**Лекція 15. Наслідування.** **Базові поняття та основні принципи**

**Відношення між класами**

***Відношення залежності***

Залежністю – називають відношення використання, згідно з яким зміна в специфікації одного елемента може вплинути на поведінку іншого елементу, що його використовує, причому зворотне не обов'язково. Найчастіше залежності застосовуються при роботі з класами, щоб відобразити в сигнатурі операції той факт, що один клас використовує інший як аргумент.

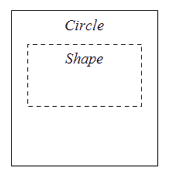
***Відношення агрегації***

Агрегацією – називають відношення включення, коли клас А включає в себе об’єкти (покажчики на об’єкти) класу В. Його ще називають відношенням частина/ціле або “has a”. Клас-агрегат вміщує колекцію покажчиків на екземпляри класів-частин.

**Композиція** – частинний і більш сильний випадок агрегації, коли зі знищенням цілого знищуються частини. Клас-композит вміщує колекцію екземплярів агрегованого класу.

***Відношення узагальнення***

Узагальненням називається відношення класифікації між загальною сутністю, суперкласом (батьківським) і більш спеціалізованим різновидом цієї сутності, що називають підкласом чи нащадком. Узагальнення називають зв’язком “is a”, від анг. is a kind of. Троянда is a kind of (це є вид) квітки.

Узагальнення і наслідування / успадкування (спеціалізація) – це протилежні напрямки одного відношення. Спадкування – відношення обернене до узагальнення. Клас-нащадок повторює структуру і поведінку іншого класу (одиночне спадкування) або других (множинне спадкування) [Буч, гл. 2]. Клас-нащадок наслідує (вміщує) всі поля та методи батьківського класу, хоча може мати і власні.

Наявність механізму спадкування відрізняє об’єктно-орієнтовані мови від просто об’єктних.

**Основні принципи наслідування**

Мета ООП полягає в повторному використовуванні створених [клас](http://moodle.ipo.kpi.ua/moodle/mod/glossary/showentry.php?courseid=508&concept=%D0%9A%D0%BB%D0%B0%D1%81)ів, що економить час і сили. Якщо існує деякий клас, то можливі ситуації, що новому класу потрібні деякі або навіть всі особливості вже існуючого класу, і необхідно додати один або декілька елементів даних або функцій. В таких випадках C++ дозволяє будувати новий об'єкт, використовуючи характеристики вже існуючого об'єкту. Іншими словами, новий об'єкт буде успадковувати елементи існуючого класу (базового класу). Коли будується новий клас з існуючого, цей новий клас часто називається похідним класом. Цей процес називається **наслідуванням** або **успадкуванням**. Наслідування є фундаментальною концепцією об'єктно-орієнтованого програмування.

**Англійською:** похідні класи - Derived classes;

Наслідування - Inheritance

***Б.Страуструп:*** *«*Похідні типи дають *простий, гнучкий і ефективний* апарат для:

* завдання класові альтернативного інтерфейсу (повторне використання коду);
* визначення класу шляхом додавання нових можливостей до базового класу без перепрограмування чи перекомпілювання».

Наслідування поділяють на просте і множинне, відкрите і закрите.

***Основні концепції наслідування***

* Якщо програми використовують успадкування, то для породження нового класу необхідний базовий клас, тобто новий клас успадковує елементи уже існуючого базового класу.
* Для ініціалізації елементів похідного класу програма повинна викликати [конструктор](http://moodle.ipo.kpi.ua/moodle/mod/glossary/showentry.php?courseid=508&concept=%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D0%BE%D1%80)и базового і похідного класів.
* Використовуючи оператор "крапку", програми можуть легко звертатися до елементів базового і похідного класів.
* На додаток до загальних ([public](http://moodle.ipo.kpi.ua/moodle/mod/glossary/showentry.php?courseid=508&concept=public" \o "Глосарій: public)) (доступним всім) і приватних ([private](http://moodle.ipo.kpi.ua/moodle/mod/glossary/showentry.php?courseid=508&concept=private" \o "Глосарій: private)) (доступним методам класу) елементів C++ надає захищені (protected) елементи, які доступні базовому і похідному класам.
* Для вирішення конфлікту імен між елементами базового і похідного класів програма може використовувати оператор глобального дозволу, указуючи перед ним ім'я базового або похідного класу.

При успадкуванні, за необхідності, можна перевизначити деякі функції батьківського класу в класі-нащадку. Звичайно у базовому класі описуються найбільш загальні властивості, які придатні для всіх похідних від нього класів. Похідний клас успадковує ці загальні властивості і додає нові, характерні тільки для нього.

**Принцип підстановки**

Відкрите наслідування встановлює між класами відношення “є” (“is a”): клас нащадок є різновидом базового класу. Всюди де використовується об‘єкт базового класу дозволяється використовувати об‘єкт похідного класу. Дане положення називається принципом підстановки.

**Форми наслідування**

Бадд[15,розділ 7.3-7.4] таким чином визначає форми наслідування.

**Спеціалізація**. Клас-нащадок є більш конкретним або спеціалізованим випадком батьківського класу. Тобто, клас-нащадок є підтипом батьківського класу.

**Специфікація**. Батьківській клас описує поведінку, яка реалізується в класі-нащадку, але залишено нереалізованим в батьківському.

**Конструювання**. Клас-нащадок використовує методи, надані батьківським класом, але не є підтипом батьківського класу (реалізація методів порушує принцип підстановки).

**Узагальнення**. Клас-нащадок модифікує або перевизначає деякі методи батьківського класу з метою отримання об’єкту більш загальної категорії.

**Розширення**. Клас-нащадок додає нові функціональні можливості до батьківського класу, але не змінює успадковану поведінку.

**Обмеження**. Клас-нащадок обмежує використання деяких методів батьківського класу.

**Варіювання**. Клас-нащадок та батьківський клас є варіаціями на одну тему, та зв’язок «клас—пцдклас» довільний.

**Комбінування**. Клас-нащадок успадковує риси більш ніж одного батьківського класу. Це— множинне успадкування; воно буде розглянуто в наступних лекціях окремо.

Переваги спадкування з єдиним предком/батьком полягає в тому, що функціональні можливості останнього класу успадковуються всіма об’єктами. Таким чином гарантується, що кожний об’єкт володіє загальним мінімальним рівнем функціональності. Мінус в тому, що єдина ієрархія "зщеплює" всі класи один з одним.

**Синтаксис наслідування**

**class Base**

**{**

**//оголошення базового класу**

**};**

**class Derived : специфікатор\_доступу Base [, специфікатор\_доступу Base2, … ]**

**{**

**//оголошення класу нащадка**

**};**

При описі класу в заголовку перечислюються усі класи, які є для нього базовими / батьківськими. Можливість звернення до елементів цих класів регулюється за допомогою специфікаторів доступу ***private, protected*** та ***public:***

**class имя : [private | protected | public] // базовий клас**

**{ /\* тіло класу \*/ };**

**Если базовых классов несколько, они перечисляются через запятую. Ключ доступа**

**может стоять перед каждым классом, например:**

**class А { ... };**

**class В { ... };**

**class С { ... };**

**class D: А. protected В. public С { ... }:**

**Специфікатори доступу**

За замовчуванням для класів використовується специфікатор доступу ***private,*** для структур ***-***  ***public***.

Для будь-якого члену класу може також застосовуватися специфікатор доступу ***protected,*** який для одиночних класів, що не входять до ієрархії, аналогічний до ***private*** . Різниця виявляється при спадкуванні, що демонструє наступна таблиця.

|  |  |  |
| --- | --- | --- |
| Специфікатор доступу | | |
| Базовий клас | В базовому класі | Доступ в класі-нащадку |
| ***private*** | ***private*** | відсутній |
| ***protected*** | ***private*** |
| ***public*** | ***private*** |
| ***protected*** | ***private*** | відсутній |
| ***protected*** | ***protected*** |
| ***public*** | ***protected*** |
| ***public*** | ***private*** | відсутній |
| ***protected*** | ***protected*** |
| ***public*** | ***public*** |

Ми бачимо, що ***private*** – члени базового класу в класі-нащадку недоступні незалежно від специфікатору доступу. Звернення до них може здійснюватися через методи базового класу.

Члени базового класу ***protected*** при спадкуванні зі специфікатором доступу ***private*** стають в класі-нащадку ***private***, в решті випадків права доступу до них не змінюються.

Доступ до ***public***– членів базового класу при спадкуванні стає відповідним специфікатору доступу

Якщо базовий клас спадкується зі специфікатором доступу ***private,*** можна вибірково зробити деякі його члени доступними в класі-нащадку, оголосивши їх в секції ***public*** в класі-нащадку за допомогою операції доступу до області видимості:

**class Base{**

**public: void f();**

**}**

**class Derived : private Base{**

**public: Base::void f();**

**}**

**Просте спадкування**

Простим називається спадкування, при якому похідний клас або клас-нащадок має один базовий клас або одного батька. Для різних методів класу існують різні правила спадкування — наприклад, конструктори та операція присвоювання в похідному класі не спадкуються, а деструктори спадкуються.

***Поля і методи при спадкуванні***

* Клас-нащадок успадковує всі поля та методи батьківського класу.
* Якщо у батьківському класі поле чи метод приватний, то нащадок не має до нього доступу.
* Допускається не тільки успадкування методів базового класу, але також додавання нових і перевизначення існуючих методів.
* Якщо ім’я поля (методу) у похідному і базовому класі співпадають, говорять про перевизначення або перекриття. Для звернення до змінної базового класу використовують операцію доступу до області видимості "**::".**

***Спеціальні методи при спадкуванні***

* Клас-нащадок успадковує всі поля методи батьківського класу крім:
  + - Конструкторів
    - Деструктора
    - Операції присвоєння
* Основне правило: у конструкторі нащадка потрібно ініціалізувати власні змінні, а для наслідуваних даних - викликати конструктор базового класу.

***Конструктори при спадкуванні***

* Якщо в конструкторі похідного класу явно не викликається конструктор базового класу, то компілятор сам викликає конструктор за замовчуванням базового класу.
* Якщо необхідно викликати конструктор базового класу такого ж виду, то конструктор вказується в рядку його ініціалізації.
* Тіло конструктора базового класу завжди виконується раніше тіла конструктора похідного класу

***Деструктори при спадкуванні***

* Деструктор похідного класу не вимагає явно викликати деструктор базового класу. У деструкторі похідного класу компілятор автоматично генерує виклики базових деструкторів
* Тіло деструктора похідного класу завжди виконується раніше тіла деструктора базового класу

Знищення конструкторів виконується в оберненому порядку до створення.

***Приклад.*** Розробити програму, що реалізує приклад взаємодії базового (class Base) і похідного (class Derived) класів. Розглянути роботу конструкторів.

**#include <iostream.h>**

**#include <conio.h>**

**#include <Windows.h>**

**using namespace std;**

**class Base**

**{public: int a, b, і; //поля базового класу**

**Base (int a=0, int b=0){this->a = a; this->b = b;**

**cout << "constructor class Base" << endl;}**

**~Base() {cout << "destructor class Base" << endl;};**

**void set\_i(int n) { і = n; }**

**int get\_i(){return i;};**

**};**

**class Derived : public Base**

**{ public: int c, j; //власні поля нащадка**

**Derived (int a=0, int b=0, int c =0):**

**Base (a, b) //явний виклик конструктора базового класу**

**{this->c = c; //присвоєння власних змінних**

**cout << "constructor class Derived" << endl;};**

**~Derived ()**

**{cout << "destructor class Derived" << endl;};**

**void set\_j(int n) { j = n; }**

**int mult() { return j\*get\_i();}**

**};**

**int main()**

**{ /\*Отримання дескриптора\*/**

**HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);**

**system("color F0"); /\*колір фону - білий, тексту - чорний\*/**

**Base b1, b2; //змінні базового класу**

**Derived d1,d2; //змінні похідного класу**

**b1 = b2; //операція = базового класу**

**cout<<"b1 = b2 = " << b1.b <<endl;**

**d1 = d2; //операція = похідного класу**

**cout<<"d1 = d2 = "<< d1.b <<endl;**

**b2 = d1; //базовий = похідний:**

**cout<<"b2 = d1 = "<< b2.b <<endl;**

**d1.Base::b = 2; //b - перевизначена тому звернення через ::**

**d1.c = 3; d2.c = 4;**

**cout<<"b2, d1 = "<< b2.b << "," << d1.b <<endl; //принцип підстановки**

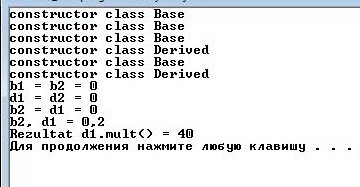
**d1.set\_i(10);**

**d1.set\_j(4);**

**cout<<"Rezultat d1.mult() = "<<d1.mult()<<endl;**

**system ("pause");**

**return 0;}**



Результат роботи програми:

**Rezultat= 40**

У програмі подано спочатку опис базового класу **Base**, а потім похідного класу **Derived**. У похідному класі оголошено, що базовий клас є відкритим для похідного, тобто всі захищені і відкриті дані і функції (але не закриті) класу **Base** доступні в класі **Derived** . За таких умов у функції **mult()** використовується відкрита функція **get\_i()** класу **Base** , однак змінна **і**, яка оголошена в закритому розділі класу **Base**, безпосередньо не може застосовуватися у класі **Derived**.

У головній програмі оголошено об’єкт **d1** класу **Derived**, однак для його обробки реалізується як функція **d1.set\_i (10)** класу **Base**, так і функція **d1.set\_j(4)** класу **Derived**. Тому при виконанні функції **d1.mult()** буде виведено 40. Якби клас А успадковувався у закритому (**private**)режимі, то усі його елементи було б заборонено використовувати в класі-нащадку.

***Контрольні запитання*.**

1. Які типи відношень між класами Ви знаєте?
2. Як пов’язані відношення агрегації і композиції?
3. Наведіть приклад відношення узагальнення.
4. Визначте основні концепції наслідування.
5. В чому полягає принцип підстановки?
6. Які форми наслідування Ви можете визначити?
7. В чому полягає просте спадкування?
8. Як визначається поведінка конструкторів та деструкторів при спадкуванні?

***Для самостійного вивчення*:** Поглибити матеріал лекції за наданою літературою. Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.

**Рекомендована література**

1. Павловская Т.А. С/С++. Программирование на языке высокого уровня. СПб.: Питер, 2003. – 461 с. URL: <http://www.ph4s.ru/bookprogramir_1.html>
2. Страуструп Б. Язык программирования С++. — СПб.; М.: Невский диалект — ЗАО “Изд-во “Бином”, 1999.
3. Джейс Либерти Освой самостоятельно С++ за 21 день: 3-е изд. пер. с англ.: Уч. пос. – М.: Издательский дом „Вильямс”, 2001. – 816 с.: ил..

4. Липпман С. Б., Лажойе Ж. Язык программирования С++: Вводный курс. — М.: ДМК, 2001. URL: <http://www.insycom.ru/html/metodmat/inf/Lipman.pdf>

5. Дейтел Х., Дейтел П. Основы программирования на С++. – М.: Бином, 1999. – 1024 с. URL: <http://ijevanlib.ysu.am/wp-content/uploads/2018/03/deytel.pdf>

6. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++ . [2-е изд.] / Буч Г. - СПб.: Невский Диалект, 1998. - 560 с.

7. Бадд Т. Объектно-ориентированное программирование в действии. [2-е изд.] – СПб.: Изд-во "Питер". 1997. URL: <http://khizha.dp.ua/library/Timothy_Budd_-_Introduction_to_OOP_(ru).pdf>

8. Скотт Мейерс*.* Эффективное использование C++. 50 рекомендаций по улучшению ваших программ и проектов. *"ДМК", 2000; 240 с.*