірності!

У висновках до звіту студент повинен розкрити призначення теми "Принципи роботи з двовимірними масивами". Висновок має бути поданий в формі пояснення сфери застосування вивченого матеріалу згідно із розумінням його студентом.

Завдання для самостійного виконання

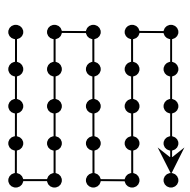
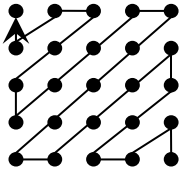
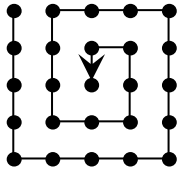
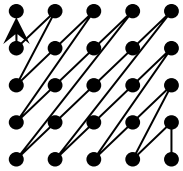
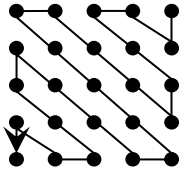
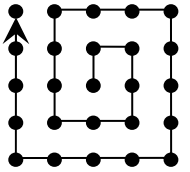
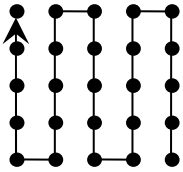
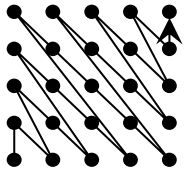
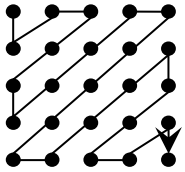
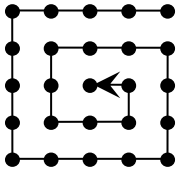
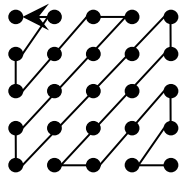
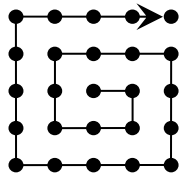
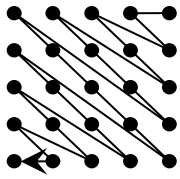
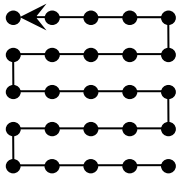
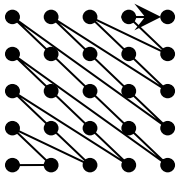
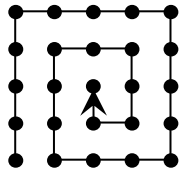
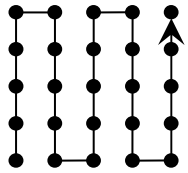
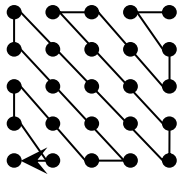
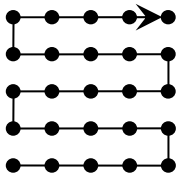
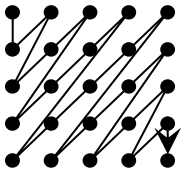
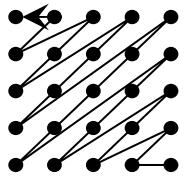
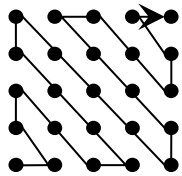
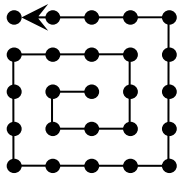
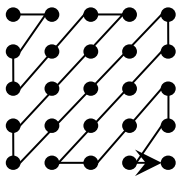
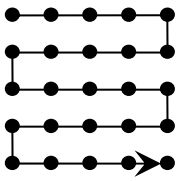
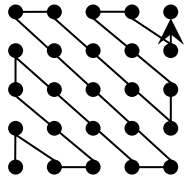
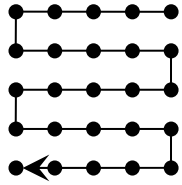
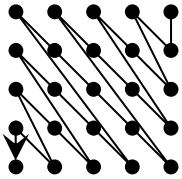
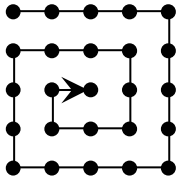
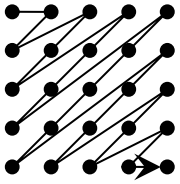
Розробити схему алгоритму та написати програму на мові С++, яка виконує (складність завдання обирається викладачем):

а) для парних варіантів читання, для непарних – запис значень елементів двовимірного масиву Arr[5,5] відповідно до вказаної у табл. 5.9 схемі. Результати як читання, так і запису повинні бути відображені на консолі. Значення елементів масиву приймаються від 1 до 25;

б) заповнення двовимірного масиву Arr[5,5] значеннями "0" та "1" згідно з зазначеною у табл. 5.10 схемою і вивід отриманої матриці на консоль.

Таблиця 5.9

Варіанти завдань складності А

1 	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 

Таблиця 5.10

Варіанти завдань складності Б

1	2	3	4	5
1 1 1 1 1	0 0 1 0 0	1 0 1 0 1	1 1 1 1 1	0 1 1 0 1
1 1 1 1 0	0 1 1 1 0	0 1 0 1 0	1 0 0 0 1	1 1 0 1 1
1 1 1 0 0	1 1 1 1 1	1 0 1 0 1	1 0 1 0 1	1 0 1 1 0
1 1 0 0 0	0 1 1 1 0	0 1 0 1 0	1 0 0 0 1	0 1 1 0 1
1 0 0 0 0	0 0 1 0 0	1 0 1 0 1	1 1 1 1 1	1 1 0 1 1

Закінчення табл. 5.10

6					7					8					9					10				
1	0	1	1	0	0	0	0	0	1	1	1	0	1	1	0	1	0	1	0	0	0	0	0	0
1	1	0	1	1	0	0	0	1	1	1	0	0	0	1	1	0	1	0	1	0	1	1	1	0
0	1	1	0	1	0	0	1	1	1	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0
1	0	1	1	0	0	1	1	1	1	1	0	0	0	1	1	0	1	0	1	0	1	1	1	0
1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	0	1	0	1	0	0	0	0	0	0

11					12					13					14					15				
1	0	0	1	0	1	0	0	0	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1
0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	1	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	1	1
1	0	0	1	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1

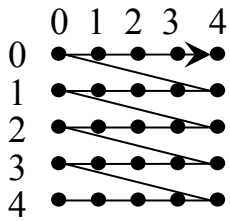
16					17					18					19					20				
0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1
1	1	1	1	1	0	0	1	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	1	1	1	0	0	1	0	0	0	1
1	1	1	1	1	0	1	0	0	1	0	1	0	0	0	1	1	1	1	0	1	1	0	0	0
0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1	0	0

21					22					23					24					25				
1	1	0	0	0	0	1	1	1	0	0	0	1	0	0	0	1	1	1	1	0	1	1	1	1
1	1	1	0	0	1	0	1	0	1	1	0	0	1	0	1	0	1	1	1	0	0	1	1	1
0	1	1	1	0	1	1	0	1	1	0	1	0	0	1	1	1	0	1	1	0	0	0	1	1
0	0	1	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	1	0	0	0	0	1
0	0	0	1	1	0	1	1	1	0	1	0	0	1	0	1	1	1	1	0	0	0	0	0	0

26					27					28					29					30				
0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	0	0	1	0	0	1	1	1	1	0
0	0	0	0	1	0	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	1	0	1
0	0	0	1	1	0	1	1	1	0	0	0	1	0	0	1	0	0	1	0	1	1	0	1	1
0	0	1	1	1	1	1	1	0	0	0	1	0	1	0	0	0	1	0	0	1	0	1	1	1
0	1	1	1	1	1	1	0	0	0	1	0	0	0	1	0	1	0	0	1	0	1	1	1	1

Приклад виконання завдання

Складність А: необхідно заповнити двовимірний масив значеннями від 1 до 25 за наведеною нижче схемою і вивести отриману матрицю на консоль:



На схемі можна побачити, що при такому алгоритмі запису значень елементи кожного рядка перебираються від 0 до 4, а самі рядки від 4 до 0.

Виходячи з цього, розробимо схему алгоритму програми (рис. 5.9).

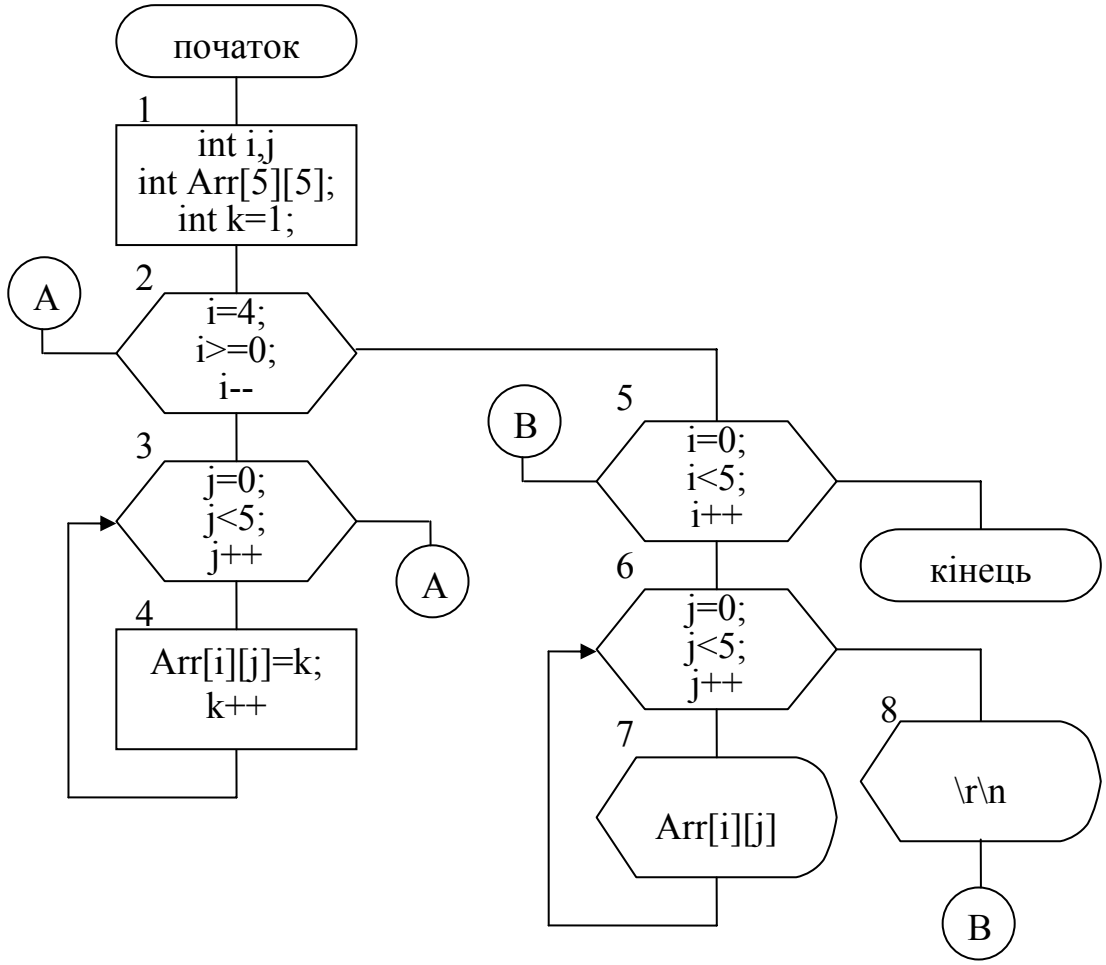


Рис. 5.9. Схема алгоритму програми, що заповнює двовимірний масив значеннями від 1 до 25

Згідно зі схемою алгоритму напишемо програму на мові С++:

```
#include <stdio.h>                    //підключення бібліотеки вводу/виводу
void main()                           //оголошення та визначення головної функції
{
      int Arr[5][5];       //оголошення цілочисельного масиву з розмірністю 5x5
      int k=1;                  //оголошення та визначення змінної лічильника
      int i, j;                       //оголошення змінних лічильників для циклів
      for(i=4; i>=0; i--)             //цикл перебору рядків у зворотному порядку
      {
            for(j=0; j<5; j++)        //цикл перебору елементів рядка
            {
                  Arr[i][j]=k;        //визначення поточного елемента
                  k++;                //збільшення лічильника
            }
      }
```

```
	for(i=0; i<5; i++)		//цикл перебору рядків
	{
		for(j=0; j<5; j++)		//цикл перебору елементів рядка
		{
			printf("%2i  ", Arr[i][j]);		//вивід поточного значення
		}
		printf("\r\n");		//перехід на новий рядок
	}
}
```

Результат виконання програми буде наступним:

```
21  22  23  24  25
16  17  18  19  20
11  12  13  14  15
 6   7   8   9  10
 1   2   3   4   5
```

Якщо в якості граничного значення для заповнення масиву використовувати змінну, що оголошена та визначена на початку програми, то алгоритм заповнення можна використовувати для правильного масиву будь-якої розмірності.

Змінимо програму, наведену вище, додавши змінну граничного значення та пов'язавши всі цикли з її значенням:

```
#include <stdio.h>			//підключення бібліотеки вводу/виводу
void main()				//оголошення та визначення головної функції
{
	int n=7;			//оголошення та визначення граничної змінної
	int Arr[7][7];		//оголошення цілочисельного масиву з розмірністю 7x7
	int k=1;			//оголошення та визначення змінної лічильника
	int i, j;			//оголошення змінних лічильників для циклів
	for(i=n-1; i>=0; i--)		//цикл перебору рядків у зворотному порядку
	{
		for(j=0; j<n; j++)		//цикл перебору елементів рядка
		{
			Arr[i][j]=k;		//визначення поточного елемента
			k++;			//збільшення лічильника
		}
	}
	for(i=0; i<n; i++)			//цикл перебору рядків
	{
		for(j=0; j<n; j++)		//цикл перебору елементів рядка
			printf("%2i  ", Arr[i][j]);		//вивід поточного значення
		printf("\r\n");			//перехід на новий рядок
	}
}
```

Результат виконання програми буде наступним:

```
43  44  45  46  47  48  49
36  37  38  39  40  41  42
29  30  31  32  33  34  35
22  23  24  25  26  27  28
15  16  17  18  19  20  21
 8   9  10  11  12  13  14
 1   2   3   4   5   6   7
```

Складність Б: необхідно заповнити масив Arr[9][9] значеннями "0" та "1" за наступною схемою:

1	1	1	1	1
1	1	0	1	1
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1

Як видно за схемою, значенням "1" заповнюються перший і останній стовпці, перший і останній рядки та діагоналі, всі інші елементи містять значення "0". Виходячи з цього, розробимо схему алгоритму роботи програми (рис. 5.10).

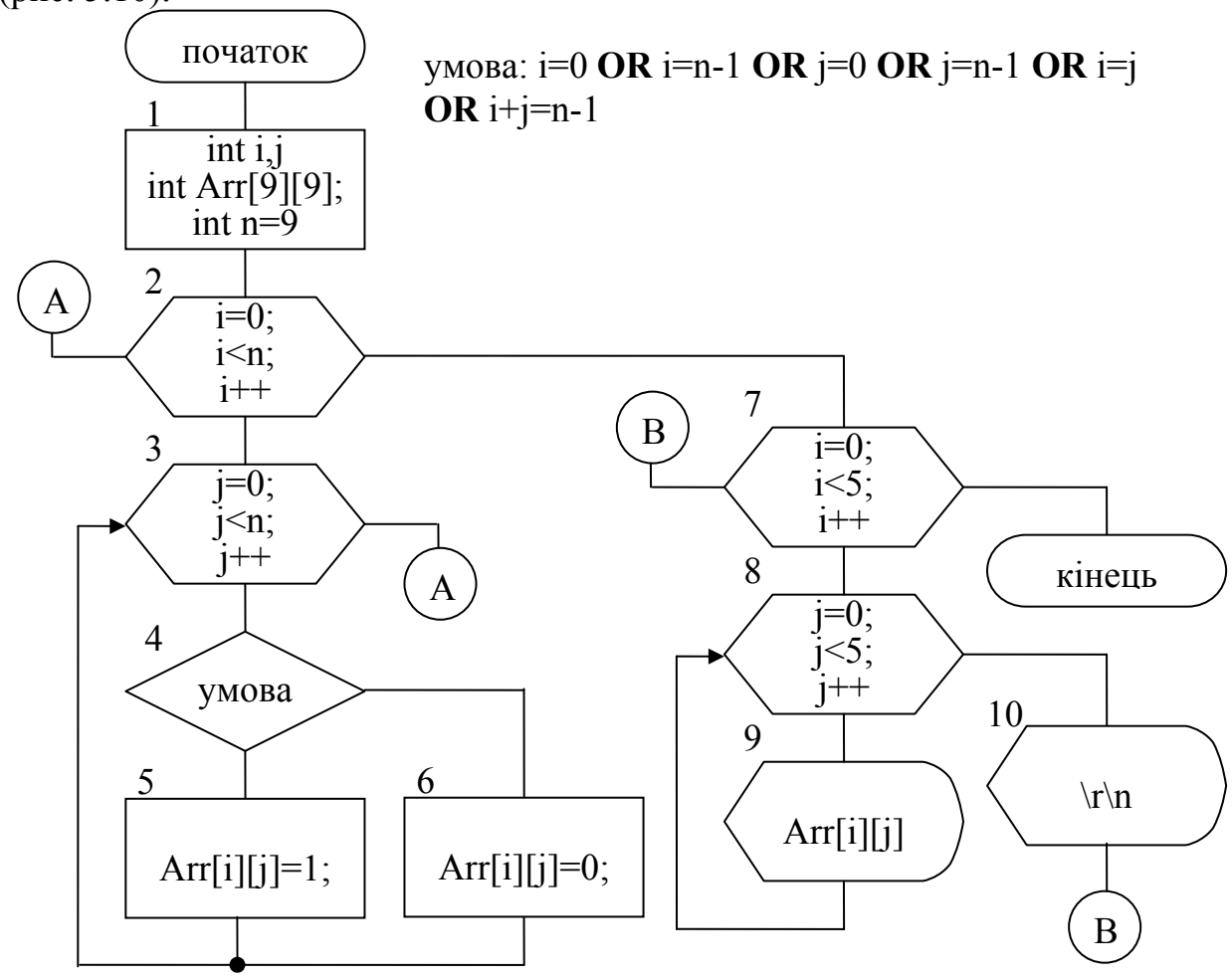


Рис. 5.10. Схема алгоритму програми, що заповнює двовимірний масив значеннями "0" та "1"

Згідно зі схемою алгоритму напишемо програму на мові С++:

```
#include<stdio.h>                    //підключення бібліотеки вводу/виводу

void main()                          //оголошення та визначення головної функції
{
        int n=9;                     //оголошення та визначення граничної змінної
        int Arr[9][9];               //оголошення цілочисельного масиву з розмірністю 9x9
        int i, j;                    //оголошення та визначення змінної лічильника
        for(i=0; i<n; i++)                   //цикл перебору рядків
                for(j=0; j<n; j++)           //цикл перебору елементів рядка
//умова для запису значення "1"
                        if(i= =0 || i= =n-1 || j= =0 || j= =n-1 || i= =j || i+j= =n-1)
                                Arr[i][j]=1;         //визначення поточного елементу
                        else                         //інакше, запис значення "0"
                                Arr[i][j]=0;         //визначення поточного елемента
        for(i=0; i<n; i++)                   //цикл перебору рядків
                for(j=0; j<n; j++)           //цикл перебору елементів рядка
                        printf("%2i  ", Arr[i][j]); //вивід поточного елемента
                printf("\r\n");              //перехід на новий рядок
}
```

Результат роботи програми буде наступним:

```
 1   1   1   1   1   1   1   1   1
 1   1   0   0   0   0   0   1   1
 1   0   1   0   0   0   1   0   1
 1   0   0   1   0   1   0   0   1
 1   0   0   0   1   0   0   0   1
 1   0   0   1   0   1   0   0   1
 1   0   1   0   0   0   1   0   1
 1   1   0   0   0   0   0   1   1
 1   1   1   1   1   1   1   1   1
```

Питання для підготовки до захисту лабораторної роботи

1) Які види багатовимірних масивів є найбільш поширеними?

2) Для чого застосовуються багатовимірні масиви?

3) За якими правилами можна пов'язувати елементи діагоналей матриці?

4) Як отримати доступ до елемента багатовимірного масиву?

5) Застосування яких операторів спрощує роботу з масивами?

6) Яка умова дозволяє вибірково заповнювати елементи правої діагоналі матриці?

5.9. Лабораторна робота №9: Змінні-покажчики та алгоритми впорядковування масивів

Мета роботи

Ознайомитись зі змінними-покажчиками і методиками сортування числових послідовностей та отримати практичні навички роботи з динамічними масивами.

Хід роботи

1) Ознайомитись з методичними вказівками до лабораторної роботи та темами "Змінні-покажчики" та "Алгоритми впорядковування";

2) згідно із завданням розробити схему алгоритму програми, що виконує впорядковування динамічного масиву;

3) за схемою алгоритму написати програму на мові С++;

4) зробити висновки;

5) підготувати звіт про виконання лабораторної роботи, що включає наступні пункти: номер, тема, мета та хід лабораторної роботи, завдання, схема алгоритму програми, лістинг програми з коментуванням кожної інструкції, результат роботи програми (скріншот), висновки.

Теоретичні відомості та рекомендації до виконання

Доступ до значень змінних може бути організовано не тільки через ім'я змінної, але і через її явну адресу в пам'яті. Для зберігання і користування адресами в мові С++ використовуються спеціальні змінні-покажчики. Їх застосування веде до спрощення організації зв'язків між різними програмними модулями і дозволяє розширювати область видимості змінних. Однією з найважливіших властивостей змінних-покажчиків є можливість створення масивів змінної довжини.

При виконанні завдання до лабораторної роботи студент повинен поглибити і закріпити знання про особливості роботи з масивами, ознайомитись із застосуванням змінних-покажчиків в мові С++ та навчитися застосовувати раніш вивчені оператори для реалізації алгоритмів сортування.

Виконання лабораторної роботи починається ознайомленням з теоретичними відомостями про змінні-покажчики, динамічні масиви і методики сортування. Потім студент приступає до розробки схеми алгоритму програми згідно із варіантом завдання. За схемою алгоритму виконується написання програми на мові С++.

Для реалізації програмного заповнення динамічного масиву невпорядкованими значеннями рекомендується використовувати наступну функцію:

```
//підключення бібліотек для реалізації функцій заповнення масиву
//невпорядкованими значеннями вектора
#include<algorithm>
#include<functional>
```

```
#include<vector>

using namespace std;                    //застосування простору імен std

//оголошення та визначення функції, що виконує  заповнення масиву  змішаними
//значеннями
void RandomShuffle(int *pA, int r, int x, int n)
{
        const int VECTOR_SIZE = r ;                 //число елементів вектора
//створення вектора
        typedef vector<int> IntVector ;
        typedef IntVector::iterator IntVectorIt ;

        IntVector Numbers(VECTOR_SIZE) ;
        IntVectorIt start, end, it ;
        for (int i=0; i<r; i++)                     //заповнення вектора значеннями
        {
                Numbers[i] = x;
                x=x+n;
        }
//визначення меж вектора
        start = Numbers.begin();
        end = Numbers.end();
//виклик функції, що перемішує значення вектора
        random_shuffle(start, end);
//заповнення динамічного масиву значеннями вектора
        for(it = start; it != end; it++)
                *(pA++)=*it;
}
```

Додавання наведеного коду необхідно виконати в програмі перед головною функцією, щоб забезпечити можливість її виклику в тілі main(). Формат виклику наступний:

```
void RandomShuffle(int *pA, int r, int x, int n),
```

де pA – змінна-покажчик на масив, який необхідно заповнити невпорядкованими значеннями;
r – розмірність масиву;
x – значення, що визначає мінімальний елемент невпорядкованої послідовності;
n – значення, що визначає крок між значеннями елементів послідовності.

У висновках до звіту студент повинен розкрити призначення теми "Змінні-покажчики та алгоритми впорядковування масивів". Висновок має бути

поданий в формі пояснення сфери застосування вивченого матеріалу згідно із розумінням його студентом.

Завдання для самостійного виконання

Розробити схему алгоритму та написати програму, яка створює динамічний масив з кількістю елементів за вибором користувача, заповнює його через консоль чи невпорядкованими значеннями з програми та виконує їх сортування згідно з методом за варіантом, що вказано в табл. 5.11.

Таблиця 5.11

Варіанти завдань

№ вар.	Методика сортування	Тип даних масиву	Принцип сортування
1	метод Шейкера	**int**	зростання
2	оптимізований метод бульбашки	**int**	спадання
3	метод вставки	**int**	зростання
4	метод Шейкера з перевіркою числа обмінів	**int**	спадання
5	метод вибірки	**int**	зростання
6	оптимізований метод бульбашки з перевіркою числа обмінів	**int**	спадання
7	метод Шелла	**int**	зростання
8	метод вибірки-вставки	**int**	спадання
9	метод Шейкера	**char**	спадання
10	оптимізований метод бульбашки	**char**	зростання
11	метод вставки	**char**	спадання
12	метод Шейкера з перевіркою числа обмінів	**char**	зростання
13	метод вибірки	**char**	спадання
14	оптимізований метод бульбашки з перевіркою числа обмінів	**char**	зростання
15	метод Шелла	**char**	спадання
16	метод вибірки-вставки	**char**	зростання
17	метод Шейкера	**unsigned int**	зростання
18	оптимізований метод бульбашки	**unsigned int**	спадання
19	метод вставки	**unsigned int**	зростання
20	метод Шейкера з перевіркою числа обмінів	**unsigned int**	спадання
21	метод вибірки	**unsigned int**	зростання
22	оптимізований метод бульбашки з перевіркою числа обмінів	**unsigned int**	спадання
23	метод Шелла	**unsigned int**	зростання
24	метод вибірки-вставки	**unsigned int**	спадання

Закінчення табл. 5.11

№ вар.	Методика сортування	Тип даних масиву	Принцип сортування
25	метод Шейкера	**unsigned char**	спадання
26	оптимізований метод бульбашки	**unsigned char**	зростання
27	метод вставки	**unsigned char**	спадання
28	метод Шейкера з перевіркою числа обмінів	**unsigned char**	зростання
29	метод вибірки	**unsigned char**	спадання
30	оптимізований метод бульбашки з перевіркою числа обмінів	**unsigned char**	зростання

Приклад виконання завдання

Розробити схему алгоритму та написати програму, яка створює динамічний масив з кількістю елементів за вибором користувача, заповнює його через консоль чи невпорядкованими значеннями з програми та виконує їх сортування згідно з неоптимізованим методом бульбашки в порядку спадання.

Розробимо схему алгоритму роботи програми (рис. 5.11).

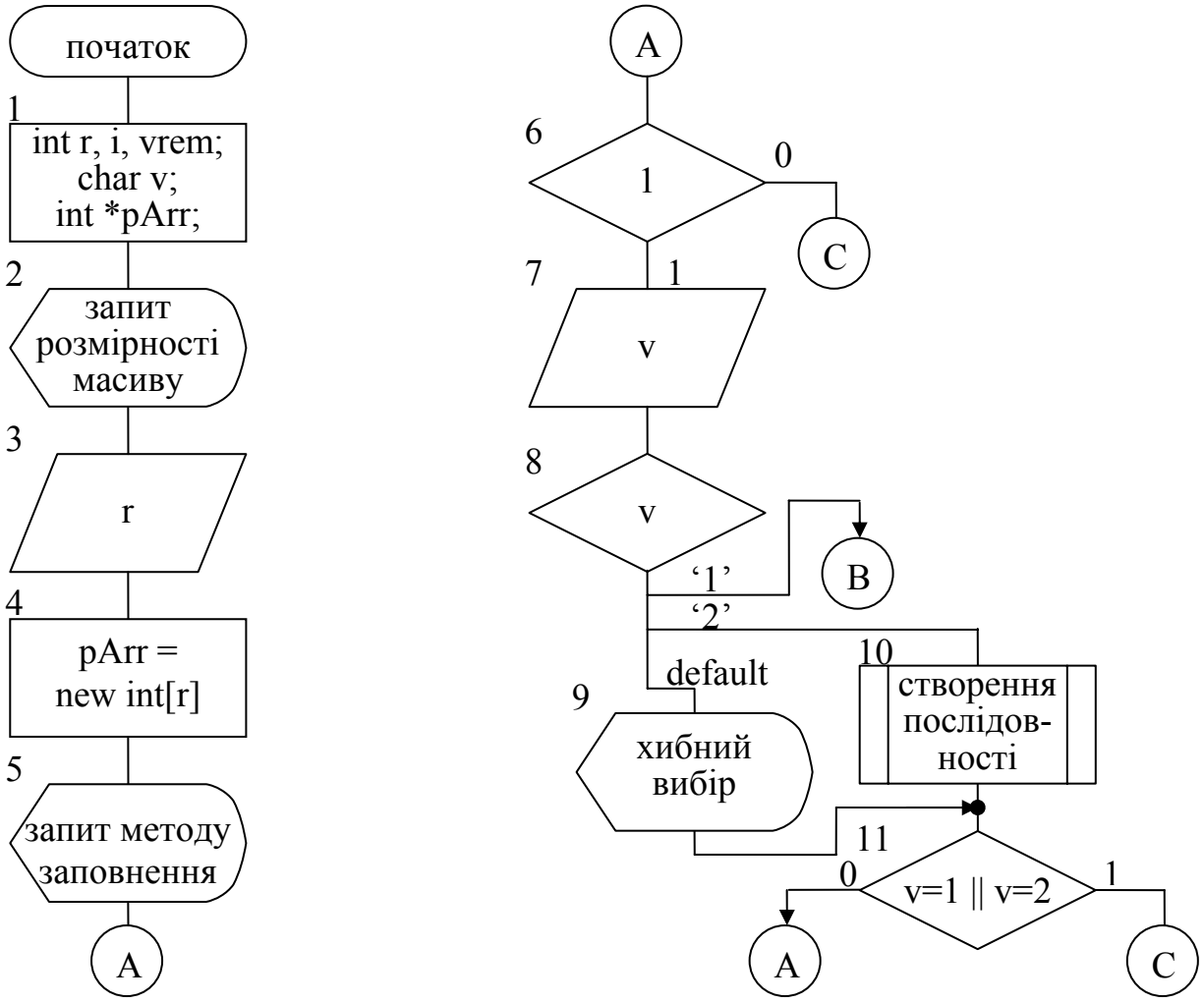


Рис. 5.11. Початок схеми алгоритму програми, сортування масиву

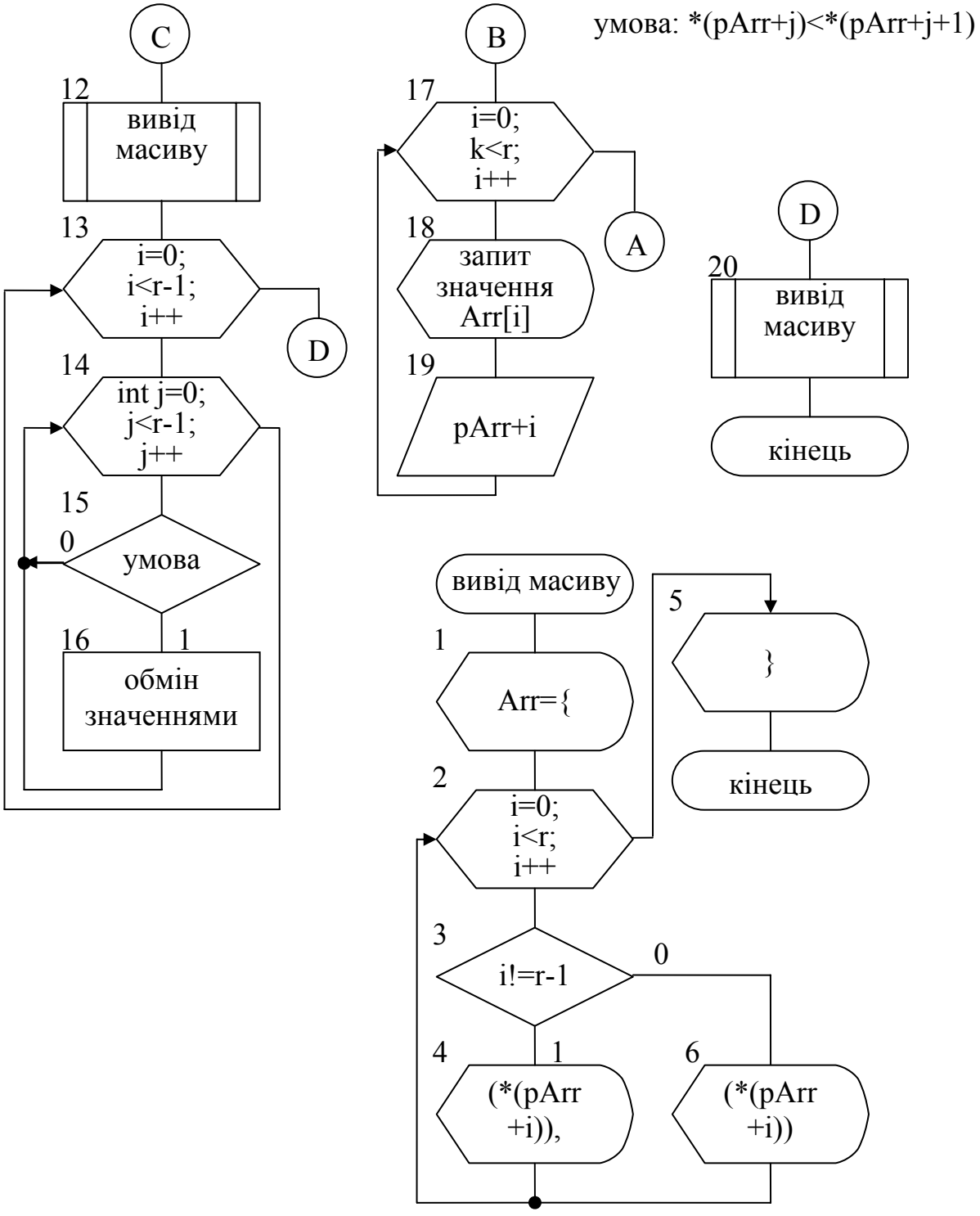


Рис. 5.11. Кінець схеми алгоритму програми, сортування масиву

Згідно зі схемою алгоритму напишемо програмний код:

```
#include<stdio.h>                    //підключення бібліотеки  вводу/виводу
// підключення бібліотек  для реалізації  функції заповнення масиву
//невпорядкованими значеннями вектора
#include<algorithm>
#include<functional>
```

```
#include<vector>

//визначення робочого простору
using namespace std;

//оголошення та визначення функції, що виконує  заповнення масиву  змішаними
//значеннями
void RandomShuffle(int *pA, int r, int x, int n)
{
	const int VECTOR_SIZE = r ;		//число елементів вектора

//створення вектора
	typedef vector<int> IntVector ;
	typedef IntVector::iterator IntVectorIt ;

	IntVector Numbers(VECTOR_SIZE) ;
	IntVectorIt start, end, it ;
	for (int i=0; i<r; i++)			//заповнення вектора значеннями
	{
		Numbers[i] = x;
		x=x+n;
	}
//визначення меж вектора
	start = Numbers.begin();
	end = Numbers.end();
	random_shuffle(start, end); //виклик функції, яка змішує значення вектора
//заповнення динамічного масиву значеннями вектора
	for(it = start; it != end; it++)
		*(pA++)=*it;
}

void main()			//оголошення та визначення головної функції
{
	int r;			//оголошення змінної розмірності масиву
	char v;			//оголошення змінної вибору методу заповнення
	int i;			//оголошення змінної лічильника елементів масиву
	int vrem;		//оголошення змінної для операції обміну
	int *pArr;		//оголошення змінної-покажчика на цілий тип даних
//запит  до користувача щодо визначення розмірності масиву
	printf("Set number of elements in the array: ");
	scanf("%i", &r);		//отримання  розмірності масиву
	pArr=new int[r];//створення динамічного масиву вказаної розмірності
//вивід константних рядків
	printf("\r\nCheck '1' for manual array initialization \r\n");
	printf("or '2' for random program array initialization\r\n");
```

```
        while(1)                                //вічний цикл
        {
//запит  до користувача щодо вибору методу  заповнення масиву
                printf("\r\nWhat is your check? ");
                scanf("%i", &v);                //зчитування вибору користувача
                switch(v)                       //перевірка введеного значення
                {
                        case 1:                 //якщо введено значення 1
                                for(i=0; i<r; i++)  //цикл перебору елементів масиву
                                {
//запит  до користувача щодо введення значення поточного елемента
                                        printf("Set Arr[%i] value: ", i);
//отримання введеного значення
                                        scanf("%i", pArr+i);
                                }
                                break;          //вихід з оператора
                        case 2:                 //якщо введено значення 2
//виклик  функції  заповнення масиву невпорядкованими значеннями
                                RandomShuffle(pArr, r, 1, 1);
                                break;          //вихід з оператора
                        default:                //якщо введено інше значення
//повідомлення користувачеві, що було зроблено хибний вибір
                                printf("The check is wrong!\r\n");
                }
//якщо користувач зробив вірний вибір, виконати вихід з циклу
                if (v==1 || v==2)
                        break;                  //вихід з оператора
        }
//вивід  невпорядкованого масиву на консоль
        printf ("\r\nArr = {");                //вивід константного рядка
        for (i=0; i<r; i++)                     //цикл перебору елементів масиву
        {
                if (i!=r-1)             //якщо поточний елемент у масиві неостанній
//вивід значення елемента масиву та символу ","
                        printf ("%i,  ", *(pArr+i));
                else                    //інакше – поточний елемент у масиві останній
                        printf ("%i", *(pArr+i)); //вивід значення елемента масиву
        }
        printf ("}\r\n");                       //вивід константного рядка
//реалізація неоптимізованого методу бульбашкового  сортування
        for(i=0; i<r-1; i++)                    //цикл для організації  проходів
                for(int j=0; j<r-1; j++)    //цикл для організації  кроків порівняння
//порівняння поточного елемента з наступним
                        if(*(pArr+j)<*(pArr+j+1))
                        {
```

```
//переприсвоїти поточне значення у тимчасову змінну
                        vrem=*(pArr+j);
//переприсвоїти значення наступного елемента поточному
                        *(pArr+j)=*(pArr+j+1);
//переприсвоїти значення тимчасової змінної наступному елементу
                        *(pArr+j+1)=vrem;
                    }
        printf ("\r\nSorted Arr = {");          //вивід константного рядка
        for (i=0; i<r; i++)                    //цикл перебору елементів масиву
        {
            if (i!=r-1)          //якщо поточний елемент у масиві неостанній
//вивід значення елемента масиву та символа ","
                printf ("%i,  ", *(pArr+i));
            else                 //інакше – поточний елемент у масиві останній
                printf ("%i", *(pArr+i)); //вивід значення елемента масиву
        }
        printf ("}\r\n");                      //вивід константного рядка
        delete [] pArr;      //звільнення виділеної для динамічного масиву пам'яті
}
```

Результат виконання програми при програмному заповненні динамічного масиву буде наступним:

```
Set number of elements in the array: 12

Check '1' for manual array initialization
or '2' for random program  array initialization

What is your check? 2

Arr = {5,  4,  1,  3,  11,  8,  9,  12,  6,  2,  7,  10}

Sorted Arr = {12,  11,  10,  9,  8,  7,  6,  5,  4,  3,  2,  1}
```

Питання для підготовки до захисту лабораторної роботи

1) Що таке змінна-покажчик?
2) Чому немає загального типу даних для змінних-покажчик?
3) У чому унікальність покажчика на порожній тип даних?
4) Чим для масиву виступає його ім'я?
5) Як можна перебирати елементи масиву, не користуючись індексами?
6) Чому адреса масиву не може бути інкрементована?
8) Для чого необхідно динамічне виділення пам'яті?
9) Як в С++ створити динамічний масив?
10) Що таке сортування?
11) Які прості методики сортування ви знаєте?
12) Які методики сортування називаються покращеними?

5.10. Лабораторна робота №10: Створення та застосування структур

Мета роботи

Ознайомитись зі створенням та застосуванням структур та отримати практичні навички роботи з їх полями.

Хід роботи

1) Ознайомитись з методичними вказівками до лабораторної роботи та темою "Складені типи даних";

2) сформувати структуру згідно з варіантом завдання;

3) розробити схему алгоритму програми, що демонструє роботу зі створеною структурою;

4) за схемою алгоритму написати програму на мові С++;

5) зробити висновки;

6) підготувати звіт про виконання лабораторної роботи, що включає наступні пункти: номер, тема, мета та хід лабораторної роботи, завдання, схема алгоритму програми, лістинг програми з коментуванням кожної інструкції, результат роботи програми (скріншот), висновки.

Теоретичні відомості та рекомендації до виконання

Як і масиви, структури дозволяють зберігати не одне значення, а декілька, проте на відміну від масивів у структурі для кожної змінної зазначено власний тип даних та унікальне ім'я. Таким чином, структури є групою змінних різного типа даних, що поєднані під одним ім'ям.

Як і масиви структури слугують для зберігання даних.

При виконанні завдання до лабораторної роботи студент повинен ознайомитися з особливостями створення структур та навчитись застосовувати їх в програмах на мові С++.

Виконання лабораторної роботи починається ознайомленням з теоретичними відомостями про оголошення та визначення структур і роботу з їх полями. Потім студент приступає до розробки схеми алгоритму програми згідно із завданням за варіантом. За схемою алгоритму виконується написання програми на мові С++.

У висновках до звіту студент повинен розкрити призначення теми "Створення та застосування структур". Висновок має бути поданий в формі пояснення сфери застосування вивченого матеріалу згідно із розумінням його студентом.

Завдання для самостійного виконання

Розробити схему алгоритму та написати програму, яка створює на основі структури за варіантом таблицю (табл. 5.12) та дозволяє заповнити її поля значеннями, що вводяться з консолі. За запитом користувача програма повинна виводити відповідний рядок або всю отриману таблицю.

Таблиця 5.12

Варіанти завдань

№ вар.	Ім'я структури (призначення)	Перше поле	Друге поле	Третє поле
1	Одяг	вид	розмір	вартість
2	Муз. композиція	назва пісні	рейтинг	виконавець
3	Телепередача	час показу	назва	рейтинг
4	Фільм	жанр	тривалість	бюджет картини
5	Студентська група	П.І.Б.	вік	середній бал
6	Книжка	автор	назва	кількість сторінок
7	Їжа	назва блюда	час приготування	калорійність
8	Напій	назва	калорійність	присмак
9	Обладнання	тип	виробник	енергоспоживання
10	Канцелярські знаряддя	вид	вартість	шифр товару
11	Військова техніка	клас	витрата палива	марка
12	Автомобілі	виробник	рік випуску	марка
13	Човни	тип	швидкість	місткість
14	Рослини	вид	середній час життя	число підвидів
15	Тварини	вид	ареал проживання	число особин
16	Комп'ютери	марка	швидкодія	рейтинг
17	Картини	художник	вартість	жанр
18	Будівлі	клас будівлі	число поверхів	район
19	Фірми	діяльність	назва	прибуток
20	Озера	країна	назва	площа
21	Планети	назва	маса	діаметр
22	Шкільні предмети	назва	час викладання	учитель
23	Мобільний телефон	фірма	модель	ціна
24	Мультиплікація	компанія	назва	тривалість
25	Країна	назва	населення	політ. режим
26	Ріки	назва	протяжність	число притоків
27	Літаки	виробник	місткість	дальність польоту
28	Оргтехніка	фірма	модель	вартість
29	Косметика	назва	ціна	номер товару
30	Робітники	професія	стаж роботи	сер. заробіток

Приклад виконання завдання

Розробити схему алгоритму та написати програму, яка створює на основі структури таблицю та дозволяє заповнити її поля значеннями, що вводяться з консолі.

№ вар.	Ім'я структури (призначення)	Перше поле	Друге поле	Третє поле
N	Співробітник	номер	ім'я	прізвище

Розробимо схему алгоритму роботи програми створення та заповнення таблиці на основі структури (рис. 5.12).

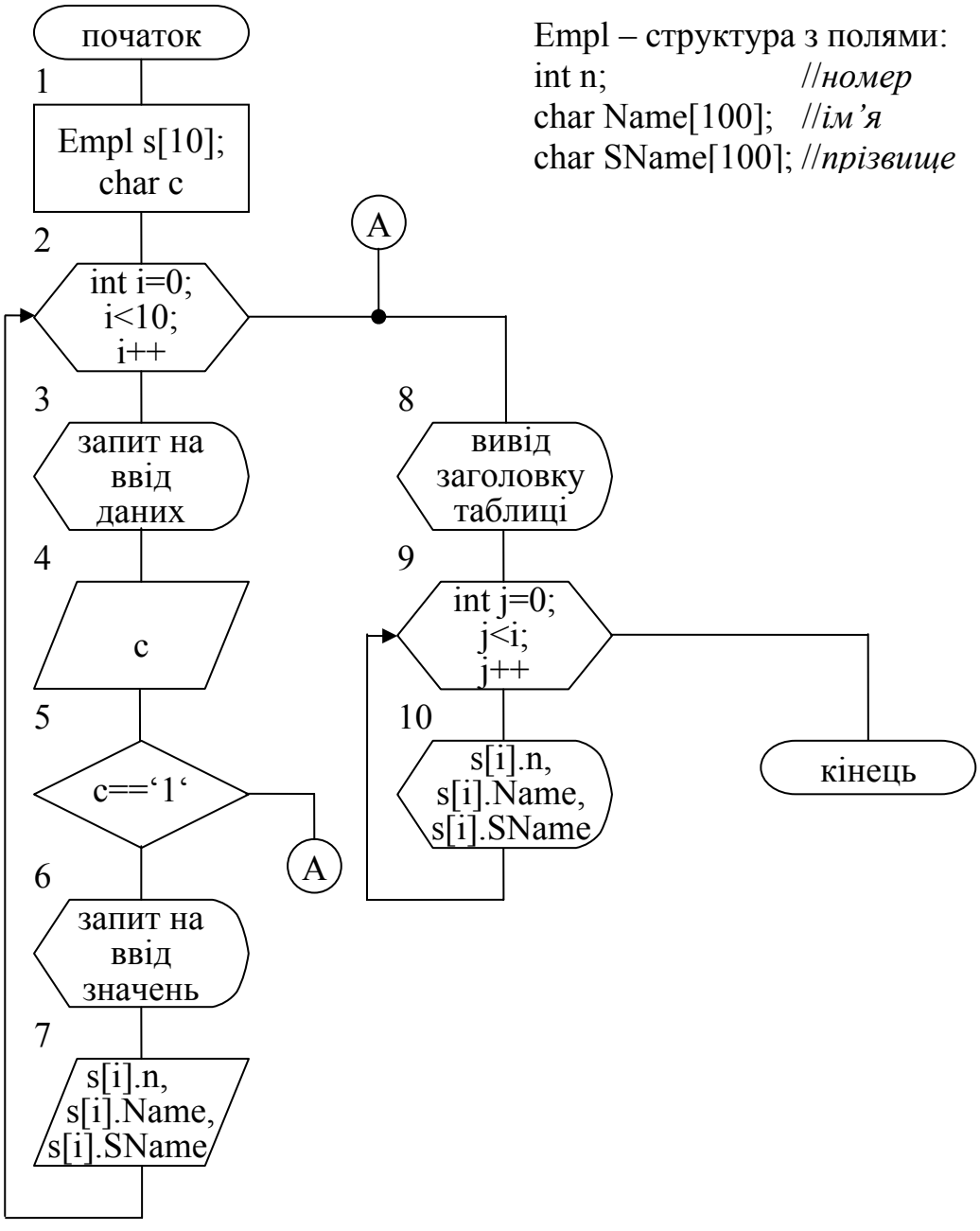


Рис. 5.12. Схема алгоритму програми роботи з таблицею

За схемою алгоритму напишемо програму на мові С++:

```
#include<stdio.h>                          //підключення бібліотеки вводу/виводу

struct Empl                 //оголошення та визначення структури співробітників
{
        int n;                                  //цілочисельне поле номера
        char Name[100];                         //поле символьний масив для імені
        char SName[100];                        //поле символьний масив для прізвища
};

void main()                             //оголошення та визначення головної функції
{
        Empl s[10];                     //оголошення масиву екземплярів структури
        char c;                     //оголошення змінної зупинки заповнення таблиці
        printf("You start table filling!\r\n");   //вивід запиту до користувача
        for(int i=0; i<10; i++)                   //цикл заповнення таблиці
        {
//вивід запиту про закінчення заповнення
                printf("Continue? 1 – No\r\n");
                scanf("%s", &c);                  //отримання значення від користувача
                if(c=='1')                        //якщо введено '1'
                        break;                          //вихід з оператора
//вивід запиту на ввід значення поля номера
                printf("Enter first field value (Number): ");
                scanf("%i", &s[i].n);             //отримання значення від користувача
//вивід запиту на ввід значення поля імені
                printf("Enter second field value (Name): ");
                scanf("%s", s[i].Name);           //отримання значення від користувача
//вивід запиту на ввід значення поля прізвища
                printf("Enter third field value (Second Name): ");
                scanf("%s", s[i].SName);          //отримання значення від користувача
        }
//вивід заповненої таблиці
        printf("Table:\r\n");                     //вивід константного рядка
        printf("N\t\tName\t\tSName\r\n");              //вивід константного рядка
        for (int j=0; j<i; j++)                   //цикл виводу полів таблиці
//вивід значень рядка таблиці
                printf("%i\t\t%s\t\t%s\r\n", s[j].n, s[j].Name, s[j].SName);
}
```

Результат виконання програмного коду буде наступним:

```
You start table filling!
Continue? 1 - No
0
Enter first field value (Number): 1
Enter second field value (Name): Sergey
Enter third field value (Second Name): Ivanov
Continue? 1 - No
0
Enter first field value (Number): 2
Enter second field value (Name): Dmitriy
Enter third field value (Second Name): Petrov
Continue? 1 - No
0
Enter first field value (Number): 3
Enter second field value (Name): Oleg
Enter third field value (Second Name): Sidorov
Continue? 1 - No
1
Table:
N               Name            SName
1               Sergey          Ivanov
2               Dmitriy         Petrov
3               Oleg            Sidorov
```

Питання для підготовки до захисту лабораторної роботи

1) Що називають складеними типами даних?
2) Дайте визначення поняттю структура?
3) Як оголошується структура?
4) Як називаються змінні, що входять до структури?
5) Як отримати доступ до змінних структури з основної програми?
6) Як сформувати у пам'яті комп'ютера на основі структури таблицю?

5.11. Лабораторна робота №11: Функції

Мета роботи

Ознайомитись з основами створення та застосування підпрограм у мові C++ та отримати практичні навички оголошення, визначення та виклику функцій.

Хід роботи

1) Ознайомитися з методичними вказівками до лабораторної роботи та темою "Функції";

2) згідно з варіантом завдання розробити схему алгоритму програми, що демонструє роботу двох функцій;

3) за схемою алгоритму написати програму на мові C++;

4) зробити висновки;

5) підготувати звіт про виконання лабораторної роботи, що включає наступні пункти: номер, тема, мета та хід лабораторної роботи, завдання, схема алгоритму програми, лістинг програми з коментуванням кожної інструкції, результат роботи програми (скріншот), висновки.

Теоретичні відомості та рекомендації до виконання

Використання функцій в мовах програмування надає можливість структуризації програми, її логічного спрощення і організації модульного підходу. Функція є програмним модулем, який приховує в собі частину програмного коду для спрощення загального розуміння програми і можливості множинного використання через простий виклик. Подібну роль у житті виконують всі технічні прилади – при їх використанні нам не доводиться замислюватися про те, з чого вони складаються і як між собою взаємодіють їх компоненти. Нас цікавлять тільки вхідні параметри, вихідні параметри та характеристики об'єкта в цілому. Те саме можна сказати і про функції – під час їх виклику користувач може передати у них ряд параметрів, а результатом виконання буде відповідна дія чи результуючі значення. Головне знати, яку саме задачу виконує функція, що використовується. Принцип модульності є одним з первинних при програмуванні на процедурних мовах, до яких належить мова C, що є основою для C++.

При виконанні завдання до лабораторної роботи студент повинен ознайомитися з особливостями оголошення, визначення та виклику функцій, а також навчитися використовувати отримані раніше знання та навички програмування для реалізації модульного підходу.

Виконання лабораторної роботи починається ознайомленням з теоретичними відомостями про можливості, що надаються мовою C++ при роботі з функціями. Потім студент приступає до розробки схеми алгоритму програми. За схемою алгоритму виконується написання програми мовою C++.

У висновках до звіту студент повинен розкрити призначення теми "Функції". Висновок має бути поданий в формі пояснення сфери застосування вивченого матеріалу згідно із розумінням його студентом.

Завдання для самостійного виконання

Розробити схему алгоритму та написати програму, що включає оголошення визначення та демонстрацію роботи через виклик двох функцій, опис яких наведено у табл. 5.13 відповідно до варіанта завдання.

Таблиця 5.13

Варіанти завдань

№ вар.	Опис функцій
1	а) Аргументи: змінна-покажчик на масив цілих значень і кількість елементів масиву. Значення, що повертається: максимальний елемент масиву. б) Аргументи: координати точки центру кола C(x, y), довжина радіуса кола (R) і кут (T) між радіусом кола, що паралельний до Ox, і відрізком CN (точка N належить колу). Значення, що повертається: координати точки кола N.
2	а) Аргументи: змінна-покажчик на масив цілих значень, кількість елементів масиву і ціле значення X. Значення, що повертається: число елементів масиву, які дорівнюють X. б) Аргументи: змінна-покажчик на масив цілих значень і кількість елементів масиву. Значення, що повертається: три максимальних елемента масиву: MAX1, MAX2, MAX3. (MAX1>=MAX2>=MAX3)
3	а) Аргументи: змінна-покажчик на масив символьних значень і кількість елементів масиву. Значення, що повертається: кількість латинських літер у масиві. б) Аргументи: координати кінців відрізка AB. Значення, що повертається: координати точки середини відрізка C.
4	а) Аргументи: змінна-покажчик на масив цілих значень і кількість елементів масиву. Значення, що повертається: мінімальний елемент масиву. б) Аргументи: K1, B1, K2, B2 – значення для рівнянь прямих Y1=K1X+B1 і B2=K2X+B2. Значення, що повертається: координати точки перетину прямих.
5	а) Аргументи: змінна-покажчик на масив цілих значень, кількість елементів масиву і ціле значення X. Значення, що повертається: позиція першого елемента, значення якого дорівнює X; якщо такого елемента немає – "-1". б) Аргументи: змінна-покажчик на масив цілих значень і кількість елементів масиву. Значення, що повертається: ціле значення, яке в масиві зустрічається найчастіше, і число елементів, що містять це значення.

Продовження табл. 5.13

№ вар.	Опис функцій
6	а) Аргументи: змінна-покажчик на масив цілих значень, кількість елементів масиву і ціле значення X. Значення, що повертається: кількість елементів масиву менших за X. б) Аргументи: координати двох протилежних вершин квадрата. Значення, що повертається: координати інших двох вершин.
7	а) Аргументи: цілі значення X та B. Значення, що повертається: X в степені B. (Функцію реалізувати без використання математичної бібліотеки) б) Аргументи: довжина діагоналі прямокутника (основа паралельна до Ox), гострий кут між діагоналями і координати центру. Значення, що повертається: координати протилежних вершин прямокутника.
8	а) Аргументи: змінна-покажчик на масив цілих значень і кількість елементів масиву. Значення, що повертається: середнє арифметичне значень елементів масиву. б) Аргументи: координати точок центрів кіл і їх радіуси (X1, У1, X2, У2, R1, R2). Значення, що повертається: координати кінців відрізка, утвореного Внаслідок перетину цих кіл і значення наявності точок перетину (дві точки перетину – "1", одна – "0", якщо точок немає – "-1").
9	а) Аргументи: дійсне число X. Значення, що повертається: X, округлене до найближчого меншого цілого. (Функцію реалізувати без використання математичної бібліотеки) б) Аргументи: координати двох вершин рівностороннього трикутника. Значення, що повертається: координати третьої вершини
10	а) Аргументи: значення X в градусах. Значення, що повертається: еквівалентне X значення У в радіанах. б) Аргументи: координати трьох вершин рівностороннього трикутника. Значення, що повертається: координати точки перетину бісектрис.
11	а) Аргументи: змінна-покажчик на масив цілих значень, кількість елементів масиву і ціле значення X. Значення, що повертається: кількість елементів масиву більших за X. б) Аргументи: координати вершин паралелограма. Значення, що повертається: координати точки перетину діагоналей.
12	а) Аргументи: довжини катетів A та B прямокутного трикутника. Значення, що повертається: довжина гіпотенузи C. б) Аргументи: координата протилежної до основи вершини рівнобедреного трикутника (основа паралельна до Ox), кут між сторонами, що дорівнюються, і їх довжина. Значення, що повертається: координати вершин при основі трикутника.

Продовження табл. 5.13

№ вар.	Опис функцій
13	а) Аргументи: дійсне число X. Значення, що повертається: X, округлене до найближчого більшого цілого. (Функцію реалізувати без використання математичної бібліотеки) б) Аргументи: змінна-покажчик на масив цілих значень і кількість елементів масиву. Значення, що повертається: три мінімальних елементи масиву: MIN1, MIN2, MIN3. (MIN1<=MIN2<= MIN3)
14	а) Аргументи: цілі значення A, B, C. Значення, що повертається: середнє з переданих значень (наприклад, B, якщо C>B>A) б) Аргументи: координати трьох вершин паралелограма. Значення, що повертається: координати четвертої вершини.
15	а) Аргументи: координати точок центрів кіл і їх радіуси (X1, У1, X2, У2, R1, R2). Значення, що повертається: Довжина відрізка, що утворено Внаслідок перетину цих кіл, і значення наявності точок перетину (дві точки перетину – "1", одна – "0", якщо точок немає – "-1") б) Аргументи: три цілих числа. Значення, що повертається: два числа з переданих, різниця між якими мінімальна. (Наприклад, якщо передані 10, 2 і 3, то мінімальна різниця між числами 2 і 3, отже, вони й будуть значеннями, що поверне функція).
16	а) Аргументи: ціле значення X. Значення, що повертається: значення У зворотне до кореню з X, округлене до 10^{-3} (Функцію реалізувати без використання математичної бібліотеки). б) Аргументи: змінна-покажчик на масив цілих значень і кількість елементів масиву. Значення, що повертається: кількість негативних і кількість позитивних елементів масиву.
17	а) Аргументи: ціле значення X. Значення, що повертається: значення У ($У=X^2/3$), округлене до 10^{-3} (Функцію реалізувати без використання математичної бібліотеки). б) Аргументи: координати точок відрізка, що є діагоналлю прямокутника. Значення, що повертається: довжини більшого і меншого радіусів еліпса, вписаного в цей прямокутник.
18	а) Аргументи: цілі значення X та У. Значення, що повертається: $C = \log_x y$. У округлити до 10^{-2}. (Функцію реалізувати без використання математичної бібліотеки). б) Аргументи: координати точки центра кола і довжина його радіусу. Значення, що повертається: координати вершин описаного квадрата, вважаючи, що його основа є паралельною до Ox.
19	а) Аргументи: кут прямокутного трикутника і довжина протилежного катета. Значення, що повертається: довжина прилеглого катета. б) Аргументи: координати точки центру кола і довжина його радіусу. Значення, що повертається: координати вершин описаного правильного трикутника з основою, що паралельна до Ox.

Закінчення табл. 5.13

№ вар.	Опис функцій
20	а) Функція зображує в консолі трикутник, використовуючи символ "X". Аргументи: висота псевдографічної фігури трикутника. Значення, що повертається: кількість видимих символів, що використані для малювання. б) Аргументи: координати точки центру кола і довжина його радіуса. Значення, що повертається: координати вершин ромба, вважаючи, що його діагональ є паралельною до Ox .
21	а) Функція здійснює сортування елементів масиву оптимізованим методом бульбашки з перевіркою числа обмінів. Аргументи: змінна-покажчик на масив цілочисельних значень та довжина масиву. Значення, що повертається: кількість проходів, використаних при сортуванні. б) Аргументи: значення довжин меншою та більшою сторін паралелограма і гострий кут між ними. Значення, що повертається: висота і ширина фігури паралелограм, вважаючи, що його основа є паралельною до Ox.
22	а) Аргументи: змінна-покажчик на масив символьних значень та кількість елементів масиву. Масив містить значення розрядів числа X у шістнадцятковій формі. Значення, що повертається: ціле значення У в десятковій формі, що є еквівалентним X. б) Аргументи: три цілих числа. Значення, що повертається: два числа з переданих, різниця між якими максимальна. (Наприклад, якщо передані 10, 2 і 3, то максимальна різниця між числами 2 і 10, отже, вони й будуть значеннями, що поверне функція).
23	а) Аргументи: кут прямокутного трикутника і довжина гіпотенузи. Значення, що повертається: довжина протилежного катета. б) Аргументи: координати центра кола і довжина його радіуса. Значення, що повертається: координати вершин п'ятикутника, вписаного в коло, вважаючи, що його основа є паралельною до Ox.
24	а) Аргументи: змінна-покажчик на масив символьних значень та кількість елементів масиву. Масив містить значення розрядів числа X у двійковій формі. Значення, що повертається: ціле значення У в десятковій формі, що є еквівалентним X. б) Аргументи: координати двох вершин при основі рівнобедреного трикутника і довжина сторін, що дорівнюються. Значення, що повертається: координата третьої вершини трикутника.
25	а) Аргументи: ціле значення X. Значення, що повертається: значення У, яке дорівнює кореню з X, округленому до 10^{-2}. (Функцію реалізувати без використання функцій математичної бібліотеки). б) Аргументи: змінна-покажчик на масив і кількість елементів масиву. Значення, що повертається: мінімальне, максимальне і найближче до середнього арифметичного значення масиву.

Приклад виконання завдання

Розробити схему алгоритму та написати програму, що включає оголошення, визначення та демонстрацію роботи через виклик двох функцій, опис яких наведено нижче:

а) аргументи функції: змінні x та y. Значення, що буде повернено: сума квадратів змінних x і y.

б) аргументи функції: покажчик на масив і число елементів масиву. Значення, що буде повернено: значення першого та останнього елементів масиву.

Згідно із завданням розробимо схему алгоритму (рис. 5.13).

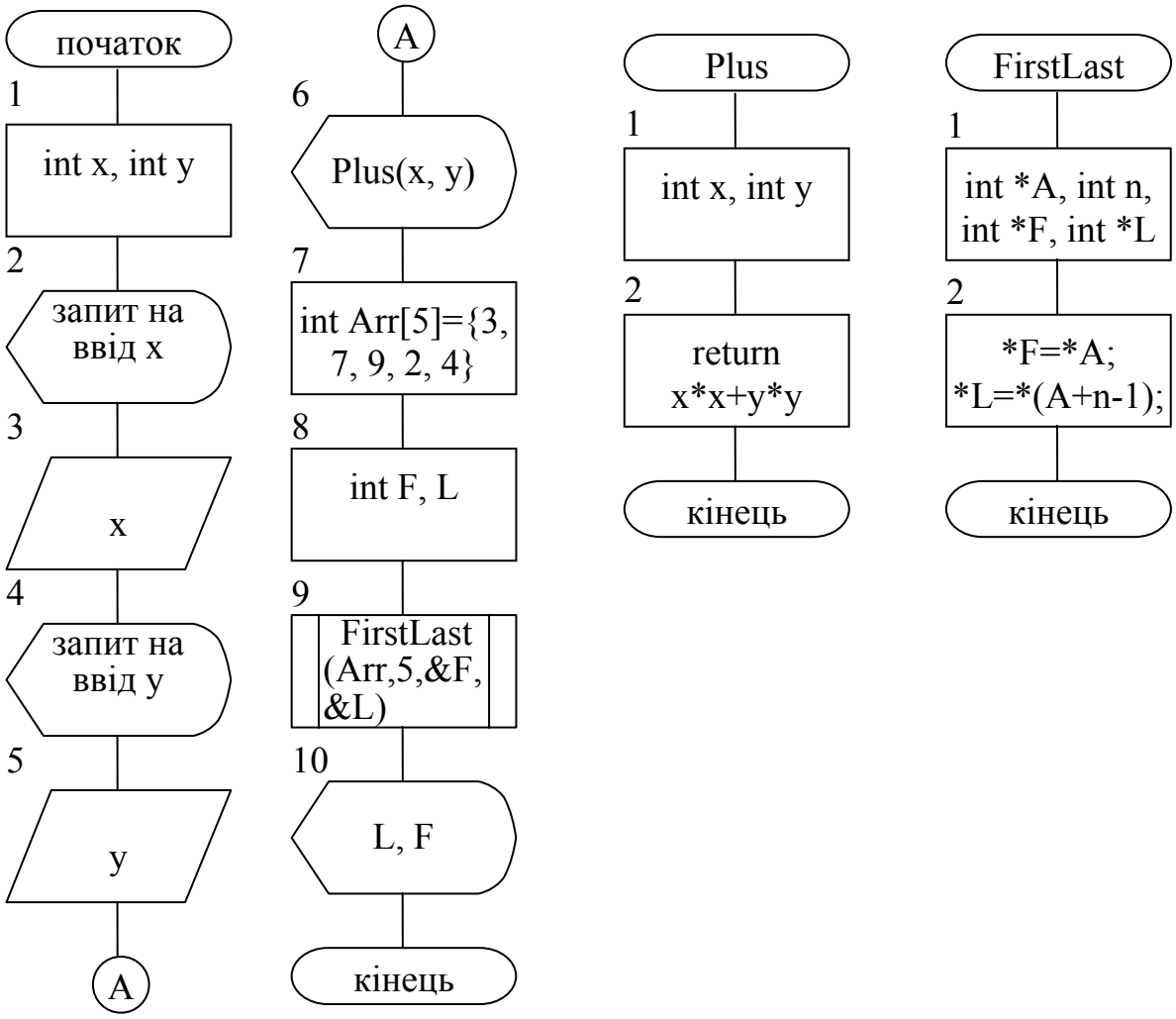


Рис. 5.13. Схема алгоритму програми, що демонструє роботу з функціями

За схемою алгоритму напишемо програму на мові C++:

```
#include<stdio.h>                    //підключення бібліотеки вводу/виводу

//оголошення  функції отримання суми квадратів x та y
int Plus(int x, int y);
```

```
//оголошення функції отримання значень першого і останнього елементів
//масиву
void FirstLast(int *A, int n, int *F, int *L);

void main()                          //оголошення та визначення головної функції
{
        int x, y;                    //оголошення цілочисельних змінних
        printf("Enter the x value:");//вивід запиту користувачу
        scanf("%i", &x);             //отримання значення від користувача
        printf("Enter the y value:");//вивід запиту користувачу
        scanf("%i", &y);             //отримання значення від користувача
//вивід результату виклику функції Pow()
        printf("X^2+Y^2=%i\r\n", Plus(x, y));
//оголошення  цілочисельного масиву  Arr з розмірністю 5 елементів
        int Arr[5]={3, 7, 9, 2, 4};
//оголошення змінних для отримання першого та останнього елементів масиву
        int F, L;
        FirstLast(Arr, 5, &F, &L);   //виклик функції FirstAndLast()
//вивід значень першого та останнього елементів масиву Arr на консоль
        printf("Arr[0]=%i; Arr[4]=%i\r\n", F, L);
}

//визначення функції отримання суми квадратів x і y
int Plus(int x, int y)
{
        return x*x+y*y;   //повернення результату розрахунку у місце виклику
}

//визначення функції отримання першого та останнього елементів масиву
void FirstLast(int *A, int n, int *F, int *L)
{
        *F=*A;                  //переприсвоювання значення через покажчик
        *L=*(A+n-1);            //переприсвоювання значення через покажчик
}
```

Результат виконання програми буде наступним:

```
Enter the x value:12
Enter the y value:31
X^2+Y^2=1105
Arr[0]=3; Arr[4]=4
```

Питання для підготовки до захисту лабораторної роботи

1) Що таке функція?
2) Що називають аргументами функції?
3) Наведіть приклад формату оголошення функції.
4) Чи можна визначення і оголошення функції робити без розділення?

5) Що таке виклик функції?

6) Яку функцію називають рекурсивною?

7) У чому полягає властивість перевантаження функції?

8) Що таке область видимості змінної?

9) Що таке клас пам'яті змінної?

10) Як на основі особливостей рекурсії можна створити цикл? У чому мінуси такого підходу?

5.12. Лабораторна робота №12: Інкапсуляція на основі класів

Мета роботи

Ознайомитись з об'єктно-орієнтованим підходом при розробці програм на мові C++ та отримати практичні навички у застосуванні інкапсуляції на основі класів.

Хід роботи

1) Ознайомитись з методичними вказівками до лабораторної роботи та темою "Об'єктно-орієнтований підхід. Класи та їх члени";

2) розробити схеми алгоритмів методів класу згідно із завданням за варіантом;

3) за схемами алгоритмів виконати оголошення та визначення класу і його членів;

4) розробити схему алгоритму програми, що демонструє роботу зі сформованим класом;

5) за схемою алгоритму написати програму на мові C++;

6) зробити висновки;

7) підготувати звіт про виконання лабораторної роботи, що включає наступні пункти: номер, тема, мета та хід лабораторної роботи, завдання, схема алгоритму програми, лістинг програми з коментуванням кожної інструкції, результат роботи програми (скріншот), висновки.

Теоретичні відомості та рекомендації до виконання

Основою об'єктно-орієнтованного програмування є поняття класу. Клас визначає природу об'єкта і є основним механізмом інкапсуляції. Клас – це логічна структура, яка поєднує у собі змінні, константи та функції, пов'язані між собою одним напрямком та ідеєю.

Основне призначення класу – спрощення написання програм, завдяки об'єктній орієнтації та інкапсуляції змінних і функцій у структурі класу.

При виконанні завдання до лабораторної роботи студент повинен засвоїти основні принципи побудови класів та набути вміння використовувати класи та їх члени у програмних проектах.

Виконання лабораторної роботи починається ознайомленням з теоретичними відомостями про основи побудови класів та методичними вказівками. Засвоївши їх, студент приступає до розробки класу згідно із завданням за варіантом. Необхідно створити основу свого класу та визначитися з кількістю методів, змінних та їх назвами. Назви методів мають відображати сутність дій, що вони виконують у програмі.

Після розробки класу та визначення його методів студент переходить до створення схем алгоритмів програми, що буде використовувати розроблений клас, та методів класу.

У висновках до звіту студент повинен розкрити призначення теми "Інкапсуляція на основі класів". Висновок має бути поданий в формі пояснення сфери застосування вивченого матеріалу згідно із розумінням його студентом.

Завдання для самостійного виконання

Розробити схему алгоритму та програму, що демонструє роботу з класом, оголошеним та визначеним згідно із завданням за варіантом. Клас застосовується для роботи з тривимірним масивом. Він повинен містити масив цілого типу даних Arr[x][y][z] та три методи (табл. 5.14):

- метод заповнення масиву за алгоритмом – конструктор класу;
- метод заповнення зрізу масиву значенням;
- метод виводу на консоль зрізу тривимірного масиву.

Тривимірний масив повинен бути побудований згідно з рис. 5.14.

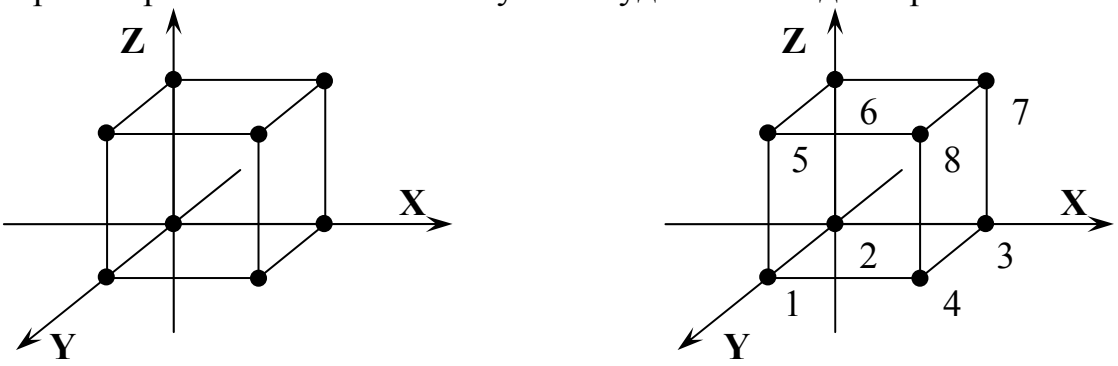


Рис. 5.14. Схематичне відображення тривимірного масиву

Таблиця 5.14

Варіанти завдань

№ вар.	Алгоритм заповнення для методу 1	Розмірність			Метод 2* (заповнення цифрою наступної фігури)	Метод 3 (відображення зрізу масиву) void
		X	Y	Z		
1	xyz	5	4	4	1278	Arr[x][2][z]
2	yxz	4	5	4	5634	Arr[2][y][z]
3	zxy	3	5	4	5832	Arr[x][y][2]
4	xzy	4	3	5	6741	Arr[3][y][z]
5	yzx	5	5	5	247	Arr[x][3][z]
6	zyx	4	4	5	245	Arr[x][y][3]
7	xyz	3	3	5	138	Arr[2][y][z]
8	yxz	5	3	4	136	Arr[x][y][3]
9	zxy	5	5	5	572	Arr[x][2][z]
10	xzy	4	5	3	574	Arr[1][y][z]
11	yzx	4	3	5	681	Arr[x][y][4]
12	zyx	5	4	4	683	Arr[4][y][z]
13	xyz	4	5	4	17	Arr[x][4][z]
14	yxz	3	5	4	64	Arr[x][2][z]
15	zxy	4	3	5	82	Arr[3][y][z]
16	xzy	5	5	5	53	Arr[x][y][3]
17	yzx	4	4	5	1278	Arr[x][2][z]
18	zyx	3	3	5	5634	Arr[2][y][z]
19	xyz	5	3	4	5832	Arr[4][y][z]

Закінчення табл. 5.14

№ вар.	Алгоритм заповнення для методу 1	Розмірність			Метод 2* (заповнення цифрою наступної фігури)	Метод 3 (відображення зрізу масиву) void
		X	Y	Z		
20	yxz	5	5	5	6741	Arr[x][3][z]
21	zxy	4	5	3	247	Arr[x][y][2]
22	xzy	4	3	5	245	Arr[3][y][z]
23	yzx	5	4	4	138	Arr[x][3][z]
24	zyx	4	5	4	136	Arr[x][2][z]
25	xyz	3	5	4	82	Arr[x][3][z]

* Метод 2 треба виконати, вважаючи, що масив має розміри 5x5x5, виключаючи ті зрізи (матриці), які відсутні за завданням. В функцію повинна передаватися цифра, за допомогою якої і буде виконано заповнення.

Вказівки до виконання завдання

Завдання до лабораторної роботи включає розробку методів згідно з варіантом. Перший метод виконує заповнення тривимірного масиву за алгоритмом, що наведено у вигляді переліку трьох площин.

Наприклад, для алгоритму **XZY** заповнення повинно виконуватися у наступній послідовності: спочатку заповнюється рядок по осі **X**, потім наступний рядок, що входить до матриці **XZ**. Коли заповнення матриці закінчується, виконується перехід до наступної матриці по вісі **Y**.

Наприклад, заповнена 0 матриця по осі **У** масиву 5x5x5, буде мати вигляд:

		x				
		0	*1*	*2*	*3*	*4*
	0	1	2	3	4	5
	1	6	7	8	9	10
z	*2*	11	12	13	14	15
	3	16	17	18	19	20
	4	21	22	23	24	25

Другий метод виконує заповнення фігури, що формується при поєднанні верхівок тривимірного масиву у послідовності за варіантом, обраною студентом цифрою. Фігура наведена за допомогою номерів верхівок тривимірного масиву згідно з рисунком у завданні. Елементи фігури входять до зрізу масиву, що створюється при формуванні площини, яка включає задану фігуру.

Наприклад, для фігури 136 масиву 3x3x3 (заповненого згідно з алгоритмом **XYZ**) необхідно записати нові значення в елементи, що зазначені на рис. 5.15.

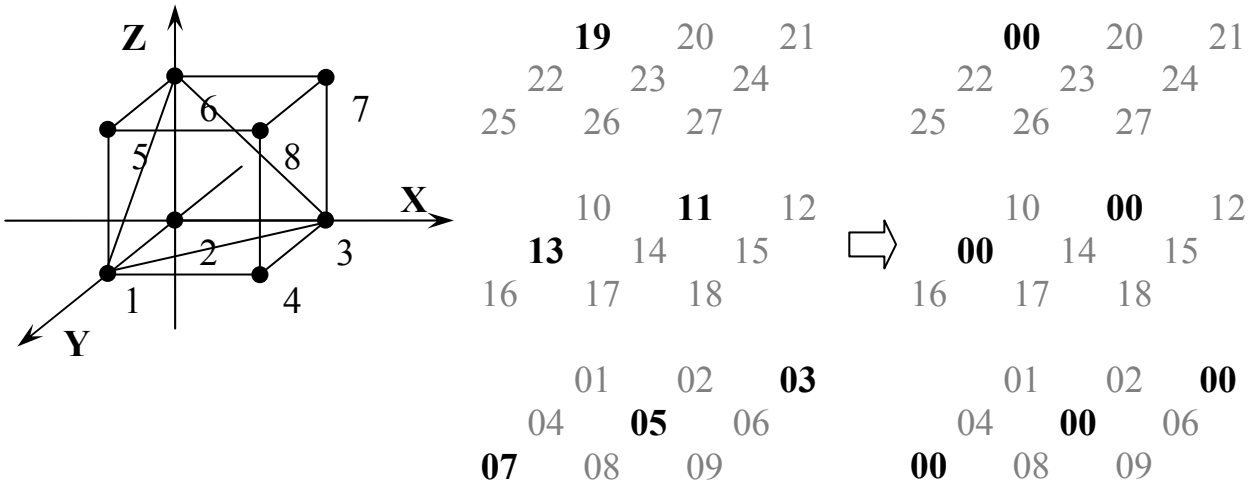


Рис. 5.15. Заповнення фігури, що задана номерами верхівок 136, цифрою

Елементи массиву, що позначено жирним, перевизначаються обраною цифрою. На рисунку цифра "0".

Третій метод виконує відображення вертикального чи горизонтального зрізу тривимірного масиву, номер зрізу вказано у таблиці згідно з варіантом.

Наприклад, для зрізу Arr[x][y][2] масиву 3x3x3 (заповненого згідно з алгоритмом **XYZ**) функція повинна виводити 3 матрицю **XY** по осі **Z**. Рахунок елементів масиву ведеться з 0, тому матриця під номером 2 і є третьою (рис. 5.16).

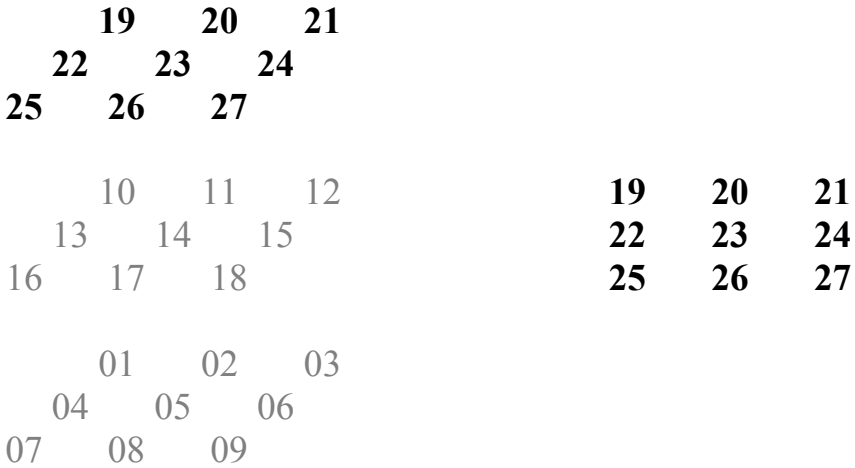


Рис. 5.16. Відображення третьої матриці по осі Z

Метод повинен вивести на консоль матрицю, що вказана у правій частині рисунка.

Для відображення тривимірного масиву можна скористатися наступною функцією:

```
//функція виводу елементів тривимірного масиву у вигляді кубу
void ShowCube(char x, char y, char z)
{
```

```
            char i, j, k;                   //оголошення змінних лічильників для циклів
            printf("\r\n\r\n\r\n");         //відступ за рядками
            for(k=z-1; k>=0; k--)           //цикл перебору елементів по осі z
            {
                  for(j=0; j<y; j++)  //цикл перебору елементів по осі y
                  {
                        for(int l=5-j; l>=1; l--)   //цикл зміщення рядка вздовж x
                              printf("   ");        //вивід 3-х символів відступу
                        for(i=0; i<x; i++)  //цикл перебору елементів по осі x
//вивід значення елемента та 9-ти символів відступу
                              printf("%0.3i       ", A[i][j][k]);
                        printf("\r\n");             //перехід на новий рядок
                  }
                  printf("\r\n\r\n\r\n");           //відступ за рядками
            }
}
```

Наведену функцію зручно додати у створений клас як четвертий метод.

Основна програма має відображати результати роботи всіх трьох методів, причому спочатку на консоль треба вивести заповнений масив за допомогою методу ShowCube().

Питання для підготовки до захисту лабораторної роботи

1) Дайте визначення поняттю об'єкт. Що таке клас?
2) Як оголошується клас?
3) Які є специфікатори доступу до членів класу?
4) Як можуть визначатися методи класу?
5) Як застосувати методи та поля класу з основної програми?
6) Що таке конструктор та деструктор класу?
7) Для чого використовуються класи?

5.13. Лабораторна робота №13: Спадкування властивостей класів

Мета роботи

Ознайомитись з властивостями процесу спадкування та отримати практичні навички у створенні похідних від базового класів.

Хід роботи

1) Ознайомитися з методичними вказівками до лабораторної роботи та темою "Спадкування";

2) розробити схеми алгоритмів методів похідного класу згідно із завданням за варіантом;

3) за схемами алгоритмів виконати оголошення та визначення похідного від створеного у лабораторній роботі №12 класу і його членів;

4) розробити схему алгоритму програми, що демонструє роботу зі сформованим класом;

5) за схемою алгоритму написати програму на мові C++;

6) зробити висновки;

7) підготувати звіт про виконання лабораторної роботи, що включає наступні пункти: номер, тема, мета та хід лабораторної роботи, завдання, схема алгоритму програми, лістинг програми з коментуванням кожної інструкції, результат роботи програми (скріншот), висновки.

Теоретичні відомості та рекомендації до виконання

Спадкування – це одне з базисних понять об'єктно-орієнтованого програмування. Воно засновується на використанні вже розроблених класів та для створення більш повних чи спеціалізованих програмних об'єктів.

Основне призначення спадкування – спрощення написання програм, завдяки застосуванню вже перевірених функцій класу без їх повторного оголошення та визначення через створення копії об'єкта у новому класі. Використання спадкування приводить до зменшення програмного коду.

При виконанні завдання до лабораторної роботи студент повинен засвоїти базові принципи побудови похідних класів та навчитися працювати з їх членами із основної програми.

Виконання лабораторної роботи починається ознайомленням з теоретичними відомостями про основи побудови похідних класів та методичними вказівками. Засвоївши їх, студент приступає до розробки класу згідно із завданням за варіантом. Базовим класом для створюваного обирається клас з лабораторної роботи №12.

Після розробки класу студент переходить до створення схеми алгоритму програми, що демонструє можливості як базового, так і похідного класів. За схемою алгоритму виконується написання програми на мові C++.

У висновках до звіту студент повинен розкрити призначення теми "Спадкування властивостей класів". Висновок має бути поданий в формі пояснення сфери застосування вивченого матеріалу згідно із розумінням його студентом.

Завдання для самостійного виконання

Розробити схему алгоритму та програму, що демонструє роботу з похідним від розробленого у лабораторній роботі №12 класом, оголошеним та визначеним згідно із завданням за варіантом. Клас повинен включати змінну, яка є прапорцем заповнення масиву, три методи, вказані у таблиці варіантів (табл. 5.15), метод поелементного заповнення тривимірного масиву та метод тасування елементів у двовимірному масиві.

Тривимірний масив повинен бути побудований згідно з рис. 5.17.

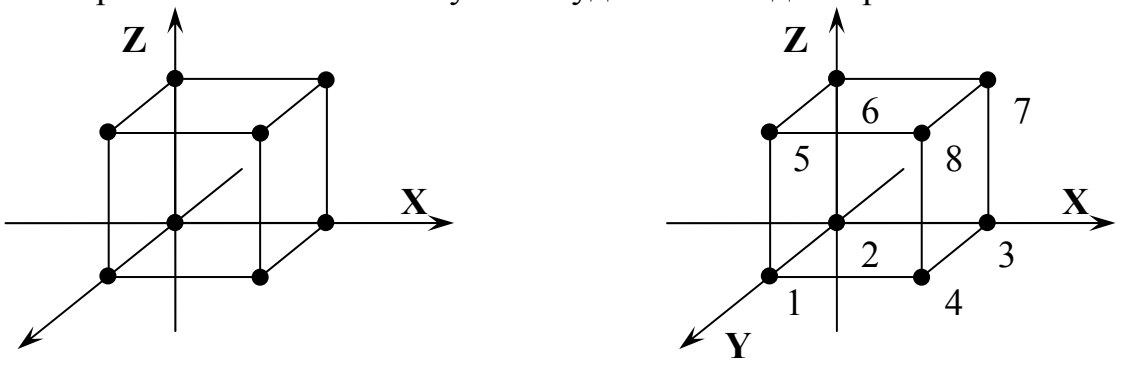


Рис. 5.17. Схематичне відображення тривимірного масиву

Таблиця 5.15

Варіанти завдань

№ вар.	Згідно з лаб. 12				Метод 1 (№)	Дод. мет. 1**	Метод 2 (№)	Дод. мет. 2**	Метод 3 (18*)
	Алгоритм заповнення масиву	x	y	z					
1	xyz	5	4	4	3*	xy	5*	xy	1278
2	yxz	4	5	4	4*	xz	6*	yz	2187
3	zxy	3	5	4	8*	z	7*	xz	7812
4	xzy	4	3	5	9*	x	10*	x	8721
5	yzx	5	5	5	13*	-	11*	y	5634
6	zyx	4	4	5	14*	-	12*	z	6543
7	xyz	3	3	5	9*	y	15*	-	3456
8	yxz	5	3	4	13*	-	16*	-	4365
9	zxy	5	5	5	14*	-	17*	-	6741
10	xzy	4	5	3	8*	z	7*	xz	7614
11	yzx	4	3	5	4*	yz	6*	yz	1476
12	zyx	5	4	4	3*	xz	5*	xy	4167
13	xyz	4	5	4	14*	-	16*	-	5832
14	yxz	3	5	4	13*	-	17*	-	8523
15	zxy	4	3	5	9*	z	5*	xz	2385
16	xzy	5	5	5	8*	y	6*	yz	3258

Закінчення табл. 5.15

№ вар.	Згідно з лаб. 12				Метод 1 (№)	Дод. мет. 1**	Метод 2 (№)	Дод. мет. 2**	Метод 3 (18*)
	Алгоритм заповнення масиву	x	y	z					
17	yzx	4	4	5	4*	xz	7*	xy	1278
18	zyx	3	3	5	3*	xy	10*	y	2187
19	xyz	5	3	4	9*	x	11*	z	7812
20	yxz	5	5	5	13*	-	7*	yz	8721
21	zxy	4	5	3	14*	-	6*	xy	5634
22	xzy	4	3	5	8*	z	5*	xz	6543
23	yzx	5	4	4	4*	yz	12*	x	3456
24	zyx	4	5	4	3*	xz	16*	-	4365
25	xyz	3	5	4	13*	-	15*	-	6741

* в цих полях вказані номери методів, що наведені нижче:
1 – метод тасування елементів масиву;
2 – метод поелементного заповнення масиву;
3/4 – метод пошуку найбільшого/найменшого значення вектора;
5/6 – метод розміщення елементів вектора в порядку зростання/спадання;
7 – метод пошуку середнього арифметичного значення елементів вектора;
8/9 – метод пошуку найбільшого/найменшого значення матриці;
10/11 – метод розміщення елементів матриці у порядку зростання/спадання;
12 – метод пошуку середнього арифметичного значення елементів матриці;
13/14 – метод пошуку найбільшого/найменшого значення тривимірного масиву;
15/16 – метод розміщення елементів тривимірного масиву у порядку зростання/зменшення;
17 – метод пошуку середнього арифметичного значення елементів тривимірного масиву;
18 – метод виводу елементів зрізу тривимірного масиву у вигляді матриці.

** у додаткових до методів полях вказано для векторів – координати матриці, у якій повинен знаходитися перший елемент вектора, для матриць – вісь, яка є перпендикулярною до неї.

Вказівки до виконання завдання

Завдання до лабораторної роботи включає розробку прапорця, трьох методів згідно з таблицею варіантів та базових методів, що мають бути присутні у всіх завданнях.

Прапорець заповнення массиву – це змінна типу **bool**. Вона має значення 1, якщо масив було заповнено після запуску програми та 0, – якщо масив ще не заповнено. Ця змінна повинна опитуватись у всіх методах похідного класу для прийняття рішення про виконання операцій з масивом (якщо змінна має

значення 0 – користувачу виводиться повідомлення про потрібність заповнення масиву. Програмний код функцій при цьому не виконується!).

Наприклад, якщо прапорець назвати flag, то для його перевірки у методах необхідно використати наступний код:

```
if(flag == 1)                          //якщо прапорець має значення "1"
{
      <дії згідно з сутністю метода>;
}
else                                   //якщо прапорець має значення "0"
{
      printf("Arrey did not fiilled!\r\n");     //вивід константного рядка
}
```

Базовими функціями, що будуть входити до похідного классу, є метод тасування елементів двовимірного масиву та метод поелементного заповнення масиву**:**

1. Метод тасування елементів масиву. Для того, щоб можна було перевірити роботу методів похідного классу, значення елементів тривимірного масиву повинні бути записані випадковим чином. Щоб заповнити тривимірний масив випадковими значеннями, використовуйте одномірний масив, який заповнюється завдяки функції з лабораторної роботи №9 (повинна бути включена до методів похідного класу):

```
//підключення  бібліотек  для реалізації функцій заповнення масиву
//невпорядкованими значеннями вектора
#include<algorithm>
#include<functional>
#include<vector>

using namespace std;                   //застосування простору імен std
//оголошення масиву з 125 елементів (відповідає тривимірному масивові 5x5x5)
int mas[125];

//оголошення та визначення функції, що виконує  заповнення масиву  змішаними
//значеннями
void RandomShuffle(int x, int n)
{
      const int VECTOR_SIZE = 125 ;    //число елементів вектора
//створення вектора
      typedef vector<int> IntVector ;
      typedef IntVector::iterator IntVectorIt ;
//оголошення та визначення змінної перебору індексів елементів масиву
      int g=0;
```

```
        IntVector Numbers(VECTOR_SIZE) ;
        IntVectorIt start, end, it ;
        for (int i=0; i<125; i++)                    //заповнення вектора значеннями
        {
                Numbers[i] = x;
                x=x+n;
        }
//визначення меж вектора
        start = Numbers.begin();
        end = Numbers.end();
//виклик функції, що перемішує значення вектора
        random_shuffle(start, end);
//заповнення масиву значеннями вектора
        for(it = start; it != end; it++)
        {
                mas[g]=*it;
                g++;
        }
}
```

Після виконання цієї функції масив mas[125] буде заповнено числами від "x" до "x+124·n", де x – це початкове значення, а n – крок зміни значення. Елементи цього масиву можна використати для послідовного заповнення тривимірного масиву, який було оголошено у базовому класі.

2. Метод поелементного заповнення масиву. Ця функція повинна мати наступний формат оголошення:

void SetArrValue(**char** x, **char** y, **char** z, **int** n);

де x – номер елемента за віссю X;
y – номер елемента за віссю Y;
z – номер елемента за віссю Z;
n – значення, яке буде присвоєно елементові масиву.

Завдяки цій функції є можливість змінювати значення елементів тривимірного масиву без використання імені масиву.

3/4. Метод пошуку найбільшого/найменшого значення вектора. Ці функції повинні мати наступний формат оголошення:

int VectorMaxValue (**char** x, **char** y);
int VectorMinValue (**char** x, **char** y);

В цих функціях x та y – координати першого елемента вектора (згідно із таблицею може бути xy, xz, yz). Перший елемент завжди знаходиться на граничній, нульовій площині тривимірного масиву. Так, для вектора, що має початок у x = 3; y = 2, початковим елементом за z буде нульовий. Таким чином, координати першого елемента вектора будуть наступними (3, 2, 0).

Значення, які повертають ці функції, відповідно найбільший та найменший елементи вектора.

5/6. Метод розміщення елементів вектора у порядку зростання/спадання. Функції повинні мати наступний формат оголошення:

void VectorMinToMax (**char** x, **char** y);
void VectorMaxToMin (**char** x, **char** y);

В цих функціях x та y – координати першого елемента вектора (див. метод 3/4).

Після виконання цього методу масив M[x], який треба буде додатково оголосити у похідному класі, повинен бути заповнений елементами з вказаного вектора згідно із завданням (у порядку зростання чи у порядку спадання).

7. Метод пошуку середнього арифметичного значення елементів вектора. Функція повинна мати наступний формат оголошення:

int VectorMidValue (**char** x, **char** y);

В цій функції x та y – координати першого елемента вектора (див. метод 3/4). Функція повертає середнє арифметичне значення елементів вектора.

8/9. Метод пошуку найбільшого/найменшого значення матриці. Функції повинні мати наступний формат оголошення:

int MatrixMaxValue (**char** x);
int MatrixMinValue (**char** x);

В цих функціях x – номер матриці двовимірного масиву. Наприклад, якщо сказано, що матриця має номер 3 по осі X (згідно із варіантом може бути x, y, z), то вона паралельна площині YZ, а її елементи відповідають масивові Arr[2][y][z].

Значення, які повертають ці функції, відповідно найбільший та найменший елементи матриці.

10/11. Метод розміщення елементів матриці у порядку зростання/спадання. Функції повинні мати наступний формат оголошення:

void MatrixMinToMax (**char** x);
void MatrixMaxToMin (**char** x);

В цих функціях x – номер матриці двовимірного масиву (див. метод 8/9).

Після виконання цього методу масив A[x][y], який треба буде додатково оголосити у похідному класі, повинен бути заповнений елементами з вказаного вектору згідно із завданням.

12. Метод пошуку середнього арифметичного значення елементів матриці. Функція повинна мати наступний формат оголошення:

int MatrixMidValue (**char** x);

В цій функції x – номер матриці двовимірного масиву (див. метод 8/9). Функція повертає середнє арифметичне значення елементів матриці.

13/14. Метод пошуку найбільшого/найменшого значення тривимірного массиву. Функції повинні мати наступний формат оголошення:

int CubeMaxValue ();
int CubeMinValue ();

Значення, які повертають ці функції, відповідно найбільший та найменший елементи масиву.

15/16. Метод розміщення елементів тривимірного масиву у порядку зростання/спадання. Функції повинні мати наступний формат оголошення:

void CubeMinToMax ();
void CubeMaxToMin ();

Після виконання цього методу масив B[x][y][z], який треба буде додатково оголосити у похідному класі, повинен бути заповнений значеннями елементів з тривимірного масиву Arr[x][y][z] згідно із завданням (новий масив заповнюється відповідно до алгоритму за завданням).

17. Метод пошуку середнього арифметичного значення елементів тривимірного масиву. Функція повинна мати наступний формат оголошення:

int CubeMidValue ();

Функція повертає середнє арифметичне значення елементів масиву.

18. Метод виводу елементів зрізу тривимірного масиву у вигляді матриці. Функція повинна мати наступний формат оголошення:

void ShowMatrix();

Функція вичитує значення елементів зрізу згідно з варіантом і розташовує їх у масиві C[x][y], який треба буде додатково оголосити у похідному класі. Зчитування значень елементів проводиться наступним чином: записана у таблиці варіантів комбінація з чотирьох цифр є верхівками чотирикутного зрізу тривимірного массиву, причому перша цифра показує розташування початкового елемента для зчитування, а перші дві цифри у сукупності – рядок, по якому має проводитися зчитування.

Наприклад, комбінація "1278" для тривимірного масиву 3x3x3, наведеного на рис. 5.18, означає, що першим елементом матриці повинно бути значення "7", а другим по лінії за рисунком – "4". Вся матриця буде мати вигляд:

07 04 01
17 14 11
27 24 21

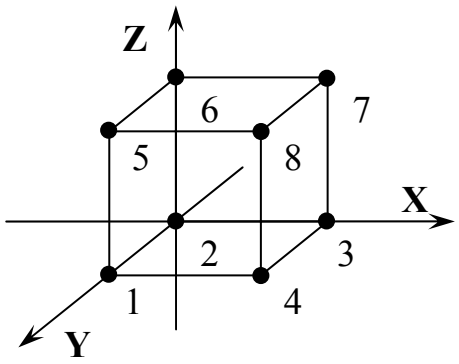


Рис. 5.18. Відображення матриці, відповідної до зрізу "1278"

Основна програма має відображати результати роботи всіх методів, причому спочатку на консоль повинен бути виведен заповнений випадковими значеннями масив за допомогою методу ShowCube() базового класу.

Питання для підготовки до захисту лабораторної роботи

1) Що таке спадкування?
2) У чому полягає призначення спадкування?
3) Які методи доступу до базового класу ви знаєте?
4) У чому полягає принцип створення похідних класів?
5) Яким чином виконується виклик методів базового класу через похідний?

5.14. Лабораторна робота №14: Клас роботи з рядками

Мета роботи

Мати уявлення про основи роботи з рядками та отримати практичні навички застосування у програмах методів класу string

Хід роботи

1) Ознайомитися з методичними вказівками до лабораторної роботи та темою "Клас роботи з рядками";

2) розробити схеми алгоритмів методів похідного класу згідно із завданням за варіантом;

3) за схемами алгоритмів виконати оголошення та визначення похідного від наведеного у методичних вказівках класу;

4) розробити схему алгоритму програми, що демонструє роботу зі сформованим класом;

5) за схемою алгоритму написати програму на мові C++;

6) зробити висновки.

7) підготувати звіт про виконання лабораторної роботи, що включає наступні пункти: номер, тема, мета та хід лабораторної роботи, завдання, схема алгоритму програми, лістинг програми з коментуванням кожної інструкції, результат роботи програми (скріншот), висновки.

Теоретичні відомості та рекомендації до виконання

Достатньо часто при програмуванні програмісти замість стандартних типів даних використовують текстовий ввід/вивід. Він більш універсальний і є одним з найкращих рішень, якщо не треба зважати на обмеженість пам'яті чи швидкість обробки інформації. Кожен символ тексту згідно із таблицею ASCII кодів є цілим значенням від 0 до 255 (1 байт). Таким чином, текстова інформація також може служити для передачі повідомлень, які базуються на інформаційній одиниці байта. Також за допомогою тексту можна вводити та виводити значення у любих системах числення, приймати, обробляти інформацію та рисувати (псевдографіка). Цифри також є символами тексту, що дозволяє обробляти і виводити значення, які значно перевищують діапазони стандартних типів даних. Текст є більш зрозумілим для людини ніж машинні коди та цифри, тому вміння працювати з текстовою інформацією є необхідним для програміста.

При виконанні завдання до лабораторної роботи студент повинен засвоїти основні принципи роботи з методами класу string, які забезпечують можливість пошуку, видалення, вставки та формування рядків, та навчитись застосовувати їх у своїх програмах.

Виконання лабораторної роботи починається ознайомленням з методичними вказівками та теоретичними відомостями про основні методи класу роботи з рядками. Засвоївши їх, студент приступає до розробки класу згідно із завданням за варіантом. Базовим для створюваного класу обирається класс, наведений у прикладі виконання завдання.

Після розробки класу студент переходить до створення схеми алгоритму програми, що буде демонструвати можливості як базового, так і похідного класів. За схемою алгоритму виконується написання програми на мові С++.

Залежність дій методів від кінця слова чи речення рядка повинна розглядатися як прив'язування програмних кодів пошуку та виявлення до особливостей тексту. Так, відомо, що кінець слова – це послідовність з літери і не літери; початок – не літери і літери чи перша літера рядка, якщо той починається з слова.

Базовий рядок вводиться студентом з клавіатури, або статично задається у програмі. Параметрами, що передаються до методів, є базовий рядок та значення змінних N, substr та str для завдань, де вони потрібні.

При роботі з рядками є зручним створювати новий рядок, у якому формується результат дій з базовим. Це дозволяє залишити базовий рядок без змін і спрощує процес створення результуючого рядка. Кожен метод повинен повертати перероблений рядок чи результат своєї роботи.

У висновках до звіту студент повинен розкрити призначення теми "Клас роботи з рядками". Висновок має бути поданий у формі пояснення сфери застосування вивченого матеріалу згідно із розумінням його студентом.

Завдання для самостійного виконання

Розробити схему алгоритму та програму, що демонструє роботу з похідним від наведеного у прикладі виконання завдання класу. Клас повинен включати три методи роботи з рядками згідно із варіантом завдання (табл. 5.16).

Таблиця 5.16

Варіанти завдань

№ вар.	**Розподілення методів за варіантами* (номери методів за табл. 5.17)**		
	1	**2**	**3**
1	1, 26, 51	25, 38, 56	13, 30, 70
2	2, 27, 52	24, 37, 57	12, 31, 69
3	3, 28, 53	23, 36, 58	11, 32, 68
4	4, 29, 54	22, 35, 59	10, 33, 67
5	5, 30, 55	21, 34, 60	9, 34, 66
6	6, 31, 56	20, 33, 61	8, 35, 65
7	7, 32, 57	19, 32, 62	7, 36, 64
8	8, 33, 58	18, 31, 63	6, 37, 63
9	9, 34, 59	17, 30, 64	5, 38, 62
10	10, 35, 60	16, 29, 65	4, 39, 61
11	11, 36, 61	15, 28, 66	3, 40, 60
12	12, 37, 62	14, 27, 67	2, 41, 59
13	13, 38, 63	13, 26, 68	1, 42, 58
14	14, 39, 64	12, 50, 69	25, 43, 57
15	15, 40, 65	11, 49, 70	24, 44, 56

Закінчення табл. 5.16

№ вар.	Розподілення методів за варіантами* (номери методів за табл. 5.17)		
	1	2	3
16	16, 41, 66	10, 48, 71	23, 45, 55
17	17, 42, 67	9, 47, 72	22, 46, 54
18	18, 43, 68	8, 46, 73	21, 47, 53
19	19, 44, 69	7, 45, 74	20, 48, 52
20	20, 45, 70	6, 44, 75	19, 49, 51
21	21, 46, 71	5, 43, 51	18, 59, 75
22	22, 47, 72	4, 42, 52	17, 29, 74
23	23, 48, 73	3, 41, 53	16, 28, 73
24	24, 49, 74	2, 40, 54	15, 27, 72
25	25, 50, 75	1, 39, 55	14, 26, 71

* стовпець, за яким виконується розподіл варіантів, обирається викладачем.

Таблиця 5.17

Опис методів класу

№ пп	Опис методу
1	Виводить ASCII коди для всіх символів рядка через відступ
2	Видаляє з тексту рядка всі літери малого регістру
3	Змінює регістр всіх символів рядка на протилежний
4	Видаляє всі символи відступу та переходу на новий рядок
5	Видаляє всі слова з кількістю символів, що є меншою за N
6	Перетворює регістр любої вказаної послідовності символів у рядку
7	Видаляє всі символи пунктуації у реченнях
8	Видаляє всі слова з кількістю символів, що є більшою за N
9	Видаляє з тексту рядка всі послідовності символів, взяті у дужки
10	Перетворює рядок таким чином, що всі речення закінчуються на велику літеру
11	Видаляє у рядку всі символи верхнього регістру
12	Видаляє з тексту рядка всі цифри
13	Перетворює рядок таким чином, що кожне речення закінчується словом великого регістру
14	Видаляє з тексту рядка всі слова, які відповідають str
15	Видаляє всі речення з кількістю символів, що є більшою за N
16	Замінює всі символи верхнього регістру рядка на символи нижнього
17	Видаляє всі речення з кількістю символів, що є меншою за N
18	Перетворює рядок таким чином, що всі слова починаються з великої літери
19	Видаляє всі речення з кількістю літер, що є меншою за N
20	Перетворює рядок таким чином, щоб регістри символів чергувалися
21	Замінює всі символи нижнього регістру рядка на символи верхнього
22	Видаляє всі речення з кількістю літер, що є більшою за N

Продовження табл. 5.17

№ пп	Опис методу
23	Перетворює рядок таким чином, що всі слова закінчуються на велику літеру
24	Видаляє всі слова, які починаються з символу чи послідовності символів substr
25	Видаляє всі речення з кількістю літер, що є меншою за N
26	Перетворює рядок таким чином, щоб його слова через одне мали різний регістр
27	Видаляє всі речення з кількістю літер, що є більшою за N
28	Перетворює рядок таким чином, щоб речення у ньому мали різний регістр через одне
29	Перетворює рядок таким чином, щоб всі слова, які мають парну кількість символів були записані наполовину символами верхнього регістру, наполовину – нижнього
30	Змінює всі речення таким чином, щоб послідовність слів у них йшла з кінця у початок
31	Шифрує та дешифрує рядок за обраним алгоритмом
32	Перетворює рядок таким чином, щоб всі речення, які мають парну кількість символів, були записані наполовину символами нижнього регістру, наполовину – верхнього
33	Змінює для всіх слів послідовність символів таким чином, щоб їх можна було читати з кінця у початок
34	Записує всі несхожі слова рядка у стовпець
35	Додає до всіх слів у дужках кількості їх символів
36	Додає до кожного слова у дужках суму ASCII кодів його символів
37	Записує всі несхожі слова, що починаються з символу чи послідовності символів substr у стовпець
38	Замінює регістр всіх слів з непарною кількістю літер
39	Шукає слово з найбільшою кількістю літер у рядку
40	Виділяє лапками всі речення з кількістю літер, що є меншою за N
41	Записує всі слова рядка у стовпець
42	Виводить у стовпець всі слова рядка, що закінчуються на велику літеру
43	Шукає найкоротше за кількістю літер речення у рядку
44	Перетворює рядок таким чином, що речення через одне будуть записані у зворотній послідовності слів
45	Шукає найдовше за кількістю літер речення рядка
46	Виділяє дужками всі речення з кількістю символів, що є меншою за N
47	Перетворює рядок таким чином, що слова через одне будуть записані у зворотній послідовності символів
48	Рахує і виводить кількість несхожих слів у рядку
49	Знаходить всі слова рядка, де кількість літер є непарною та змінює регістр середньої букви на верхній

Закінчення табл. 5.17

№ пп	Опис методу
50	Знаходить середнє арифметичне кількості слів у реченнях рядка і додає в дужках отримане число у кінець рядка
51	Виділяє дужками всі речення з кількістю символів, що є більшою за N
52	Виконує пошук слів у рядку, відповідних заданному, та виводить у стовпець позиції знайдених послідовностей
53	Шукає слово з найбільшою кількістю приголосних літер
54	Додає у дужках до кожного речення кількість його літер
55	Шукає слово з найменшою кількістю голосних літер
56	Замінює всі символи нижнього підкреслювання на відступи
57	Записує всі речення з непарною кількістю слів верхнім регістром
58	Знаходить суму ASCII кодів всіх літер кожного речення і додає її перед точкою у дужках
59	Виводить у стовпець всі слова рядка, що починаються з великої літери
60	Додає до всіх слів вказану послідовність символів
61	Виводить у стовпець всі речення, що закінчуються на велику літеру
62	Виділяє лапками всі речення з кількістю літер, що є більшою за N
63	Виводить у стовпець всі речення, що починаються з великої літери
64	Записує всі речення рядка у стовпець
65	Додає у дужках до кожного речення кількість його слів
66	Знаходить найдовше слово з непарним числом літер у рядку
67	Виділяє лапками всі слова рядка, які відповідають заданому
68	Замінює у всьому рядку послідовність символів на задану
69	Виводить кількість слів рядка з парним числом літер та кількість слів – з непарним
70	Змінює всі слова рядку з кількістю літер, що є меншою за N, таким чином, щоб всі букви були замінені на протилежні згідно з алфавітом (Наприклад, для латиниці: 'A' <–> 'Z'; 'B' <–> 'Y'; 'C' <–> 'X' і т.і.)
71	Змінює регістр всіх слів з кількістю літер, що є меншою за N
72	Змінює регістр всіх слів з кількістю літер, що є більшою за N
73	Записує всі речення з парною кількістю слів верхнім регістром
74	Знаходить середнє арифметичне кількості літер у словах рядка і додає отримане число на початок рядка у лапках
75	Виводить у стовпець всі несхожі символи рядка. Після кожного символу у дужках записує число, що відповідає кількості таких символів на рядок

Приклад виконання завдання

Розробити клас, що включає три методи роботи з рядками:

– метод зміни регістру символу;

– метод виділення у рядку символу чи сукупності символів;

– метод отримання суми ASCII кодів символів рядка.

Виконаємо оголошення та визначення заданого класу:

```
#include <string>                    //підключення бібліотеки "string.h"
using namespace std;                 //застосування простору імен std

class  CBaseString                                  //оголошення класу BaseString
{
      public:                                       //відкритий доступ до членів
//метод зміни регістру символа
            char ChangeCharReg(unsigned char c);
//метод виділення у рядку символа чи сукупності символів
            string SelectString(string s, string subs);
//метод отримання суми ASCII кодів рядка
            int GetStringWeight(string subs);
};

//Метод зміни регістру символа. c – символ, для якого буде змінено регістр.
//Значення, що буде повернуто – змінений символ.
char CBaseString::ChangeCharReg(unsigned char c)
{
      if (c>=65 && c<=90)            //якщо це велика літера латиниці
            return c+32;             //повернути ASCII код, збільшений на 32
      else if (c>=97 && c<=122)      //якщо це маленька літера латиниці
            return c-32;             //повернути ASCII код, зменшений на 32
      else if (c>=128 && c<=143)     //якщо це велика літера кирилиці від А до П
            return c+32;             //повернути ASCII код, збільшений на 32
      else if (c>=144 && c<=159)     //якщо це велика літера кирилиці від Р до Я
            return c+80;             //повернути ASCII код, збільшений на 80
      else if (c>=160 && c<=175) //якщо це маленька літера кирилиці від а до п
            return c-32;             //повернути ASCII код, зменшений на 32
      else if (c>=224 && c<=239) //якщо це маленька літера кирилиці від р до я
            return c-80;             //повернути ASCII код, зменшений на 80
      else if(c==240)                //якщо це велика літера кирилиці Ё
            return c+1;              //повернути ASCII код, збільшений на 1
      else if(c==241)                //якщо це маленька літера кирилиці ё
            return c-1;              //повернути ASCII код, зменшений на 1
      else                           //якщо це не літера латиниці або кирилиці
            return c;                //повернути незмінений символ
}
//Метод виділення у рядку символу чи сукупності символів дужками.  s – рядок,
//у якому необхідно знайти послідовності символів, subs – послідовність
//символів, яка має бути виділена дужками. Значення, що буде повернуто –
//рядок з виділеними за допомогою дужок послідовностями символів.
string CBaseString::SelectString(string s, string subs)
{
```

```
	string str;	//оголошення тимчасової змінної для отримання результатів
//оголошення змінної для отримання номера першого символа послідовності у
//рядку
	int n;
//циклічний пошук всіх заданих послідовностей у рядку
	while ((n=s.find(subs))>-1)	//доки метод find() повертає не -1
	{
//додати до str символи з початку s до послідовності, що шукається
		str+=s.substr(0,n);
		str+='('+subs+')';		//виділити послідовності дужками
//визначити s значенням рядка після знайденої послідовності
		s=s.substr(n+subs.length(),s.length()-n-subs.length()+1);
	}
//додати до str залишок символів рядка s, що не має у собі заданої
//послідовності
	str+=s;
	return str;			//повернути отриманий рядок
}
//Метод отримання суми ASCII кодів рядка. subs – сукупність символів, для
//якої необхідно знайти суму ASCII кодів. Значення, що буде повернуто – сума
//ASCII кодів.
int CBaseString::GetStringWeight(string subs)
{
	int sum=0;	//оголошення змінної для рахування суми ASCII кодів рядка
//цикл, у якому опрацьовується кожний символ
	for (int i=0; i<subs.length(); i++)
	{
		sum+=subs.at(i);	//додати до змінної значення ASCII коду символа
	}
	return sum;			//повернути отриману суму
}
```

Питання для підготовки до захисту лабораторної роботи

1) Для чого може використовуватися текстова інформація у програмуванні?

2) Який клас дозволяє програмісту працювати з рядками?

3) Як виконується підключення класу роботи з рядками до програми?

4) Які методи класу роботи з рядками виконують пошук підрядків?

5) Які методи класу роботи з рядками дозволяють працювати з підрядками?

6) Який метод дозволяє отримати змінну-покажчик на масив, який вміщує текст рядка?

7) Який метод дозволяє отримати любий символ рядка за його номером?

5.15. Лабораторна робота №15: Робота з файлами та багатофайлові проекти

Мета роботи

Ознайомитись з принципами роботи з файловими даними та отримати практичні навички застосування файлових функцій.

Хід роботи

1) Ознайомитись з методичними вказівками до лабораторної роботи та темами "Багатофайлові проекти" і "Робота з файлами";

2) розробити схеми алгоритмів методів класу згідно із завданням за варіантом;

3) за схемами алгоритму виконати оголошення та визначення класу роботи з текстовими файлами;

4) розробити схему алгоритму програми, що демонструє роботу зі сформованим класом;

5) за схемою алгоритму розробити програму на мові С++;

6) розділити програмний код на заголовний та програмний файли;

6) зробити висновки;

7) підготувати звіт про виконання лабораторної роботи, що включає наступні пункти: номер, тема, мета та хід лабораторної роботи, завдання, схема алгоритму програми, лістинг програми з коментуванням кожної інструкції, результат роботи програми (скріншот), висновки.

Теоретичні відомості та рекомендації до виконання

Файли виступають основною одиницею зберігання інформації на жорстких дисках. Майже вся робота з персональним комп'ютером зводиться до створення, копіювання, видалення, переміщення файлів та до запису/зчитування вкладеної у них інформації. Файли також дозволяють зберігати поточні дані виконання програм та кінцеві результати проектів для повторного їх зчитування та використання у майбутньому. Тому кожному програмісту необхідно мати навики застосування файлів у своїх проектах.

Достатньо часто деякі класи, функції та змінні відокремлюються у заголовкових файлах з розширенням ".h". Це виконується для забезпечення можливості повторного використання вже розроблених програмних фрагментів без їх повного копіювання у нові проекти через підключення заголовних файлів до основної програми у якості бібліотек. При цьому програмісту немає потреби осягати особливості написання тих чи інших функцій – він їх лише використовує. Завдяки цьому забезпечується можливість створення програмного продукту групою програмістів, кожен з котрих виконує лише свою частину згідно зі заздалегідь запланованими правилами формування функцій.

При виконанні завдання до лабораторної роботи студент повинен засвоїти основні принципи потокової роботи з файлами та навчитися створювати багатофайлові програмні проекти.

Виконання лабораторної роботи починається ознайомленням з методичними вказівками та теоретичними відомостями про можливості потокової роботи з файлами, забезпечені мовою С++. Засвоївши їх, студент приступає до розробки схеми алгоритму програми згідно із завданням за варіантом. За схемою алгоритму виконується оголошення та визначення класу та написання програми, що демонструє роботу з методами двома методами роботи з файлом.

Після перевірки роботи програми її код розділяється на два файли, щоб перетворити проект на багатофайловий.

У висновках до звіту студент повинен розкрити призначення теми "Робота з файлами та багатофайлові проекти". Висновок має бути поданий в формі пояснення сфери застосування вивченого матеріалу згідно із розумінням його студентом.

Завдання для самостійного виконання

Розробити схему алгоритму та програму, що демонструє використання класу, який містить два методи роботи з текстовою інформацією: форматування тексту та запис результату у файл "OUTPUT1.TXT" (табл. 5.18); створення з літер тексту псевдографічного зображення та запис результату у файл "OUTPUT2.TXT" (табл. 5.19). Текст для передавання через аргументи у методи класу повинен бути взятий з файлу "INPUT.TXT" (шлях до файлу вводиться з консолі). Програмний код поділити на два файли – заголовний та програмний.

Таблиця 5.18

Варіанти завдань до створення першого методу класу

№ вар.	Опис методу
1	Виконати коригування тексту, видаляючи символи переходу на новий рядок, які не завершують абзац. Вважати, що абзац починається з червоного рядка – символа табуляції.
2	Виконати коригування тексту, видаляючи повторювані й зайві знаки пробілів. Зайвими вважати відступи, поставлені перед абзацом.
3	Виконати коригування тексту, вставляючи символи пробілу для правильного розділення слів та знаків пунктуації у реченнях. Якщо пробіл вже є, то коригувати не потрібно.
4	Виконати коригування тексту, видаляючи з нього всі знаки переносів, враховуючи, що вони зустрічаються тільки в кінці рядка.
5	Виконати коригування тексту, додаючи для назв глав виділення пропусками перед і після одного рядка. Якщо пропуск рядка вже є, то коригувати даний фрагмент не потрібно. Вважати, що назва розділу починається зі слова "глава" і відповідної цифри.
6	Виконати коригування тексту таким чином, щоб у кожному рядку було до N символів і при цьому були відсутні розриви слів.

Продовження табл. 5.18

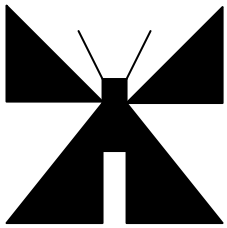
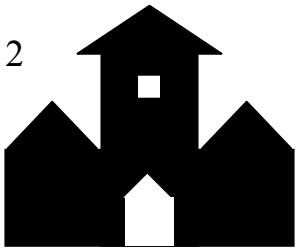
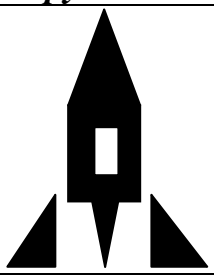
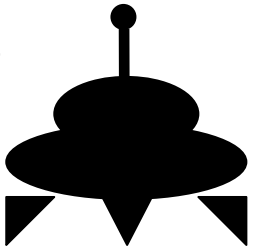
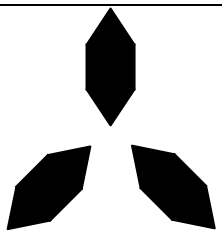
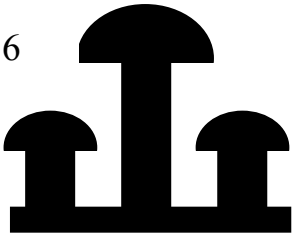
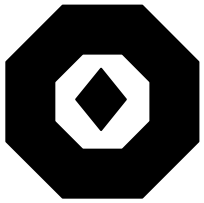
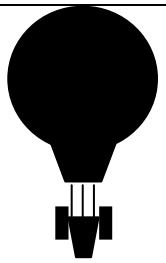
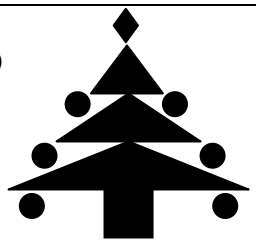
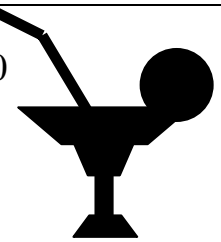
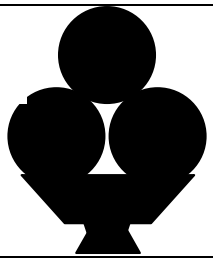
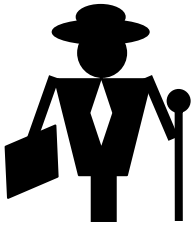
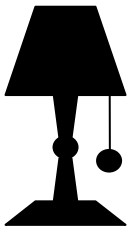
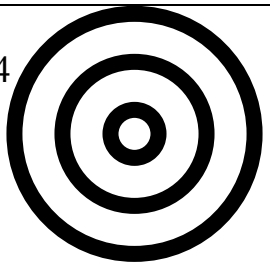
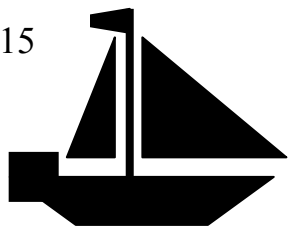
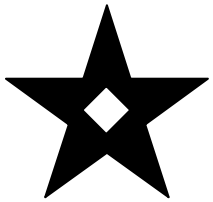
№ вар.	Опис методу
7	Виконати коригування тексту таким чином, щоб всі назви глав були написані прописними буквами. Вважати, що назви глав в тексті виділено пропусками перед і після одного рядка.
8	Виконати коригування тексту, видаляючи з нього всі зайві відступи рядків. Відступи, що використовуються для виділення назв глав, зайвими не вважаються. Назви глав починаються зі слова "глава" і відповідної цифри.
9	Виконати коригування тексту таким чином, щоб усі абзаци починалися з червоного рядка – символу табуляції.
10	Виконати коригування тексту, видаляючи з нього всі символи, які не відносяться до текстової інформації.
11	Виконати коригування тексту таким чином, щоб всі слова могли починатися з великої або малої літери, а всі інші літери були тільки малими.
12	Виконати коригування тексту, видаляючи всі тире, що поставлені перед початком рядка, крім тих, що починають абзац. Вважати, що абзаци починаються з червоного рядка – символу табуляції.
13	Виконати коригування тексту таким чином, щоб усі цифри в тексті були виділені пробілами. Для цифр, які починають речення, коригування не потрібне.
14	Виконати коригування тексту, видаляючи з нього всі зайві точки. Точки і трикрапки в кінці речень видаляти не потрібно.
15	Виконати коригування тексту, видаляючи всі символи табуляцій, крім тих, що починають абзаци і є червоним рядком.
16	Виконати для тексту формування в його кінці змісту без зазначення номерів сторінок (перелік глав). Вважати, що назви глав починаються зі слова "глава" і відповідної цифри.
17	Виконати для тексту нормалізацію всіх виносок таким чином, щоб вони містили число символів '*', яке постійно збільшується. Коригування повинно проходити окремо для виносок і їх розшифровок.
18	Виконати коригування тексту, додаючи в нього номери сторінок зліва знизу. Вважати, що один лист може містити 65 рядків, включаючи номер.
19	Виконати коригування тексту, додаючи для кожного рядка поле по 5 символів, так щоб в рядок містилося N символів і були відсутні розриви слів.
20	Виконати коригування тексту, видаляючи з нього всі назви глав. Вважати, що назви глав починаються зі слова "глава" і відповідної цифри.
21	Виконати коригування тексту, виділяючи всі слова, що містять нелітеральні символи за допомогою квадратних дужок.

Закінчення табл. 5.18

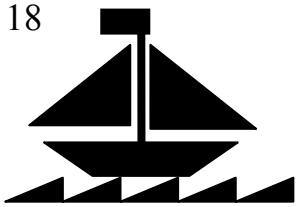
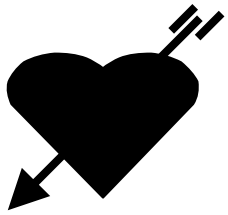
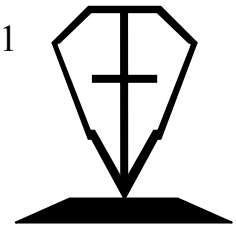
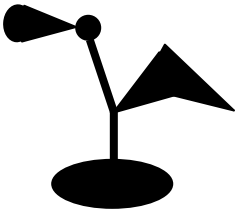
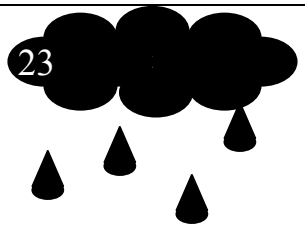
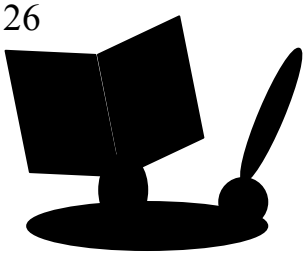
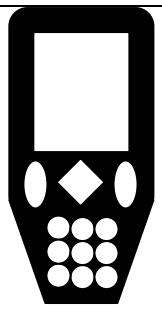
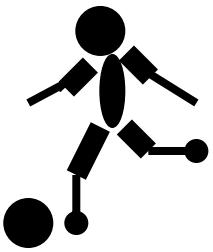
№ вар.	Опис методу
22	Виконати для тексту формування на початку, після назви, списку номерів глав із зазначенням номерів сторінок, вважаючи, що лист може містити 70 рядків.
23	Виконати для тексту на початку кожного листа вставку колонтитулів, що містять назву тексту. Вважати, що один лист містить 68 рядків.
24	Виконати коригування тексту таким чином, щоб кожна глава починалася з нового аркуша, вважаючи, що глави починаються зі слова "глава" і відповідної цифри. Лист може містити до 60 рядків.
25	Виконати коригування тексту, видаляючи з нього всі виноски та їх розшифровки. Вважати, що виноска – це сукупність декількох символів '*', а розшифровка – речення, що починається з декількох символів '*', написане після тире.

Таблиця 5.19

Варіанти завдань до створення другого методу класу

1 	2 	3 	4 
5 	6 	7 	8 
9 	10 	11 	12 
13 	14 	15 	16 

Закінчення табл. 5.19

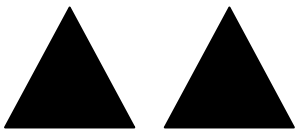
17 	18 	19 	20 
21 	22 	23 	24 
25 	26 	27 	28 

Приклад виконання завдання

Розробити програму, що демонструє використання класу, який містить два методи роботи з текстовою інформацією:

– видалення з тексту символів ',' та запис результату до файлу "OUTPUT1.TXT";

– створення з літер тексту псевдографічного зображення



та запис результату до файлу "OUTPUT2.TXT".

Текст для передавання через аргументи у методи класу треба взяти з файлу "INPUT.TXT". Програмний код поділити на два файли – заголовний та програмний.

Розробимо програму на мові C++.

Заголовний файл "main.h":

```
#include <stdio.h>                  //підключення бібліотеки вводу/виводу
#include <string>                   //підключення бібліотеки роботи з рядком

using namespace std;                //застосування простору імен
```

```
class CFWork                        //оголошення класу роботи з файлами
{
    public:                         //відкритий доступ до членів
//оголошення методу формування псевдографічного зображення
        void PGraph(string s);
        void Format(string s); //оголошення методу видалення ком з тексту
};
```

Файл програми "main.cpp":

```
#include "main.h"       //підключення заголовного файлу з оголошенням класу

//визначення методу формування псевдографічного зображення
void CFWork::PGraph(string s)
{
    int a=0;            //оголошення та визначення цілої змінної лічильника
    string st;                  //оголошення об'єкта рядка
    for (int i=1; i<=4; i++)    //цикл за рівнем малюнка (4 – кількість рядків)
    {
            for(int j=1; j<=((7-(2*i-1))/2); j++)   //цикл додавання відступів
                st+=' ';                            //додавання символа " "
            for(j=1; j<=(2*i-1); j++)           //цикл додавання символу рядка
                st+=s.at(a++);                  //додавання символу рядка
            for(j=1; j<=((7-(2*i-1))+2); j++)       //цикл додавання відступів
                st+=' ';                            //додавання символа " "
            for(j=1; j<=(2*i-1); j++)           //цикл додавання символу рядка
                st+=s.at(a++);                  //додавання символу рядка
            for(j=1; j<=((7-(2*i-1))/2); j++)       //цикл додавання відступів
                st+=' ';                            //додавання символа " "
            st+="\r\n";         //додавання переходу на новий рядок
    }
//оголошення змінної-покажчика на структуру роботи з файлом
    FILE *f;
    f=fopen("output2.txt", "w");                //створення файлу для запису
    fprintf(f, "%s", st.c_str());               //запис рядка до файлу
    fclose(f);                                  //закриття файлу
}

//визначення методу видалення ком з тексту
void CFWork::Format(string s)
{
    string ns;                  //оголошення об'єкта рядка
    for(int i=0; i<s.length(); i++)         //цикл перебору символів рядка
        if(s.at(i)!=',')                    //якщо символ не є комою
            ns+=s.at(i);                    //додати символ до нового рядка
```

```
//оголошення змінної-покажчика на структуру роботи з файлом
        FILE *f;
        f=fopen("output1.txt", "w");          //створення файлу для запису
        fprintf(f, "%s", ns.c_str());         //запис рядка до файлу
        fclose(f);                            //закриття файлу
}

void main()                          //оголошення та визначення головної функції
{
//оголошення змінної-покажчика на структуру роботи з файлом
        FILE *f;
        string st;                   //оголошення об'єкта класу роботи з рядками
        int n=0;//оголошення та визначення змінної лічильника номера символа

        f=fopen("input.txt", "r");            //відкриття файлу для читання
        while (!feof(f))                      //цикл до визначення кінця рядка
        {
                st+=fgetc(f);                 //формуємо рядок
        }
        fclose(f);                            //закриття файлу
        CFWork fw;                            //оголошення об'єкта класу
//виклик методу формування псевдографічного зображення
        fw.PGraph(st);
        fw.Format(st);               //виклик методу видалення ком з тексту
}
```

Результатом роботи програми буде створення двох вихідних файлів.

Питання для підготовки до захисту лабораторної роботи

1) За допомогою якої методики мови С++ є можливість потокового вводу/виводу інформації файлу?

2) Для чого може використовуватись відокремлення оголошень класів, функцій та змінних у заголовний файл?

3) Як можна виконати виклик функції, що оголошена в іншому файлі?

4) За допомогою яких функцій виконується потокове зчитування та запис тексту у файл?

5) Яким чином виконується переміщення курсору файлу?

ЧАСТИНА V
ТЕСТОВІ ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ ПЕРЕВІРКИ ЗАСВОЄННЯ МАТЕРІАЛУ

Матеріал даної частини призначено для перевірки засвоєного під час роботи з навчальним посібником. Тестові завдання наведені у відповідності до викладених у теоретичних частинах тем. Вони включають як відкриту (обрати правильну відповідь з наведених), так і закриту (надати відповідь на запитання, виконати визначення поняття, знайти помилку у програмі) форми тестування. Тестові завдання є еквівалентними до тих, що використовуються на модульних контрольних та екзаменаційній роботах. Відповіді на питання тестів оцінюються у балах, 1 – для відкритої і 2 – для закритої форми. Результуюча оцінка виставляється за наступним принципом: відмінно – 90%, добре – 70% та задовільно – 50% від максимальної кількості балів.

22. ТЕСТОВІ ЗАВДАННЯ

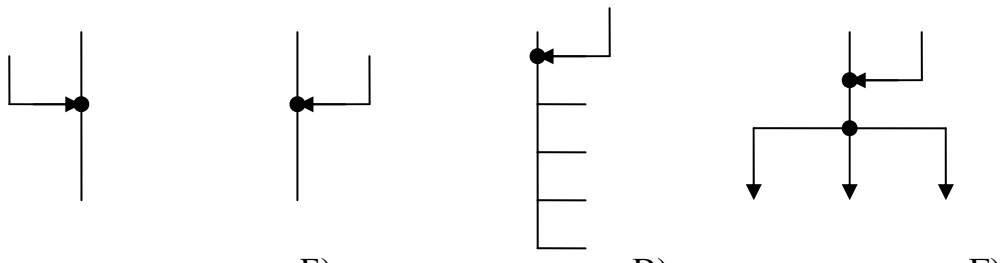
1. Мова С є…

А) процедурною мовою програмування
Б) агентно-орієнтованою мовою програмування
В) об'єктно-орієнтованою мовою програмування
Г) функціональною мовою програмування
Д) правильної відповіді немає

2. Який спеціальний символ схем алгоритмів використовується для відображення складних циклів з передумовою?

А) блок рішення
Б) блок дисплей
В) блок дані
Г) блок процес
Д) правильної відповіді немає

3. Який з варіантів застосування ліній поєднання символів схем алгоритмів є правильним?



А) Б) В) Г)
Д) правильний фрагмент схеми алгоритму відсутній

4. Який з наведених типів даних називають цілочисельним?

А) double
Б) bool
В) int
Г) float
Д) правильної відповіді немає

5. Як на тип даних впливає використання службового слова signed?

А) збільшує діапазон значень типу даних вдвічі
Б) зміщує значення у позитивний діапазон
В) зменшує діапазон значень типу даних вдвічі
Г) зміщує значення до негативного діапазону
Д) правильної відповіді немає

6. Яким є результат обчислення будь-якої логічної операції?

А) будь-яке ціле значення
Б) будь-яке дійсне значення
В) нуль або одиниця
Г) будь-яке символьне значення
Д) правильної відповіді немає

7. Який символ використовується для відображення в схемах алгоритмів оператора вибору switch?

А) блок підготовки
Б) блок процесу
В) блок даних
Г) блок рішення
Д) правильної відповіді немає

8. У чому полягає відмінність між операторами циклів з передумовою for і while?

А) відмінностей немає
Б) оператор while більш надійний, тому що працює з будь-якими умовними виразами, а for – ні
В) на основі оператора for можна організовувати вічний цикл
Г) оператор for при оголошенні відразу дозволяє визначити початкове значення, крок і кінцеву умову для змінної
Д) правильної відповіді немає

9. Який з варіантів програмних кодів є правильним для знаходження суми чисел від 1 до N?

А)
```
void main()
{
  int i,Sum=0,N;
  scanf("%i",&N);
  while (i<=N)
  {
    Sum+= i++;
  }
}
```

Б)
```
void main()
{
  int i,Sum=1,N;
  scanf("%i",&N);
  while (i<=N)
  {
    Sum+= i++;
  }
}
```

В)
```
void main()
{
  int i,Sum=0,N;
  scanf("%i",&N);
  while (i<=N)
  {
    Sum+= ++i;
  }
}
```

Г)
```
void main()
{
  int i,Sum=1,N;
  scanf("%i",&N);
  while (i<N)
  {
    Sum+=++ i;
  }
}
```

Д) правильної відповіді немає

10. Багатовимірний масив – це ...

А) масив, в якому безліч значень різних типів даних
Б) масив, який включає прості масиви та матриці
В) масив з безліччю розмірностей
Г) масив, побудований у просторі
Д) правильної відповіді немає

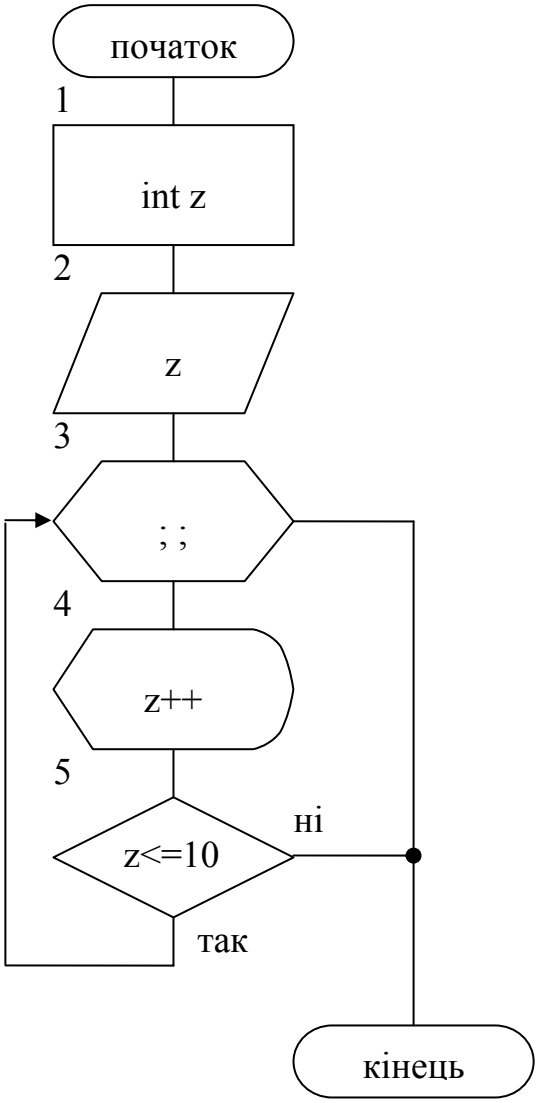
11. Яким буде результат виконання наступного програмного коду?

```
int k=-1;
for (k=0; k<=5; k+=1)
{
        k=k+++1;
}
printf ("%i",k);
```

12. Знайдіть помилку в програмному коді. Програма повинна виконувати підрахунок суми значень від 11 до 100 включно.

```
void main()
{
        int i=10, Rez=0;
        while(i++<100);
        {
                Rez+=i;
        }
        printf("%d", Rez);
}
```

13. За заданою схемою алгоритму напишіть лістинг програми.



14. Знайдіть результат виразу та подайте його в десятковій формі

(13<<3)%3+(12&120)

15. Програма створює в консолі псевдографічне зображення рівнобедреного трикутника.

```
     x
    xxx
   xxxxx
  xxxxxxx
 xxxxxxxxx
xxxxxxxxxxx
```

```
#include <stdio.h>
void main()
{
        printf("\r\n");
```

```
        int n=7;
        for (int i=1; i<=n; i++)
        {
                for (int j=1; j<=((2*n-1)-2*i+1)/2; j++)
                        printf(" ");
                for (j=1; j<=(2*i-1); j++)
                        printf("X");
                printf("\r\n");
        }
}
```

Яким повинен бути вираз і як його треба змінити для того, щоб зображення стало наступним?

16. Який з наведених нижче двовимірних масивів заповнено у відповідності з умовою "0: i+j > 4"?

А)

0	0	0	0	0
1	0	0	0	0
1	1	0	0	0
1	1	1	0	0
1	1	1	1	0

Б)

1	1	1	1	1
1	1	1	1	0
1	1	1	0	0
1	1	0	0	0
1	0	0	0	0

В)

1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
0	0	0	0	1

Г)

0	0	0	0	1
0	0	0	1	1
0	0	1	1	1
0	1	1	1	1
1	1	1	1	1

Д) правильної відповіді немає

17. При додаванні до змінної-покажчика одиниці, значення адреси, що зберігається в ній ...

А) збільшується залежно від типу даних, визначеного при оголошенні
Б) збільшується на одиницю
В) не збільшується, адреса є константою
Г) збільшується на два
Д) правильної відповіді немає

18. Який з наведених нижче програмних фрагментів правильний для створення, заповнення, виведення на консоль і видалення динамічного масиву типу даних int?

А)
```
int *p = new int [10];
p = p + 9;
for (int i=0;i<10;i++)
  *(p--)=10 – i;
p++;
for (i=0;i<10;i++)
  printf("%i\t",*(p+i));
delete [] p;
```

Б)
```
int *p = new int [10];
for (int i=0;i<10;i++)
  *(p++)=10 – i;
p++;
for (i=0;i<10;i++)
  printf("%i\t",*(p+i));
delete [] p;
```

В)
```
int *p = new int [10];
for (int i=0;i<10;i++)
  *(p--)=10 – i;
p++;
for (i=0;i<10;i++)
  printf("%i\t",*(p+i));
delete [] p;
```

Г)
```
int *p = new int [10];
p = p + 9;
for (int i=0;i<10;i++)
  *(p++)=10 – i;
p++;
for (i=0;i<10;i++)
  printf("%i\t",*(p+i));
delete [] p;
```

Д) правильної відповіді немає

19. Вдосконалення методу Шейкера полягає в ...

А) зупинці сортування за відсутності обмінів протягом проходу
Б) зменшення кількості кроків на кожному проході
В) зменшення кроку між порівнюваними елементами на кожному проході
Г) зміні напрямку сортування на кожному проході
Д) правильної відповіді немає

20. При використанні оператора "–>" після змінної-покажчика, що містить адресу структури, можна ...

А) виконувати спеціальні функції
Б) виконувати операції з полями структури
В) виконувати сортування полів структури
Г) виконувати оголошення екземплярів структури
Д) правильної відповіді немає

21. Який з наведених нижче програмних фрагментів правильний для заповнення структури через змінну-покажчик?

А)
```
struct cat
{
        int Arr[10];
        float f;
}ct;
cat *c;
c = &cat;
for(int i=0;i<10;i++)
        c.Arr[i]=i*10;
c.f=i*1.1;
```

Б)
```
struct cat
{
        int Arr[10];
        float f;
};
cat ct;
cat *c;
c = &ct;
for(int i=0;i<10;i--)
        c->Arr[i]=i*10;
c->f=i*1.1;
```

В)
```
struct cat
{
        int Arr[10];
        float f;
}ct;
cat *c;
c = &ct;
for(int i=0;i<10;i++)
        c->Arr[i]=i*10;
c->f=i*1.1;
```

Г)
```
struct cat
{
        int Arr[10];
        float f;
};
cat ct;
cat *c;
c = &cat;
for(int i=0;i<10;i--)
        c->Arr[i]=i*10;
c->f=i*1.1;
```

Д) правильної відповіді немає

22. Який з наведених нижче виразів виводить слова "Hello world" в стовпець?

А) cout << "Hello" << flush << "world" << endl;
Б) cout << "Hello" << flush << "world" << flush;
В) cin >> "Hello" >> flush >> "world" >> endl;
Г) cin >> "Hello" >> flush >> "world" >> flush;
Д) правильної відповіді немає

23. Властивість функції, що дозволяє виконувати виклик зі свого тіла, називається ...

А) перевантаженням
Б) оголошенням
В) рекурсією
Г) визначенням
Д) правильної відповіді немає

24. У програмі може бути відсутнім оголошення функції, якщо ...

А) її визначення виконано перед всіма програмними викликами
Б) функція має скорочений список аргументів
В) функція має тип параметра, що повертається, void
Г) в програмі немає її визначення, а присутні тільки програмні виклики
Д) правильної відповіді немає

25. Який з наведених нижче програмних лістингів правильний для створення і використання функції, що виводить на консоль результат ділення двох чисел?

А)
```
#include <stdio.h>
int Div(int x, int y);
void main()
{int d=Div(5,2);}
void Div(int x, int y)
{
  if (y!=0)
    printf("%i/%i=%f",x,y,((float)x)/y);
  else
    printf ("Error – div 0!");
}
```

Б)
```
#include <stdio.h>
void Div(int x, int y);
void main()
{Div(5,2);}
void Div(int x, int y)
{
  if (y!=0)
    printf("%i/%i=%f",x,y,((float)x)/y);
  else
    printf ("Error – div 0!");
}
```

В)
```
#include <stdio.h>
void Div(int, int);
void main()
{int d=Div(5,2);}
void Div(int x, int y)
{
  if (y!=0)
    printf("%i/%i=%f",x,y,((float)x)/y);
  else
    printf ("Error – div 0!");
}
```

Г)
```
#include <stdio.h>
int Div(int, int);
void main()
{Div(5,2);}
void Div(int x, int y)
{
  if (y!=0)
    printf("%i/%i=%f",x,y,((float)x)/y);
  else
    printf ("Error – div 0!");
}
```

Д) правильної відповіді немає.

26. Яка умова буде використовуватися для програмного заповнення виділеної діагоналі матриці обраним значенням?

i/j	0	1	2	3	4
0	0,0	0,1	0,2	0,3	0,4
1	1,0	1,1	1,2	1,3	1,4
2	2,0	2,1	2,2	2,3	2,4
3	3,0	3,1	3,2	3,3	3,4
4	4,0	4,1	4,2	4,3	4,4

27. Як зміниться порядок елементів масиву на третьому кроці його сортування за спаданням методом вибірки?

Arr[10] = {12,-3,-7,5,1,9, 6,8,0,1};

28. Знайдіть помилку і поясніть, у чому вона полягає, для програмного коду створення та використання функції, що повертає добуток квадратів двох чисел.

```
#include <stdio.h>

void mult(int x, int y)
{
	return x*x*y*y;
}

void main()
{
	printf("%i^2*%i^2=%i",3,7,mult(3,7));
}
```

29. Поля класу – це ...

А) функції, що є членами класу
Б) дані, котрі видимі в класі
В) функції, котрі видимі в класі
Г) дані, що є членами класу
Д) правильної відповіді немає

30. Список ініціалізації використовується для того, щоб ...

А) визначити значення полів класу в конструкторі
Б) визначити поля класу значеннями аргументів, що передаються в метод
В) викликати методи класу під час виклику конструктора
Г) визначити змінні програми під час виклику конструктора класу
Д) Правильної відповіді немає

31. Який з наведених нижче програмних фрагментів правильний для роботи з класом?

А)
```
class CFirst
{
        void func()
        {printf("1\r\n");}
        public:
                void func2()
                {printf("2\r\n");}
        protected:
                void func3()
                {printf("3\r\n");}
};
void main()
{
        CFirst f;
        f.func2();
}
```

Б)
```
class CFirst
{
        public
                void func()
                {printf("1\r\n");}
        private
                void func2()
                {printf("2\r\n");}
        protected
                void func3()
                {printf("3\r\n");}
};
void main()
{
        CFirst f;
        f.func();
}
```

В)
```
class CFirst
{
        void func()
        {printf("1\r\n");}
        private:
                void func2()
                {printf("2\r\n");}
        protected:
                void func3()
                {printf("3\r\n");}
};
void main()
{
        CFirst f;
        f.func();
}
```

Г)
```
class CFirst
{
        private
                void func()
                {printf("1\r\n");}
        public
                void func2()
                {printf("2\r\n");}
        protected
                void func3()
                {printf("3\r\n");}
};
void main()
{
        CFirst f;
        f.func3();
}
```

Д) правильної відповіді немає

32. Клас, методи і поля якого успадковуються похідним класом, називається ...

А) віртуальним класом
Б) абстрактним класом
В) базовим класом
Г) похідним класом
Д) правильного відповіді немає

33. Який з наведених нижче програмних фрагментів є правильним для роботи з класом?

А)
```
class CFirst
{
        public:
                int z;
                void func();
                void CFirst():z(7)
                {}
        private:
                void func2();
};
void CFirst:func()
{cout<<z<<endl;}
void CFirst:func2()
{cout<<z+1<<endl;}
void main()
{
        CFirst f;
        f.func2();
}
```

Б)
```
class CFirst
{
        public:
                int z;
                void func();
                CFirst():z(7)
                {}
        private:
                void func2();
};
void CFirst::func()
{cout<<z<<endl;}
void CFirst::func2()
{cout<<z+1<<endl;}
void main()
{
        CFirst f(2);
        f.func();
}
```

В)
```
class CFirst
{
        public:
                int z;
                void func();
                CFirst():z(7)
                {}
        private:
                void func2();
}f;
void CFirst::func()
{cout<<z<<endl;}
void CFirst::func2()
{cout<<z+1<<endl;}
void main()
{f.func();}
```

Г)
```
class CFirst
{
        public:
                int z;
                void func();
                void CFirst():z(7)
                {}
        private:
                void func2();
}f;
void CFirst:func()
{cout<<z<<endl;}
void CFirst:func2()
{cout<<z+1<<endl;}
void main()
{f.func ();}
```

Д) правильної відповіді немає

34. Виклик функції похідного класу, яка є перевантаженням функції з таким самим ім'ям, числом і типом аргументів базового класу, з методу класу спадкоємця ...

А) виконується через об'єкт базового класу
Б) виконується через оператор "->"
В) виконується через об'єкт похідного класу
Г) виконується через оператор "::"
Д) правильної відповіді немає

35. Який з наведених нижче програмних фрагментів правильний для роботи з похідним класом.

А)
```
class CCl
{
	public
		int z;
		CCl()
		{z=1;}
		void func();
	private
		void func2();
	protected
		void func3();
};
void CCl:func()
{cout<<z<<endl;}
void CCl:func2()
{printf("%i",z+1);}
void CCl:func3()
{printf("%i",z-1);}
class CSecond:private CCl
{
	public:
		void func4();
		void func5();
};
void CSecond:func4()
{printf("%i",z*2);}
void CSecond:func5()
{func3();}
void main()
{
	CSecond sec;
	sec.func5();
}
```

Б)
```
class CCl
{
	public
		int z;
		CCl()
		{z=1;}
		void func();
	private
		void func2();
	protected
		void func3();
};
void CCl::func()
{cout<<z<<endl;}
void CCl::func2()
{printf("%i",z+1);}
void CCl::func3()
{printf("%i",z-1);}
class CSecond:public CCl
{
	public:
		void func4();
		void func5();
};
void CSecond::func4()
{printf("%i",z*2);}
void CSecond::func5()
{func3();}
void main()
{
	CSecond sec;
	sec.func5();
}
```

В)

```
class CCl
{
	public:
		int z;
		CCl()
		{z=1;}
		void func();
	private:
		void func2();
	protected:
		void func3();
};
void CCl:func()
{cout<<z<<endl;}
void CCl::func2()
{printf("%i",z+1);}
void CCl:func3()
{printf("%i",z-1);}
class CSecond:private CCl
{
	public:
		void func4();
		void func5();
}
void CSecond:func4()
{printf("%i",z*2);}
void CSecond::func5()
{func3();}
void main()
{
	CSecond sec;
	sec.func5();
}
```

Г)

```
class CCl
{
	public:
		int z;
		CCl()
		{z=1;}
		void func();
	private:
		void func2();
	protected:
		void func3();
};
void CCl::func()
{cout<<z<<endl;}
void CCl::func2()
{printf("%i",z+1);}
void CCl::func3()
{printf("%i",z-1);}
class CSecond:public CCl
{
	public:
		void func4();
		void func5();
};
void CSecond::func4()
{printf("%i",z*2);}
void CSecond::func5()
{func3();}
void main()
{
	CSecond sec;
	sec.func5();
}
```

Д) правильної відповіді немає

36. Змінна-покажчик на базовий клас може зберігати адреси ...

А) об'єктів тільки базового класу
Б) об'єктів як базового, так і похідних від нього класів
В) об'єктів лише похідного класу
Г) об'єктів будь-яких класів

Д) правильної відповіді немає

37. Чиста віртуальна функція – це…

38. Яке значення відобразиться на консолі після виконання наступного програмного коду?

```
#include <iostream>
using namespace std;
class CF
{
	public:
		int c;
		CF()
		{c=2;}
		void func1()
		{
			c++;
			func2();
		}
		~CF()
		{c=0;}
	private:
		void func2()
		{c+=2;}
	protected:
		void func3()
		{c-=2;}
};
```

```
class CS:public CF
{
	public:
		CS(int nc)
		{c=nc;
		c+=3;}
		void func4()
		{c--;
		func6();}
		~CS()
		{c=3;}
	private:
		void func5()
		{c-=3;}
	protected:
		void func6()
		{c--;
		func3();}
};
void main()
{
	CS sec(5);
	sec.func4();
	cout<<sec.c<<endl;
}
```

39. Похідний клас – це…

40. Специфікатор доступу protected визначає ...

41. Яка послідовність цифр виведеться на консоль при виконанні наступного програмного коду?

```
#include <iostream>
using namespace std;
class CCl
{
        public:
                CCl()
                {
                        cout<<"1";
                        func3();
                }
                void func1()
                {cout<<"2";}
                ~CCl()
                {cout<<"3";}
        private:
                void func2()
                {cout<<"4";}
        protected:
                void func3()
                {cout<<"5";}
};
```

```
class CSec:public CCl
{
        public:
                CSec()
                {cout<<"6";}
                void func4()
                {cout<<"7";}
                ~CSec()
                {func6();
                cout<<"8";}
        private:
                void func5()
                {cout<<"9";}
        protected:
                void func6()
                {cout<<"0";}
};
void main()
{       CSec sec;
        sec.func1();
}
```

А) 1562083 В) 1506283 Д) правильної відповіді немає
Б) 5106283 Г) 5160238

42. Знайдіть помилку і поясніть, у чому вона полягає, для програмного коду роботи з математичним класом.

```
#include <iostream.h>
#include <math.h>
class CMat
{ int p;
  public:
    CMat(int pw) p(pw)
    {}
    CMat() p(0)
    {}
    float Sum(float x, float y)
    {return (x+y)*pow(10,p);}
    float Sub(float x, float y)
    {return (x-y)*pow(10,p);}
    float Mult(float x, float y)
    {return x*y*pow(10,p)*pow(10,p);}
    float Div(float x, float y)
    { if(y!=0) return x/y;
      else return 0;
    }
};
void main()
{ CMat m1(-3),m2(2),m3;
  CMat *pm;
  pm=&m1;
  cout<<pm->Sum(7,2)<<endl;
  pm=&m2;
  cout<<pm->Sub(7,2.3)<<endl;
  pm=&m3;
  cout<<pm->Mult(11,123)<<endl;
}
```

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

Застосування у сукупності з навчальним посібником наступної літератури дозволить значно розширити та поглибити знання та вміння щодо програмування мовою С++.

1. Лафоре, Р. Объектно-ориентированное программирование в С++. Класика Computer Science [Текст]: пер. с англ./ Р. Лафоре. – 4-е изд. – СПб.: Питер, 2005. – 924 с.: ил.
2. Шилдт, Г. Полный справочник по С [Текст]: пер. с англ./ Г. Шилдт. – 4-е изд. – М.: Издательский дом "Вильямс", 2004 – 704 с.: ил.
3. Шилдт, Г. Полный справочник по С++ [Текст]: пер. с англ./ Г. Шилдт. – 4-е изд. – М.: Издательский дом "Вильямс", 2006 – 800 с.: ил.
4. Шилдт, Г. С++. Руководство для начинающих [Текст]: пер. с англ./ Г. Шилдт. – М.: Издательский дом "Вильямс", 2006 – 672 с.: ил.
5. Страуструп, Б. Язык программирования С++. Специальное издание [Текст]: пер. с англ./ Б. Страуструп – М.: "Бином"-"Невский диалект", 2012 – 1136 с.: ил.
6. Березин, В.И. Начальный курс С и С++ [Текст]/ В.И. Березин, С.Б. Березин – М.: Диалог-Мифы, 2003. – 288 с.
7. Архангельский, А. Я. Программирование в С++ Builder 6 и 2006 [Текст] А.Я. Архангельский, М. А.Тагин – М.: Бином-Пресс, 2007. – 1184с.
8. Седжвик, Р. Фундаментальные алгоритмы на С. Анализ/Структуры данных/Сортировка/Поиск [Текст]: пер. с англ./ Р. Седжвик – СПб: ООО "ДиаСофт", 2003. – 672 с.
9. Солтер, Николас А. С++ для профессионалов. [Текст]: пер. с англ./ Николас А. Солтер, Скотт Дж. Клепер – М.: Издательский дом "Вильямс", 2006. – 912 с.: ил.

ПЕРЕЛІК СКОРОЧЕНЬ

АКС – Автоматизація комп'ютерних систем

ASCII – American Standard Code for Information Interchange – Американський стандартний код для інформаційного обміну

BIN – Binary – Двійкова система числення

C89 – Стандарт ANSI/ISO мови С 1989 року

C99 – Стандарт ANSI/ISO мови С 1999 року

DEC – Decimal – Десяткова система числення

HEX – Hexadecimal – Шістнадцяткова система числення

OCT – Octal – Вісімкова система числення

ДОДАТОК А
Основні положення "ГОСТ 19.701-90" – Схеми алгоритмів, програм даних і систем. Умовні позначення і правила виконання

Справжній стандарт поширюється на умовні позначення (символи) в схемах алгоритмів, програм, даних, систем і встановлює правила виконання схем, які використовуються для відображення різних видів задач, обробки даних і засоби їх рішення.

Стандарт не поширюється на форму записів і позначень, які розміщуються всередині символів чи поряд з ними і слугують для уточнення виконуваних ними функцій.

Вимоги стандарту є обов'язковими.

1. ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Схеми алгоритмів, програм, даних і систем (далі – схеми) складаються з символів, що мають задане значення, короткого пояснювального тексту і ліній, які їх з'єднують.

1.2 Схеми можуть використовуватись на різних рівнях деталізації, причому число рівнів залежить від розмірів і складності задачі обробки даних. Рівень деталізації повинен бути таким, щоб різні частини і взаємозв'язок між ними були зрозумілі в цілому.

1.3 У справжньому стандарті визначені символи, які використовуються в документації по обробці даних, і наведено керівництво умовними позначеннями для використання їх у:

1) схемах даних;

2) схемах програм;

3) схемах роботи системи;

4) схемах взаємодії програм;

5) схемах ресурсів системи.

1.4 В стандарті використовуються наступні поняття:

1) основний символ – символ, який використовується в тих випадках, коли тип (вид) процесу чи носія даних невідомий чи немає необхідності в описі фактичного носія даних;

2) специфічний символ – символ, який використовується у тих випадках, коли відомий точний тип (вид) процесу чи носія даних або коли необхідно описати фактичний носій даних;

3) схема – графічне подання визначення, аналізу чи методу рішення задачі, в якому використовуються символи для відображення операцій, даних, потоку, обладнання і т.і.

2. ОПИСАННЯ СХЕМ

2.1. Схема даних

2.1.1. Схеми відображають шлях даних при рішенні задач і визначають етапи обробки, а також різні використовувані носії даних.

2.1.2. Схема даних складається з:

1) символів даних (символи даних можуть також вказувати вид носія даних);

2) символів процесу, що потрібно виконати над даними (символи процесу можуть також вказувати функції, виконувані обчислювальною машиною);

3) символів ліній, що вказують потоки даних між процесами і (чи) носіями даних;

4) спеціальних символів, які використовується для полегшення написання і читання схеми.

2.1.3. Символи даних передують і слідують за символами процесу. Схема даних починається і закінчується символами даних (за виключення спеціальних символів, вказаних в п 3.4).

2.2. Схема програми

2.2.1. Схеми програм відображають послідовність операцій у програмі.

2.2.2. Схема програми складається з:

1) символів процесу, що вказують фактичні операції обробки даних (включаючи символи, які визначають шлях, якого потрібно дотримуватися з урахуванням логічних умов);

2) лінійних символів, що вказують потік керування;

3) спеціальних символів, що використовуються для полегшення написання і читання схеми.

2.3. Схема роботи системи

2.3.1. Схеми роботи системи відображають керування операціями і потік даних у системі.

2.3.2. Схема роботи системи складається з:

1) символів даних, що вказують на наявність даних (символи даних можуть також вказувати на вид носія даних);

2) символів процесу, що вказують на операції, які потрібно виконати над даними, а також, що визначають логічний шлях, якого потрібно дотримуватись;

3) лінійних символів, що вказують на потоки даних між процесами і (чи) носіями даних, а також потік керування між процесами;

4) спеціальних символів, що використовуються для полегшення написання блок-схеми.

2.4. Схема взаємодії програм

2.4.1. Схеми взаємодії програм відображають шлях активацій програм і взаємодій з відповідними даними. Кожна програма у схемі взаємодії програм показується тільки один раз (в схемі роботи системи програма може зображуватися більше ніж в одному потоці керування).

2.4.2. Схема взаємодії програм складається з:

1) символів даних, що вказують на наявність даних;

2) символів процесу, що вказують на операції, які потрібно виконати над даними;

3) лінійних символів, що відображають потік між процесами і даними, а також ініціації процесів;

4) спеціальних символів, що використовуються для полегшення написання і читання схеми.

2.5. Схема ресурсів системи

2.5.1. Схеми ресурсів системи відображають конфігурацію блоків даних і блоків, що обробляють дані, які потрібні для рішення задачі чи набору задач.

2.5.2. Схема ресурсів системи складається з:

1) символів даних, що відображають вхідні, вихідні і запам'ятовуючі пристрої обчислювальної машини;

2) символів процесу, що відображають процесори (центральні процесори, канали тощо);

3) лінійних символів, що відображають передачу даних між пристроями вводу-виводу і процесорами, а також передачу керування між процесорами;

4) спеціальних символів, що використовуються для полегшення написання і читання схеми.

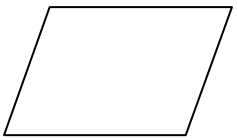
3. ОПИСАННЯ СИМВОЛІВ

3.1. Символи даних

3.1.1. Основні символи даних

3.1.1.1. Дані

Символ відображає дані, носій даних не визначений.

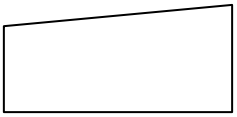


3.1.1.2. Документ

Символ відображає дані, подані на носії в зручній для читання формі.

3.1.1.3. Ручний ввід

Символ відображає дані, які вводяться вручну під час обробки з пристроїв будь-якого типу (клавіатура, перемикачі, кнопки, світлове перо, смужки зі штриховим кодом).



3.1.1.4. Дисплей

Символ відображає дані, подані у людиночитаємій формі на носії у вигляді відображуючого пристрою (екран для візуального спостереження, індикатори виводу інформації).

3.1.1.5. Процес

Символ відображає функцію обробки даних будь-якого вигляду (виконання певної операції чи групи операцій, які приводять до заміни значень, форми чи розміщення інформації або до визначення, за яким з декількох направлень потоку потрібно рухатись).

3.1.1.6. Наперед визначений процес

Символ відображає наперед визначений процес, який складається з однієї чи декількох операцій чи кроків програми, які визначені в іншому місці (у підпрограмі, модулі).

3.1.1.7. Ручна операція

Символ відображає будь-який процес, який виконує людина.

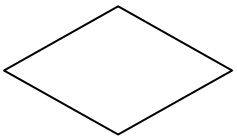
3.1.1.8. Підготовка

Символ відображає модифікацію команди чи групи команд з ціллю вплинути на деяку подальшу функцію (встановлення перемикача, модифікація індексного регістру чи ініціалізація програми).

3.1.1.9. Рішення

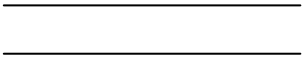
Символ відображає рішення чи функцію типу перемикача, яка має один вхід і декілька альтернативних виходів, один і тільки один з яких може бути активований після обчислення умов, визначених усередині цього символа.

Відповідні результати обчислення можуть бути записані по сусідству з лініями, які відображають ці шляхи.

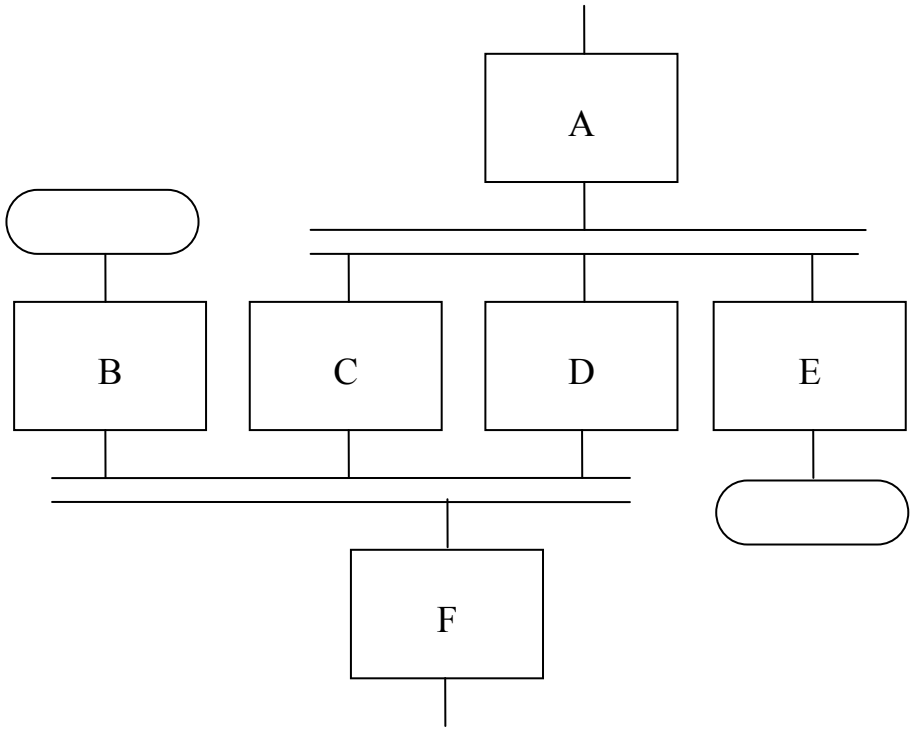


3.1.1.10. Паралельні дії

Символ відображає синхронізацію двох чи більше паралельних операцій.



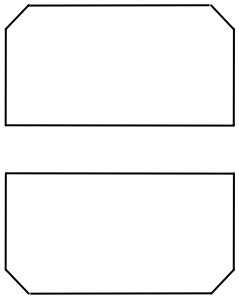
Приклад:



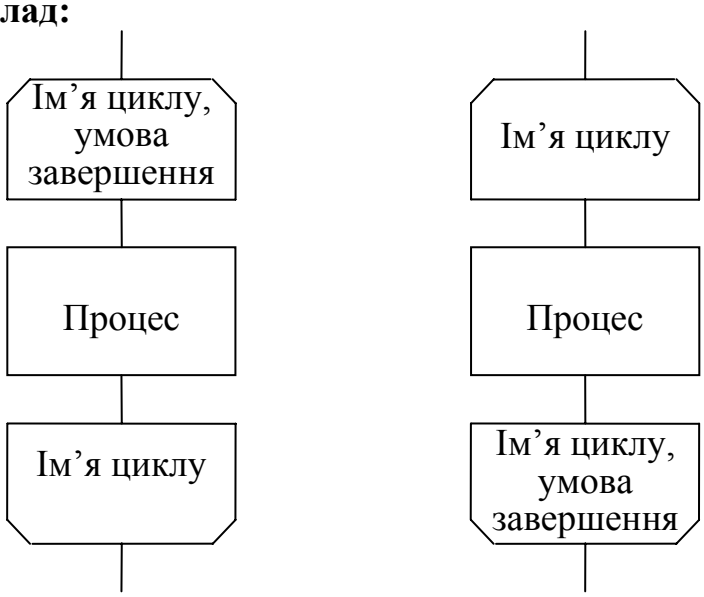
Примітка: Процеси C, D і E не можуть початися до тих пір, поки не завершиться процес A; аналогічно процес F повинен чекати завершення процесів B, C і D, однак процес C може початися і (чи) завершитися перш, ніж відповідно почнеться і (чи) завершиться процес D.

3.1.1.11. Межа циклу

Символ, який складається з двох частин, відображає початок і кінець циклу. Обидві частини символа мають один і той самий ідентифікатор. Умови для ініціалізації, приросту, завершення тощо розміщуються всередині символа, на початку чи у кінці в залежності від розташування операції, яка перевіряє умову.



Приклад:



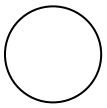
3.1.1.12. Основний символ лінії

Символ відображає потік даних чи керування

Примітка: при необхідності чи для підвищення зручності читання можуть бути додані стрілки-покажчики.

3.1.1.13. Сполучувач

Символ відображає вихід у частину схеми і вхід з іншої частини цієї схеми і використовується для розриву лінії і продовження її в іншому місці. Відповідні символи-сполучувачі повинні мати одне і те саме унікальне позначення.



Примітка: цим унікальним позначенням звичайно є заголовні латинські літери.

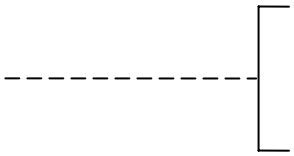
3.1.1.14. Термінатор

Символ відображає вихід у зовнішнє середовище і вхід із зовнішнього середовища (початок чи кінець схеми програми, зовнішні використання і джерело чи пункт призначення даних).

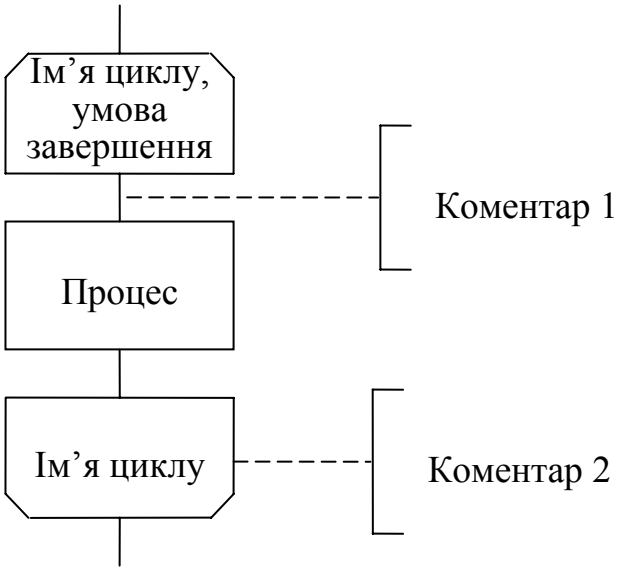
Примітка: при використанні даного символу в якості початку/кінця блок-схеми функції замість слова початок у першому блоці міститься ім'я функції, що розглядається.

3.1.1.15. Коментар

Символ використовують для додавання описових коментарів чи пояснювальних записів з метою пояснення чи приміток. Пунктирні лінії в символі коментару зв'язані з відповідним символом чи можуть обводити групу символів. Текст коментарів чи приміток повинен бути розміщений біля фігури, до якої відноситься.



Приклад:



4. ПРАВИЛА ЗАСТОСУВАННЯ СИМВОЛІВ І ВИКОНАННЯ СХЕМ

4.1. Правила застосування символів

4.1.1. Символ слугує для графічної ідентифікації функції, яку він відображає, незалежно від тексту усередині цього символу.

4.1.2. Символи в схемі повинні бути розміщені рівномірно. Потрібно дотримуватися розумної довжини з'єднань і мінімального числа довгих ліній.

4.1.3. Більшість символів задумані таким чином, щоб дати можливість включення тексту усередину символа. Форми символів, які встановлені справжнім стандартом, повинні служити керівництвом для символів, що фактично використовуються. Не повинні змінюватися кути і інші параметри, які впливають на відповідну форму символів. Символи повинні бути, за можливістю, одного розміру.

Символи можуть бути накреслені в будь-якій орієнтації, але, за можливістю, переважною є горизонтальна орієнтація. Дзеркальне зображення форми символа означає одну і ту саму функцію, але не є переважним.

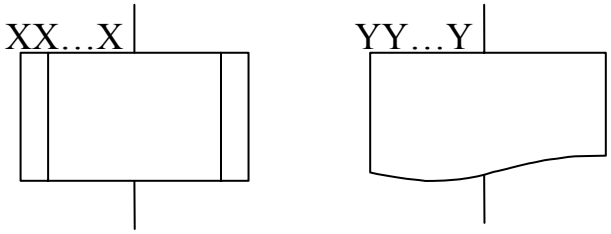
4.1.4. Мінімальну кількість тексту, необхідну для розуміння функції даного символу, потрібно розміщувати всередині даного символа. Текст для читання повинен записуватись зліва направо і зверху вниз незалежно від напрямку потоку.

Якщо об'єм тексту, який розміщується всередині символу, більший за його розміри, потрібно використовувати символ коментар.

Якщо використання символів коментар може заплутати чи зруйнувати хід схеми, текст потрібно розміщувати на окремому аркуші і давати перехресне посилання на символ.

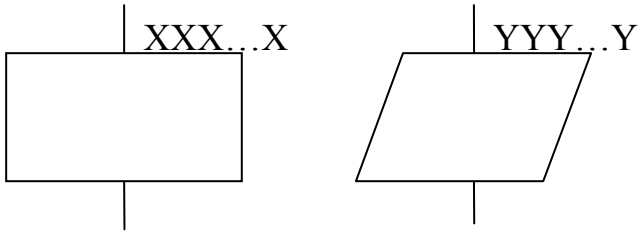
4.1.5. В схемах може використовуватись ідентифікатор символів. Це зв'язаний з даним символом ідентифікатор, який визначає символ для використання у довідкових цілях в інших елементах документації (наприклад, у лістингу програми). Ідентифікатор символа повинен розташовуватися зліва над символом.

Приклад:



4.1.6. В схемах може використовуватись описання символів – будь-яка інша інформація, наприклад, для відображення спеціального застосування символа з перехресним посиланням чи для покращення розуміння функції як частини схеми. Описання символа повинно бути розташовано справа над символом.

Приклад:



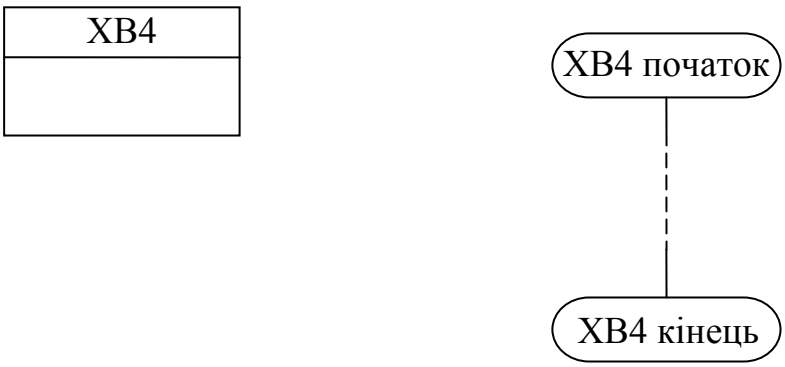
4.1.7. В схемах роботи системи символи, які відображують носії даних, у багатьох випадках вказують на способи вводу-виводу. Для використання в якості посилання на документацію текст на схемі для символів, що відображають способи виводу, повинен розміщуватись справа над символом, а текст для символів, що відображають способи вводу – справа під символом.

4.1.8. В схемах може використовуватись докладне подання, яке позначується за допомогою символу зі смугою для процесу чи даних. Символ зі смугою вказує, що в цьому ж комплекті документації в іншому місці є більш докладне подання.

Символ зі смугою являє собою будь-який символ, усередині якого у верхній частині проведена горизонтальна лінія. Між цією лінією і верхньою лінією символа розміщено ідентифікатор, який вказує на детальне подання даного символа.

В якості першого і останнього символа детального подання повинен бути використаний символ покажчика кінця. Перший символ покажчика кінця повинен містити посилання, яке є також в символі зі смугою.

Символ зі смугою **Детальне подання**



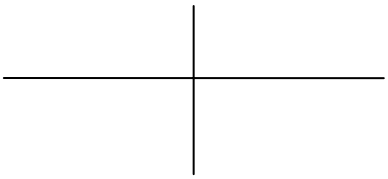
4.2. Правила виконання з'єднань

4.2.1. Потоки даних чи потоки керування в схемах позначаються лініями. Напрямок потоку зліва направо і зверху вниз вважається стандартним.

У випадках, коли необхідно внести більшу ясність у схему (наприклад, при з'єднаннях), на лініях використовуються стрілки. Якщо потік має напрямок, який відрізняється від стандартного, стрілки повинні вказувати цей напрямок.

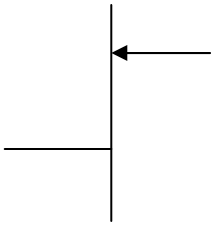
4.2.2. В схемах потрібно уникати перетинання ліній. Лінії, що перетинаються, не мають логічного зв'язку між собою, тому зміни напрямку в точках перетину не допускаються.

Приклад:



4.2.3. Дві чи більше вхідних ліній, можуть об'єднуватися в одну вихідну лінію. Якщо дві чи більше ліній об'єднуються в одну, місце об'єднання повинно бути зміщене.

Приклад:

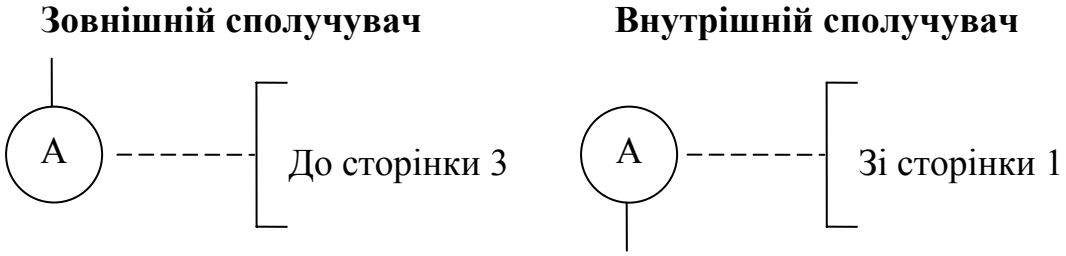


4.2.4. Лінії в схемах повинні підходити до символа зліва чи зверху, а виходити справа чи знизу. Лінії повинні бути спрямовані до центру символа.

4.2.5. При необхідності лінії в схемах можна розривати для уникнення зайвих перетинів чи занадто довгих ліній, а також, якщо схема складається з декількох аркушів. Сполучувач на початку розриву називається зовнішнім, а сполучувач у кінці разриву – внутрішнім.

4.2.6. Посилання до сторінок можуть бути наведені спільно з символом коментарію для їх сполучників.

Приклад:



4.3. Спеціальні умовні позначення

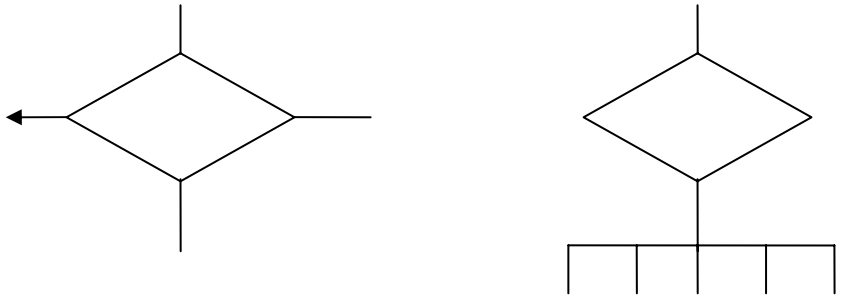
4.3.1. Декілька виходів

4.3.1.1. Декілька виходів з символа потрібно показувати:

1) декількома лініями від даного символа до інших символів;

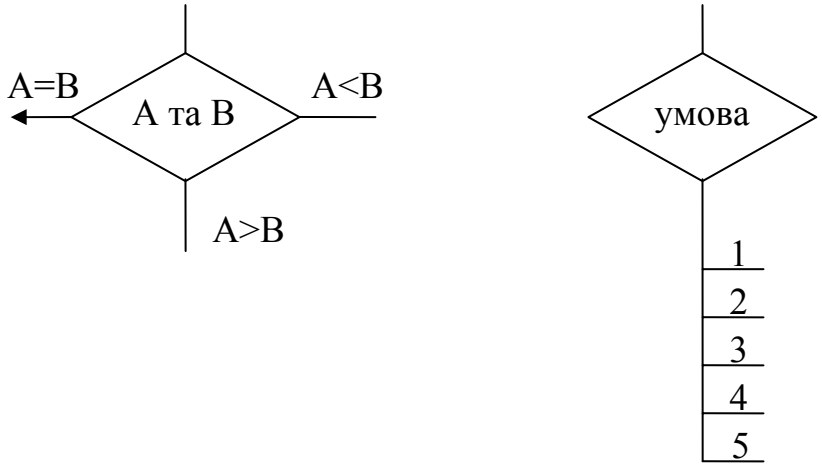
2) однією лінією від даного символа, яка потім розгалужується у відповідне число ліній.

Приклад:



4.3.1.2. Кожний вихід з символа повинен супроводжуватись відповідними значеннями умов, щоб показати логічний шлях, який він подає, з тим, щоб ці умови і відповідні посилання були ідентифіковані.

Приклад:



4.3.2. Подання, що повторюється

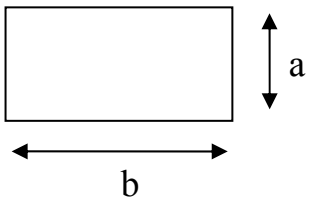
4.3.2.1. Замість одного символу з відповідним текстом можуть бути використані декілька символів з перекриттям зображення, кожний з яких містить описовий текст (використання чи формування декількох носіїв даних чи файлів, виробництво великої кількості копій друкованих звітів).

4.3.2.2. Коли декілька символів являють собою впорядковану множину, це упорядкування повинно розташовуватись від переднього (першого) до заднього (останнього).

4.3.2.3. Лінії можуть входити чи виходити з будь-якої точки перекритих символів, однак вимоги п. 4.2.4 повинні бути дотриманими. Пріоритет чи послідовний порядок декількох символів не змінюється точкою, в яку лінія входить чи з якої виходить.

5. ВІДНОШЕННЯ ГЕОМЕТРИЧНИХ ЕЛЕМЕНТІВ СИМВОЛІВ

5.1. Розмір висоти символу *a* повинен вибиратися з ряду 10, 15, 20 мм. Допускається збільшувати розмір *a* на число, що є кратним 5. Розмір ширини символа *b* дорівнює 1,5*a*.



Примітка: При ручному виконанні схем алгоритмів і програм для обов'язкових символів 1 – 5, 11, 12, 16, 29 і рекомендованих символів 3 і 4 допускається встановлювати *b* таким, що дорівнює 2*a.* Обов'язкові символи 7 – 10, 14 і рекомендований символ 8 допускається подавати у вигляді рівнобедреного прямокутного трикутника з катетом *a.*

5.2. При виконанні умовних графічних позначень автоматизованим методом розміри геометричних елементів символів округляються до значень, які визначаються технічними можливостями пристроїв, що використовуються.

5.3. Мінімальна відстань між символами вибирається з рахунку 0,5*a*. Мінімально допустима відстань між лініями з'єднання – не менш 3 мм.

ДОДАТОК Б
Приклад оформлення лабораторної роботи

Лабораторна робота №4

Тема: Розгалуження програм на основі операторів умови

Мета роботи

Ознайомитись з операторами умови та отримати практичні навички реалізації розгалуження програм.

Хід роботи

1) Ознайомитись з методичними вказівками до лабораторної роботи та темою "Розгалуження";

2) відповідно до завдання за варіантом розробити схему алгоритму програми, що працює з операторами умови;

3) за схемою алгоритму написати програму на мові С++ та отримати результати її роботи;

4) зробити висновки;

5) підготувати звіт про виконання лабораторної роботи, що включає наступні пункти: номер, тема, мета та хід лабораторної роботи, завдання, схема алгоритму програми, лістинг програми з коментуванням кожної інструкції, результат роботи програми (скріншот), висновки.

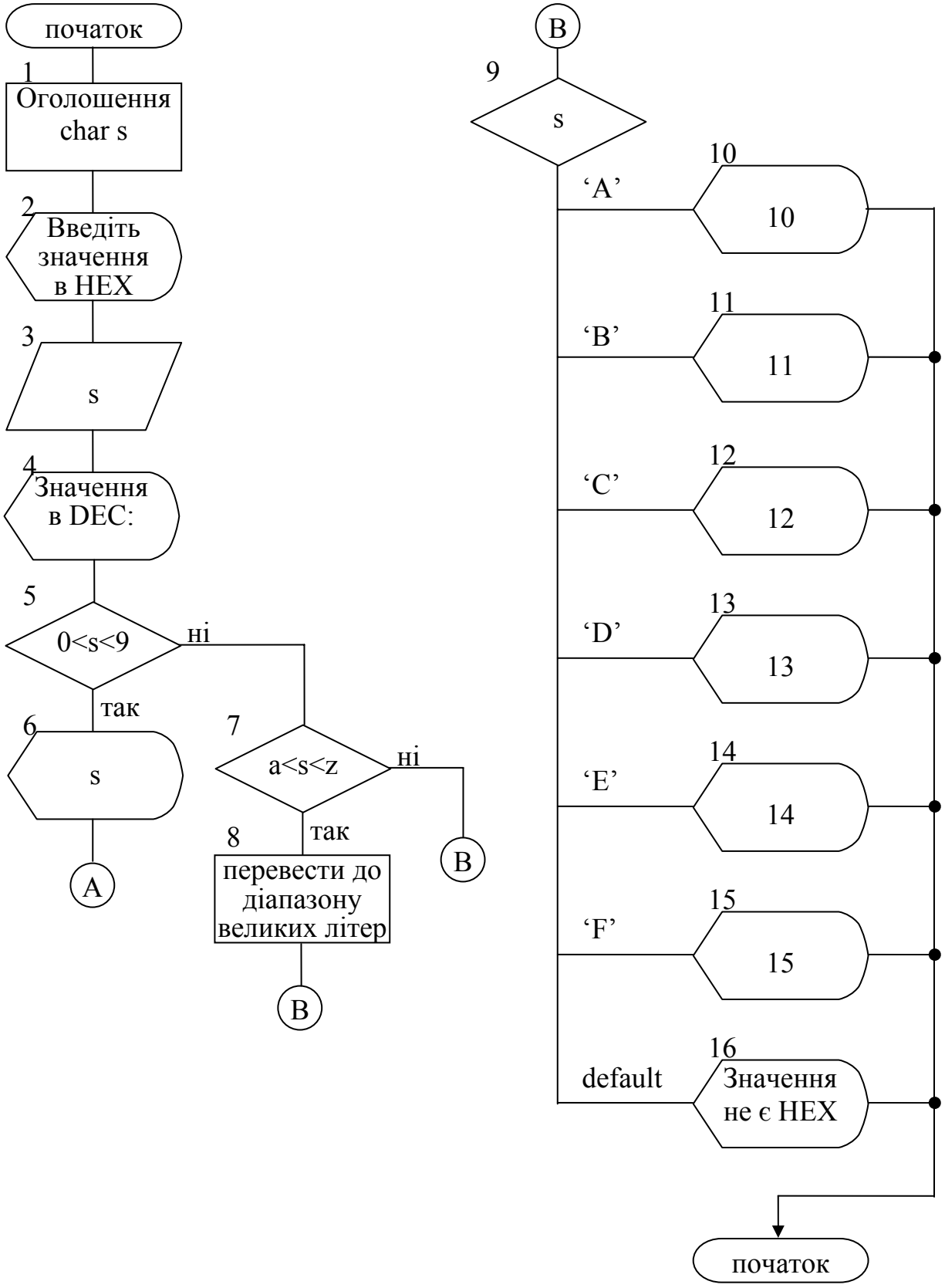
Завдання

Розробити схему алгоритму та написати програму, що містить розгалуження програмного коду.

Варіант N:

Програма зчитує з консолі однорозрядне шістнадцяткове число та виводить відповідне йому в десятковій формі.

Схема алгоритму програми



Лістинг програми

```
#include <stdio.h>                    //підключення бібліотеки вводу/виводу

void main()                           //підключення бібліотеки математичних функцій
{
        char s;                       //оголошення символьної змінної
        printf("Enter HEX value: ");  //вивід запиту до користувача
        scanf("%s", &s);              //отримання значення від користувача
        printf("DEC value is: "); //вивід константного рядка
        if(s>='0' && s<='9')          //якщо значення є цифрою
                printf("%c\r\n", s);//вивід значення на консоль
        else                          //інакше – значення не цифра
        {
                if(s>='a' && s<='z')  //якщо значення є маленькою літерою
                        s -=32;       //перетворити у велику літеру
                switch(s)             //перевірка значення змінної
                {
                        case 'A':     //якщо символ 'A'
                                printf ("10\r\n"); //вивід на консоль цифри 10
                                break;      //вихід з оператора
                        case 'B':     //якщо символ 'B'
                                printf ("11\r\n"); //вивід на консоль цифри 11
                                break;      //вихід з оператора
                        case 'C':     //якщо символ 'C'
                                printf("12\r\n"); //вивід на консоль цифри 12
                                break;      //вихід з оператора
                        case 'D':     //якщо символ 'D'
                                printf("13\r\n"); //вивід на консоль цифри 13
                                break;      //вихід з оператора
                        case 'E':     //якщо символ 'E'
                                printf("14\r\n"); //вивід на консоль цифри 14
                                break;      //вихід з оператора
                        case 'F':     //якщо символ 'F'
                                printf("15\r\n"); //вивід на консоль цифри 15
                                break;      //вихід з оператора
                        default:      //якщо інший символ
//вивід константного рядка
                                printf("It is not HEX value!\r\n");
                }
        }
}
```

Результати роботи програми

```
Enter HEX value: a
DEC value is: 10
```

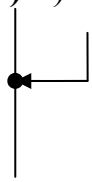
Висновки: Розгалуження слугує для того, щоб зробити програмний код більш гнучким до вимогливостей користувача та вирішення логічних задач. Мова С++ містить два оператори умови: if-else та switch. Кожен з них має своє призначення. Оператор if-else є більш універсальним, бо на ньому може бути реалізовано будь-яке розгалуження відповідно до умови, що задана явним чином. Оператор switch працює лише з цілочисельними змінними і це обмежує його застосування. Призначення switch – перевірка значень змінних.

Додаток В
Відповіді до тестових питань частини V

1) А) процедурною мовою програмування

2) Д) правильної відповіді немає

3) Б)



4) В) int

5) Д) правильної відповіді немає

6) В) нуль або одиниця

7) Г) блок рішення

8) Г) оператор for при оголошенні відразу дозволяє визначити початкове значення, крок і кінцеву умову для змінної

9) Д) правильної відповіді немає

10) В) масив з безліччю розмірностей

11) Вивід на консоль значення 6

12) Помилка в тому, що після закриваючої круглої скобки умовного виразу оператора циклу з передумовою поставлено ";". Це призведе до того, що програмний блок тіла стане незалежним від оператора і його буде виконано лише один раз. Повинно бути так:

```
while(i++<100)
```

13)

```
#include <stdio.h>
void main()
{
        int z;
        scanf("%i", &z);
        for(;;)
        {
```

```
            z++;
            if(z >10)
                  break;
      }
}
```

14) 10

15) Формулу для виводу символа "X" треба змінити на наступну:
((2*n-1)+2*i+1)/2

16) Б)

1	1	1	1	1
1	1	1	1	0
1	1	1	0	0
1	1	0	0	0
1	0	0	0	0

17) А) збільшується залежно від типу даних, визначеного при оголошенні

18) А)

```
int *p = new int [10];
p = p + 9;
for (int i=0;i<10;i++)
      *(p--)=10 – i;
p++;
for (i=0;i<10;i++)
      printf("%i\t",*(p+i));
delete [] p;
```

19) А) зупинці сортування за відсутності обмінів протягом проходу

20) Б) виконувати операції з полями структури

21) В)

```
struct cat
{
      int Arr[10];
      float f;
}ct;
cat *c;
c = &ct;
for(int i=0;i<10;i++)
{     c->Arr[i]=i*10;
}
c->f=i*1.1;
```

22) Д) правильної відповіді немає

23) В) рекурсією

24) А) її визначення виконано перед всіма програмними викликами

25) Б)

```
#include <stdio.h>
void Div(int x, int y);
void main()
{Div(5,2);}
void Div(int x, int y)
{
        if (y!=0)
                printf("%i/%i=%f",x,y,((float)x)/y);
        else
                printf ("Error – div 0!");
}
```

26) i+j ==7

27)

крок	**масив {12,-3,-7,5,1,9, 6,8,0,1}**									
1	12	-3	-7	5	1	9	6	8	0	1
2	12	9	-7	5	1	-3	6	8	0	1
3	12	9	8	5	1	-3	6	-7	0	1

28) Помилка в тому, що функція, яка повертає значення, не може мати тип даних void. Повинно бути так:

```
int mult(int x, int y)
```

29) Г) дані, члени класу

30) А) визначити значення полів класу в конструкторі

31) А)

```
class CFirst
{
        void func()
        {printf("1\r\n");}
        public:
                void func2()
                {printf("2\r\n");}
        protected:
```

```
        void func3()
        {printf("3\r\n");}
};
void main()
{
    CFirst f;
    f.func2();
}
```

32) В) базовим класом

33) В)

```
class CFirst
{
    public:
        int z;
        void func();
        CFirst():z(7)
        {}
    private:
        void func2();
}f;
void CFirst::func()
{cout<<z<<endl;}
void CFirst::func2()
{cout<<z+1<<endl;}
void main()
{f.func();}
```

34) Д) правильної відповіді немає

35) Г)

```
class CCl
{
    public:
        int z;
        CCl()
        {z=1;}
        void func();
    private:
        void func2();
    protected:
        void func3();
};
```

```
void CCl::func()
{cout<<z<<endl;}
void CCl::func2()
{printf("%i",z+1);}
void CCl::func3()
{printf("%i",z-1);}
class CSecond:public CCl
{
	public:
		void func4();
		void func5();
};
void CSecond::func4()
{printf("%i",z*2);}
void CSecond::func5()
{func3();}
void main()
{
	CSecond sec;
	sec.func5();
}
```

36) Б) об'єктів як базового, так і похідних від нього класів

37) Чиста віртуальна функція – це функція, при оголошенні котрої після закриваючої круглої скобки списку аргументів зазначено "=0"

38) 4

39) Похідний клас – це клас, що успадковує властивості від базового класу за рахунок того, що створює в собі копію його об'єкта.

40) Специфікатор доступу protected визначає захищений доступ до членів класу, що обмежує їх застосування лише методами поточного класу та його спадкоємця

41) А) 1562083

42) Помилка в тому, що неправильно використано список ініціалізації при оголошенні обох конструкторів класу. Повинно бути так:

```
CMat(int pw):p(pw)
CMat():p(0)
```

ПРЕДМЕТНИЙ ПОКАЖЧИК

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

Навчальне видання

Ткачов Віктор Васильович
Огієнко Павло Юрійович
Макітренко Роман Васильович

КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ ТА ПРОГРАМУВАННЯ

Навчальний посібник

Том 2
ДОДАТКОВІ ВІДОМОСТІ ТА ПРАКТИЧНІ ЗАВДАННЯ

Редактор Є.М. Ільченко

Підписано до друку 22.08.2012 р. Формат 30х42/4
Папір офсет. Ризографія. Ум. друк. арк. 10,7
Обл.-вид. арк. 14,9. Тираж 100 пр. Зам. №

Підготовлено до друку та видрукувано
у Державному ВНЗ "Національний гірничий університет"
Свідоцтво про внесення до Державного реєстру ДК № 1842
від 11.06.2004 р.

49005, м. Дніпропетровськ, просп. К.Маркса, 19