ЛАБОРАТОРНА РОБОТА №4

Тема: Успадковування класів

Мета: ознайомитись зі способами та механізмами успадкування класів та навчитись використовувати їх для побудови об’єктно-орієнтованих програм.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Класи використовуються для моделювання концепцій реального та програмного світу. Однак жодна концепція не існує ізольовано. Вона співіснує зі спорідненими концепціями і саме цьому зв'язку вона є сильною. Поняття похідного класу і пов'язані з ним механізми мови призначені для вираження ієрархічних відносин, тобто для відображення спільності класів. Наприклад, концепції кола і трикутника пов'язані тим, що обидва об’єкти є фігурами. Концепція фігури є спільною для них. Тому ми повинні явно визначити, що класи *Circle* і *Triangle* мають загальний базовий клас *Shape.* Представлення понять «коло» і «трикутник» в програмі без введення поняття «фігура» означало б втрату чогось істотного.

Базові та похідні класи

Розглянемо наведений раніше приклад. І коло, і трикутник є фігурами, однак якщо не використовувати спеціальних засобів для явного опису цього факту, компілятор сам не зможе зробити подібний висновок. Відповідно, ми не зможемо, наприклад, помістити покажчики на об'єкти класів *Circle* і *Triangle* в один список. Відношення між класами у якому існують породжуючі (базові) класи називають **успадкуванням.**

Клас може бути *породжено* з іншого класу, який називається *базовим* класом *похідного* класу. Клас може бути породжено від одного або декількох класів. Породжені класи успадковують властивості базових класів, включаючи дані та функції члени. Крім того, в похідних класах можуть бути оголошені **додаткові** дані та функції члени.

Оголошення похідного класу має наступний синтаксис:

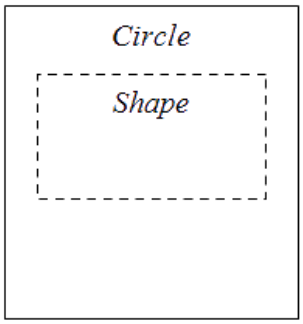
class <*ім'я похідного* ***класу***> **:** <*специфікатор доступу*> <*назва базового класу*>

[, <*Специфікатор доступу*> <*назва базового класу*>]

(...);

Відношення успадкування між класами може бути представлено у графічному вигляді наступним чином:

Популярною та ефективною реалізацією поняття похідних класів є представлення об'єкта похідного класу у вигляді об'єкта базового класу та інформації, що відноситься тільки до похідного класу.



Похідний клас може сам, у свою чергу, служити базовим класом.

Якщо члени базового класу не були перевизначені в похідних класах, тоді вони позначаються і трактуються так само, як і члени похідного класу. В такому випадку кажуть, що члени базового класу **успадковуються** похідним класом. Операція вирішення області **видимості ::** може вживатися для явного посилання на член базового класу. Це забезпечує доступ до імені, яке може бути перевизначеним в похідних класах.

```
class Base
{
public:
      int a, b;
};

class Derived : public Base
{
public:
      int b, c;
};

Derived d;
d.a      = 1;        // Ініціалізація a, успадкованого з класу Base
d.Base::b = 2;       // Ініціалізація b з класу Base
d.b      = 3;        // Ініціалізація b, оголошеного в класі Derived
d.c      = 4;
Base *bp  = &d;  // Перетворимо покажчик на клас Derived у покажчик на
Base
```

Клас називається *безпосереднім базовим класом,* якщо він згадується при оголошенні похідного класу, і *непрямим базовим класом,* якщо він є базовим класом для одного з базових класів оголошуємо класу.

```
class A
{
public:
      void f();
};
class B : public A
{
};
// Клас B є безпосереднім базовим класом для класу C,
// а клас A – непрямим базовим класом для класу C
class C : public B
{
```

```
public:
    void f();
    void ff();
};
```

У записі *<ім'я класу>::<ім'я>* – *ім'я класу* може бути ім'ям непрямого базового класу. *Ім'я класу* визначає клас, в якому починається пошук *імені.*

```
void C::ff()
{
    f();    // Виклик функції f() з класу C
    A::f(); // Виклик функції f() з класу A
    B::f(); // Знову виклик функції f() з класу A,
        // тому що у класі B функцію f() не визначено
}
```

Специфікатори доступу для базових класів

Специфікатор доступу може приймати значення:

- *public* - у цьому випадку публічні члени базового класу стають публічними членами похідного класу, а захищені члени базового класу стають захищеними членами похідного класу;
- *protected* - у цьому випадку публічні і захищені члени базового класу стають захищеними членами похідного класу;
- *private* - у цьому випадку публічні і захищені члени базового класу стають приватними членами похідного класу.

Приватні члени класу недоступні у похідних класах, якщо тільки він не оголошений як дружній.

У випадку коли специфікатор доступу не вказано (за замовчанням) використовується специфікатор доступу *private.*

```
class Base
{
private:
    int a;
protected:
    int b;
public:
    int c;
};
class Derived1 : public Base
{...  // A недоступний
    // B - захищений член класу Derived1
    // C - публічний член класу Derived1
};
class Derived2: protected Base
{...  // A недоступний
    // B і C - захищені члени класу Derived2
};
class Derived3: private Base
{...  // A недоступний
    // B і C - приватні члени класу Derived3
};
```

Конструктори і деструктор

Похідні класи можуть також потребувати конструктора. **Конструктори не успадковуються, вони повинні бути визначені в самому класі.** Якщо базовий клас має конструктор, він повинен бути викликаний. Конструктори замовчуванням можуть бути викликані неявно. Однак якщо всі конструктори базового класу вимагають зазначення аргументів, то конструктор цього базового класу повинен бути викликаний явним чином. Аргументи конструктора базового класу вказуються у визначенні конструктора похідного класу.

```
class Base
{
private:
	int a;
protected:
	int b;
public:
	Base(int aa, int bb) : a(aa), b(bb) {}
};

class Derived : public Base
{
private:
	int c;
public:
	Derived(int aa, int bb, int cc)
		: Base(aa, bb)  // Ініціалізація базового класу
		, c(cc)		// Ініціалізація членів похідного класу
	{
	}
};
```

Конструктор похідного класу може мати ініціалізатори для своїх власних членів та членів базового класу, але він не може безпосередньо ініціалізувати члени базового класу.

```
class Derived : public Base
{
private:
	int c;
public:
	Derived(int aa, int bb, int cc)
		: a(aa), b(bb)		// Помилка - a і b не оголошено в класі Derived
		, c(cc)
	{
	}
};
```

Об'єкти класу створюються знизу вгору: спочатку базовий клас, потім похідний, а знищуються в протилежному порядку. Дані базових класів конструюються в порядку їх оголошення в класі і знищуються в зворотному порядку.

Використання захищених членів класу

Проста модель приховування даних «відкритий / закритий» добре працює для конкретних типів. Однак при використанні похідних класів існує два види користувачів класу: похідні класи та всі інші функції. Модель «відкритий / закритий» дозволяє програмісту розрізняти розробників і всіх інших, але вона не передбачає особливого обслуговування для похідних класів.

Мова C++ дозволяє здійснювати гнучке управління доступом до членів класу за рахунок використання трьох специфікаторів доступу. Однак захищеними членами класу можна більше зловживати, ніж приватні. Вміщення значної частини даних у загальний клас, доступний для всіх похідних класів, призводить до ризику руйнування цих даних. Більш того, так само як і відкриті, захищені дані не просто реструктурувати через складності знаходження всіх випадків їх використання. Таким чином, захищені дані часто приводять до проблем супроводу.

На щастя, немає необхідності використовувати захищені дані. Члени класів за замовчуванням закриті і, як правило, це є найкращим варіантом. При розробці похідного класу ви може використовувати інтерфейс базового класу для роботи з приватними членами базового класу.

Зверніть увагу, що всі ці заперечення не мають великого значення для захищених *функцій.* Захищеність - це прекрасний спосіб визначення операцій для використання в похідних класах.

.

ПОРЯДОК ВИКОНАННЯ РОБОТИ

1. Ознайомитись з теоретичним матеріалом про наслідування класів.
2. Напишіть програму згідно наступного завдання 1.

Завдання 1. Уявіть собі видавничу компанію, яка торгує книгами і аудіо-записами цих книг. Створіть клас publication, в якому зберігаються назва (рядок) і ціна (тип float) книги. Від цього класу успадковуються ще два класи: book, який містить інформацію про кількість сторінок у книзі (типу int), і type, який містить час запису книги у хвилинах (тип float). У кожному з цих трьох класів повинен бути метод getdata(), через який можна отримувати дані від користувача з клавіатури, і putdata(), призначений для виведення цих даних.

Напишіть функцію main() програми для перевірки класів book і type. Створіть їх об'єкти в програмі і запросіть користувача ввести і вивести дані з використанням методів getdataQ і putdata().

3. Напишіть програму згідно завдання 2.

Завдання 2. До класів з попереднього завдання (попередньо зберігши окремо код) додайте базовий клас sales, в якому міститься масив, що складається з трьох значень типу float, куди можна записати загальну вартість проданих книг за останні три місяці. Включіть в клас методи getdata() для отримання значень вартості від користувача і putdata() для виведення цих цифр. Змініть класи book і type так, щоб

вони стали похідними обох класів: publication і sales. Об'єкти класів book і type повинні вводити і виводити дані про продажі разом з іншими своїми даними. Напишіть функцію main() для створення об'єктів класів book і type, щоб протестувати можливості введення/виведення даних.

4. Проаналізувати результати роботи програми у наведеному прикладі.

ПРИКЛАД ПРОГРАМИ

Створити клас ГЕОМЕТРИЧНА ФІГУРА з полями – координатами, кольором та стилем. Визначити конструктори, деструктор. Створити похідний клас КОЛО з полями: координати, колір, стиль, радіус. Визначити конструктори за замовчуванням і з різним числом параметрів, деструктор, функції малювання кола.

```
#include <windows.h>
#include <conio.h>
#include <math.h>


enum fstyle { SOLID_FILL, X_FILL, O_FILL,
        LIGHT_FILL, MEDIUM_FILL, DARK_FILL };


enum color {
        cBLACK=0, cDARK_BLUE=1, cDARK_GREEN=2, cDARK_CYAN=3,
        cDARK_RED=4, cDARK_MAGENTA=5, cBROWN=6, cLIGHT_GRAY=7,
        cDARK_GRAY=8, cBLUE=9, cGREEN=10, cCYAN=11,
        cRED=12, cMAGENTA=13, cYELLOW=14, cWHITE=15 };

void init_graphics();
void clear_screen();
void set_cursor_pos(int x, int y);
void set_color(color fg, color bg = cBLACK);
void set_fill_style(fstyle);
void draw_circle(int x, int y, int rad);


HANDLE hConsole; //дескриптор консолі
char fill_char; //символ заповнення

////////////////////////////////////////////////////////

void init_graphics()
{
        COORD console_size = {80, 25};
        //відкрити канал введення/виведення на консоль
        hConsole = CreateFile(TEXT("CONOUT$"), GENERIC_WRITE | GENERIC_READ,
                FILE_SHARE_READ | FILE_SHARE_WRITE,
                0L, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0L);
        //задати розмір екрану 80x25
        SetConsoleScreenBufferSize(hConsole, console_size);

        //текст білим по чорному
        SetConsoleTextAttribute( hConsole, (WORD)((0 << 4) | 15) );
        fill_char = '\xDB'; //символ заповнення за замовчуванням
        clear_screen();
}

//--------------------------------------------------------
```

```
void clear_screen()
{
        set_cursor_pos(1, 25);
        for(int j=0; j<25; j++)
                _putch('\n');
        set_cursor_pos(1, 1);
}

//------------------------------------------------------------

void set_cursor_pos(int x, int y)
{
        COORD cursor_pos; // початок у верхньому лівому куті
        cursor_pos.X = x - 1; // Windows починає з (0, 0)
        cursor_pos.Y = y - 1; // ми почнемо з (1, 1)
        SetConsoleCursorPosition(hConsole, cursor_pos);
}

//------------------------------------------------------------

void set_color(color foreground, color background)
{
        SetConsoleTextAttribute( hConsole, (WORD)((background << 4) | foreground) );
}

//------------------------------------------------------------

void set_fill_style(fstyle fs)
{
        switch(fs)
        {
        case SOLID_FILL: fill_char = '\xDB'; break;
        case DARK_FILL: fill_char = '\xB0'; break;
        case MEDIUM_FILL: fill_char = '\xB1'; break;
        case LIGHT_FILL: fill_char = '\xB2'; break;
        case X_FILL: fill_char = 'X'; break;
        case O_FILL: fill_char = 'O'; break;
        }

}

//------------------------------------------------------------

void draw_circle(int xC, int yC, int radius)
{
        double theta, increment, xF, pi=3.14159;
        int x, xN, yN;

        increment = 0.8 / static_cast<double>(radius);
        for(theta=0;theta<=pi/2;theta+=increment)
        {
                xF = radius * cos(theta);
                xN = static_cast<int>(xF * 2 / 1);
                yN = static_cast<int>(radius * sin(theta) + 0.5);
                x = xC-xN;
                while(x <= xC+xN)
                {
                        set_cursor_pos(x, yC-yN); _putch(fill_char); //верхня лінія
                        set_cursor_pos(x++,yC+yN); _putch(fill_char); //нижня лінія
                }
        }
}



//////////////////////////////////////////////////////////////
```

```
class shape // базовий класс
{
protected:
        int xCo, yCo; // координати фігури
        color fillcolor; // колір
        fstyle fillstyle; // стиль
public:
        // конструктор без параметрів
        shape () : xCo(0), yCo(0), fillcolor(cWHITE), fillstyle(SOLID_FILL)
        { }
        // конструктор з параметрами
        shape (int x, int y, color fc, fstyle fs) : xCo(x), yCo(y), fillcolor(fc), fillstyle
(fs)
        { }
        // деструктор
        ~shape ()
        { }
        // функция задання кольору і стилю
        const void draw()
        {
                set_color (fillcolor);
                set_fill_style( fillstyle );
        }
};

////////////////////////////////////////////////////////

class circle : public shape
{
private:
        int radius; // радіус, а xCo і yCo будуть координатами центру
public:
        // конструктор без параметрів
        circle() : shape()
        { }
        // конструктор з параметрами
        circle (int x, int y, int r, color fc, fstyle fs) : shape (x,y,fc,fs), radius(r)
        { }
        // деструктор
        ~circle ()
        { }
        // функція малювання кола
        const void draw()
        {
                shape::draw();
                draw_circle (xCo, yCo, radius);
        }
};

int main ()
{
        init_graphics (); // ініціалізуємо систему відображення графіки
        circle cir (40, 12, 5, cGREEN, X_FILL); // створюємо коло
        cir.draw (); // малюємо

        set_cursor_pos(1, 25); // переводимо курсор у низ екрану
        set_color (cLIGHT_GRAY);
        system("pause");
        return 0;
}
```

5. Для заданого індивідуального завдання написати програму з використанням наслідування.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ

1. Створити клас ПАРА ЦІЛИХ ЧИСЕЛ. Визначити конструктори, деструктор, функції доступу до полів, введення-виведення та порівняння пар (пара p1 більша за пару p2, якщо перше число p1 більше за перше число p2 або перші числа рівні і друге число p1 більше за друге число p2). Створити похідний клас ДРОБОВЕ ЧИСЛО з полями: ціла частина, дробова частина. Визначити конструктори за замовчуванням і з різним числом параметрів, деструктор, функції доступу до полів, введення-виведення та порівняння дробових чисел (на рівність/нерівність, більшість/меншість).

2. Створити клас ТРІЙКА ЧИСЕЛ. Визначити конструктори, деструктор, функції доступу до полів, введення-виведення та обчислення суми чисел. Створити похідний клас ТРИКУТНИК з полями-сторонами. Визначити конструктори за замовчуванням і з різним числом параметрів, деструктор, функції доступу до полів, введення-виведення, обчислення кутів трикутника.

3. Створити клас ПАРА ЦІЛИХ ЧИСЕЛ. Визначити конструктори, деструктор, функції доступу до полів, введення-виведення та додавання пар за формулою: (a,b)+(c,d)=(a+c,b+d). Створити похідний клас БАГАТОРОЗРЯДНЕ ЧИСЛО з полями: старша частина, молодша частина. Визначити конструктори за замовчуванням і з різним числом параметрів, деструктор, функції доступу до полів, введення-виведення, додавання, віднімання та множення багаторозрядних чисел.

4. Створити клас ПАРА ЧИСЕЛ. Визначити конструктори, деструктор, функції доступу до полів, введення-виведення та обчислення добутку чисел. Створити похідний клас ПРЯМОКУТНИЙ ТРИКУТНИК з полями-катетами. Визначити конструктори за замовчуванням і з різним числом параметрів, деструктор, функції доступу до полів, введення-виведення, обчислення площі та гіпотенузи трикутника.

5. Створити клас ТРІЙКА ЦІЛИХ ЧИСЕЛ. Визначити конструктори, деструктор, функції доступу до полів, введення-виведення та збільшення значення кожного з чисел на 1. Створити похідний клас ДАТА з полями: рік, місяць і день. Визначити конструктори за замовчуванням і з різним числом параметрів, деструктор, функції доступу до полів, введення-виведення та збільшення значення року, місяця або дня на 1.

6. Створити клас ТРИКУТНИК з полями-сторонами. Визначити конструктори, деструктор, функції доступу до полів, введення-виведення, обчислення периметра. Створити похідний клас РІВНОБІЧНИЙ ТРИКУТНИК. Визначити конструктори за замовчуванням і з різним числом параметрів, деструктор, функції доступу до полів, введення-виведення, обчислення площі.

7. Створити клас ПАРА ЧИСЕЛ. Визначити конструктори, деструктор, функції доступу до полів, введення-виведення. Створити похідний клас ГЕОМЕТРИЧНА ПРОГРЕСІЯ з полями: перший елемент та відношення прогресії. Визначити конструктори за замовчуванням і з різним числом параметрів, деструктор, функції доступу до полів, введення-виведення, обчислення суми для заданої кількості елементів прогресії.

8. Створити клас ТРІЙКА ЧИСЕЛ. Визначити конструктори, деструктор, функції доступу до полів, введення-виведення. Створити похідний клас ПАРАЛЕЛЕПІПЕД з полями-сторонами. Визначити конструктори за замовчуванням і з різним числом параметрів, деструктор, функції доступу до полів, введення-виведення, обчислення об'єму та площі поверхні.

9. Створити клас ТРІЙКА ЧИСЕЛ. Визначити конструктори, деструктор, функції доступу до полів, введення-виведення. Створити похідний клас КВАДРАТНЕ РІВНЯННЯ з полями-коефіцієнтами. Визначити конструктори за замовчуванням і з різним числом параметрів, деструктор, функції доступу до полів, введення-виведення, обчислення коренів рівняння.

10. Створити клас ЧОТИРИКУТНИК з полями – координатами вершин. Визначити конструктори, деструктор, функції доступу до полів, введення-виведення та обчислення добутку чисел. Створити похідний клас ПАРАЛЕЛОГРАМ. Визначити конструктори за замовчуванням і з різним числом параметрів, деструктор, функції доступу до полів, введення-виведення, обчислення площі паралелограма.

11. Створити клас ЧЕТВІРКА ЧИСЕЛ. Визначити конструктори, деструктор, функції доступу до полів, введення-виведення. Створити похідний клас КУБІЧНЕ РІВНЯННЯ з полями-коефіцієнтами. Визначити конструктори за замовчуванням і з різним числом параметрів, деструктор, функції доступу до полів, введення-виведення, обчислення коренів рівняння.

12. Створити клас ТРІЙКА ЦІЛИХ ЧИСЕЛ. Визначити конструктори, деструктор, функції доступу до полів, введення-виведення та порівняння трійок (трійка t1 більша за трійку t2, якщо перше число t1 більше за перше число t2 або перші числа рівні і друге число t1 більше за друге число t2 і т.д.). Створити похідний клас ДАТА з полями: рік, місяць і день. Визначити конструктори за замовчуванням і з різним числом параметрів, деструктор, функції доступу до полів, введення-виведення та порівняння дат (на рівність/нерівність, більшість/меншість).

13. Створити клас ЧОТИРИКУТНИК з полями – координатами вершин. Визначити конструктори, деструктор, функції доступу до полів, введення-виведення та обчислення добутку чисел. Створити похідний клас ТРАПЕЦІЯ. Визначити конструктори за замовчуванням і з різним числом параметрів, деструктор, функції доступу до полів, введення-виведення, обчислення площі трапеції.

14. Створити клас ТРІЙКА ЧИСЕЛ. Визначити конструктори, деструктор, функції доступу до полів, введення-виведення. Створити похідний клас БІКВАДРАТНЕ РІВНЯННЯ з полями-коефіцієнтами. Визначити конструктори за замовчуванням і з різним числом параметрів, деструктор, функції доступу до полів, введення-виведення, обчислення коренів рівняння.

15. Створити клас ПАРА ЦІЛИХ ЧИСЕЛ. Визначити конструктори, деструктор, функції доступу до полів, введення-виведення. Створити похідний клас КУТ НА ПЛОЩИНІ з полями: градуси, хвилини. Визначити конструктори за замовчуванням і з різним числом параметрів, деструктор, функції доступу до полів, введення-виведення, збільшення/зменшення на задану величину, обчислення тангенсу.