Міністерство освіти і науки Украни
Національний Авіаційний Університет

М.О. СИДОРОВ

ОБ’ЄКТНО ОРІЄНТОВАНА ПАРАДИГМА

КИЇВ 2009

УДК 004 4(042 4) ББКз973 . 0я7

С 347

Рецензенти: *С.Л.Кривий* - д-р фіз-мат. наук, проф., *ОМ. Ходзінський -* канд. фіз-мат. наук, *ЮМ. Крамар -* канд. техн. наук, доц.

*Затверджено* *методично-редакційною радою* **^ ш а л ь н о ­**

**го** *авіаційного університету (протокол № 14 від 03.07.2008 р.).*

***4***

|  |  |  |
| --- | --- | --- |
| С 347 | С И О ^ є к т ^ р і є н т о в а н а парадигма | : конспект лекцій / М О . Сидо- |
|  | ров. - К. : Вид во Нац. авіац. ун-ту «НАУ-друк», 2009. - 80 с. |
|  | **Конспект лекцій містить матеріали, що розкривають основні положен-** |
|  | **ня об'єктно орієнтованої парадигми.** | **псгигч** |
|  | **Для студентів напрямів 6.050103 «Програмна інженерія» та 6.050101** |
|  | **«Комп'ютерні науки».** |  |

|  |  |
| --- | --- |
| **УДК** | **004.4(042.4)** |
| **ББК** | **3973.0 я7** |

|  |  |  |
| --- | --- | --- |
|  | **З М І С Т** |  |
| **Вступ** |  | **4** |
| **Модуль І .** | **О С Н О В И О Б ' Є К Т Н О - О Р І Є Н Т О В А Н О Г О** |  |
| **П Р О Г Р А М У В А Н Н Я** | **5** |
| **1.1. Схеми** | **реалізації постпідпрограмного рівня** |  |
| **інкапсуляції** |  | **5** |
| **1.2. Принципи об'єктно-орієнтованого програмування** | **7** |
| **1.3. Будова класу** | **10** |
| **1.4. Приховування інформації. Роздільна компіляція** | **12** |
| **1.5. Область дії об'єкта. Правила доступу** |  |
| **до компонентів класу** | **16** |
| **1.6. Приховування інформації. Використання класів** | **23** |
| **1.7. Доступ до рі іуаіе-частини класу.** |  |
| **«Дружні» підпрограми** | **26** |
| **1.8. Оператор привласнення для об'єктів** | **27** |
| **1.9. Спеціальні методи класу** | **28** |
| **1.10. Виклик конструкторів і деструкторів** | **32** |
| **1.1 !. Динамічний розподіл пам'яті** | **34** |
| **1.12. Особливості застосування параметрів у методах** | **37** |
| **1.13. Об'єкти як члени класу. Композиція** | **47** |
| **Модуль 2 . П Р И Н Ц И П И О Б ' Є К Т Н О - О Р І Є Н Т О В А Н О Г О** |  |
| **П Р О Г Р А М У В А Н Н Я** | **52** |
| **2.1. Спадковість** | **52** |
| **2.2. Поліморфізм** | **66** |
| **2.3. Поліморфізм. Пізнє зв'язування** | **76** |
| **Список літератури** | **80** |

**© СидоровМ.О.,2009**

**В С Т У П**

|  |  |  |
| --- | --- | --- |
| **К о н ц е п ц і я о б ' є к т н о - о р і є н т о в а н о г о** | **п р о г р а м у в а н н я** | **істо ­** |
| **р и ч н о пов'язана** | **з к о н ц е п ц і є ю** | **універсальної** | **мови п р о г р а м у ­** |
| **вання, я к а і н т е н с и в н о** | **розроблялася в** | **50 -** | **6 0 - х р о к а х** | **XX** | **ст.** |
| **О д н і є ю** | **з т а к и х** | **б у л а м о в а** | **м о д е л ю в а н н я** | **S i m u l a - 6 7 ,** | **створена** |
| **т а к ,** | **щ о** | **вона** | **дозволяла с т в о р ю в а т и с п е ц і а л і з о в а н і** | **м о в и** | **я к ї ї** |
| **п і д м н о ж и н и .** | **З а с о б о м** | **для с т в о р е н н я** | **с п е ц і а л і з о в а н и х** | **м о в** |
| **слугував м е х а н і з м** | **класів,** | **об'єкти я к и х** | **в и к о р и с т о в у в а л и с я в** |
| **п р о г р а м а х . О к р і м** | **цього, в** | **мові** | **S i m u l a - 6 7** | **за д о п о м о г о ю** | **пре ­** |
| **фіксів, п о с у т і , забезпечувався** | **м е х а н і з м с п а д к о в о с т і . П о в н і с ­** |
| **тю** | **к о н ц е п ц і я об'єктно - орієнтованої** | **м о в и** | **і** | **в і д п о в і д н и й** | **м е ­** |
| **т о д** | **п р о г р а м у в а н н я** | **б у л и** | **реалізовані** | **в** | **мові** | **S m a l l t a l k** | **8 0 . Д о** |
| **т о г о ж ,** | **реалізовано** | **к о н ц е п ц і ю** | **б у л о** | **н е** | **р о з ш и р е н н я м** | **засобів** |
| **п р о ц е д у р н о г о ( п і д п р о г р а м н о г о )** | **рівня і н к а п с у л я ц і ї** | **м о в и** | **п р о ­** |
| **г р а м у в а н н я ,** | **як** | **це** | **п е р е д б а ч е н о** | **в б і л ь ш о с т і** | **м о в ( C + + , Java,** |
| **С # ,** | **A D A 9 5 ,** | **ОЬегоп),** | **а с т в о р е н н я м** | **с у т о о б ' є к т н о -** |

**о р і є н т о в а н о г о с е р е д о в и щ а п р о г р а м у в а н н я .**

**І**

**М о д у л ь І**

**ОСНОВИ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ**

|  |  |  |
| --- | --- | --- |
| **1 . 1 .** | **Схеми реалізації постпідпрограмного** |  |
|  | **рівня інкапсуляції** |  |  |
| **Н а с ь о г о д н і** | **в і д о м і дві с х е м и** | **реалізації п о с т п і д п р о г р а м н о г о р і ­** |
| **вня і н к а п с у л я ц і ї** | **- к о м п о з и ц і й н а** | **і к л а с и ф і к а ц і й н а . О б и д в і** | **і с н у ю т ь** |
| **о д н о ч а с н о .** | **П е р ш а** | **в и к о р и с т о в у є т ь с я н а м о д у л ь н о м у рівні** | **і н к а п с у ­** |
| **ляції, а** | **д р у г а -** | **на** | **к л а с н о м у . О б и д в і с х е м и б у д у ю т ь с я** | **на основі** |
| **п о н я т т я** | **« в і д н о ш е н н я » . В і д м і н н о с т і виявляються в р і з н і й** | **і н т е р п р е ­** |

**т а ц і ї п о н я т т я « в і д н о ш е н н я » в с х е м а х як о р г а н і з о в у ю ч і с е р е д о в и щ а**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **( б і б л і о т е к ,** | **к о л е к ц і й ) з п р о г р а м н и х** | **к о н с т р у к ц і й ( м о д у л і в** | **або** | **кла ­** |  |
| **сів),** | **т а к** | **і в и к о р и с т о в у ю ч и х . У** | **табл .** | **1.1 наведено** | **р і з н и ц ю** | **в і н т е р ­** |  |
| **п р е т а ц і ї** | **в і д н о с и н** | **у с х е м а х . Т у т п і д** | **м о д у л е м** |  | **розуміється** | **будь - яка** |  |
| **п р о г р а м н а** | **к о н с т р у к ц і я** | **п о с т п і д п р о г р а м н о г о** |  | **рівня** | **і н к а п с у л я ц і ї** |  |
| **( н а п р и к л а д , модуль** | **у** | **мові** | **M o d u l a - 2 , п а к е т у** | **мові** | **A d a , або клас в** |  |
| **C + + ) .** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ***Композиційна*** | ***схема.*** | **П р о г р а м н и й** | **м о д у л ь** | **у** |  | **ц і й с х е м і** | **назива ­** |  |
| **ється** | **м о д у л е м .** | **С е р е д о в и щ е** | **м о д у л і в будується** | **н а** | **і н т е р п р е т а ц і ї** |  |
| **в і д н о с и н** | **м і ж** | **н и м и** | **я к** | **« м і с т и т ь » . Ц е означає,** |  | **я к щ о** | **н о в и й модуль** |  |
| **о п и с у є т ь с я** | **н а основі** | **т о г о ,** | **щ о** | **існує,** | **т о й о г о** | **з а с т о с у в а н н я** | **з д і й с н ю ­** |  |
| **ється** | **ш л я х о м** | **й о г о в к л ю ч е н н я** | **у вміст н о в о г о м о д у л я .** |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ***Таблиця*** | ***1.1*** |  |
|  |  | **Р і з н и ц я** | **в інтерпретації відносин *у* к о м п о з и ц і й н і й** |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | **і** | **класи(** | **і і к а ц і й н і й схемах** |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **Режим використання** |  |  |  | **Інтерпретація відносини** |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | **Т и п схеми** |  |  |  |  |  |
|  |  |  | **відносин** |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | **композиційна** |  |  | **класифікаційна** |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Організація безлічі модулів** |  |  | **«Містить»** |  |  | **«Є»** |  |  |  |  |  |
| **Використання** | **модулів** |  |  |  | **«Використовує»** |  |  | **«Є», «Бути»** |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**З а с т о с у в а н н я о д н о г о м о д у л я і н ш и м в к о м п о з и ц і й н і й с х е м і з д і й ­ с н ю є т ь с я ш л я х о м і н т е р п р е т а ц і ї в і д н о ш е н н я я к « в и к о р и с т о в у є » . В і д н о с и н и реалізуються, н а п р и к л а д , через м е х а н і з м и і м п о р т у -**

**5**

**експорту - (import - export: Модула - 2, Модула - 3), або викорис­ тання - (use: Ада).**

***Класифікаційна с х е м а .* Програмний модуль у цій схемі назива­ється - клас. Середовище класів організовується шляхом інтерпре­ тації відношення «є». Це означає, що створюваний клас с класом типу, що вже існує, тобто на основі якого він створюється. Вико­ ристання класу здійснюється шляхом інтерпретації відношення «бути». Відношення реалізується через механізм спадковості, напри­ клад, «бути» представником класу в програмі.**

**Нарис . 1.1 *а, б* показано приклади інтерпретації відношення для обох схем при організації множини модулів.**

**Автомобіль** **Автомобіль**

**ісіюріниіі**

**засіб**

***а*** ***б***

**Рис. 1.1. Інтерпретація відношення для композиційної *( а )* і класифікаційної (б) схем**

**Унаслідок вказаної інтерпретації відносин між модулями, ви­ користовуючи композиційну схему, в програмі реально існують всі модулі, використовувані в цій програмі (як створені, так і ті, на основі яких вони створювалися). Використовуючи класифікаційну схему, можемо скористатися наявними конструкціями (об'єктами), які є представниками класів, на основі яких вони створювалися в програмі.**

**На класном) рівні дозволяється використання композиційної схеми, оскільки її застосування не суперечить класифікаційній схемі. До того ж клас, в якому є інтерпретації відношення «міс­ тить», називається контейнером. Зворотне використання неможли­ ве, оскільки потребує реалізації додаткових механізмів. Концепту­ ально обидва т і Ш модулів існують глобально і не параметризу-**

**ються перед або в процесі використання. Проте в мовах програ­ мування передбачаються спеціальні механізми, що забезпечують параметризацію модулів до їх використання. У композиційній схе­ мі - це родовий механізм (generic), а в класифікаційній - механізм шаблонів (template). Обидва механізми будуються на основі відкритих підпрограм, і тому параметризація здійснюється до компіляції.**

**На основі класифікаційної схеми в інженерії програмного за­ безпечення реалізуються три види дій, які широко використову­ ються в розробленні програмного забезпечення:**

* **об'єктно-орієнтований аналіз вимог - у результаті його вико­ нання визначається, «що повинне робити» програмне забезпечення;**
* **об'єктно-орієнтоване проектування - у результаті його вико­ нання визначається «яким повинно бути» програмне забезпечення, щоб воно робило те, «що воно повинне робити»;**
* **об'єктно-орієнтоване програмування - у результаті його ви­ конання визначається, «яким є програмне забезпечення» відповідно до визначеного, «яким воно повинно бути».**

**Далі розглядається застосування класифікаційної схеми в про­ грамуванні, тобто об'єктно-орієнтоване програмування. Для цього розроблені спеціальні мови об'єктно-орієнтованого програмування, які зазвичай будуються шляхом розширення підпрограмних (про­ цедурних) мов механізмами об'ектно-орієнтованого програмуван­ ня. Механізми підтримують реалізацію основних принципів об'єктно-орієнтованого програмування. Так реалізовано більшість мов, наприклад, мови Object Pascal, C++, С#, Java, Ada 95, Oberon. Проте є мови програмування, які забезпечують тільки класний рівень інкапсуляції. Це, наприклад, мови SmallTalk-80, Eiffel, які є «чистими» мовами об'єктно-орієнтованого програмування.**

**1.2. Принципи об'єктно-орієнтованого програмування**

**До основних принципів об'єктно-орієнтованого програмування належать такі: абстракція, інкапсуляція, спадковість і поліморфізм. Окрім них можуть застосовуватися додаткові принципи, якими є параметризація, багатократне і повторне використання.**

***Абстракція* - це принцип (метод) наукового дослідження, за­стосування якого забезпечує формування уявлення про властивості**

7

**п р е д м е т а ш л я х о м у я в н о г о в и д а л е н н я й о г о д е т а л е й . П р и к л а д о м**

|  |  |  |
| --- | --- | --- |
| **застосування** | **а б с т р а к ц і ї є** | **б а г а т о р і в н е в и й « п о г л я д » на п р о г р а м и .** |
| *Інкапсуляція* | *-* | **це і н ж е н е р н и й п р и н ц и п , я к и й** | **забезпечує** | **ство ­** |
| **р е н н я о б о л о н к и** | **н а в к о л о** | **« р е ч о в и н » і у т в о р е н н я** | **відповідної** | **к а п с у ­** |

**ли ( п р о г р а м н о ї к о н с т р у к ц і ї ) . В о б ' є к т н о - о р і є н т о в а н о м у п р о г р а м у ­**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ванні к а п с у л а є** | **к л а с о м ,** | **а « р е ч о в и н а » -** | **п р о г р а м н и м и** | **к о н с т р у к ц і я ­** |
| **м и н и ж ч и х рівнів** | **і н к а п с у л я ц і ї ( п і д п р о г р а м и ,** | **о п е р а т о р и , л е к с е м и ) .** |
| *Спадковість* **-** | **це** | **е в о л ю ц і й н и й п р и н ц и п ,** | **я к и й забезпечує** | **пере ­** |
| **д а ч у властивостей** | **від** | **п о п е р е д н и к а д о п о с л і д о в н и к а . К е р у ю ч и с ь** |
| **п р и н ц и п о м с п а д к о в о с т і ,** | **з класів м о ж н а б у д у в а т и ієрархії,** | **які** | **н а з и ­** |
| **ваються** | **к л а с и ф і к а ц і я м и . С а м е т о м у о б ' с к т н о - о р і є н т о в а н а** | **с х е м а щ е** |
| **називається** | **к л а с и ф і к а ц і є ю ,** | **а о б ' є к т н о - о р і є н т о в а н и й** | **п і д х і д** | **- к л а ­** |
| **с и ф і к а ц і й н и м .** |  |  |  |  |  |  |  |  |  |  |
| В **о б ' є к т н о - о р і є н т о в а н о м у п р о г р а м у в а н н і** | **розрізняються два т и ­** |
| **пи с п а д к о в о с т і - о д и н и ч н е ( п р о с т е ) і м н о ж и н н е .** |  |  |  |
| **П р и** | *одиничній спадковості* **н а с т у п н и й** | **клас** | **( п о с л і д о в н и к ) ,** | **я к и й** |
| **називається** | **п о х і д н и м ,** | **є п і д к л а с о м т і л ь к и** | **о д н о г о к л а с у - п о п е р е д -** |
| **н и к а , я к и й** | **називається** | *базовим* **( б а т ь к і в с ь к и м ) (рис .** | 1.2). |  |  |
|  |  |  |  |  | **Базовий клас** |  |  |  |  |  |  |
|  |  | **Похідний клас** |  | **Похідний клас** |  |  |
|  |  |  |  | **Рис.** 1.2. | **Одинична спадковість** |  |  |  |
| **П р и** | *множинній* | *спадковості* | **п о х і д н и й** | **клас** | **м о ж е** | **у с п а д к о в у в а ­** |
| **ти властивості р і з н и х базових класів (рис .** | **1.3).** |  |  |  |
|  |  |  |  |  | **Базовий клас** |  | **Одинична спад­** |  |
|  |  |  |  |  |  |  |  | **ковість** |  |  |  |

**Базовий клас**

**Базовий клас**

**М н о ж и н н а спадковість**

**Похідний клас**

**Рис.** 1.3. **М н о ж и н н а спадковість**

**X**

**С х е м а м н о ж и н н о ї с п а д к о в о с т і у с к л а д н ю є я к п р о ц е с п р о г р а м у ­**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **вання, т а к і р о з у м і н н я** |  | **п р о г р а м ,** | **т о м у вона** | **реалізується не** | **у** | **всіх** |
| **мовах . Н а п р и к л а д ,** | **цю** | **с х е м у** | **не реалізовано** | **в м о в а х** | **O b j e c t** | **Pascal,** |
| **O b e r o n , A d a** 95, | **C # , Java, але реалізовано в м о в і C + + .** |  |  |  |
| **К е р у ю ч и с ь п р и н ц и п о м с п а д к о в о с т і , м о ж н а с т в о р ю в а т и к л а с и ,** |
| **щ о м а ю т ь властивості** | **п о п е р е д н и к і в** | **і р о з ш и р ю ю т ь** | **ї х ш л я х о м** | **д о ­** |
| **давання властивостей .** |  |  |  |  |  |  |  |  |  |
| *Поліморфізм* **-** | **це** | **п р и н ц и п ,** | **я к и й** | **забезпечує п р о г р а м н і й** | **к о н с т ­** |
| **р у к ц і ї (зазвичай** | **п о ч и н а ю ч и з** | **п і д п р о г р а м ) властивість м а т и** | **декіль ­** |
| **к а форм** | **( р е а л і з а ц і й ) .** | **В с т а н о в л е н н я** | **зв'язку** | **м і ж і н т е р ф е й с о м** | **під ­** |
| **п р о г р а м и** | **( в и к л и к** | **п і д п р о г р а м и ) і** | **к о н к р е т н о ю ф о р м о ю м о ж е** | **здійс ­** |
| **н ю в а т и с я в різні п е р і о д и** | **о б р о б к и** | **п р о г р а м и і** | **визначається к о н т е к с ­** |
| **т о м в и к л и к у .** | **П р о с т и м** | **п р о я в о м** | **п о л і м о р ф і з м у є** | **реалізація** | **за** |

**д о п о м о г о ю п і д п р о г р а м в и з н а ч е н и х о п е р а ц і й в м о в а х п р о г р а м у в а н ­ ня і о п е р а т о р а п р и в л а с н е н н я . П е р е в а н т а ж е н н я п і д п р о г р а м т а к о ж є**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **п р о я в о м п о л і м о р ф і з м у .** | **О б и д в а** | **прояви** | **представляють** | **о б м е ж е н и й** |
| **( с т а т и ч н и й )** | **п о л і м о р ф і з м ,** | **о с к і л ь к и** | **зв'язок м і ж** | **п о з н а ч е н н я м ( і н т е р ­** |
| **ф е й с о м ) і** | **р е а л і з а ц і є ю** |  | **п і д п р о г р а м и** | **( ф о р м о ю )** | **встановлюється** | **під** |
| **час к о м п і л я ц і ї . З** | **м е т о ю** | **р о з ш и р е н н я** | **з а с т о с у в а н н я** | **п о л і м о р ф і з м у , в** |
| **о б ' є к т н о - о р і є н т о в а н е** | **п р о г р а м у в а н н я** | **вводиться** |  | **п о н я т т я** | **зв'язу ­** |
| **вання .** | *Зв'язування* | *-* | **це** |  | **м е х а н і з м ,** | **я к и й** | **забезпечує** | **в с т а н о в л е н н я** |
| **зв'язку** | **м і ж** | **і н т е р ф е й с о м** | **п р о г р а м н о ї** | **к о н с т р у к ц і ї** | **і** | **о д н і є ю** | **з** | **форм** |
| **( р е а л і з а ц і й )** | **к о н с т р у к ц і ї ,** | **п е р е д б а ч е н и х** | **для неї.** | **Розглядається** | **два** |
| **види зв'язування** | *- раннє* | *і пізніше.* | **Зв'язування,** | **яке** | **в и к о н у є т ь с я** | **в** |
| **п е р і о д і к о м п і л я ц і ї** | **п р о г р а м и , називається** | **р а н н і м ,** | **а** | **в** | **п е р і о д і** | **в и к о ­** |
| **н а н н я п р о г р а м и** | **- п і з н і м .** | **О ч е в и д н о ,** | **щ о з а** |  | **д о п о м о г о ю** | **пізнього** |
| **з в ' я з у в а н н я** |  | **м о ж н а** | **з м і н и т и** | **зв'язок** |  | **м і ж** | **і н т е р ф е й с о м** | **і** | **ф о р м о ю в** |
| **періоді** | **в и к о н а н н я** | **п р о г р а м и . Т о м у** | **« с п р а в ж н і й » п о л і м о р ф і з м реалі ­** |
| **зується** | **за д о п о м о г о ю** | **м е х а н і з м у** | **п і з н ь о г о з в ' я з у в а н н я** | **і с п е ц і а л ь н и х** |
| **п і д п р о г р а м ,** | **які** |  | **називаютьс я** | *віртуальними* | *підпрограмами* | **( ф у н к ­** |
| **ції, м е т о д и ) . Т а к** | **само,** | **як і** | **р о з ш и р е н н я** | **мов** | **п р о г р а м у в а н н я** | **п р и н ­** |
| **ц и п а м и о б ' є к т н о - о р і є н т о в а н о г о** | **п р о г р а м у в а н н я ,** | **пізнє** | **з в ' я з у в а н н я** |
| **вводиться** | **в** | **більшість** |  | **мов,** | **як** | **р о з ш и р е н н я .** |  | **П р о т е** | **« ч и с т і »** | **м о в и** |
| **о б ' є к т н о - о р і є н т о в а н о г о** | **п р о г р а м у в а н н я ( н а п р и к л а д ,** | **мова** | **S m a l l T a l k -** |
| **8 0 ) п і д т р и м у ю т ь т і л ь к и** | **пізнє зв'язування .** |  |  |  |  |  |  |  |  |  |  |
| *Параметризація -* | **це** |  | **п р и н ц и п ,** | **я к и й** | **забезпечує** | **у п р а в л і н н я** | **з м і ­** |

**с т о м т е к с т у п р о г р а м н о ї к о н с т р у к ц і ї . Ц е з д і й с н ю є т ь с я д в о м а ш л я ­ х а м и . П о - п е р ш е , ш л я х о м з а с т о с у в а н н я с п е ц і а л ь н и х операторів**

**9**

**періоду макрогенерації здійснюється управління формуванням тексту програмної конструкції. По-друге, шляхом використання параметрів виконується передача в програмну конструкцію значень для обробки. Як на модульному, так і на класному рівнях інкапсу­ ляції використання параметризації спрямоване на універсалізацію програмної конструкції. Для параметризації класів використову­ ється механізм відкритих підпрограм, який реалізується за допомо­ гою шаблонів. Ефективність параметризації обмежена можливос­ тями відкритих підпрограм.**

*Багатократне* *використання* **- це принцип, який забезпечує**

**створення типових програмних компонентів, які можна використо­ вувати багато разів. Цим принципом в об'єктно-орієнтованому програмуванні керуються, проектуючи класи. Компоненти багато­ кратного використання в об'єктно-орієнтованому програмуванні «затираються» в бібліотеках класів, шаблонів, контейнерів, які по­ ставляються разом з інтегрованим середовищем програмування.**

*Повторне використання* **- це принцип, який забезпечує вико­**

**ристання існуючого програмного забезпечення в новостворювано­ му. Принцип застосовується для підвищення ефективності розроб­ лення програмного забезпечення. Ним також слід керуватися, роз­ робляючи класи.**

**1.3. Будова класу**

**реалізація ресурсів, оголошених у класі. Таким чином, якщо тіло класу має продовження, то в основній його частині описується інтерфейс, а в продовженні - реалізація ресурсів, що надаються класом. Щ о б пов'язати реалізацію ресурсу з відповідним класом, у продовженні опису класу слід використовувати оператор дозволу контексту («::»).**

**Оскільки клас, як правило, є абстрактним типом даних, то тіло класу повинно містити опис таких компонентів:**

* **значень, що надаються абстрактним типом даних (класом);**
* **операцій (підпрограм), що забезпечують обробку значень абстрактного типу даних (класу).**

**Опис значень визначає множину значень, що обробляються класом, а опис підпрограм - множину операцій, спрямованих на обробку значень, визначених у класі. До того ж, для представлення значень (називаються даними-членами) використовуються описи визначених типів і типів, що визначаються, а для представлення операцій використовуються закриті підпрограми, які називаються функціями-членами або методами.**

**У тілі класу можна розрізняти частини, які позначаються спеці­ альними ключовими словами (наприклад, public, private). Наприк­ лад, у мові C + + можна дати такий приклад опису та використання класу:**

**Основною програмною конструкцією (капсулою) об'єктно-оріснтованого рівня інкапсуляції є клас. Будь-який клас, як програ­ мна конструкція складається з позначення, інтерфейсу і тіла. По ­ значення є ідентифікатором. Інтерфейс описує ресурси, що нада­ ються класом. Тіло складається з конструкцій, що описують реалі­ зацію поданих класом ресурсів. Залежно від конкретної мови програмування, будова класу може відрізнятися. Наприклад, у мо­ вах програмування C++, С# клас складається з позначення і тіла. Позначення класу - це ідентифікатор, який іде за зарезервованим словом «class». Тіло класу міститься між обмежувальними симво­ лами «{ « і »}» та містить як інтерфейс, так і ресурси, що реалізу­ ються класом. Тіло може мати продовження і за обмежуючими символами (мова C++) . У цьому продовженні зазвичай описується**

**class PlaneData public:**

**char \* g e t i d (char\* s); void s e t i d (char\* s); int get\_speed(int i); void set\_speed(int i); void get\_altitude(int i);**

**private:**

**char id[80]; int speed; int altitude;**

**Основна частина тіла (інтерфейс класу)**

**Ю**

**h a r \* g e t \_ i d ( c h a r \* s )**

|  |  |  |  |
| --- | --- | --- | --- |
| **j "** | **Реалізація** | **О с н о в н а ч а с т и н а** |  |
|  |  |  |
| **v o i d** | **set-id ( c h a r \* s n o n e P a u i \*** | **т і л а (реалізація** |  |
|  | ***У*** | **ресурсів)** |  |

**v o i d set \_ a l t i t u d e ( i n t i)**

**m a i n ( )**

|  |  |  |
| --- | --- | --- |
| **і** |  |  |
|  | **О п и с о б ' є к т а** |  |
| **Plane D a t a Plane;** | **Plane т и п у** |  |
| **P l a n e D a t a ( в и к о ­** |  |
|  |  |
|  | **р и с т а н н я к л а с у )** |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Н а** | **основі** | **класів** | **с т в о р ю ю т ь с я** | **к о** | **н с т р у к ц і ї ,** | **які** | **називаються** | **об'­** |
| **є к т а м и** | **( P l a n e в н а в е д е н о м у п р и к л а д і ) . Д о т о г о** | **ж ,** | **п о з н а ч е н н я класу** |
| **використовується** | **як** | **п о з н а ч е н н я** | **т и п у ,** | **а п о з н а ч е н н я** | **об'єкта,** | **як** |
| **п о з н а ч е н н я змінної,** | **п р и** | **в і д о м о м у о п и с і** | **змінної . Т о м у ,** | **об'єкт** | **-** | **це** |
| **п р о г р а м н а к о н с т р у к ц і я ,** | **я к а створюється** | **н а основі класу я к** | **й о г о** |
| **п р и м і р н и к і** | **має в** | **своєму складі** | **всі** | **к о м п о н е н т и** | **класу . О б ' є к т ,** | **що** |
| **представляє** | **клас,** | **д і й с н о** | **і с н у є в** | **п р о г р а м і і м о ж е д і я т и . На основі** |
| **о д н о г о класу** | **м о ж е б у т и с т в о р е н о с к і л ь к и** | **з а в г о д н о об'єктів .** |  |  |

**1.4. Приховування інформації. Роздільна компіляція**

|  |  |  |
| --- | --- | --- |
| **Д л я реалізації** | **ф у н д а м е н т а л ь н о г о п р и н ц и п у п о б у д о в и** | **а б с т р а к ­** |
| **т н и х** | **т и п і в** | **д а н и х -** | **п р и х о в у в а н н я інформації, суть я к о г о** | **полягає в** |
| **т о м у ,** | **щ о б** | **п р и х о в а т и від к о р и с т у в а ч а а б с т р а к т н о г о т и н у д а н и х** |

**деталі реалізації його ресурсів, які представлені в інтерфейсі на клас­ ному рівні інкапсуляції, т а к само, як і на модульному, використову ­ ється роздільна компіляція . Нагадаємо, що суть її полягає в т о м у , що користувачеві абстрактного т и п у д а н и х надається л и ш е інтерфейс у вигляді п о ч а т к о в и х кодів мови програмування, а реалізація ресурсів**

|  |  |
| --- | --- |
| **компілюється роздільно в** | **об'єктний код. Т а к и м ч и н о м , розробляючи** |
| **п р о г р а м н е забезпечення,** | **к о м п і л ю ю т ь с я** | **інтерфейси, а реалізації ре ­** |
| **сурсів підключаються на етапі збору програм (рис.** 1.4). |  |  |  |
| У | **м о в і** | C++ **п р и х о в у в а н н я** | **і н ф о р м а ц і ї з д і й с н ю є т ь с я** | **ш л я х о м** |
| **р о з м і щ е н н я** | **інтерфейсу класу** | **в** | **о с н о в н і й ч а с т и н і тіла, в** | **заголовно ­** |
| **му файлі з р о з ш и р е н н я м** | **«.h», а реалізація ресурсів інтерфейсу** | **роз ­** |
| **м і щ у є т ь с я в** | **п р о д о в ж е н н і** | **опису** | **т і л а у** | **файлі** | **з р о з ш и р е н н я м «.срр»** |
| **після** | **п р и в е д е н н я** | **у формат о б ' є к т н о г о** | **к о д у .** | **П і д к л ю ч е н н я** | **заголов ­** |
| **н о г о** | **файлу** | **( в и к о р и с т а н н я класу** | **в п р о г р а м і )** | **з д і й с н ю є т ь с я** | **як** | **м а к ­** |
| **росу** | **з д о п о м о г о ю** | **о п е р а т о р а періоду к о м п і л я ц і ї «include»** | **(рис .** | 1.4). |
| **Р о з г л я н е м о** | **п р и к л а д** | **в и к о р и с т а н н я** | **роздільної к о м п і л я ц і ї ( м о в а** |
| C++): |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **class** | **T i m e** |  |  |  |  |  |  |  |  |  |
|  | ***/*** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | ***\*** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **public:** |  |  |  |  |  |  |  |  |  |  |
|  |  | **v o i d s e t \_ T i m e ( i n t , int,** | **int);** |  |  |  |  |  |  |
|  |  | **void d i s p l a y \_ T i m e ( ) ;** |  |  |  |  |  |  |  |
|  | **private:** |  |  |  |  |  |  |  |  |  |  |
|  |  | **int** | **hour;** |  |  |  |  |  |  |  |  |  |
|  |  | **int** | **m i n ;** |  |  |  |  |  |  |  |  |  |
|  |  | **int sec;** |  |  |  |  |  |  |  |  |  |  |
|  | }; |  |  |  |  |  |  |  |  |  |  |  |  |
| **Н а в е д е н и й** | **о п и с інтерфейсу** | **класу T i m e** | **п о м і щ а є т ь с я** | **в** | **заголов ­** |
| **н и й** | **файл з** | **п о з н а ч е н н я м** | **«time . h» . Д а л і , в** | **п р о д о в ж е н н і** | **о п и с у** | **т і л а** |
| **о п и с у є м о р е а л і з а ц і ю ресурсів,** | **п о к а з а н и х в** | **інтерфейсі:** |  |  |  |
|  | **v o i d T i m e** | **:: s e t \_ T i m e ( i n t h, int m, int s)** |  |  |  |
|  | { |  |  |  |  |  |  |  |  |  |  |  |  |
|  | } |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **v o i d T i m e** | **:: display** | **T i m e Q** |  |  |  |  |  |  |

{

І

*\2* **13**

**Наведений опис розміщується в початковому файлі Тіте . срр, структура якого є такою:**

**# include «Тіте.Іт»**

**< реалізація ресурсів>**

**Створений файл з позначенням «Тіте.срр» компілюється в об'­ єктний код (див. рис.** 1**.4).**

**Нарешті, описується програма користувача, яка застосовує об'єкти - екземпляри класу Time (файл main у такому прикладі):**

**# include «Time.h» void main ()**

{

**Time my Time;**

**myTime . set = T i m e ( l 3 , 30, 5); my Time . DisplayTimeQ;**

}

**Таким чином, якщо припустити, що на момент часу написання основної програми файл «Тіте.срр» представляє об'єктний код, то бачимо, що реалізація всіх ресурсів прихована від користувача. Винятком є підпрограми-методи, реалізовані як «іпііпе», котрі зав­ жди повинні знаходитись в основному описі тіла, в заголовному файлі. У файлі «Тіте.срр» за допомогою використання оператора «include» перевіряється правильність сигнатур методів в інтерфейсі**

* **реалізації класу.**
	+ **файлі з позначенням «МаіпТіте . срр» за допомогою викори­ стання оператора «include» встановлюється розмір пам'яті, необ­ хідний для створюваного об'єкта класу Time (тільки для значень - даних, що містяться в об'єкті).**

**Зважаючи на необхідність приховування інформації про реалі­ зацію ресурсів класу, і використовуючи для цього роздільну компі­ ляцію, можна для мови С + + вказати таку файлову структуру класу, як показано на рис.** 1**.5.**

**У мові С# реалізація приховування інформації принципово ін­ ша (рис. ! .6).**

**Початковий текст компілюється в керований модуль з розши­ ренням .netmodule (.dll), у якому розміщується реалізація ресурсів.**

**!4** **15**

|  |  |  |
| --- | --- | --- |
| **щ о представлена н а** | **є д и н і й п р о м і ж н і й мові п л а т ф о р м и** | **. Net** |
| **( C o m m o n Intermediate** | **L a n g u a g e ) , а інтерфейс представлено за** | **д о ­** |

**п о м о г о ю м е х а н і з м у м е т а д а н и х . П е р е в а г о ю т а к о ї с т р у к т у р и п р о г р а ­**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **м н о г о** | **м о д у л я** | **є те, що метадані с т в о р ю ю т ь с я** | **не** | **п р о г р а м і с т о м , а** |  |
| **с е р е д о в и щ е м ,** | **а в т о м а т и ч н о о н о в л ю ю т ь с я при** | **п е р е к о м п і л я ц і ї п о ч а ­** |  |
| **т к о в о г о** | **т е к с т у** | **реалізації** | **ресурсів, знаходяться** | **в о д н о м у** | **файлі з** |  |
| **реалізацією . І с н у ю т ь спеціальні і н с т р у м е н т и** | **їх** | **ч и т а н н я і** | **в и к о р и с ­** |  |
| **т а н н я п р о г р а м і с т о м і п р о г р а м н и м и з а с о б а м и .** |  |  |  |  |  |
|  |  |  |  | **інтерфейс класу** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | **Видима** |  |  |
|  |  |  |  | **Заголовний** |  | **частина класу** |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | **файл h** |  |  |  |  |  |  |
| **Зв'язок між** |  |  |  |  |  |  |  |  |
| **частинами** |  |  |  |  |  |  |  |  |
|  |  |  |  | **Реалізація** |  |  |  |  |  |  |
|  |  |  |  | **ресурсів класу #** |  |  |  |  |  |  |
|  |  |  |  |  | **include".h"** |  |  | **Прихована частина** |  |
|  |  |  |  |  |  |  |  |  | **класу** |  |  |
|  |  |  |  | **Похідний файл** |  |  |  |  |  |  |
|  |  |  |  | **(об'єктний код)** |  |  |  |  |  |  |
|  |  |  |  |  | **(•СРР)** |  |  |  |  |  |  |
|  |  |  |  | **Рис.** 1.5. | **Файлова структура класу** |  |  |  |
| **1 . 5 . Область дії об'єкта. Правила доступу** |  |  |  |
|  |  | **до компонентів класу** |  |  |  |  |  |  |
| **Д а н і** | **і** | **м е т о д и - ч л е н и** | **класу м а ю т ь область** | **дії** | **- клас.** | **У класі** |  |
| **й о г о** | **члени** | **м о ж у т ь б у т и** | **д о с т у п н і з а** | **п о з н а ч е н н я м .** | **Д а н і і** | **п і д п р о г ­** |  |
| **р а м и ,** | **що не є** | **ч л е н а м и класу, м а ю т ь** | **область дії - файл . Н а п р и к л а д ,** |  |
| **я к щ о** | **о г о л о ш е н о об'єкт:** |  |  |  |  |  |  |  |
| **C o u n t counter;,** |  |  |  |  |  |  |  |
| **т о поза** | **к л а с о м ч л е н и** | **класу м о ж у т ь б у т и** | **д о с т у п н і т а к и м и с п о ­** |  |
| **с о б а м и :** |  |  |  |  |  |  |  |  |  |  |
| **-** | **через** | **п о з н а ч е н н я об'єкта:** |  |  |  |  |  |  |
|  | **counter . x = І;** |  |  |  |  |  |  |  |
|  | **counter. print();** |  |  |  |  |  |  |  |

**lb**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **-** | **за** | **д о п о м о г о ю** | **вказівної змінної,** | **н а п р и к л а д , о г о л о ш е н а** | **і і н і -** |
| **ціалізована вказівна з м і н н а , щ о т и п і з о в а н а :** |  |  |  |  |  |  |  |
| **count** | **\*** | **pcounter;** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **counter =** | **& c o u n t e r ;** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Т о д і ,** | **члени класу** | **д о с т у п н і т а к и м ч и н о м :** |  |  |  |  |  |  |  |
| **pcounter —» х = 3;** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **pcounter — \* print ( ) ;** |  |  |  |  |  |  |  |  |  |  |  |  |
| **- за д о п о м о г о ю п о с и л а н н я , н а п р и к л а д , о г о л о ш е н о п о с и л а н н я :** |  |
| **& ccounter = counter;.** |  |  |  |  |  |  |  |  |  |  |  |
| **Т о д і ,** | **члени класу** | **д о с т у п н і т а к и м ч и н о м :** |  |  |  |  |  |  |  |
| **ccounter . x = 1 0 ;** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ccountei .print ( ) ;** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Н а** | **область в и д и м о с т і членів** | **к л а с у** | **р о з п о в с ю д ж у є т ь с я** | **дія** | **м е м ­** |
| **б р а н н о г о** | **е ф е к т у .** | **Н а п р и к л а д , я к щ о** | **в** | **класі** | **о п и с а н о** | **д а н е** | **і** | **д а н е** | **з** |
| **т а к и м** | **же** | **п о з н а ч е н н я м** | **в і д о б р а ж е н о** | **у** | **м е т о д і** | **класу,** | **то** | **тіло** | **класу** | **і** |
| **тіло методу** | **діють,** | **я к** | **вкладені** | **б л о к и .** | **П р о т е м е м б р а н н и й** | **ефект** |
| **м о ж н а п о д о л а т и ,** | **в и к о р и с т о в у ю ч и** | **о п е р а ц і ю** | **дозволу** | **для** |  | **д о с т у п у** |
| **до глобальної** | **з м і н н о ї** | **(«::»). Н а п р и к л а д ,** | **у н а с т у п н о м у** | **п р и к л а д і за** |
| **д о п о м о г о ю** | **в к а з а н о г о** |  | **о п е р а т о р а** | **з д і й с н ю є т ь с я д о с т у п** | **д о** |  | **змінної,** |
| **я к а о п и с а н а в** | **ч а с т и н і** | **private:** |  |  |  |  |  |  |  |  |  |  |  |
|  | **public:** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **int v a r p ( )** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | { |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | **int** | **var;** |  |  |  |  |  |  |  |  |  |  |  |

**:: v a r = 1 0 . 0 ;**

}

**private:**

**float var;**

**Н а с т у п н и й п р и к л а д , о п и с у є область дії класу: class C o u n t**

**18**

};

**public:**

**int x;**

**void a d d x ( i n t і)**

**: н й** **ннэг.р J u r o r s o w л н м і т а к О** **в н к Ы г е е**

**х = х + і;**

}

};

**int m a i n ()**

{

**Count counter**

**counter . x = 1;**

**counter . addx ( 1 ) ;**

**C o u n t \*pcounter;**

**pcounter = & c o u n t e r ;**

**pcounter -> x = 3;**

**pcounter - > a d d x ( 1 0 ) ;**

**& c c o u n t e r = counter;**

**ccounter . x = 2;**

**c c o u n t e r . a d d x ( 3 ) ;**

}

**class C o u n t**

**{**

**public int x;**

**public void addx(int i)**

**x = x + i;**

}

**int m a i n ( )**

{

**C o u n t counter;**

**counter = n e w C o u n t Q ;**

**counter . x = 1;**

**c o u n t e r . a d d x ( l ) ;**

}

**19**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **К о ж е н об'єкт** | **к л а с у** | **має** | **власну** | **к о п і ю всіх д а н и х - ч л е н і в** | **класу .** |
| **Бувають в и п а д к и ,** | **к о л и** | **о д н а** | **к о п і я** | **з м і н н о ї п о в и н н а спільно** | **в и к о ­** |

**р и с т о в у в а т и с я в с і м а о б ' є к т а м и к л а с у . Т о д і з а с т о с о в у є т ь с я с т а т и ч ­**

|  |  |  |  |
| --- | --- | --- | --- |
| **н а з м і н н а** | **к л а с у . О ч е в и д н о ,** | **щ о с т а т и ч н і ч л е н и класу** | **і с н у ю т ь т о д і ,** |
| **к о л и н е** | **існує н і я к и х об'єктів** | **ц ь о г о** | **класу .** | **О г о л о ш е н н я с т а т и ч н и х** |
| **членів** | **класу** | **починається з** | **к л ю ч о в о г о** | **слова static.** | **С т а т и ч н і** | **д а н і -** |
| **ч л е н и** | **м а ю т ь** | **область д і ї к л а с** | **і** | **п о в и н н і** | **і н і ц і а л і з у в а т и с я о д и н** | **р а з в** |
| **о б л а с т і** | **д і ї** | **файлу . Д о с т у п** | **д о** | **в і д к р и т и х** | **с т а т и ч н и х** | **ч л е н і в** | **класу** |
| **м о ж л и в и й** | **з а д о п о м о г о ю б у д ь - я к о г о об'єкта** | **класу** | **або з а д о п о м о ­** |
| **г о ю і м е н і** | **к л а с у** | **і б і н а р н о ї о п е р а ц і ї** | **дозволу** | **області** | **дії. Д л я** | **д о с т у ­** |
| **п у д о в і д к р и т о г о** | **с т а т и ч н о г о** | **члена,** | **п е р е д** | **д а н и м - ч л е н о м с т а в и т ь с я** |

**п о з н а ч е н н я к л а с у і п о т і м б і н а р н а о п е р а ц і я д о з в о л у о б л а с т і д і ї ( C + + ) , а б о п о з н а ч е н н я к л а с у і с и м в о л у к р а п к а ( С # ) . Н а п р и к л а д , у мові C + + :**

**class E m p l**

{

**public:**

**setCount(int);**

**static int g e t C o u n t ( ) ; private:**

**static int count;**

};

**int m a i n ( )**

**{**

**E m p l \* p E m p l ;**

**p E m p l = n e w E m p l ;**

**p E m p l > s e t C o u n t ( l O ) ; delete p E m p l ;**

**count = g e t C o u n t ( ) ;**

}

**П р и к л а д у мові C # :**

**class E m p l**

**{**

**public setCount(int)**

{ •••};

**public static int g e t C o u n t ( )**

**{ • • • } ;**

**private static int count = 1 0 ;**

**};**

**int m a i n ( )**

{

**E m p l e m p l ;**

**m p l •-= n e w E m p l Q ; e m p l . s e t C o u n t ( l O ) ;**

**int count = E m p l . g e t C o u n t ( ) ;**

}

**П р о г р а м н і к о н с т р у к ц і ї , я к і м о ж у т ь в и к о р и с т о в у в а т и м е т о д и і**

|  |  |  |
| --- | --- | --- |
| **з н а ч е н н я** | **об'єкта, с т в о р е н о г о н а основі класу, називаються** | **к о м п о ­** |
| **н е н т а м и** | **класу і** | **діляться н а т р и** | **г р у п и :** |  |  |  |  |  |  |  |
| **- р е с у р с и п о х і д н о г о к л а с у ;** |  |  |  |  |  |  |  |  |  |
| **-** | **« д р у з і » класу** | **- п і д п р о г р а м и і** | **к л а с и ,** | **о г о л о ш е н і с п е ц і а л ь н и м** |
| **ч и н о м ;** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **-** | **з в и ч а й н і п і д п р о г р а м и** | **- не «є»,** | **ні р е с у р с а м и** |  | **класу,** | **ні** | **« д р у з я ­** |
| **м и » класу .** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Д л я о б м е ж е н н я** | **д о с т у п у** | **до** | **к о м п о н е н т і в базового класу** | **і забез ­** |
| **п е ч е н н я е ф е к т и в н о ї** | **о р г а н і з а ц і ї** | **взаємодії** | **об'єктів,** | **я к і с т в о р ю ю т ь с я** |
| **на основі** | **класів,** | **д о с т у п до** | **к о м п о н е н т і в** | **базового** | **класу,** | **і о т ж е ,** | **до** |
| **к о м п о н е н т і в об'єкта, с т в о р е н о г о н а** | **основі** | **цього** | **класу,** | **р е г у л ю є т ь ­** |
| **с я з а** | **д о п о м о г о ю с п е ц і а л ь н и х** | **правил . Ц і** | **правила** | **в и з н а ч а ю т ь** | **об ­** |
| **ласть** | **в и д и м о с т і** | **членів** | **класу і** | **т и м с а м и м** | **область** | **дії. П р а в и л а** | **за­** |
| **с т о с о в у ю т ь с я до** | **ч а с т и н** | **класу . У зв'язку з ц и м у тілі класу зазвичай** |
| **передбачається т р и т а к і** | **ч а с т и н и :** |  |  |  |  |  |  |  |  |

* **з а к р и т а ( p r i v a t e ) ;**
* **з а х и щ е н а (protected);**

|  |  |  |
| --- | --- | --- |
| **-** | **в і д к р и т а** | **(public) .** |
| **За у м о в ч а н н я м** | **всі** | **ч л е н и класу є з а к р и т и м и (private) .** |
| **В и з н а ч а ю т ь с я** | **т а к і** | **правила д о с т у п у д о в к а з а н и х ч а с т и н класу** |
| **(рис** | 1.7): |  |  |  |
| **-** | **ч а с т и н и ,** | **о г о л о ш е н і , як з а к р и т і , д о с т у п н і м е т о д а м д а н о г о кла ­** |

**су і д р у з я м класу;**

**20** **21**

**- ч а с т и н и , о г о л о ш е н і , я к з а х и щ е н і , д о с т у п н і м е т о д а м д а н о г о класу м е т о д а м п о х і д н о г о класу і д р у з я м к л а с у ;**

**- ' ч а с т и н и , о г о л о ш е н і , я к в і д к р и т і , д о с т у п н і б у д ь - я к и м к о н с т ­ р у к ц і я м п р о г р а м и .**



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Рис. 1.7. Схема взаємодії конструкцій** |  |  |  |
| **П е р е д б а ч а є т ь с я ,** | **що** | **ч л е н и ,** | **представлені у** | **в і д к р и т і й** | **ч а с т и н і ,** |
| **п о к а з у ю т ь** | **д о с т у п н і** | **р е с у р с и** | **класу** | **( і н т е р ф е й с ) . Ч л е н и** | **з а к р и т о ї** |
| **ч а с т и н и** | **і** | **реалізація** | **членів в і д к р и т и х ч а с т и н н е д о с т у п н і** | **к л і є н т а м** |
| **класу і** | **представляють р е а л і з а ц і ю ресурсів класу .** |  |  |  |
| **П р о т е** | **м е т о д и ц ь о г о** | **класу** | **і** | **« д р у з і » м о ж у т ь** | **м а т и** | **д о с т у п до чле ­** |
| **нів з а к р и т и х** | **ч а с т и н** | **класу, а** |  | **о п и с** | **д а н и х з а к р и т о ї** | **ч а с т и н и** | **в и д н о** |
| **к о р и с т у в а ч е в і** | **класу . Ц е й недолік о р г а н і з а ц і ї класів** | **п о р у ш у є** | **п р и н ­** |
| **ц и п п р и х о в у в а н н я інформації . Н а п р и к л а д :** |  |  |  |  |
| **#inc!ude** | **« t i m e l .h»** |  |  |  |  |  |  |  |  |  |
| **int m a i n** | **( )** |  |  |  |  |  |  |  |  |  |
| { |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | **T i m e t ;** |  |  |  |  |  |  |  |  |  |
|  |  | **/7 П о м и л к а : ' T i m e : :hour'** | **н е д о с т у п н о** |  |  |  |
|  |  | **t h o u r =** | **7:** |  |  |  |  |  |  |  |  |
| } |  |  |  |  |  |  |  |  |  |  |  |  |

**Н а п р и к л а д class Airplanes**

{

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **private:** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | **A i r p l a n e \_ D a t a** | **Database** | **[ 1 0 0 ] ;** |  |  |  |  |
|  |  | }; |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **class AN 1 4 0** | **: Airplanes** |  |  |  |  |  |  |  |  |  |  |  |
|  |  | { |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **void p r o c e s s A N 1 4 0 ( i n t ) ;** |  |  |  |  | **\_** |  |  |  |
|  |  | **i ;** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **void** | **A N 140** | **: : p r o c e s s \_ A N 1 4 0 ( i n t i )** |  |  |  |  |  |  |
|  | { |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **A i r p l a n e J D a t a** | **My D a t a =** | **Database** | **[ i j ;** | **//** | **п о м и л к а д о с т у ­** |
|  | **пу** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | }; |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Д о т о г о ж , я к о ю о г о л о ш е н и й** | **об'єкт** | **M y A N 1 4 0 класу** | **A N 140, т о** |
| **він** | **м і с т и т и м е** | **пам'ять** | **для** | **д а н и х** | **Database** | **ч л е н а базового класу** |
| **Airplanes .** | **Я к б и** |  | **Database** | **був** |  | **ч л е н о м** | **з а х и щ е н о ї** | **ч а с т и н и , т о д о** |
| **нього м о ж н а було б звертатися** | **з п о х і д н о г о класу . О д н а к н а д і й н і ш е ,** |
| **я к щ о п о х і д н и й** | **клас** | **м а н і п у л ю є** | **у с п а д к о в а н и м и** | **к о м п о н е н т а м и ,** |
| **в и к о р и с т о в у ю ч и загальні або з а х и щ е н і ч а с т и н и .** |  |  |  |  |
| **Ч а с т и н и класу** |  | **м о ж у т ь** | **б у т и** | **р о з т а ш о в а н і** | **у б у д ь - я к о м у** | **п о р я д к у ,** |
| **п р о т е ї х** | **р е к о м е н д у є т ь с я р о з м і щ у в а т и** | **так: с п о ч а т к у** | **в і д к р и т і , п о т і м** |
| **з а х и щ е н і і, н а р е ш т і з а к р и т і .** |  |  |  |  |  |  |  |  |  |  |  |  |
| **1**.6. **Приховування інформації. Використання класів** |
| **З а с т о с у в а н н я роздільної** | **к о м п і л я ц і ї ,** | **як** | **в ж е н а г о л о ш у в а л о с я , не** |
| **дозволяє** | **п о в н і с т ю** |  | **п р и х о в а т и від** | **к о р и с т у в а ч а** | **і н ф о р м а ц і ю** | **п р о опис** |
| **реалізації** | **ресурсів** | **класу .** | **В і д к р и т о ю** | **залишається** | **p r i v a t e - ч а с т и н а** |
| **класу . Щ о б п о в н і с т ю п р и х о в а т и** | **о п и с** | **з а к р и т о ї ч а с т и н и** | **в и к о р и с т о ­** |
| **вуються** | **спеціальні** | **п р и й о м и ,** | **о д и н** | **з** | **я к и х** | **реалізується** | **ш л я х о м** |
| **с т в о р е н н я д о д а т к о в о г о** | **с п е ц і а л ь н о г о** | **класу, я к и й називається Proxy .** |
| **С х е м а з а с т о с у в а н н я д о д а т к о в о г о класу** | **п о к а з а н а на рис . 1.8.** |
| **Т а к и м** | **ч и н о м ,** |  | **створюється** |  | **з а г о л о в н и й** | **файл** | **x . h,** | **щ о м і с т и т ь** |
| **клас** | **з о п и с о м** | **ресурсів,** | **що н а д а ю т ь с я** | **к о р и с т у в а ч е в і ,** | **р е а л і з а ц і ю** |
| **я к и х н е о б х і д н о** | **п р и х о в а т и .** | **Д л я** | **п о в н о г о** | **п р и х о в у в а н н я** | **інформації,** |
| **в к л ю ч а ю ч и** | **і p r i v a t e - ч а с т и н у** | **класу** | **в** | **x.h,** | **створюється д р у г и й клас** |

**23**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **i.h,** | **я к и й називається** | **proxy .** | **О с н о в н а ч а с т и н а тіла** | **цього класу в** |  |
| **частині public п о в т о р ю є о п и с** | **методів,** | **які надаються** | **к о р и с т у в а ч е в і ,** |  |
| **а для п р и х о в у в а н н я p r i v a t e - ч а с т и н и** | **класу з** | **x.h** | **в и к о р и с т о в у є т ь с я** |  |
| **член - дане - п о к а ж ч и к ,** | **що т и п і з у є т ь с я** | **к л а с о м ,** | **п р е д с т а в л е н и м у x.h.** |  |
| **Д л я п о в н о г о** | **п р и х о в у в а н н я і н ф о р м а ц і ї у файлі** | **i.h** | **використовується** |  |
| **не** | **о п е р а т о р** | **include, а** | **п о п е р е д ж у ю ч е** | **о г о л о ш е н н я** | **класу виду class** |  |
| **х; де х — п о з н а ч е н н я класу з x.h.** |  |  |  |  |  |  |
|  | **x.h** |  |  | **i.h** | **Ргоху-кл** | **#** | **include "i . h"** |  |
|  |  |  |  |  |  |
|  | **Public** | **Повторює** | **Public:** 43 а с т о с о в Ує |  |  |  |
|  |  |  |  |  |  |
|  | **k Private:** |  |  | **Private:** |  |  |  |  |  |

**Реалізує ре­**

**сурси** **Доступ до private-частини x.h**

**Д о ресурсів**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | **Рис.** 1.8. | **Приховування** | **інформації** |  |  |
| **Реалізація ресурсів класу з i.h описується в** | **п р о д о в ж е н н і о п и с у** |
| **тіла класу** | **і** | **р о з м і щ у є т ь с я у** | **файлі** | **і.срр, я к и й** | **ш л я х о м роздільної** |
| **к о м п і л я ц і ї** | **зберігається** | **в** | **о б ' є к т н о м у** | **к о д і . Д о с т у п** | **до** | **private -** |
| **ч а с т и н и** | **класу з** | **x.h в о п и с і** | **ресурсів** | **в** | **і.срр** | **з д і й с н ю є т ь с я через п о ­** |
| **к а ж ч и к** | **р Х ,** | **щ о** | **типізується .** |  | **У п р о г р а м і к о р и с т у в а ч а** | **класу** | **( m a i n )** |
| **в и к о р и с т о в у є т ь с я т і л ь к и** | **з а г о л о в н и й** | **файл** | **i.h,** | **через** | **це** | **private -** |
| **ч а с т и н а** | **класу** | **з** | **x.h п о в н і с т ю** | **п р и х о в а н а** | **від** | **к о р и с т у в а ч а .** |  |
| **С х е м н а** | **реалізація о п и с а н о г о п р и й о м у м о ж е** | **м а т и т а к и й** | **вигляд** |

* **мові C + + :**
	+ **реалізація р е с у р с н о г о класу: //x . h**

**class Resources**

**{**

**public:**

**24**

**void set V a l u e ( i n t V a l )**

**value = V a l ;**

**nit g e t V a l u e f )**

***і***

**return value;**

**)**

**private;**

**int value;**

};

**реалізація p r o x y класу:**

**// i.h**

**class Resources ::class Proxy**

**{**

**public:**

**v o i d S e t V a l u e ( i n t v a l ) ;**

**int g e t V a l u e ( ) ;**

**private:**

**Resources \* p X ;**

**} ;**

* + **i.cpp**
* **include «i.h»**
* **include «x.h»**

**void Resources :: s e t V a l u e (int v a l )**

**p X - » s e t V a l u e ( V a l ) ;**

**int Resources :: g e t V a l u e ( V**

{

* **return p X - > g e t V a l u e ( ) ;**

**25**

**р е а л і з а ц ія m a i n п і д п р о г р а м и :**

**# include «і.Ii» m a i n ( )**

{

**int n V ;**

**Resources obj ( 1 0 ) ; obj . s e t V a l u e ( 5 ) ;**

**nv = obj . g e t V a l u e ( ) ;**

}

**1 . 7 . Доступ до private-частини класу. «Дружні» підпрограми**

|  |  |  |
| --- | --- | --- |
| **У** | **мові C + + передбачається** | **о б х і д м е х а н і з м у п р и х о в у в а н н я і н ­** |
| **ф о р м а ц і ї** | **ш л я х о м з а с т о с у в а н н я** | **с п е ц і а л ь н и х** | **п і д п р о г р а м , я к і нази ­** |
| **ваються** | **« д р у ж н і м и » ф у н к ц і я м и** | **( п і д п р о г р а м а м и ) .** | **У класі** | **п і д п р о г ­** |
| **р а м и** | **о п и с у ю т ь с я п р о т о т и п а м и** | **з** | **к л ю ч о в и м** | **словом** | **friend,** | **а визна ­** |
| **чаються поза к л а с о м , д о ресурсів я к и х вони м а ю т ь д о с т у п .** |  |
| **Н а п р и к л а д :** |  |  |  |  |  |
|  | **class** | **W i t h F r e n d** |  |  |  |  |  |
|  | **іf** |  |  |  |  |  |  |  |
|  | **f r i e n d** | **v o i d s e t X ( W i t h F r i e n d** | **& , int);** |  |  |  |
|  |  |  | **public:** |  |  |  |  |  |
|  | **private:** |  |  |  |  |  |
|  |  |  | **int x;** |  |  |  |  |  |
|  | **&** |  |  | **•** | **.4** |  |  |  |
|  | **void s e t X ( W i t h F r i e n d & c ,** | **int v a l )** |  |  |  |

**і**

**c.x = val;**

}

**26**

**m a i n ( )**

***і*і**

**W i t h Friend o b j ; s e t X ( o b j , 1 0 ) ;**

}

|  |  |  |  |
| --- | --- | --- | --- |
| **З п р и к л а д у б а ч и м о ,** | **що зв'язок « д р у ж н ь о ї » п і д п р о г р а м и** | **з** | ***pri­*** |
| ***vate* ч а с т и н о ю класу** | **встановлюється** | **через** | **п о з н а ч е н н я об'єкта** | **як** |
| **ф а к т и ч н и й** | **параметр,** | **а** | **о п е р а т о р дозволу** | **в и д и м о с т і (::) у** | **цьому** |
| **в и п а д к у не** | **використовується .** |  |  |  |  |
| **О к р і м п і д п р о г р а м** | **« д р у з я м и » класу** | **м о ж у т ь б у т и і н ш і к л а с и ,** | **щ о** |

**м а ю т ь т а к и й ж е д о с т у п д о p r i v a t e - ч а с т и н и , я к і « д р у ж н і » ф у н к ц і ї :**

**class O n e**

**{**

**public:**

**friend class T w o ;**

***}"***

**М е х а н і з м « д р у ж н о с т і » не має властивостей с и м е т р и ч н о с т і і**

**т р а н з и т и в н о с т і .**

**1 . 8 . Оператор привласнення для об'єктів**

**У мовах C + + і С# передбачається ф о р м а о п е р а т о р а привлас ­**

**нення, яка м о ж е в и к о р и с т о в у в а т и с я для п р и в л а с н е н н я значень од ­**

**н о г о об'єкта і н ш о м у . О б и д в а об'єкти м а ю т ь н а л е ж а т и д о о д н о г о**

**класу .**

**Н а п р и к л а д , у мові С # :**

**class AssStat**

**і**

**і**

**public:**

**private:**

**27**

**int n V a l ; char c V a l ;**

};

**m a i n**

{

**AssStat obj** 1; **AssStat** obj2;

**оЬі2 = оЬ) 1;**

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **У** | **мові С + +** | **н а в е д е н а ф о р м а** | **о п е р а т о р а п р и в л а с н е н н я** | **н а з и в а ­** |
| **ється** | **п о б і т о в и м** | **к о п і ю в а н н я м** | **за** | **у м о в ч а н н я м .** |  |  |
| **Реалізується** | **п р и в л а с н е н н я** | **п о б і т о в и м к о п і ю в а н н я м ,** | **т о м у , я к щ о** |
| **в класі в и к о р и с т о в у є т ь с я д и н а м і ч н и й** | **розподіл пам'яті,** | **то** | **правиль ­** |
| **не в и к о н а н н я о п е р а т о р а п р и в л а с н е н н я** | **не гарантується .** |  |  |

**1.9. Спеціальні методи класу**

**З н а ч е н н я д а н и х - ч л е н і в к л а с у м о ж н а в с т а н о в и т и з а д о п о м о г о ю**

|  |  |  |  |
| --- | --- | --- | --- |
| **м е т о д і в класу після** | **с т в о р е н н я** | **об'єкта . Т а к о ж** | **м о ж н а в и к о н а т и п о ­** |
| **т р і б н і** | **для** | **з н и щ е н н я** | **об'єкта** | **дії.** | **П р о т е** | **в** | **к л а с а х п е р е д б а ч а ю т ь с я** |
| **о п и с і** | **в и к о р и с т а н н я** | **двох т и п і в** | **с п е ц і а л ь н и х** | **методів,** | **які** | **назива ­** |
| **ються** | **« к о н с т р у к т о р и »** | **і « д е с т р у к т о р и » ,** |  | **я к і** | **ч а с т к о в о в и к о н у ю т ь** |
| **вказані** | **дії.** |  |  |  |  |  |  |  |  |  |
| *Конструктори* **-** | **спеціальні** | **м е т о д и ,** | **р е а л і з у ю т ь с я** | **я к** | **з а к р и т і** |
| **п і д п р о г р а м и , с т в о р ю ю т ь е к з е м п л я р и класу . С у т ь с т в о р е н н я** | **об'єкта** |
| **полягає в** | **т о м у , щ о б з а х о п и т и** | **пам'ять д л я** | **об'єкта ( я к** | **правило, ра ­** |

**з о м ) т а і н і ц і а л і з у в а т и п о ч а т к о в и м и з н а ч е н н я м и к о м п о н е н т и - д а н и х**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **об'єкта, що о п и с а н і** | **в класі,** | **на** | **основі** | **я к о г о** | **створюється** | **об'єкт .** |
| **Т а к и м ч и н о м , к о л и** | **н е о б х і д н о** | **с т в о р и т и** | **об'єкт,** | **в и к л и к а є т ь с я** | **к о н с ­** |
| **т р у к т о р класу .** |  |  |  |  |  |  |  |
| **У м о в а х п р о г р а м у в а н н я С + +** | **і** | **С# і с н у ю т ь т а к і правила для о г о ­** |
| **л о ш е н н я к о н с т р у к т о р і в :** |  |  |  |  |  |  |
| **- п о з н а ч е н н я к о н с т р у к т о р а** | **має б у т и** | **т а к и м** | **ж е , як і п о з н а ч е н н я** |

**класу; - к о н с т р у к т о р м о ж е м а т и с п и с о к п а р а м е т р і в .**

**Р о з р і з н я ю т ь т р и т и п и к о н с т р у к т о р і в — з а у м о в ч а н н я м , к о п і ю ­**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вання, д о д а т к о в и й .** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| *Конструктор* | *за умовчанням.* | **У** | **мові** | **C + +** | **він називається** |  | **void -** |
| **к о н с т р у к т о р та** | **реалізується** | **за д о п о м о г о ю** | **закритої підпрограми без** |
| **параметрів або із с п и с к о м параметрів, п р и з н а ч е н и х за у м о в ч а н н я м .** |
| *Конструктор* | *копіювання.* |  | **У** | **мові** | **C + +** | **він** | **називається** | **сору -** |
| **к о н с т р у к т о р** | **та** | **реалізується за д о п о м о г о ю** | **з а к р и т о ї п і д п р о г р а м и з** |
| **о д н и м п а р а м е т р о м .** | **П а р а м е т р** | **є п о с и л а н н я м ,** | **щ о т и п і з о в а н и й** | **кла ­** |
| **с о м . Д о т о г о** | **ж , з н а ч е н н я** | **к о м п о н е н т і в - д а н и х** | **с т в о р ю в а н о г о** | **об'єкта** |
| **к о п і ю ю т ь с я з** | **у ж е с т в о р е н о г о** | **об'єкта класу** | **в к а з а н о г о** | **т и п у** | **в** | **ново ­** |
| **с т в о р ю в а н и й .** | **К о п і ю в а н н я ,** | **яке** | **з д і й с н ю є** | **« ш т а т н и й »** | **к о н с т р у к т о р -** |
| **к о п і ю в а н н я ,** | **називається** | **п о в е р х н е в и м** | **і** | **засноване** | **на** | **о п е р а т о р і** |
| **п р и в л а с н е н н я .** | **С у т ь** | **й о г о** | **р о б о т и** | **полягає** | **в** | **т а к о м у :** | **я к щ о** | **д а н і -** |
| **к о м п о н е н т и об'єкта** | **є в к а з і в н и м и** | **з м і н н и м и** | **та** | **і н і ц і й о в а н і** | **п о с и л а н ­** |
| **н я м и** | **на д и н а м і ч н у** | **пам'ять** | **в** | **г р у п і , а** | **к о п і ю в а н н я з д і й с н и л о с я , то** |
| **к о л и** | **о д и н об'єкт з н и щ у є т ь с я** | **-** | **в** | **результаті** | **звільняється пам'ять,** |
| **з а й н я т а з н а ч е н н я м и ,** | **але** | **д р у г и й** | **об'єкт** | **( к о п і я ) в к а з у в а т и м е** | **н а зві ­** |
| **л ь н е н у д і л я н к у** | **пам'яті, щ о** | **з в и ч а й н о н е п р и п у с т и м о . Щ о б** | **у н и к н у т и** |
| **ц ь о г о** | **застосовується** | **г л и б о к е к о п і ю в а н н я ,** | **к о л и з н а ч е н н я ,** | **р о з м і щ е ­** |
| **ні в д и н а м і ч н і й** | **пам'яті в** | **к у п і ,** | **переносяться** | **на н о в і д і л я н к и** | **пам'яті** |
| **о б ' є к т а , я к и й** | **створюється .** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Н а п р и к л а д , у м о в і C + + :** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **class А** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **{** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **public:** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **( ) ;**
* **( А & ) ;**

|  |  |
| --- | --- |
| **int** | **g e t I n t A ( ) ;** |
| **private:** |  |
| **int** | **\* i i n t A ;** |
| **int** | **\* i n t B ;** |
| **};** |  |

**К о н с т р у к т о р з г л и б о к и м к о п і ю в а н н я м : А :: А ( А & г )**

{

**28** **20**

|  |  |
| --- | --- |
| **intA = n e w** | **int;** |
| **intB = new** | **int;** |
| **\* i n t A = r . g e t l n t A ( ) ;** |
| **\* i n t B = \* ( r . i n t B ) ;** |
| } |  |
| **Наприклад, у мові** C#: |
| { |  |
| **public A ( )** |  |

**l n i t ( ) ;**

}

**public A ( A r )**

**itsA = r . getltsAO: itsB = r.itsB;**

}

|  |  |  |
| --- | --- | --- |
| **public** | **int** | **g e t l t s A O** |
| **{** |  |  |
| **return** | **itsA;** |
| *\* |  |  |
| **private** | **int itsA;** |
| **private** | **int itsB;** |
| **private** | **void I n i t ( )** |
| { |  |  |
| **i t s A =** | **1 ;** |
| **itsB** | **=** 2; |
| } |  |  |
| **};** |  |  |

***Додатковий конструктор* - забезпечує список будь-яких фор­мальних параметрів, за допомогою якого компоненти-дані об'єкта ініціалізувалися значеннями фактичних параметрів.**

**Реалізуючи конструктори класу, використовуємо механізм пе­ ревантаження підпрограм, тому всі конструктори класу мають од­ накове позначення, але різні реалізації.**

**ЗО**

**Якщо в класі не оголошений конструктор, то він створюється, як правило, автоматично у формі конструктора за умовчанням для цього класу.**

**Всі конструктори реалізуються у формі процедур.**

***Деструктори* - це спеціальні методи класу, які ліквідують ек­земпляри класу (об'єкти), коли настає кінець періоду їх існування. Суть ліквідації полягає у звільненні пам'яті зайнятою під об'єкт конструктором у групі.**

У **мовах** C++ **і** С# **для деструкторів передбачені такі правила:**

* **позначення деструктора повинне збігатися з позначенням класу і перед ним має стояти символ «~»;**
* **деструктор у класі передбачається тільки один;**
* **деструктор реалізується у формі процедури без параметрів;**
* **якщо у класі не оголошений деструктор, то він створюється автоматично;**
* **якщо настав кінець області існування об'єкта і деструктор не викликаний вручну, то він викликається автоматично.**

**Окрім звільнення пам'яті, деструктор виконує всі дії, протиле­ жні діям відповідного конструктора.**

**Наприклад, у мові** C++:

|  |  |
| --- | --- |
| **class** | **myclassQ;** |
| **{ public:** |
| **m y c l a s s ( )** |
| { |  |
| **m n l** | **= 0;** |

}

**m y class (myclass & ( o b j ) )**

**{**

**m n l = o b j . m n l ;**

}

**v o i d set m n l ( i n t N e w V a l )**

{

**m n l = N e w V a l ;**

**«дій aMQr(M<t3lft:'.iftt' •flrr.tft'flo х»:налхаои j í h h f n t b t ' j HQO'ofvrcraiio)! int g e í m n l O ;**

**return r n \_ n l ;**

**ЗІ**

}

**~ m y c l a s s ( ) ;**

**private:**

**int m n l ;**

};

**m a i n ( )**

**{**

**int n V a l ;**

**myclass o b j l ; /7 о г о л о ш е н н я о б ' є к т а класу myclass**

**obj.set m \_ n l ( 1 0 ) ;**

**n V a l = g e t r n \_ n l ( ) ;**

**myclass o b j 2 ;**

**myclass (obj 1 ) ;**

**n V a l = o b j 2 . get m \_ n l ( ) ;**

}

**4**

**1 . 1 0 . Виклик конструкторів і деструкторів**

**В и к л и к и к о н с т р у к т о р і в і д е с т р у к т о р і в , я к щ о в о н и явно не зада ­**

**н і , б у д у ю т ь с я неявно ( к о м п і л я т о р о м ) . П о с л і д о в н і с т ь в и к о н а н н я**

**в и к л и к і в з а л е ж и т ь від т о г о , в я к і й п о с л і д о в н о с т і з д і й с н ю ю т ь с я**

**п р о ц е с и входу і виходу в області дії, в я к и х с т в о р ю ю т ь с я о б ' є к т и . У**

**загальному в и п а д к у в и к л и к и д е с т р у к т о р і в в и к о н у ю т ь с я в п о р я д к у ,**

**з в о р о т н о м у в и к л и к а м к о н с т р у к т о р а . П р о т е послідовність в и к л и к і в**

**д е с т р у к т о р і в м о ж е з м і н ю в а т и с я .**

**К о н с т р у к т о р и об'єктів, о г о л о ш е н і в г л о б а л ь н і й області дії, в и ­**

**к л и к а ю т ь с я р а н і ш е , н і ж будь - яка п і д п р о г р а м а ц ь о г о файлу ( в к л ю ­**

**ч а ю ч и m a i n ) т а п о ч и н а є в и к о н у в а т и с я . В і д п о в і д н і д е с т р у к т о р и в и к л и к а ю т ь с я т о д і , к о л и з а в е р ш у є т ь с я п і д п р о г р а м а m a i n .**

|  |  |  |
| --- | --- | --- |
| **К о н с т р у к т о р и** | **а в т о м а т и ч н и х л о к а л ь н и х** | **об'єктів в и к л и к а ю т ь с я** |
| **т о д і , к о л и п р о ц е с** | **в и к о н а н н я досягає м і с ц я** | **о г о л о ш е н н я об'єктів .** |

**В і д п о в і д н і д е с т р у к т о р и в и к л и к а ю т ь с я , к о л и п р о ц е с в и к о н а н н я**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **з а л и ш а є область** | **дії об'єктів** | **( н а п р и к л а д ,** | **блок, у я к о м у ці** | **об'єкти** |
| **о г о л о ш е н і ) . К о н с т р у к т о р и** | **і** | **д е с т р у к т о р и** | **для а в т о м а т и ч н и х** | **об'єктів** |
| **в и к л и к а ю т ь с я к о ж н о г о разу** | **п р и вході і** | **виході з області дії.** |  |  |
| **К о н с т р у к т о р и** | **с т а т и ч н и х** | **л о к а л ь н и х** | **об'єктів в и к л и к а ю т ь с я** | **від­** |
| **разу ж , я к тільки** | **процес в и к о н а н н я досягає м і с ц я , д е об'єкти** | **б у л и** |

**в п е р ш е о г о л о ш е н і . В і д п о в і д н і д е с т р у к т о р и в и к л и к а ю т ь с я , к о л и завершується п і д п р о г р а м а m a i n .**

**32**

**П о с л і д о в н і с т ь в и к л и к і в к о н с т р у к т о р і в і д е с т р у к т о р і в об'єктів**

**к л а с у G A n d D в д е к і л ь к о х о б л а с т я х д і ї п о к а з у є т а к а п р о г р а м а :**

**class G A n d D**

**{**

**public:**

**G A n d D ( i n t ) ; // к о н с т р у к т о р**

**~ G A n d D ( ) ; // д е с т р у к т о р**

**private:**

**I** **nt objectd;**

**char \*message;**

};

**G A n d D :: C A n d D (int o b N , char \* m e s s a g e P )**

''H **'""i> - '** **\* \* " < ™ ™** **v** **\* )**

**object = o b j N :**

**message = messageP;**

**C A n d D : : ~ C A n d D ( )**

}

**C A n d D first ( I « ( г л о б а ч ь н о с т в о р е н и й д о m a i n ) » ;**

**v e i n m a i n ( )**

{

**C A n d D second (2 « ( л о к а л ь н и й а в т о м а т и ч н и й у m a i n ) » ) ;**

**Static C A n d D third ( 3 « ( л о к а л ь н и й с т а т и ч н и й у m a i n ) » ) ;**

**createf);**

**v o i d c r e a t e ( v o i d )**

{

**C A n d D f i f t h ( 5 « ( л о к а л ь н и й а в т о м а т и ч н и й у c r e a t e ) » ) ;**

**static C A n d D sixth ( 6 « ( л о к а л ь н и й с т а т и ч н и й у crea -**

**t e ) » ) ;**

**C A n d D seventh (** 7 **« ( л о к а л ь н и й а в т о м а т и ч н и й у crea -**

**t e ) » ) ;**

**}**

**Результат в и к о н а н н я :**

**created I - глобально с т в о р е н и й до m a i n ;**

**created 2 - л о к а л ь н и й а в т о м а т и ч н и й у m a i n ;**

**created 3 - л о к а л ь н и й с т а т и ч н и й у m a i n ;**

**33**

|  |  |
| --- | --- |
| **created 5** | **- л о к а л ь н и й а в т о м а т и ч н и й у create;** |
| **created** | **6** | **- л о к а л ь н и й с т а т и ч н и й у create;** |
| **created** | **7** | **- л о к а л ь н и й** | **а в т о м а т и ч н и й** | **у create;** |
| **deleted** | **7** | **- л о к а л ь н и й** | **а в т о м а т и ч н и й у create;** |
| **deleted** | **5** | **- л о к а л ь н и й а в т о м а т и ч н и й у create;** |
| **created** | **4** | **- л о к а л ь н и й** | **а в т о м а т и ч н и й** | **у** | **m a i n ;** |
| **deleted** | **4** | **- л о к а л ь н и й** | **а в т о м а т и ч н и й** | **у** | **m a i n ;** |
| **deleted 2 - л о к а л ь н и й** | **а в т о м а т и ч н и й у** | **m a i n ;** |
| **deleted 6 - л о к а л ь н и й** | **с т а т и ч н и й у** | **create;** |
| **deleted** | **3 - л о к а л ь н и й** | **с т а т и ч н и й у** | **m a i n ;** |
| **deleted** | **1** | **- глобально с т в о р е н и й до m a i n .** |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **В и к л и к и к о н с т р у к т о р і в** | **м о в о ю С #** | **м о ж л и в і я к для р о з м і р н и х ,** |
| **т а к** | **і для п о с и л а л ь н и х т и п і в** | **і задаються** | **явно .** |
| ***л*** | **Д л я розмірного т и п у виклик к о н с т р у к т о р а припускає л и ш е ініціа ­** |

**лізацію об'єкта нульовими з н а ч е н н я м и без захоплення пам'яті в к у п і . Д л я п о с и л а л ь н о г о т и п у к о н с т р у к т о р С # п р а ц ю є п о д і б н о д о**

**к о н с т р у к т о р а С + + . 1 й о г о в и к л и к о б ' є к т о м п о с и л а л ь н о г о т и п у обов'язковий перед в и к о р и с т а н н я м .**

**1 . 1 1 . Динамічний розподіл пам'яті**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **М о в и С + + т а С #** | **дозволяють** | **у п р а в л я т и** | **в и д і л е н н я м пам'яті** | **і ї ї** |
| **п о в е р н е н н я м для б у д ь - я к о г о** | **в и з н а ч е н о г о к о р и с т у в а ч е м** | **об'єкта .** |  |
| ***Динамічний розподіл*** | ***пам'яті*** | ***для*** | ***об'єктів у*** | ***мові С+*** | ***+*** | **-** | **виді ­** |

**л е н н я і п о в е р н е н н я в и к о н у ю т ь с я за д о п о м о г о ю о п е р а ц і й n e w і delete . Н а п р и к л а д , т а к е о г о л о ш е н н я і о п е р а т о р :**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **T i m e \* t i m e P t r** |  |  |  |  |  |  |  |
| **t i m e P t r = n e w T i m e ;** |  |  |  |  |  |  |  |
| **де T i m e -** | **п о з н а ч е н н я класу .** |  |  |  |  |  |  |
| **О п е р а ц і я** | **n e w с т в о р ю є** | **об'єкт** | **в і д п о в і д н о г о р о з м і р у** | **для** | **т и п у** |
| **T i m e , а** | **п о т і м** | **в и к л и к а є т ь с я** | **к о н с т р у к т о р** | **за** | **у м о в ч а н н я м** | **для** | **цього** |
| **об'єкта .** | **N e w** | **повертає п о к а ж ч и к** | **т и п у ,** | **щ о** | **в к а з а н и й** | **п р а в о р у ч від** |
| **о п е р а ц і ї n e w ( т о б т о T i m e \* ) .** |  |  |  |  |  |  |  |
| **Щ о б** | **звільнити прост ір ,** | **виділений д и н а м і ч н о для** | **об'єкта,** | **в и к о ­** |
| **ристовується о п е р а ц і я delete в т а к о м у вигляді: D e l e t e timePtr .** |  |
| **П е р е д в и к о н а н н я м п р и в е д е н о г о о п е р а т о р а с п о ч а т к у в и к л и к а ­** |
| **ється д е с т р у к т о р для об'єкта, н а я к и й у к а з у є tumePtr, п о т і м** | **delete** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **звільняє пам'ять, пов'язану з** | **о б ' є к т о м** | **або** | **о п е р а т о р delete** | **с п о ч а т к у** |
| **в и к л и к а є д е с т р у к т о р для об'єкта, а п о т і м звільняє пам'ять .** |  |  |  |
|  | ***Динамічний*** | ***розподіл*** | ***пам'яті*** | ***для*** | ***об'єктів*** | ***у*** | ***мові*** | ***С#*** | **—** | **я к щ о** | **в** |
| **мові C + + я к з а х о п л е н н я , т а к** | **і звільнення д и н а м і ч н о ї пам'яті,** | **з а й н я ­** |
| **тої** | **під об'єкти, в и к о н у ю т ь с я** | **вручну,** | **то в** | **мові С# о п е р а т о р** | **delete** |
| **не** | **в и к о р и с т о в у є т ь с я ,** | **а в и р і ш е н н я** | **завдання звільнення** | **пам'яті « п е ­** |
| **рекладається»** | **на с и с т е м у « з б и р а н н я** | **с м і т т я » ( з б и р а л ь н и к** | **с м і т т я** | **в** |
| **. N e t ) . С у т ь р о б о т и с и с т е м и** | **полягає** | **в т о м у ,** | **щ о** | **вона** | **а в т о м а т и ч н о** |
| **звільняє пам'ять у н е в і д о м и й** | **м о м е н т** | **часу для п о в т о р н о г о** | **в и к о р и ­** |
| **с т а н н я без в т р у ч а н н я** | **п р о г р а м і с т а . Т а к и й** | **п р о ц е с** | **називається** | **неде -** |

**т е р м і н о в а н н м з б и р а н н я м с м і т т я . В і н а к т и в і з у є т ь с я т і л ь к и з п о т р е б и**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **або в** | **с п е ц і а л ь н и х** | **в и п а д к а х** | **і** | **п р и малій з а в а н т а ж е н о с т і процесора .** |
| **Т о м у ,** | **як правило,** | **д е с т р у к т о р** | **класів** | **в и к л и к а є т ь с я** | **не** | **п р и виході з** |
| **області д і ї об'єкта, а з н а ч н о** | **п і з н і ш е ,** | **але обов'язково** | **перед завер ­** |
| **ш е н н я м п р о г р а м и .** |  |  |  |  |  |  |  |  |
| **Ч е р е з н е д е т е р м і н о в а н и й** | **в и к л и к д е с т р у к т о р и н е варто** | **в и к о р и с ­** |
| **т о в у в а т и для в и к о н а н н я д і й ,** | **які п о в и н н і б у т и** | **прив'язані** | **до певної** |
| **т о ч к и** | **п р о г р а м и . Д л я цього** | **в и к о р и с т о в у ю т ь с я** | **спеціальні м е т о д и ,** |
| **щ о в и к л и к а ю т ь с я ,** | **к о л и об'єкт** | **перестає в и к о р и с т о в у в а т и с я .** |
| **Іноді в и н и к а ю т ь ситуації, коли необхідно виконати збирання** |
| **сміття в заданий** | **м о м е н т часу. Д л я управління ц и м у** | **бібліотеці класів** |
| **. Net представлений клас G C** | **і з** | **с т а т и ч н и м м е т о д о м** | **Collect,** | **проте н а** |
| **практиці запускати процес збирання сміття не рекомендується .** |
| **Н а п р и к л а д і п р о г р а м и д е м о н с т р у є т ь с я н е д е т е р м і н о в а н е з б и ­** |
| **р а н н я** | **с м і т т я в . Net:** |  |  |  |  |  |  |  |
|  | **class Destruct** |  |  |  |  |  |  |  |  |
|  | ***і*і** |  |  |  |  |  |  |  |  |  |
|  | **static public int x = 0;** |  |  |  |  |  |
|  | **public** | **Destruct()** |  |  |  |  |  |  |  |
|  | **{** |  |  |  |  |  |  |  |  |  |
|  | **x =** | **x +** | **1 ;** |  |  |  |  |  |  |  |

}

**~ D e s t r u c t ( )**

***і***

**x -- x - 1 ;**

}

34 35

**created 5 - локальний автоматичний у create; created 6 - локальний статичний у create; created 7 - локальний автоматичний у create; deleted 7 - локальний автоматичний у create; deleted 5 - локальний автоматичний у create; created 4 - локальний автоматичний у main; deleted 4 - локальний автоматичний у main; deleted 2 - локальний автоматичний у main; deleted 6 - локальний статичний у create; deleted 3 - локальний статичний у main; deleted 1 - глобально створений до main.**

**Виклики конструкторів мовою С# можливі як для розмірних, так і для посилальних типів і задаються явно.**

**Для розмірного типу виклик конструктора припускає лише ініціа­ лізацію об'єкта нульовими значеннями без захоплення пам'яті в купі.**

**Для посилального типу конструктор С# працює подібно до конструктора С + + . І його виклик об'єктом посилального типу обов'язковий перед використанням.**

|  |  |  |  |
| --- | --- | --- | --- |
| **1 . 1 1 . Динамічний розподіл** | **пам'яті** |  |  |
| **Мови C + + та С#** | **дозволяють управляти** | **виділенням** | **пам'яті і її** |
| **поверненням для будь-якого визначеного користувачем об'єкта.** |
| ***Динамічний розподіл*** | ***пам'яті для*** | ***об'єктів у*** | ***мові* C + +** | **- виді­** |

**лення і повернення виконуються за допомогою операцій new і delete. Наприклад, таке оголошення і оператор:**

**Time \* timePtr timePtr = new Time;**

**де Time - позначення класу.**

**Операція new створює об'єкт відповідного розміру для типу Time, а потім викликається конструктор за умовчанням для цього об'єкта. New повертає покажчик типу, що вказаний праворуч від операції new (тобто Time\*) .**

**Щоб звільнити простір, виділений динамічно для об'єкта, вико­ ристовується операція delete в такому вигляді: Delete timePtr.**

**Перед виконанням приведеного оператора спочатку виклика­ ється деструктор для об'єкта, на який указує tumePtr, потім delete**

**звільняє пам'ять, пов'язану з об'єктом або оператор delete спочатку викликає деструктор для об'єкта, а потім звільняє пам'ять.**

***Динамічний*** ***розподіл пам'яті для об'єктів у мові Сії —* якщо в**

**мові C + + як захоплення, так і звільнення динамічної пам'яті, зайня­ тої під об'єкти, виконуються вручну, то в мові С# оператор delete не використовується, а вирішення завдання звільнення пам'яті «пе­ рекладається» на систему «збирання сміття» (збиральник сміття в**

**.Net). Суть роботи системи полягає в тому, що вона автоматично звільняє пам'ять у невідомий момент часу для повторного викори­ стання без втручання програміста. Такий процес називається неде-термінованим збиранням сміття. Він активізується тільки з потреби або в спеціальних випадках і при малій завантаженості процесора. Тому, як правило, деструктор класів викликається не при виході з області дії об'єкта, а значно пізніше, але обов'язково перед завер­ шенням програми.**

**Через недетермінований виклик деструктори не варто викорис­ товувати для виконання дій, які повинні бути прив'язані до певної точки програми. Для цього використовуються спеціальні методи, що викликаються, коли об'єкт перестає використовуватися.**

**Іноді виникають ситуації, коли необхідно виконати збирання сміття в заданий момент часу. Для управління цим у бібліотеці класів**

**.Net представлений клас GC із статичним методом Collect, проте на практиці запускати процес збирання сміття не рекомендується.**

**На прикладі програми демонструється недетерміноване зби­ рання сміття в .Net:**

**class Destruct *f***

**static public int x =** 0;

**public Destruct()**

{

**x = x +** 1;

}

**~Destruct()**

{

**x = x - 1;**

}

34 35

**public void generator()**

**{**

**Destruct про = new Destruct();**

**// дії, пов'язані із закінченням використання**

**об'єкта**

}

}

**class Program**

{

**static void Main(string[] args)**

{

**int count;**

**Destruct obj = new Destruct();**

**for (count = 1; count < 100000; count = count + 1)**

**{**

**obj.generator();**

**if (count % 100 = 0) Console.WriteLine("Count { 0 } " , Destruct.x);**

}

* **примусове очищення сміття GC.Collect();**
* **очікування виконання деструктора**
* **що всі підлягають видаленню об'єктів GC . WaitForPendi ngFi nalizersQ;**

**Console.WtiteLine(" Final count { 0 } " , Destruct.x);**

}

}

**У процесі виконання програми в певні моменти часу запуска­ тиметься збиральник сміття і видалятиме певну кількість об'єктів, перед цим виконуючи їх деструктори. Після створення кожної сот­ ні об'єктів поточна кількість невидалених об'єктів виводиться на екран. У кінці виконання програми примусово запускається збира­ льник сміття, який видаляє всі об'єкти, окрім першого.**

36

**1.12. Особливості застосування параметрів у методах**

з7™.Т **С + + в** **™ ° Р** **и с т о в** **У** **є т** **ь с я один спосіб передачі параметрів**

**- за значенням. Для того, щоб внесені зміни в значення об'єкта**

нГр**^оад8** **°Г О Ч Є Н Н Я** **'** **В И К** **°** **Р И С Т 0 В** **-** **У Є Т Ь С Я П** **Ч » " « Р ™п у посилання.**

**class Point**

**public:**

**Pointf)**

{

**x = l ; У = 2 ;**

**o p e n \_ x = 10; open\_y = 20;**

**int o p e n x ;**

**int open\_y;**

**private:**

**int x;**

**int y;**

};

**void fun 1 (Point p) f**

**p . openx = 22; // не змінено**

**} 5**

**void** fun2**(Point & p )**

**p . openx = 22; // змінено**

};

**Point point;**

**fun 1 (point); fun2(point);**

37

**При передаванні об'єкта за значенням створюється його копія-об'єкт, для якої викликається конструктор копіювання і деструктор. Якщо в класі об'єкта конструктор копіювання не визначений, мож­ ливий побічний ефект, суть якого полягає в такому. Якщо об'єкі. що передається як параметр, виділяє і використовує динамічну пам'ять у групі, а потім звільняє при її знищенні, то локальна копія - при виході з методу звільнятиме ту ж пам'ять при тому, що аргу­ мент ще існуватиме.**

**На практиці, якщо як параметри застосовуються об'єкти класів, то для економії пам'яті при передачі їх в підпрограми або методи прийнято використовувати посилання. Проте це може викликати небажані зміни об'єкта Щоб уникнути змінювання значення об'єк­ та існує декілька рішень.**

**По-перше, описати посилання на об'єкт з модифікатором const, при цьому операції читання всередині методу дозволені, а зміни забороняються компілятором. Наприклад:**

**void fun3(Point const & р )**

**{**

**int і = p.openx;**

**// p.open\_x - 33; error!**

**По-друге, описати сам об'єкт з модифікатором const, при цьому спроби змінити об'єкт будуть помилковими. Проте компілятори C + + часто не дозволяють ніяких викликів методів таких об'єктів, навіть якщо ці методи тільки використовують значення, не зміню­ ючи їх (на конструктори і деструктори це правило не поширюється, оскільки навіть для об'єкта-константи необхідна ініціалізація його членів-даних і виконання дій до того, як його видалено з пам'яті). Вирішити цю проблему можна, описуючи модифікатором const не сам об'єкт, а методи його класу. Наприклад:**

**int some\_class::member\_fun() consl ) return date member\_value;}**

**Як параметр підпрограми може виступати не проста змінна-об'єкт, а покажчик. До того ж. у методі не можна змінити значення покажчика, а тільки те, на що він посилається. Наприклад:**

**void fun4(Point \*р)**

**р->ореп\_х - 33; // changed**

**р = new Point;** **// not changed**

**Заборонити змінювати члени-дані через покажчик можна, ви­ користовуючи модифікатор const перед типом покажчика. До того** ж, **значення покажчика методу може змінитися. Наприк-лад,**

**void** funS(const **Point \*р, Point \*r)**

**// неконстантний покажчик на константні дані**

**//р->ореп\_х = 44; //error! р = г;**

***} •***

**Якщо ж вказати модифікатор const перед позначенням параме­ тра, сам покажчик не модифікується, а зміниться лише значення, на яке він указує:**

**void Гші6( Point \* const р, Point \*r)**

***?Л* // константний покажчик на неконстантні дані**

**р->ореп х = 44;**

**//р = r;** **//error!**

};

**Можна використовувати обидва модифікатори const, тоді пара­ метр може використовуватися лише для читання даних. Наприк­ лад,**

**void run'7(const Point \* const p, Point \*r)**

**// константний покажчик на константні дані**

{

**//р->ореп\_х = 44; //error! //р = r; //error!**

};

**У мові С# існує два способи передачі параметрів: за значенням і за посиланням. При передачі методу значення аргументу розмір­ ного типу за умовчанням має місце передача за значенням. Для передачі розмірного аргументу за посиланням використовуються**

**39**

**модифікатори ref і out. Модифікатор ref використовується для пе­ редавання значення в метод з обов'язковою попередньою ініціалі­ зацією, а модифікатор out - для повернення значення з методу, при цьому початкова ініціалізація необов'язкова і швидше зайва, але всередині методу параметр повинен набути значення, що поверта­ ється. Обидва модифікатори вказуються як при описі методів перед ідентифікаторами типів формальних параметрів, так і перед аргу­ ментами при виклику. Наприклад:**

**public void sqr(ref int і)**

**int j = 25; obj.sqr(ret'j);**

**public int get\_parts(double n, out double frac)**

**{**

**int whole = (int)n; frac = n - whole; return whole;**

}

**int i; double f;**

**і = obj.get\_parts(55.16, out f);**

**Під час передавання методу значення посилального типу воно передається за значенням, тому, якщо перевизначити посилання, зміни не торкнуться об'єкта в оточенні, методу, що став аргумен­ том. Проте це не стосується змін, що вносяться до значень, на які є посилання (так само як покажчик у мові C++) . Тому всі зміни зна­ чень елементів-даних відіб'ються на об'єкті - фактичному параметрі.**

**Модифікатори ref і out також можуть застосовуватися до поси­ лальних параметрів. У такому разі відбувається передача значення за посиланням. Це зазвичай дозволяє послатися на інший об'єкт і повернути значення в оточення виклику методу:**

**class RefSwap**

**40**

**public void swap(ref RefSwap pari, ref RefSwap par2 )**

{

**RefSwap tmp; tmp = pari; pari = par2; par2 = tmp;**

}

*Фактичні* *параметри за умовчанням.* **Зазвичай, при виклику**

**підпрограм список фактичних параметрів точно відповідає списку формальних параметрів, указаному в описі підпрограм. Проте, якщо в цьому описі для деяких параметрів задати значення за умо­ вчанням, то списки формальних і фактичних параметрів можуть не збігатися за кількістю параметрів (C++) . Тобто, якщо фактичний параметр за умовчанням не вказано у виклику методу, то до нього автоматично передається значення цього параметра за умовчанням. Такі параметри повинні бути останніми в списку методу. Тому, якщо викликається метод з двома і більше параметрами за умов­ чанням і, якщо необхідно пропустити параметр у списку, який не є останнім, то решта всіх параметрів, що знаходяться праворуч, по­ винна також пропускатися. Значення параметрів за умовчанням слід указувати в декларації або описі методу:**

**class Parallelepiped**

**{**

**public:**

**int boxVolume(int length = 1, int width = 1, int height = 1)**

{

**return length \* width \* height;**

}

**void main()**

**{**

**Parallelepiped obj;**

**obj.boxVolume(l, 2, 3); obj.boxVolume(l, 2);**

**obj.boxVolume(l);**

}

**41**

**У мові С# можна здійснювати емуляцію параметрів за умов­ чанням, викликаючи всередині методів і конструкторів їх переза-**

|  |  |
| --- | --- |
| **вантажені аналоги. Наприклад:** |  |
| **class** | **X Y C o o r d** |  |  |
| **{** |  |  |  |  |
| **public XYCoord(int x, int у) { xx = x; yy = y; }** |
| **public XYCoord(int x** | **) : this(x, x) { }** |
| **public int SumCoord(int x, int y) {** | **return x + y; }** |
| **public int SumCoord(int x) { return SumCoord(x, x); }** |
| **private** | **int xx;** |  |  |
| **private** | **int yy;** |  |  |
| } |  |  |  |  |
| *Передача* | *змінної кількості* | *параметрів.* **У** | **мовах C + + і С# є** |

**можливим задавати різне число фактичних параметрів одній і тій же підпрограмі (методу).**

**У мові C + + прикладом такої підпрограми є варіант функції main, що приймає аргументи int arge і char\*argv** [0]. **При цьому argv** [0J **посилається на перше значення в командному рядку - позна­** **чення програми, останні - на параметри** програми, **що задаються в**

**командному рядку. Передача різної кількості параметрів у довільну функцію або метод зводиться до передачі загального числа параме­ трів у вигляді об'єкта цілого типу і їх значень у вигляді масиву значень відповідного типу:**

**int sum(int arge, int argv[])**

**{**

**int i, s =** 0;

**for(i =** 0; **і < arge; і = і +** 1) **s = s + argv[i];**

**return s;**

**};**

**int \*a = new int[3]; sum(l.a); sum(2,a);**

**sum(3, a);**

**У мові C# будь-який метод можна описати так, щоб він міг приймати довільну кількість параметрів. Для цього застосовується спеціальний вид параметра з модифікатором params. Цим парамет­ ром є масив різного числа значень, у тому числі і нульового. У такому методі можна передавати список аргументів будь-якої дов­ жини . Наприклад:**

**static int sum(params int[] nums)**

**{**

**int і s =** 0;

**for (і =** 0; **і < nums.Length; і = і +** 1 )

**s = s + numsfi];**

**return s;**

}

**static void Main(string[] ares)**

**{**

**sum();**

**sum**(1);

**s u t n ( l , 2 , 3);**

}

*Повернення значень з методів.* **Для повернення значень з мето­**

**дів або підпрограм існують такі способи:**

**- повернення через параметр, переданий з використанням по­ силання ( C + + ) або переданий за посиланням з модифікатором out <С#);**

**- повернення результату виконання методу за допомогою оператора return.**

**У мові C + + при поверненні з функції автоматично створюється тимчасовий об'єкт, що зберігає значення, яке повертається. Після повернення цей об'єкт знищується. При цьому можуть виникнути ті /к проблеми з деструктором, що і при передачі об'єкта за значен­ ням. Для вирішення цієї проблеми слід так само явно визначити в** класі **конструктор копіювання або ж перезавантажити оператор**

**привласнення. Наприклад: Point fun8()**

**{**

42

43

**Point tmp;**

**tmp.open x = 44; return tmp;**

};

**Point\* mn9()**

**{**

**return new Point;**

**};**

У **мові** C# **в тих випадках, коли метод повертає створений ло­**

**кально об'єкт, існування цього об'єкта продовжується до тих пір, доки на нього є хоча б одне посилання. Інакше він підпадає під дію збиральника сміття. Таким чином, локальний об'єкт не руйнується після завершення методу, що його створив, а значить - відсутній**

**t** **побічний ефект. Наприклад:**

**public Point fun8()**

{

**Point tmp = new Point();**

**tmp.open \_x = 44;**

**return tmp;**

}

**public Pointfun9()**

**return new Point();**

**\* , , } , . .** **.**

У **мові** C++, **окрім значень простих і вказівних типів операто­**

**ром return, можна повернути посилання на об'єкт. При використан­ ні посилання може виникнути побічний ефект, якщо ініціалізувати це посилання закритим елементом класу. Незважаючи на те. що подібне використання оператора повернення з підпрограми не су­ перечить синтаксису мови, компілятор видає відповідне попере­ дження. Наприклад:**

**public:**

**i n t & bedFun()**

{

**44**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **return** | **x;** |  |
| } |  |  |  |
| **Point** | **point;** |  |  |
| **int &i** | **= point.bedFunQ;** |  |
| **і = 55;** | **// private x is changed** | **-** |
| У **прикладі демонструється неприпустима операція отримання** |
| **доступу із зовні до закритого елемента, класу.** |
| *Статичні* | *члени-методи,* **як і статичні** | **члени-дані, можна вико­** |

**ристовувати, не пов'язуючи їх з певним об'єктом класу, в якому вони описані. До статичних методів, як і до даних, звернення вико­**

|  |  |  |
| --- | --- | --- |
| **нується через позначення класу і** | **операцію** | **дозволу видимості** |
| **( C + + ) або крапку** (С#). **Існують такі** | **особливості** | **використання ста­** |
| **тичних методів:** |  |  |

* **статичний метод не має посилання this;**
* **статичний метод може викликати безпосередньо тільки ста­ тичні методи цього ж класу;**
* **статичний метод може звертатися безпосередньо тільки до статичних даних;**
* **статичний метод може викликати нестатичні методи і звер-**

**іагися до нестатичних даних, використовуючи об'єкт цього класу. І Іаприклад:**

С#

**class S A N D D**

**{**

**private static int si = 10; private int di;**

**private static void s t a t M e t h l Q {•}**

**private void d i n M e t h l ( ) {•}**

**public static void s t a t M e t h 2 ( S A N D D obj)**

{

**int tmp; tmp = si;**

**45**

**t mp = d i ;** **//error!**

**s t a t M e t h l ( ) ;**

**d i n M e t h l ( ) ; //error!**

**tmp = obj . di; o b j d i n M e t h l O ;**

***Ініціалізатори*** ***даних-членів класу.* Усі дані-члени можуть ініці-**

**алізуватися з використанням ініціалізаторів члена класу. Констант­ ні дані - члени і дані, які є посиланнями, обов'язково повинні ініці-алізуватися з використанням ініціалізаторів членів класу. Напри­ клад, у мові C++:**

**class Increment**

**{**

**public:**

**void Increment (int, int); private:**

**int count;**

**const int increment;**

};

**Increment:: Increment (int c, int i): count (c), increment (i)**

{

}

**Визначення конструктора Increment використовує список ініці­ алізаторів членів класу для ініціалізації даних-членів класу. Список ініціалізаторів відділяється від списку параметрів конструктора за допомогою двокрапки (:). Кожний ініціалізатор члена у класі скла­ дається з позначення даного-члена класу, за яким слідує початкове значення в круглих дужках. У наведеному прикладі count ініціалі-зувався значенням параметра з конструктора, a increment ініціалі-зувався значенням «і». Декілька ініціалізаторів членів класу відді­ ляються один від одного комами. Список ініціалізаторів членів класу виконується до виконання тіла конструктора.**

**1.13. Об'єкти як члени класу. Композиція**

**Як члени-дані класу можуть виступати інші об'єкти. Коли об'єкт член створюється, автоматично викликається його конструк­ тор. Об'єкти-члени створюються в тому порядку, в якому вони оголошені і до того, як будуть створені об'єкти класу, який їх міс­ тить. Розглянемо приклад:**

**class Date**

***і*і**

**public:**

**Date (int month = 1, int day = 1, int year = 1900); Date (Date & d ) ;**

**~Date();**

**private:**

**int month: int day; int \_year;**

**Date::Date(int month, int day, int year)**

{

**month = month; day = day;**

**\_ year = year; cout«"\n created ";**

**c o u t « \_ m o n t h « " . " « \_ d a y « " . " « \_ y e a r ;**

**};**

**Date::Date (Date & d )**

{

|  |  |
| --- | --- |
| **month** | **= d . m o n t h ;** |
| **\_ day =** | **d.\_day;** |

**y e a r = d . y e a r ; cout«"\n created copy";**

**c o u t « \_ m o n t h « " . " « \_ d a y « " . " « \_ y e a r ;**

**};**

**47**

**Date:: ~Date()**

**cout«"\n deleted ";**

**c o u t « \_ m o n t h « " . " « d a y « " . " « vear;**

**class Employee**

**{**

**public:**

**Employee (char \*. char \*. Date, Date):**

**- E m p l o y e e ( ) ;**

**private:**

**char \* firstName;**

**char \* lastName;**

**const Date birthDate;**

**const Date hireDate;**

**};**

**Employee::Employee** **(char \* fname, char \***

**dateOfBirth,**

**Date** **dateOfHire): \_ birthDate** **(dateOfBirth),**

**(dateOfHire)**

**{**

**first.Name = fname;**

**lastName = lname;**

**c o u t « " \ n created Employee";**

***i>***

**Employee :: ~Employee()**

{

**cout«"\n deleted Employee";**

**int main()**

{

48

**cout«"employ is created";**

**Employee employ("Vasa", "Pupkin", Date(12, 3, 1977),**

**Date(l, 3,2008));**

**c o u t « " \ n employ has already been created";**

**return 0;**

**}** **^ # ^ , l U R O T , №**

/\*

**employ is created created 1.3.2008 created 12.3.1977 created copy 12.3.1977 created copy 1.3.2008 created Employee deleted 12.3.1977 deleted 1.3.2008**

**employ has already been created deleted Employee**

**deleted 1.3.2008 deleted 12.3.1977**

\*/

**У конструкторі Employee перші два параметри використову­ ються в тілі конструктора для ініціалізації змінних ftName і IName, а останні два параметри передаються за допомогою ініціалізаторів членів класу конструктору класу Date. Ініціалізатори членів класів указують на те, що значення параметрів Employee передаються конструктору об'єктів-членів класу Date. Значення параметра dateOfBirth передається конструктору об'єкта birtDate, а значення параметра - dateOfHire передається конструктору об'єкта hireDate. Клас Date не надає конструктора, який має параметри типу Date, тому компілятор забезпечує кожен клас копією конструктора за умовчанням. Він копіює кожен елемент ініціалізованого об'єкта. Програміст може визначати призначені для користувача копії конструкторів за умовчанням.**

**Видно, що першими створюються об'єкти-фактичні параметри типу Date, до того ж список параметрів обробляється в зворотному порядку. Потім починає створюватися об'єкт Employee таким чи-**

49

н о м : спочатку його об'єкти - членн, потім сам об'єкт - контейнер . П р и виході з програми об'єкт Employee знищується в зворотному по­ рядку - по - перше викликається к о н с т р у к т о р самого об'єкта, а потім

* о н с т р у к т о р и об'єктів-членів.
	+ мові *CM,* оскільки всі об'єкти класів є об'єктами посилальних ти ­ пів, то як для об'єкта-контейнера, так і для об'єктів його членів необ­

хідно викликати відповідні конструктори . Так само, як і в мові С + + при створенні об'єкта-контейнера, в першу чергу викликаються конс ­ труктори об'єктів-членів. Порядок виклику деструкторів у мові С# не розглядається, оскільки він залежить від дій збиральника сміття:

public class Date

{

public DateQ

{

\_ day = 1;

month = 1;

\_ y e a r = 1999;

}

public Date(int month, int day. int year)

{

month = month;

day = day;

\_ year = year;

Console . WriteLine("created { 0 } . { 1 } . { 2 } " , month, day,

\_ year);

}

public Date(Date d)

*і*

***x***

month = d . m o n t h ; day = d.\_day;

\_ year = d.\_year;

Console .WriteLine("created { 0 } . { 1 } . { 2 } " , month, \_ dav,

\_ year);

}

private int month; private int day; private int \_year;

}

public class Employee

**ir**

public Employ ee( string fhame, string Iname, Date

dateOfBirth, Date dateOfHire)

,. {

firstName = fhame; J a s t N a m e = lname;

birthDate = dateOfflirth; hireDate = dateOfHire;

Console . WriteLine("created employ");

!

private string firstName; private siring J a s t N a m e ; private Date birthDate; private Date hireDate;

}

static void Main(string[] args)

**• o » ' , t** ***I*** '. <; ~' i l l **-j'**. i ' *i* *-i i* *•'•* *\'.i****"Y-A*** <-.<!\*'!•"/•>• ***J . -***

***\***

Console . WriteLine("employ is created");

Employee employ = new Employee("Vasa", "Pupkin", new

I uu-( **1** 2 , 3, 1977), new Date( 1. 3. 2008));

Console. WriteLineC'employ has already been created");

*I\**

employ is created

created 12.3.1977

created 1.3.2008

50 51

created employ

employ has already beenereated \*/

У бібліотеках об'єктно-орієнтованих мов програмування існує безліч спеціальних класів-контейнерів. У мові C++ це можуть бути класи стандартної бібліотеки шаблонів Standard Template Library (STL), а в мові C# — це класи колекцій.

Модуль 2

**ПРИНЦИПИ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ**

1. **Спадковість**

Одним з методів управління складністю програмного об'єкта є декомпозиція - ділення об'єкта на такі частини, якими можна ефек­ тивно управляти. Відомо дві методології декомпознції - структурна і об'єктно-орієнтована. Обидві методології побудовано на понятті об'єкта і ставлять за мету отримання структури об'єкта після деко-мио пі ції у вигляді дерева або графи взаємозв'язаних компонентів.

Сірую урна методологія декомпознції будує дерево за принци­ пом розділення цілого на частини («має» або «складається з»). Це відповідає композиційному підходу.

Об'єктно-орієнтована методологія будує дерево декомпознції, яке обов'язково враховує спадковість батьківських властивостей (структури і поведінки) дочірніми частинами дерева в ієрархії. У разі об'єктно-орієнтованого програмування це відповідає відно­ шенню «бути» або «є», а дерево презентується класифікацією.

Таким чином, ***с п а д к о в і с т ь*** - це таке відношення між об'єктами (класами), коли один об'єкт (клас) повторює структуру і поведінку іншого.

Об'єкти декомпозиції, що розташовуються на верхніх рівнях

|  |
| --- |
| класифікаційного дерева, не є конкретними представниками час­ |
| тин | об'єкта, що | декомпозуються і називаються | ***абстрактним^*** |
| ***класами.*** | На їх основі | будуються інші класи і, нарешті, конкретні |
|  | **І** | 52 |  |

об'єкти, які є виконуваними частинами програми і називаються об'єктами або екземплярами класів.

Таким чином, класи у верхньому рівні ієрархії класифікації є базовими або такими, що «поставляють» структури і поведінку для решти всіх класів і об'єктів. Більшість мов програмування мають і ередовища програмування, в яких представлений один базовий клас, а з нього породжується решта всіх класів середовища програ­ мування.

Таким чином, якщо клас розглядати як обслуговувач (сервер), *ТО* у нього може бути два види клієнтів:

* похідні класи;
* конкретні екземпляри класу - об'єкти.

Щоб зрозуміти, як працює або як влаштований похідний клас •бо екземпляр, треба вивчати всі базові (батьківські) класи. Це є і' (ним з головних недоліків об'єктно-орієнтованого програмування.

Спадковість припускає не тільки повторення структури і пове­ дінки батьківських класів, але і їх розширення або обмеження. У Першому випадку додаються нові властивості структури і поведін-і и похідного класу, відмінні від базового, а в другому - забирають-I ч або перерозподіляються існуючі структури і поведінка.

Описуючи спадковість, використовуємо правила спадковості р| .них частин базового класу, які також позначаються ключовими

і.нами public, private, protected. Цими правилами є такі:

всі відкриті і захищені частини базового класу стають відкри-піми для похідних (public);

- всі відкриті і захищені частини базового класу стають закри-ГИМи частинами похідного класу (private);

всі відкриті і захищені частини базового класу стають захи­ щеними частинами похідного класу (protected).

До того ж, не можна успадковувати конструктори, деструктори, підношення друзів.

Незважаючи на те, що при закритій спадковості всі відкриті і **1**1 пінені члени базового класу стають закритими в похідному кла-| можна відновити початковий рівень доступу до одного або декі-**> і** "\ успадкованих членів. Це можна виконати, використовуючиоголошення рівня доступу в похідному класі.

53

**Оголошення рівня доступу має такий синтаксис: <позначення базового класу>::<член класу> Наприклад, у мові C++:**

**class BaseClass**

**{**

**public:**

**int і;**

**v o i d M e t h o d l ( ) { ; protected:**

**i n t j ;**

**void M e t h o d 2 ( ) { ;**

**};**

**class DerivedClass:private BaseClass**

**{**

**public:**

**BaseClass::i;**

**BaseClass::Methodl;**

**protected:**

**BaseClass::Method2;**

**До того ж, можна відновлювати лише відкриті і захищені члени базового класу і не можна знизити або підвищити рівень доступу до члена класу (наприклад, захищені члени відновлюються тільки захищеними). Для кожного відновлюваного члена оголошується свій рівень доступу.**

**У мові С# об'єкти класів ієрархії спадковості можна неявно приводити до типу базового класу:**

**shapel = circle 1;**

**shape 1 .Show(); //помилка!**

**Також об'єкти класів ієрархії спадковості можна явно приводи­ ти до типу успадкованого класу. Проте, по-перше, це повинні бути п о к а ж ч и к и , а по-друге, слід мати на увазі, що у отриманого об'єкта к о р е к т н о ініціалізували лише члени базового класу:**

**0281іаре \*рз1іаре1; •2СігсІе\*рсігс1е1;**

**рсігсіеі =(П2Сігс1е\*)р5Ітаре1;**

**Відома одинична (проста) і множинна спадковість.**

|  |  |
| --- | --- |
| *Одинична спадковість* | *-* **це така спадковість, коли похідний** |
| **клас** | **успадковує властивості лише одного батьківського** | **класу** |
| **(рис.** | **2.1). Одиничній** | **спадковості відповідає деревоподібне** | **пред­** |

**ставлення наочної області.**

**Базовий клас**



|  |  |
| --- | --- |
| **Похідний** | **Похідний** |
| **клас** | **клас** |

**Рис.** 2.1**. Одинична спадковість**

**Порядок виконання конструкторів визначається таким чином: спочатку викликаються вгору (за ієрархією) всі конструктори базо­ вих класів; в кінці викликаються конструктори похідних класів.**

**Наприклад, у мові C + + :**

**class One**

**{**

**int пА, пВ. пС;**

**D2Shape \*pshapel;**

|  |  |
| --- | --- |
| **D2Circle** | **\* p c i r c l e l ;** |
| **pshapel** | **= p c i r c l e l ;** |

**//pshape 1 - >Show();**

* **помилка!**

**54**

**public:**

**One (int nP!; int nP2; int nP3;)**

**{**

**55**

**n A = n P l ;**

**n B = п Р 2 ;**

**п С = п Р З ;**

**class T w o : p u b l i c O n e**

{

**i n t n V ;**

**p u b l i c :**

**T w o ( i n t n P ) : O n e ( n P , n P + 1 0 ; n P )**

***x***

**n V a l = n P ;**

}

**T w o ( i n t n P l , i n t n P 2 ) ;**

***\;***

**T w o : : T w o ( i n t n P l , i n t n P 2 ) ; O n e ( n P l , n P 2 , 2 0 )**

**{**

**n V a l = n P l - n P 2 ;**

}

**Д е с т р у к т о р и в п о х і д н о м у класі в и к л и к а ю т ь с я у п о р я д к у , зво­**

**р о т н о м у в и к л и к а м к о н с т р у к т о р і в .**

**Н а в е д е м о п р и к л а д о п и с у базового класу D 2 S h a p e і двох й о г о**

**с п а д к о є м ц і в D 2 C i r c l e і D 2 S q u a r e в мові С # :**

**e n u m C o l o r { R E D , B L U E . G R E E N . W H I T E . B L A C K ,**

**Y E L L O W , G R E Y } ;**

**class D 2 S h a p e**

**{**

**p r i v a t e i n t p r i v M e m b e r ;**

**p r o t e c t e d v o i d p r o t e c t M e t h o d ( ) { }**

**p u b l i c C o l o r P e n ;**

**p u b l i c C o l o r B r u s h ;**

**p u b l i c v o i d S h o w C o l o r ( )**

**{**

**t**

**p r o t e c t M e t h o d ( ) ;**

**C o n s o l e . W r i t e L i n e ( " P e n a n d B r u s h o f s h a p e are { 0 }**

**I I } " , Pen, B r u s h ) ;**

**}**

}

**class D 2 C i r c l e : D 2 S h a p e**

**{**

**p u b l i c d o u b l e R a d i u s ;**

**p u b l i c d o u b l e S q r ( )**

{

**r e t u r n 3 . 14 \* R a d i u s \* R a d i u s ;**

***i***

**p u b l i c v o i d S h o w ( )**

{

**p r o t e c t M e t h o d ( ) ;**

**C o n s o l e . W r i t e L i n e ( " C i r c l e : r a d i u s = { 0 } , s q r = { 1 } " ,**

**R a d i u s , S q r ( ) ) ;**

**S h o w C o l o r ( ) ;**

}

**class D 2 S q u a r e : D 2 S h a p e**

***i***

***x***

**p u b l i c d o u b l e S i d e ;**

**p u b l i c d o u b l e S q r ( )**

{

**r e t u r n S i d e \* S i d e ;**

}

**p u b l i c v o i d S h o w ( )**

**!**

**C o n s o l e . W r i t e L i n e ( " S q u a r e : s i d e = { 0 } , s q r = { 1 } " , S i d e ,**

**t q r ( ) ) ;**

}

**i**

***i***

57

Класи - спадкоємці, о к р і м своїх власних членів, в к л ю ч а ю т ь від­ криті і захищені члени батьківського класу, його закриті члени при цьому не успадковуються (рис . 2.2). Захищені члени м о ж н а вико ­ ристовувати тільки методам класу - спадкоємця . Цей базовий клас і класи - спадкоємці м о ж н а використовувати для створення об'єктів . Наприклад:

|  |
| --- |
| D2Shape shape 1 = new D2Shape(); |
| shape 1 . ShowColor(); |
| D2Circle circlel | = new D2Circle(); |
| circle 1.Radius = | 45; |
| circlel .pen = Color . GREY; |
| circlel | .brush = | C o l o r . G R E E N : |
| circle | l . ShowQ; |  |

4 //circlel . protectMethodO; error!!!!

D2Square square 1 = new D2Square();

square 1 .Side = 22;

square 1.pen = C o l o r . B L A C K ;

square 1 .brush = Color . Y E L L O W ;

square 1 .Show();

square! . ShowColor();

С т в о р ю ю ч и об'єкт-спадкоємець, викликаємо й о г о к о н с т р у к т о р і

к о н с т р у к т о р базового класу. Я к щ о базовий клас не має к о н с т р у к ­

тора за умовчанням, слід викликати к о н с т р у к т о р з параметрами,

в и к о р и с т о в у ю ч и ключове слово base в оголошенні конструктора

п о х і д н о г о класу:

public D2Shape(Color р, Color b)

{

pen = p; brush = b;

}

public D2Circle(Color p, Color b) : base(p, b)

{

}

**D2Circle**

**D2Circle** *Улґ* **object**

**D2Square у object**

**Рис. 2.2. Структура класів ОгБИаре, 02СігсІе і 028яиаге**

М о ж н а створити багаторівневу ієрархію спадковості . П р и цьо­ му перераховані в описі класу - спадкоємця базові класи успадкову ­ ються безпосередньо, останні успадковуються побічно . У констру ­ кторі спадкоємця можна викликати тільки к о н с т р у к т о р класу, ус ­ падкованого прямо .

Порядок виконання к о н с т р у к т о р і в визначається т а к и м ч и н о м :

|  |  |  |
| --- | --- | --- |
| спочатку викликаються вгору за ієрархією всі | к о н с т р у к | т о р и базо­ |
| вих класів; в к і н ц і викликаються к о н с т р у к т о р и | п о х і д н и х | класів. |

58 59

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **У мові** | **С#** | **в к л а с і - н а щ а д к у** | **м о ж н а** | **п е р е в и з н а ч и т и ( п р и х о в а т и )** |
| **у с п а д к о в а н и й** | **член** | **базового класу ( з м і н н у ,** | **м е т о д ) . Т о д і** | **в** | **об'єктів** |
| **б а т ь к і в с ь к о г о** | **і п о х і д н о г о** | **класів** | **в и к о р и с т о в у в а т и м у т ь с я** | **різні вер ­** |
| **сії членів .** | **Щ о б з а п о б і г т и** | **п о м и л к о в о м у п е р е в и з н а ч е н н ю .** | **к о м п і л я ­** |
| **т о р** | **видає** | **п о п е р е д ж е н н я .** | **Щ о б** | **к о м п і л я т о р** | **н е** | **реагував** | **н а т а к и й** |
| **к о д ,** | **у к л а с і - с п а д к о є м ц е в і** | **в и к о р и с т о в у є т ь с я зарезервоване** | **п о з н а ­** |
| **ч е н н я n e w . Н а п р и к л а д :** |  |  |  |  |  |  |  |  |  |  |
|  | **p u b l i c n e w v o i d S h o w C o l o r Q** |  |  |  |  |  |  |  |  |
|  | **p u b l i c n e w i n t I; .** |  |  |  |  |  |  |  |  |  |  |  |
| **Р а з о м** | **з** | **ц и м , є** | **м о ж л и в і с т ь у** | **м е т о д а х к л а с у - с п а д к о є м ц я** | **зверну ­** |
| **тися** | **д о п р и х о в а н о г о члена . Н а п р и к л а д :** |  |  |  |  |  |  |  |
|  | **І = base . !;** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **B a s e . S h o w C o l o r O ;** |  |  |  |  |  |  |  |  |  |  |
| **ц** | **О б ' є к т и** | **класів** | **ієрархії с п а д к о в о с т і** | **м о ж н а неявно** | **п р и в о д и т и до** |
| **т и п у** | **базового** | **класу .** | **А л е** | **т о д і для** | **ц ь о г о об'єкта** | **б у д у т ь** | **д о с т у п н і** |
| **члени т і л ь к и** | **цього базового класу . Н а п р и к л а д :** |  |  |  |  |  |
|  | **s h a p e 1 = c i r c l e 1;** |  |  |  |  |  |  |  |  |  |  |  |
|  | **s h a p e l . S h o w ( ) ;** | **/ / п о м и л к а !** |  |  |  |  |  |  |  |  |
| **О б ' є к т и** | **класів** | **ієрархії** | **с п а д к о в о с т і** | **м о ж н а** | **явно** | **п р и в о д и т и д о** |
| **т и п у** | **у с п а д к о в а н о г о** | **класу . П р о т е слід м а т и** | **н а увазі,** | **щ о у о т р и м а ­** |
| **н о г о** | **об'єкта** | **к о р е к т н о** | **ініціалізовані** | **л и ш е** | **ч л е н и** | **базового** | **класу .** |
| **Н а п р и к л а д :** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **c i r c l e ! = ( D 2 C i r c l e ) s h a p e l ;** |  |  |  |  |  |  |  |  |  |
| **М н о ж и н н а** | **спадковість** | **- ц е** | **т а к а спадковість,** | **к о л и** | **п о х і д н и й** |
| **клас** | **у с п а д к о в у є о д н о ч а с н о** | **властивості** | **і п о в е д і н к у** | **д в о х** | **і б і л ь ш е** |
| **класів (рис . 2 . 3) .** |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **Базовий клас 1** |  |  |  | **Базовий клас 2** |  |  |

**Похідний клас**

**Рис. 2.3. М н о ж и н н а спадковість**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **П р и к л а д** | **в і д п о в і д н о г о т е к с т у в мові** | **C + +** | **є** | **т а к и м :** |  |  |
|  | **class** | **b a s e l** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **{** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **p u b l i c :** |  |  |  |  |  |  |  |  |  | **-** |  |  |  |
|  |  |  | **i n t** | **1 1 :** |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **b a s e l ( ) { } ;** |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **b a s e l ( i n t i ) { 1 1 = i ; }** |  |  |  |  |  |  |  |  |
|  |  |  | **v o i d M e t h o d 1 ( ) { }** |  |  |  |  |  |  |  |  |
|  | ! ; |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **class** | **b a s e 2** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **p u b l i c :** |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **i n t 12;** |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **b a s e 2 ( ) { } ;** |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **b a s e 2 ( i n t i ) { 12** | **= i;}** |  |  |  |  |  |  |  |  |
|  |  |  | **v o i d M e t h o d 2 ( ) { }** |  |  |  |  |  |  |  |  |
|  | }; |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **class** | **d e r i v e d :** | **p u b l i c b a s e l ,** | **p u b l i c base2** |  |  |  |  |  |  |
|  | { |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **p u b l i c :** |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **i n t 13;** |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **d e r i v e d ( ) { }** |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **d e r i v e d ( i n t i ) : b a s e l ( i + 1 0 ) , b a s e 2 ( i + 2 0 )** |  |  |  |  |
|  |  |  |  |  | **{13 = i;}** |  |  |  |  |  |  |  |  |  |
|  |  |  | **v o i d M e t h o d 3 ( ) { } ;** |  |  |  |  |  |  |  |  |
|  | **};** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **У б і л ь ш о с т і** | **реалізацій** | **мов** | **п р о г р а м у в а н н я** | **м н о ж и н н а** | **с п а д к о ­** |
| **вість** | **часто о б м е ж у є т ь с я** | **д в о м а** | **б а з о в и м и к л а с а м и** | **або** | **взагалі не** |
| **реалізується,** | **н а п р и к л а д ,** | **у** | **м о в і** | **С # . Це** | **пов'язано** | **не** | **з т р у д н о щ а м и** |
| **реалізації,** | **а з** | **п р о б л е м а м и** | **з а с т о с у в а н н я** | **п о х і д н и х** | **класів,** | **побудова ­** |
| **н и х на** | **м н о ж и н н і й** | **с п а д к о в о с т і** | **( п о г і р ш у є т ь с я зрозумілість** | **і н а д і й ­** |
| **ність** | **п р о г р а м ) .** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Розглядаються** | **т р и** | **с х е м и** | **м н о ж и н н о ї** | **с п а д к о в о с т і** | **н а основі** |
| **п р я м и х** | **базових** | **класів** | **(рис . 2 . 4 ) , н е п р я м и х** | **р о з д і л е н и х** | **базових** |

**класів (рис . 2 . 5 ) , н е п р я м о г о в і р т у а л ь н о г о базового класу (рис . 2 . 6 ) . 61**

**Class A** **Class В**

**Class C**

**Рис. 2.4. М н о ж и н н а спадковість на основі прямих базових класів**

**Class A** **class А**



|  |  |  |
| --- | --- | --- |
| **Class В** | **Class** С |  |
|  |  |
| **Class D** |  |
| **Рис. 2.5. М н о ж и н н а спадковість** | **на основі непрямих базових класів** |  |
| **Class** | **А** |  |
| **Class В** | **Class** С |  |
|  |  |

**Class U**

**Рис. 2.6. М н о ж и н н а спадковість на основі непрямого віртуального базового класу**

**class derived 1: public basel**

**{**

**public:**

**int 14;**

**void M e t h o d 4 ( ) { ;**

};

**class derived2: public base!**

{

**public:**

**int 15;**

**void M e t h o d 5 ( ) { } ;**

**};**

**class derived12:public d e r i v e d l , public derived2**

{

**public:**

**int sum;**

**void Method6()**

{

**sum = 14 + 15;**

**//sum = 11; error**

**sum = derivedl ::I1 + derived2::I I;**

!;

**};**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| У | **цьому** | **п р и к л а д і** | **клас** | *basel* | *-* **н е п р я м и й** | **р о з д і л е н и й базовий** |
| **клас.** | **К л а с** *derived* | *12* | **має** | **дві** | **к о п і ї** | **членів класу** | *basel,* **що** | **входять у** |
| **класи** | *derivedl* | **і** | *derived2.* | **За** | **т а к о ї м н о ж и н н о ї** | **с п а д к о в о с т і** | **м о ж е** |
| **в и н и к н у т и н е о д н о з н а ч н і с т ь** | **-** | **в** | **у с п а д к о в а н о м у класі** | *derivedl!* | **не ­** |
| **м о ж л и в о в и з н а ч и т и ,** |  | **до** | **м е т о д у** | *II,* | **я к о г о з** | **класів** | *derivedl* | **або** |
| *derivedl* | **відбувається** | **звернення .** | **Т о м у** | **перед** | **п о з н а ч е н н я м члена** |
| **обов'язково в и к о р и с т о в у є т ь с я** | **п о з н а ч е н н я класу .** |  |  |
| **Н а п р и к л а д , у** | **мові** | **C + + :** |  |  |  |  |  |  |  |

**class derived3: virtual public basel**

{

**public:**

**int 14; derived3():basel(3){};**

**void M e t h o d 4 ( ) { ;**

**}:**

**ciass derivecW: virtual public basel**

{

**public:**

**int 15;**

**deri ved4(): base 1 (4) { } ; void M e t h o d 5 ( ) { ;**

**62** **63**

};

**class derived34:public derived3, public derived4**

{

**public:**

**int sum;**

**v o i d Method6()**

**{**

**sum = 14 + 15;**

**sum = I I ;**

**};**

**};**

**derived34 d34;** **/ / I I is not initialized**

**derived3 d 3 ; / / I l = 3**

**У цьому прикладі клас *b a s e l -* непрямий віртуальний базовий**

**клас. Якщо успадковуються одночасно обидва класи *d e r i v e d !* і**

***d e r i v e d ! ,* то члени їх загального базового віртуального класу *b a s e l***

**не дублюються в класі-спадкоємцеві *derivedl2* і не ініціалізувалися**

**в конструкторі, а отже, - відсутня неоднозначність при зверненні до члена //. Тому, перед позначенням такого члена позначення класу вже не використовується (тут class А - непрямий віртуальний базовий клас).**

**Можливе комбінування другої і третьої схем, а саме викорис­ тання віртуальних і невіртуальних базових класів (рис. 2.7).**

Class А

**Рис.** 2.7. **Комбінована схема спадковості**

**Тут class А використовується, як непрямий віртуальний базо­ вий клас і непрямий базовий клас.**

**Опис і виклик конструкторів при множинній спадковості здійс­ нюються так само як і при одиночному. Виклик деструкторів здій­ снюється в такому ж порядку, як і при одиночній спадковості, тоб­ то в зворотному викликам конструкторів.**

**Так само, як і при одиничній спадковості (Гонструктору базово­ го класу повинні передаватися фактичні параметри опису, які вста­ новлюються в похідному класі. Розглянемо приклад множинної спадковості. Припустимо, що є така схема (рис. 2.8):**

**circle** **table**

**Рис.** 2.8. **Схема множинної спадковості**

**Тоді можна описати такі класи у мові C++:**

**Class circle**

**{**

**float radius; public: circle(float e);**

**{radius r;**

}

**float area()**

**{**

**return radius\* radius\*3.14;**

}

};

**class table f**

**float hight; public:**

**table(float h)**

{

***ї*** 65

**night = h;**

}

**float hight()**

{

**return hight;**

{

**};**

**class RoundTable public: circle; public: table;**

**{**

**int colour; public:**

**RoundTable(float r, float h, int з): Circle, table(h)**

**{**

**colour=c**

|  |  |  |  |
| --- | --- | --- | --- |
| *'* | **>•** | **!** |  |
|  |  |

**void inain()**

**{**

**Round Table MyTable(15 . 0;10 . 0** ; 2 ); **float h**

**r;**

**int з;**

**h = M y t a b l e hight(); r = Mytable . area();**

**з = colour. з():**

**};**

**Множинної спадковості слід уникати шляхом ретельнішого проектування класифікації.**

**2.2. Поліморфізм**

*Поліморфізм* **забезпечує програмній конструкції властивість**

**володіти декількома формами (реалізаціями).**

**Передбачається наперед визначене і визначуване застосування поліморфізму. Простим прикладом наперед визначеного полімор­ фізму є реалізація арифметичних операцій і оператора привласнен­ ня в мовах програмування. Прикладом визначуваного поліморфіз­ му є перевантаження підпрограм.**

**Незалежно від способу застосування поліморфізм будується на основі механізму зв'язування виклику підпрограми з однією з її конкретних форм (реалізація, тіло). Реалізація цього механізму здійснюється, як вже зазначалося, двома шляхами, тому розрізня­ ють раннє і пізнє зв'язування. \_**

**Раннє зв'язування здійснюється під час компіляції. Встановле­ ний зв'язок між викликом конструкції і його формою не змінюється під час виконання програми. Застосування механізму раннього зв'язування веде до створення надійних, але недостатньо гнучких програм. Раннє зв'язування ефективне, оскільки для виконання програми потрібно менше часу.**

**Пізнє зв'язування здійснюється під час виконання програми, тому зв'язок між викликом програмної конструкції і її реалізацією може змінюватися. Використання пізнього зв'язування веде до побудови менш ефективних за часом виконання, але гнучкіших програм.**

**Бажано використовувати раннє зв'язування, а пізнє використо­ вувати з потреби. У «суто» об'єктно-орієнтованих мовах, напри­ клад, SmallTalk, використовується тільки механізм пізнього зв'я­ зування.**

**Таким чином, для реалізації принципу поліморфізму потрібно визначити програмну конструкцію, передбачити для неї декілька форм реалізації і обрати механізм зв'язування.**

*Раннє* *зв'язування або перевантаження.* **У програмуванні, ран­**

**нє зв'язування може застосовуватися як для закритих підпрограм, так і для операцій і операторів. У мовах процедурного програму­ вання поліморфізм визначається для наперед визначених конструк­ цій. У об'єктно-орієнтованих мовах перевантаження може визнача­ тися як для підпрограм, так і операцій (операторів). Вибір форми програмної конструкції при ранньому зв'язуванні визначається результатом аналізу інтерфейсу (списку параметрів програмної конструкції).**

*Перевантаження* *підпрограм.* **Реалізація підпрограми вибира­**

**ється під час компіляції. Наприклад, у мові C + + :**

**class abs\_val**

**public:**

67

**i n t n u m b e r í i n t v a l )**

**її t** ***гШто~r* н і ш г ю ч ш і я іл ш і яИНИ«%Є«'8Г< м .**

**r e t u m ( a b s ( v a l ) ) ;**

}

**d o u b l e n u m b e r ( d o u b l e ( v a l ) )**

{

**r e t u r n ( f a b s ( v a l l ) ) ;**

}

**};**

**m a i n ( )**

{

**i n t м у V a l I n t ;**

**d o u b l e m y V a l D o u b l e ;**

**a b s v a l m y a b s ;**

**j** **my V a l l n t = m y \_ a b s . n u m b e r ( - 1 0 )**

**m y V a l D o u b l e = m y \_ a b c . n u m b e r ( - 1 0 . 0 ) ;**

*\*vnu****&ám\*mb*** *в* ***мп*****«****У****г « ' « г** **ж д е ц и т у \* о к ж < і » м <**

**У** **класах, о к р і м к о н с т р у к т о р і в , м о ж у т ь б у т и п е р е в а н т а ж е н і**

**будь - які п і д п р о г р а м и - м е т о д и . Д е с т р у к т о р и н е п е р е в а н т а ж у ю т ь с я ,**

**о с к і л ь к и , н е м а ю ч и п а р а м е т р і в , в о н и н е м о ж у т ь в и к о р и с т о в у в а т и с я в м е х а н і з м а х п е р е в а н т а ж е н н я .**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Перевантаження* |  | *операцій* | *і* | *операторів.* | **У** | **мовах** | **п р о г р а м у в а н ­** |
| **ня п е р е в а н т а ж е н н я ,** | **я к е засноване на р а н н ь о м у зв'язуванні,** | **як** | **пра ­** |
| **вило,** | **в и з н а ч е н е** | **для** | **б і л ь ш о с т і о п е р а ц і й** | **і** | **операторів .** | **Н а п р и к л а д ,** |
| **п о з н а ч е н н я а р и ф м е т и ч н и х** | **о п е р а ц і й** | **без** | **п е р е в и з н а ч е н н я** | **в и к о р и с ­** |
| **товується для** | **р і з н и х** | **н а п е р е д в и з н а ч е н и х** | **т и п і в .** |  |  |  |  |  |
| **У** | **м о в а х C + +** | **і** | **С# на м е х а н і з м і** | **р а н н ь о г о** | **зв'язування** | **будується** |
| **п е р е в а н т а ж е н н я** | **о п е р а ц і й** | **і** | **операторів .** | **Ц е й** | **т и п** | **п е р е в а н т а ж е н н я** |
| **вводиться в м о в у** | **і** | **в и к о р и с т о в у є т ь с я** | **з** | **м е т о ю с к о р о ч е н н я** | **п о з н а ­** |
| **чень п і д п р о г р а м ,** | **що** | **реалізовують відомі** | **в** | **мові о п е р а ц і ї** | **і** | **о п е р а т о ­** |
| **р и для т и п і в .** | **Н а п р и к л а д ,** | **в и к о р и с т а н н я** | **п о з н а ч е н н я о п е р а ц і ї** | **скла ­** |
| **д а н н я** | **« + » для** | **п о з н а ч е н н я** | **о п е р а ц і ї** | **н а д** | **з н а ч е н н я м и і н д е к с о в а н о г о** |
| **т и п у .** | **Я к п о з н а ч е н н я** | **о п е р а ц і й або операторів** | **беруться** | **п о з н а ч е н н я** |
| **в и з н а ч е н и х у** | **мові** | **о п е р а ц і й** | **або операторів ( + , -** | **/, »,** | **«** | **= ) .** | **Біль­** |
| **ш і с т ь о п е р а ц і й м о ж н а п е р е з а в а н т а ж у в а т и .** |  |  |  |  |  |  |  |
| **З** | **погляду** | **с т и л і с т и к и п р о г р а м у в а н н я ,** | **п р и** | **описі** | **п е р е з а в а н т а ж е -** |
| **н и х к о н с т р у к ц і й** | **слід** | **враховувати в и м о г и** | **з р о з у м і л о с т і п р о г р а м . Ц е** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **означає,** | **щ о п о з н а ч е н н я** | **п е р е з а в а н т а ж е н и х** | **о п е р а ц і й** | **або** | **операторів** |
| **слід в и б и р а т и** | **так,** | **щ о б** | **вони відповідали с е м а н т и ц і** | **ф о р м и п і д п р о г ­** |
| **р а м и .** |  |  |  |  |  |  |  |  |  |  |
| **С т а р ш и н с т в о і а с о ц і а т и в н і с т ь о п е р а ц і й не м о ж у т ь б у т и з м і н е н і** |
| **п е р е в а н т а ж е н н я м . С т в о р ю в а т и** | **нові** | **п о з н а ч е н н я о п е р а ц і й** | **н е м о ж н а .** |
| *Перевантаження* | *підпрограм-членів* | *класів.* | **Ц е й т и п** | **п е р е в а н ­** |
| **т а ж е н н я** | **в ж е** | **д е м о н с т р у в а в с я** | **н а** | **п р и к л а д і к о н с т р у к т о р і в** | **класу .** |
| **О к р і м к о н с т р у к т о р і в , у** | **класі** | **м о ж у т ь б у т и** | **п е р е в а н т а ж е н і** | **будь - які** |

**п і д п р о г р а м и . Д е с т р у к т о р и н е п е р е в а н т а ж у ю т ь с я , о с к і л ь к и н е м а ю ­ ч и п а р а м е т р і в , в о н и н е м о ж у т ь в и к о р и с т о в у в а т и м е х а н і з м п е р е в а н ­ т а ж е н н я .**

**П о з н а ч е н н я п і д п р о г р а м и п е р е в а н т а ж е н о ю в класі діє л и ш е в цьому класі . Н а п р и к л а д , т а к и й к л а с у мові C + + :**

**class Е х**

**{**

**p u b l i c :**

**v o i d V a l u e ( i n t v )**

{

**v a l u e = v;**

}

**i n t V a l u e ( )**

{

**r e t u r n V a l u e ;**

}

**p r i v a t e :**

**i n t v a l u e ;**

} ;

**v o i d m a i n ( )**

{

**i n t n l ; E x o b j ;**

**o b j . v a l u e ( 1 0 ) ; nl = o b j . v a l u e ( ) ;**

**68** 69

**Механізм перевантаження не успадковується в ієрархії класів. Наприклад, у мові C + + :**

**class Ех** 1

і

**public:**

**int add(int і)**

{

**r e t u r n ( i + l ) ;**

}

***h***

{**Class** Ex2: **public Ex** 1

**public:**

**int add(float і)**

{

**return(i**+10);

}

};

**void main ()**

{

**int**Ex2**n l ; obj;**

**nl = obj.add**(2);

}

**У цьому прикладі буде викликана підпрограма add, яка описана в похідному класі** Ех2. **Оскільки мова C + + не точно типізована, то**

**після приведення типів відбудеться виклик підпрограми. У мовах, що точно типізуються, буде видано повідомлення про помилку невідповідності типів. Застосовуючи оператор розширення види­ мості («::»), можна виконати підпрограму, описану в базовому кла­ сі, наприклад, E x l : : add**(2).

***Перевантаження*** ***бінарних*** ***операторів*** ***і*** ***операцій.*** **Для бінарних**

**операторів список параметрів перезавантаженого оператора міс­ тить тільки один параметр. Значення цього параметра набуде об'єкт, що розташований праворуч від позначення оператора. Зна­ чення об'єкта, позначення якого розташоване зліва, задається неяв­ но. На це вказує тип значення, що повертається.**

**Якщо операція не повинна змінити значення результату вико­ нання оператора, то для розміщення результатів використовуються іюкальні об'єкти типу відповідного класу. Є оператори, які повер­ тають значення, відмінне від типу класу (наприклад, операції від­ ношення і логічні). \_**

**Для позначення перевантаження застосовують ключове слово operator. Наприклад, перевантаження в мові C++:**

**Class coord**

{

**public:**

**coord()**

{ 0;

* 1. **=** 0;

**У =**

}

**coord (int i, int j)**

**x = i; y = j ;**

**void get \_ xy(int & i, int & j)**

***<***

**i = x;**

}

**coord operator + (coord obj); private:**

**int x,y;**

**i •**

**coord coord :: operator + (coord obj)**

{

**coord** **temp;**

**temp.x -- x + obj.x; temp.y = y + obj.y;**

**return temp;**

}

70 71

|  |  |  |
| --- | --- | --- |
| **v o i d** | **main** | **()** |
| { |  |  |
|  | **coord** | **obOl(IO . lO);** |
|  | **coord** | **ob02(5,3);** |
|  | **coord** | **ob03 ( ) ;** |
| **I** | **nt X, y;** |
|  | **ob03 = ob01 + o b 0 2** |
|  | **getxyix; y)** |

**Я к щ о необхідно, щоб лівий операнд у бінарній операції був ти­ пу, відмінного від класу, у якому описується перезавантаженні) оператор, то слід використовувати перевантажені оператори як «дружні» підпрограми. Наприклад, у такому тексті в мові** *С++:*

**4**

**Class А**

**{**

**public:**

**A operator + (А); private:**

**int val;**

***\.***

**А А :: operator + (А т \_ х )**

**{**

* **А 1 ( ) ;**
* **А2(3);**

**А1 = А2 + «xyz»;**

**А1 = «xyz» + А2:**

}

**Тут, в останньому операторі привласнення компілятор виявить синтаксичну помилку, оскільки тип першого операнда не є типом класу А. При цьому перетворення за умовчанням не дозволяється. Якщо перевантажений оператор описати як «дружній», наприклад:**

**trend A operator + (A m х, А т \_ х )**

{

**return A(m \_ x . val = m \_ y . val);**

**то для першого операнда буде виконано неявне приведення типів. У мові С# для бінарних операторів, в описі, список параметрів**

**перевантаженого оператора містити два параметри, але один з них повинен бути типу, для якого цей перевантажений оператор визна­ чається. Тому, не можна перевантажувати оператор для цілих або рядкових типів. Оскільки операторна функція статична, необхідно явно вказувати всі операнди, наприклад:**

|  |  |
| --- | --- |
| **public class** | **Coord** |
| **і** |  |
| **public** | **Coord()** |

**I**

**x = 0; У = 0;**

}

**public Coord(int i, int j)**

**{**

**x = i;**

**y = j ;**

}

**public Coord A d d ( C o o r d obj)**

**{**

**Coord tmp = new Coord(); tmp.x = x + obj.x;**

**tmp.y = y + obj.y; return tmp;**

*\*

**public static Coord operator +(Coord obj 1, Coord obj2)**

{

**Coord t m p = new CoordQ; tmp.x = obj 1 .x + obj2.x; tmp.y = o b j l . y + obj2.y; return tmp;**

}

**private int x; private int y;**

**72**

**s t a t i c v o i d M a i n ( s t r i n g [ ] a r g s )**

**{**

**C o o r d o l = n e w C o o r d ( 4 , 5 ) :** **/ / o l = { 4 , 5 }**

**C o o r d o 2 = n e w C o o r d ( 1 , 1 ) ;** **/ / o 2 = { 1 , 1 }**

**C o o r d о З = о I . A d d ( o 2 ) ;** **/ / o 3 = { 5 . 6 }**

**/ / а б о**

**o 2 = o l + o 2 ;** **/ / o 2 = { 5 , 6 {**

}

***Перевантаження*** ***унарних*** ***операцій*** ***і*** ***операторів.*** **П е р е в а н т а ­**

**ж е н н я** **у н а р н и х** **о п е р а ц і й** **з д і й с н ю є т ь с я** **т а к** **с а м о ,** **я к** **і** **б і н а р н и х .**

**П р о т е** **в и к о р и с т о в у є т ь с я** **п о р о ж н і й** **с п и с о к** **п а р а м е т р і в ,** **о с к і л ь к и**

**у н а р н и й о п е р а т о р з м і н ю є с в і й о п е р а н д , я к и й о п и с у є т ь с я я к з н а ч е н ­**

**і ї я . щ о п о в е р т а є т ь с я . Н а п р и к л а д :**

• •

**c l a s s** **А**

**{**

**p u b l i c :**

**А** **( і т и і )**

**t**

**v a l = і :**

}

**A** **o p e r a t o r ! ( ) ;**

**{**

**r e t u r n A ( ! v a l ) ;**

***х***

***і***

**p r i v a t e :**

**i n t v a l ;**

};

**v o i d m a i n ( )**

{

**A** **o b j ( 3 ) ;**

**o b j = ! o b j ;**

}

**О ч е в и д н о ,** **щ о м о ж н а о д н о ч а с н о п е р е з а в а н т а ж у в а т и** **і** **у н а р н і** **і**

**б і н а р н і** **о п е р а ц і ї .**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **У** | **м о в і** | **С #** | **п е р е в а н т а ж е н н я** | **у н а р н и х** | **о п е р а ц і й** | **п р а к т и ч н о** | **н е** | **в і д ­** |
| **р і з н я є т ь с я** | **в і д** | **б і н а р н и х** | **з а** | **в и н я т к о м** | **т о г о ,** | **щ о** | **о п е р а т о р н а ф у н к ц і я** |
| **м а є** | **т і л ь к и** | **о д и н** | **п а р а м е т р** | **т и п у** | **к л а с у ,** | **в** | **я к о м у** | **в о н а в и з н а ч а є т ь с я .** |
| **Н а п р и к л а д , :** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **p u b l i c** | **s t a t i c** | **C o o r d** | **o p e r a t o r** | **! ( C o o r d** | **o p )** |  |  |  |  |  |  |  |  |
|  |  | **{** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | **o p . x** | **=** | **- o p . x :** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **o p . y** | **=** |  | **- o p . y ;** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **r e t u** | **r n** | **o** | **p ;** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **В** | **у м о в н и х** | **в и р а з а х** | **с т р у к т у р н и х** | **о п е р а т о р і в** | **у** | **м о в і** | **С #** | **н а** | **в і д м і ­** |
| **н у** | **в і д** | **м о в и** | **C + +** | **н е о б х і д н о** | **я в н о** | **в и к о р и с т о в у в а т и** | **о п е р а ц і ю** | **в і д н о ­** |
| **ш е н н я** | **( « і »** | **- п о м и л к а ,** | **« і** | **=** | **= t r u e »** | **-** | **п р а в и л ь н о ) .** |  |  |  |  |  |  |
|  | ***Особливості*** |  | ***перевантаження*** | ***операторів*** |  | ***і*** |  | ***операцій.*** | **В** | **с і** |  | **п е** | **р е ­** |
| **в а н т а ж е н і** | **о п е р а т о р и** | **р е а л і з у ю т ь с я** |  | **я к** | **ф у н к ц і ї .** | **Д е я к і** | **о п е р а т о р и** |
| **м а ю т ь** | **-** | **а р н і с т ь ,** | **щ о з б і г а є т ь с я** | **з** | **- а р н і с т ю** | **в и з н а ч е н и х** | **о п е р а т о р і в .** |
| **І н ш і м а ю т ь** | **о с о б л и в о с т і .** | **Н а п р и к л а д ,** | **о п е р а ц і я** | **і н д е к с у в а н н я** | **( « [ ] » ) в** |
| **м о в і** | **C + +** |  | **т е ж** | **р о з г л я д а є т ь с я** | **т і л ь к и** | **я к б і н а р н и й** | **о п е р а т о р** | **т а к о г о** |

**в и г л я д у :**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **l o n g o p e r a t o r [ ]** | **( c h a r \* ) ; .** |  |  |  |  |  |  |  |
|  |  |  |  | **Н а в п а к и ,** | **о п е р а т о р** | **в и к л и к у** | **п і д п р о г р а м и** | **« ( ) »** | **в** | **м о в і** | **C + +** | **м о ж е** |
| **м а т и** |  | **б і л ь ш е д в о х** | **п а р а м е т р і в .** | **Н а п р и к л а д :** |  |  |  |  |  |
| **c** | **l** | **a** | **s** | **s** |  | **А** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **(** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| *X* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **p** | **u** | **b** | **l** | **i** | **c** | **:** |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | **i n t** | **х o p e r a t o r** | **( )** | **( i n t x ,** | **i n t** | **y ,** | **i n t z )** |  |  |  |  |  |
|  |  |  |  |  |  | { |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | **r e t u r n v a l u e** | **f x ]** | **[ у ]** | **[ z ] ;** |  |  |  |  |  |  |
|  |  |  |  |  |  | } |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **p** | **r** | **i** | **v** | **a** | **t** | **e :** |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | **i n t** | **v a ] u e [ 1 0 ] [ 1 0 ] [ 1 0 ] ;** |  |  |  |  |  |  |  |  |

**I i**

74 75

**v o id main ()**

{

**A obj; int x; i n t y ; int z; x = 2; у = 2;**

**z = 2**

**obj (x, v, z) = obj(x + 2. у - 1, z)+ obj(x, v, z):**

}

**У цьому прикладі, в операторові привласнення використову­ ється операція «+» для визначеного типу int.**

**2.3. Поліморфізм. Пізнє зв'язування**

**Вказівна змінна типу базового класу може використовуватися без явного приведення, як покажчик на будь-який клас, похідний від даного класу. Для використання вказівних змінних типу похід­ ного класу, що типізуються, до частин базового класу потрібне явне приведення типу. Адресна арифметика для вказівних змінних, що типізуються базовим класом при збільшенні значення посилан­ ня на наступний, переадресовує на нижній за ієрархією похідний клас. Використовуючи покажчики змінних, що типізуються базо­ вим класом для доступу до похідного класу, від даного базового класу, доступ дозволяється лише до методів успадкованого класу. Наприклад, у мові C++:**

**void main ()**

**{**

**int х; іпу у;**

**double R; double D; point \* pPoint;**

**Circle Ocircle( 100,20,3.2); x = pPoint — • getx ();**

76

**у = pPoint — • gety ();**

**pPoint — • print ();**

**pPoint = circle;**

**R = pPoint — *p .* getR();**

**D = pPoint — • getD();**

**A = pPoint —». getA();** **-**

}

}

**У наведеному прикладі клас Circle є похідним від класу Point. Методи getR(), getD() належать тільки похідному класу Circle. То­ му, зважаючи на описану вище поведінку вказівної змінної, що типізується, для операторів привласнень змінних R і D будуть ви­ дані синтаксичні помилки незважаючи на те, що pPoint указує на об'єкт Cricle . Пояснюється це тим, що методи getR(), getD() не є методами базового класу.**

***Віртуальні функції.* Використання покажчика, що типізуєтьсябазовим класом, дозволяє створювати гнучкіші програми, але мас, як було видно, обмежене використання.**

**Для подолання цього обмеження використовують віртуальні підпрограми. Віртуальна підпрограма, по суті, є безліччю підпрог­ рам з однаковою сигнатурою, але різними формами, які використо­ вують у класифікації. До того ж, одну з підпрограм цієї множини обов'язково потрібно визначити в базовому класі з ключовим сло­ вом virtual, а інші — в похідних класах. Зазвичай клас, що містить віртуальну підпрограму, називається поліморфним. Наприклад, у мові C + + :**

**class Point**

**{**

**public:**

**virtual void print (); private:**

**int x; int y;**

77

**1 .**

***) 1***

**class circle : public point**

{

**public:**

**virtual void print (); private :**

**double radius;**

**};**

**void main ()**

{

**point M y P o i n t (30, 20); point \*pPoint;**

**circle \* M y C i r c l e (100, 20, 3.2); circle \*pCircle;**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **pPoint** | **= &** | **MyPoint;** |  |  |
| **Ppoint** | **— •** | **print ();** |  |  |
| **M y P o i n t .print (); //** |  |  |
| **pCircle -"- &** | **My Cir c I e ;** |  |
| **pCircle —•** | **print ();** |  | **// статичне зв'язування** |
| **MyCircIe . print ();** |  |  |
| **pPoint** | **= &** | **MyCircIe;** |  |  |
| **pPoint —• print ();** | **//** | **пізнє зв'язування print () з Circle** |

};

**Зазвичай, ієрархія класів будується так, що методи самого верхнього класу є настільки невизначеними, що їх докладне опи­ сання позбавлене сенсу. Тому, в таких класах описують віртуальні підпрограми без тіла. Вони називаються чистими віртуальними підпрограмами (pure virtual function) . Наприклад, у мові C++:**

**virtual void draw () const = 0;.**

78

**Клас, що містить такі методи, не може слугувати для поро­ дження об'єктів, тому він називається абстрактним або абстракт­ ним базовим класом. Він використовується в спадковості і батьків­ ські класи визначають не довизначені методи абстрактного класу. Абстрактні класи можуть скласти декілька верхніх рівнів ієрархії класифікації. Кожен батьківський клас перевизначає чисту віртуа­ льну функцію абстрактного класу. Таким чином абстрактний клас описує загальний інтерфейс з ресурсами, які довизначаються похі­ дними класами. До того ж, що не можна створити об'єкт абстракт­ ного класу - можна описати вказівну змінну, що типізується абст­ рактним класом і використовувати її для посилання на методи об'­ єктів будь-яких класів, похідних від абстрактного. Зв'язування здійснюватиметься динамічно.**

79

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | *СПИСОК Л1ТЕРА* | *ТУРИ* |  |
| **1.** | *Себеста Р. У.* **Основные концепции языков программирова­** |
| **ния / Р.У. Себеста. - М. : Вильяме, 2001 . - 672 с.** |  |
| **2.** | *Бен-Apu М.* | **Языки программирования. Практический** | **срав­** |
| **нительный анализ / М. Бен-Ари. - М . : Мир, 2000. - 366 с.** |  |
| **3.** | *Дейл ХМ.* | **Как программировать** | **на** | **С + + / Х . М .** | **Дейл,** |
| **П.-М. Дж . Дейтел. - О О О Бином-Пресс, 2005. -** | **1248 с.** |  |

1. *Шилд Г.* **С#: учебный курс / Г. Шилд. - СПв. : Питер; К. :**

**Издательская группа BHV, 2002. - 512 с.**

1. *Дж. Майо* **С# : Исскуство программирования. Энциклопе­**

**дия программиста / Дж . Майо / пер. с англ. - СПб . : ООО ДиаСофтЮП, 2 0 0 2 . - 6 5 6 с.**

|  |  |
| --- | --- |
| **4** | **6.** *Робинсон С. С#* **для профессионалов. - Т 1,2. - М. ; Лори.** |
| **2 0 0 3 . - 9 9 5 с.** |
|  | **7.** *Sidorov N.A.* **Basics of programming languages: Lecture Course** |
| **/ N.A.** | **Sidorov. - K.: N A U , 2003. - 130 p.** |

**í**

Навчальне видання

СИДОРОВ Микола Олександрович

**ОБ'ЄКТНО-ОРІЄНТОВАНА ПАРАДИГМА**

Конспект лекцій

**НТБ НРУ**



|  |  |
| --- | --- |
| **У ч . В . 2** | **U 1 2 . М** |

Редактор *С.М. Зозуля*

Технічний редактор *А.І. Лавринович* Коректор 0 . 0 . *Крусь*

Комп'ютерна верстка *Н.В. Чорної*

Підп. до друку 15.06.09. Формат 60x84/16. Папір офс. Офс. друк. Ум. друк арк. 4,65. Обл.-вид. арк. 5,0.

Тираж 100 пр. Замовлення № 147-1.

Видавництво Національного авіаційного університету «НАУ-друк» 03680. Київ -58, проспект Космонавта Комарова 1.

Свідоцтво про внесення до Державного реєстру ДК" № 977 від 05.07. 2002

80