ЛАБОРАТОРНА РОБОТА №6
СПАДКУВАННЯ

Мета роботи: отримати практичні навички створення ієрархії класів.

Основні теоретичні відомості

Спадкування – один з трьох фундаментальних принципів об'єктно-орієнтованого програмування, оскільки саме завдяки йому можливе створення ієрархічних класифікацій. Використовуючи спадкування, можна створити клас, який визначає найбільш загальні характеристики всіх пов'язаних з ним елементів. Цей клас потім може бути успадкований іншими, спеціалізованими класами з додаванням в кожен з них своїх, унікальних особливостей.

У стандартній термінології мови C++ клас, який успадковується, називається базовим. Клас, який успадковує базовий клас, називається похідним. Похідний клас можна використовувати в якості базового для іншого похідного класу. Таким шляхом і будується багаторівнева ієрархія класів.

Розглянемо приклад. Є клас Human, який в найзагальніших рисах визначає будь-яку людину. Кожна людина має рік народження і им’я. Саме ці дані зберігають відповідні поля класу Human.

```
class Human
{
 int  year_of_birth; // рік народження
 string name; // ім'я
 public:
    void set_year(int year) { year_of_birth = year; }
    int get_year() { return year_of_birth; }
    void set_name(string text) { name = text; }
    string get_name() { return name; }
};
```

Це загальне визначення людини можна використовувати для визначення інших осіб. Наприклад, студента і викладача. В наступному фрагменті шляхом спадкування класу Human створюється клас Student (студент).

```
class Student : public Human
{
 int kurs; // курс
 string group; // група
 public:
    void set_kurs(int num) { kurs = num; }
    int get_kurs() { return kurs; }
    void set_group(string text) { group = text; }
    string get_gropu() { return group; }
 void show ();
};
```

Той факт, що клас Student наслідує клас Human, означає, що клас Student наслідує весь вміст класу Human. До вмісту класу Student клас Human додає поле kurs і group, а також функції-члени, яки визначають ці поля.

Загальний формат для забезпечення спадкування має такий вигляд.

клас ім'я_похідного_класса: доступ ім'я _базового_класу

{

тіло нового класу

}

Тут елемент *доступ* необов'язковий. При необхідності він може бути виражений одним з модифікаторів доступу: *public*, *private* або *protected*. Поки в визначеннях всіх успадкованих класів будемо використовувати модифікатор *public*. Це означає, що всі public - члени базового класу також будуть public - членами похідного класу. Отже, в попередньому прикладі члени класу Student мають доступ до відкритих функцій-членів класу Human, як ніби ці функції були оголошені в тілі класу Human. Однак клас Student не має доступу до private - членів класу Human. Так, для класу Student закритий доступ до члена даних (поля) imya і year_of_birth.

Розглянемо програму, яка використовує механізм спадкування для створення двох похідних класів від класу Human: Student і Teacher.

Ієрархічна схема спадкування зображена на діаграмі класів (рис. 6.1)

Похідні класи Teacher і Student є нащадками класу Human.

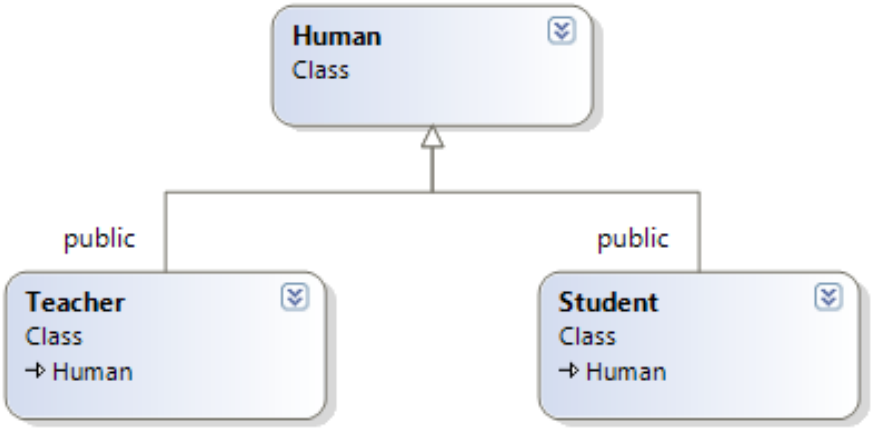


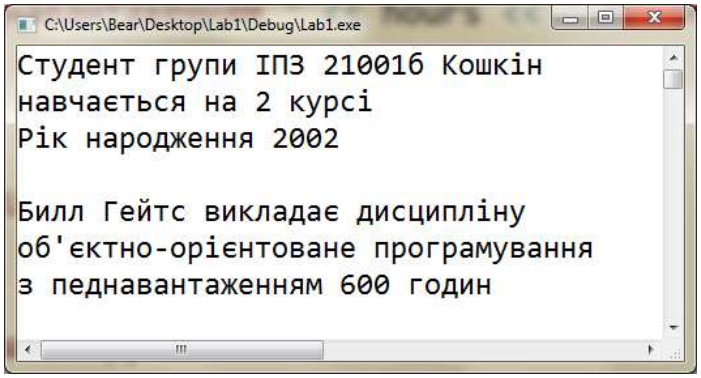
Рис.6.1. Ієрархічна схема спадкування.

```
// Визначаємо базовий клас людина
class Human
{
 int  year_of_birth; // рік народження
 string name; // ім'я
 public:
    void set_year(int year) { year_of_birth = year; }
    int get_year() { return year_of_birth; }
    void set_name(string text) { name = text; }
    string get_name() { return name; }
};
// Визначаємо клас студент
class Student : public Human
{
 int kurs; // курс
 string group; // група
 public:
    void set_kurs(int num) { kurs = num; }
    int get_kurs() { return kurs; }
    void set_group(string text) { group = text; }
    string get_gropu() { return group; }
 void show ();
};
// Визначаємо клас викладач
class Teacher : public Human
{
 int hours; // педнавантаження в год.
 string discipline; // дисципліна
 public:
    void set_hours(int num) { hours = num; }
    int get_hours() { return hours; }
    void set_discipline(string text) { discipline = text; }
    string get_discipline() { return discipline; }
 void show ();
};
```

```
void Student::show()
{
 cout<<"Студент групи "<<group<<" "<<get_name()<<endl;
 cout<<"навчається на "<<kurs<<" курсі \n";
 cout<<"Рік народження "<<get_year()<<endl<<endl;
}

void Teacher::show()
{
 cout<<get_name()<<" викладає дисципліну \n"<<discipline<<endl;
 cout<<"з піднавантаженням "<<hours <<" годин"<<endl<<endl;
}
int main()
{
 Student s;
 Teacher t;
 s.set_name("Кошкін");
 s.set_year(2002);
 s.set_group("ІПЗ 210016");
 s.set_kurs(2);
 t.set_name("Билл Гейтс");
 t.set_year(1955);
 t.set_hours(600);
 t.set_discipline("об'єктно-орієнтоване програмування");
 s.show();
 t.show();
}
```

При виконанні ця програма генерує такі результати.



Як видно за результатами виконання цієї програми, основна перевага спадкування полягає в тому, що воно дозволяє створити базовий клас, який потім можна включити до складу більш спеціалізованих класів.

Управління доступом до членів базового класу

Якщо один клас успадковує інший, члени базового класу стають членами похідного. Статус доступу членів базового класу в похідному класі визначається специфікатором доступу, що використовуються для спадкування базового класу. Специфікатор доступу базового класу виражається одним з ключових слів: *public*, *private* або *protected*. Якщо специфікатор доступу не вказано, то за замовчуванням використовується специфікатор *private*, якщо мова йде про спадкування типу *class*. Якщо успадковується тип *struct*, то при відсутності явно заданого специфікатора доступу за замовчуванням використовується специфікатор *public*.

Якщо базовий клас успадковується як *public*-клас, його *public*-члени стають *public*-членами похідного класу.

У всіх випадках *private*-члени базового класу залишаються закритими в рамках цього класу і не доступні для членів похідного.

Наприклад, в наступній програмі public-члени класу base стають public-членами класу derived. Отже, вони будуть доступні і для інших частин програми.

```
class base {
int i, j;
public:
void set (int a, int b) {i = a; j = b; }
void show () {cout << i << " " << j << " "; }
};

class derived: public base {
int k;
public:
derived (int x) {k = x; }
void showk () {cout << k << " "; }
};

int main ()
{
derived ob (3);
ob.set (1, 2); // доступ до членів класу base
ob.show (); // доступ до членів класу base
ob.showk (); // доступ до члена класу derived
}
```

Оскільки функції set() і show() (члени класу base) успадковані класом derived як public-члени, їх можна викликати для об'єкта типу derived в функції main(). Оскільки члени даних *i* та *j* визначені як private-члени, вони залишаються закритими в рамках свого класу base.

Використання захищених членів

Член класу може бути оголошений не тільки відкритим (public) або закритим (private), але і захищеним (protected). Крім того, базовий клас в цілому може бути успадкований з використанням специфікатора protected.

Використовуючи специфікатор protected, можна створити члени класу, які закриті в рамках свого класу, але які може успадкувати похідний клас, причому з отриманням доступу до них.

Розглянемо наступний приклад програми.

```
class base {
protected:
int i, j; // Ці члени закриті в класі base, але доступні
          //для класу derived.
public:
void set (int a, int b) {i = a; j = b; }
void show () {cout << i << " " << j << " "; }
};

class derived: public base {
int k;
public:
// Клас derived має доступ до членів класу base і i j.
void setk () {k = i * j; }
void showk () {cout << k << ""; }
};

int main ()
{
derived ob;
ob.set (2, 3); // OK, класу derived це дозволено.
ob.show (); // OK, класу derived це дозволено.
ob.setk ();
ob.showk ();
}
```

При оголошенні члена класу відкритим (з використанням ключового слова public) до нього можна отримати доступ з будь-якої іншої частини програми. Якщо член класу оголошується закритим (за допомогою специфікатор private), до нього можуть отримувати доступ тільки члени того ж класу. Більш того, до закритих членам базового класу не мають доступу навіть похідні класи. Якщо ж член класу оголошується захищеним (protected-членом), до нього можуть отримувати доступ тільки члени того ж або похідних класів. Таким чином, специфікатор protected дозволяє успадковувати члени, але залишає їх закритими в рамках ієрархії класів.

Якщо базовий клас успадковується з використанням ключового слова public, його public-члени стають public-членами похідного класу, а його protected-члени – protected-членами похідного класу.

Якщо базовий клас успадковується з використанням специфікатора protected, його public і protected-члени стають protected-членами похідного класу.

Якщо базовий клас успадковується з використанням ключового слова private, його public- і protected-члени стають private-членами похідного класу.

У всіх випадках private-члени базового класу залишаються закритими в рамках цього класу і не успадковуються.

Спадкування декількох базових класів

Похідний клас може успадковувати два або більше базових класів (так зване множинне спадкування).

```
// Приклад використання декількох базових класів.
class base1 {
protected:
int x;
public:
void showx () {cout << x << ""; }
};

class base2 {
protected:
int y;
public:
void showy () {cout << y << ""; }
};
```

```
// Спадкування двох базових класів.
class derived: public base1, public base2 {
public:
void set (int i, int j) {x = i; y = j; }
};

int main ()
{
derived ob;
ob.set (10, 20); // член класу derived
ob.showx (); // функція з класу base1
ob.showy (); // функція з класу base2
}
```

На рисунку 6.2 показана ієрархічна схема цих класів.

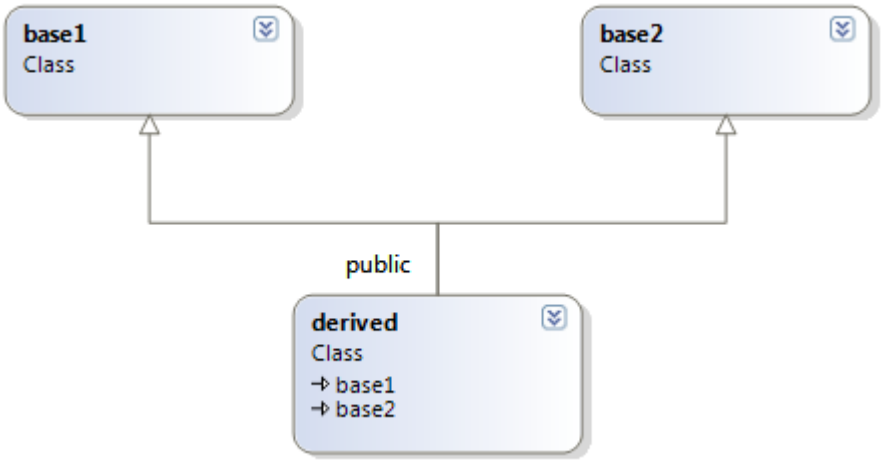


Рис. 6.2. Клас derived успадковує обидва класа base1 і base2.

Як видно з цього прикладу, щоб забезпечити спадкування декількох базових класів, необхідно через кому перерахувати їх імена у вигляді списку. При цьому потрібно вказати специфікатор доступу для кожного успадкованого базового класу.

Завдання до лабораторної роботи

1. Визначити ієрархію класів (відповідно до варіанта). Для визначення ієрархії класів зв'язати ставленням спадкування класи, наведені для заданого варіанта, з цих класів обрати один, який буде стояти на чолі ієрархії.

2. Реалізувати класи. Визначити в класах всі необхідні конструктори і деструктор.

3. Написати демонстраційну програму, в якій створюються об'єкти різних класів.

4. Побудувати діаграму класів.

Варіанти завдань:

1. студент, викладач, персона, зав кафедрою;

2. службовець, персона, робітник, інженер;

3. робочий, кадр, інженер, головний інженер;

4. дуб, рослина, дерево, очерет;

5. організація, страхова компанія, будівна компанія, метробуд;

6. журнал, книга, друковане видання, підручник;

7. тест, іспит, випускний іспит, випробування;

8. місце, область, місто, мегаполіс;

9. іграшка, продукт, товар, молочний продукт;

10. квитанція, накладна, документ, чек;

11. автомобіль, поїзд, транспортний засіб, таксі;

12. двигун, двигун внутрішнього згоряння, дизель, турбореактивний двигун;

13. республіка, монархія, королівство, держава;

14. ссавці, парнокопитні, птиці, тварина;

15. корабель, пароплав, вітрильник, яхта.

Контрольні питання

1. У чому полягає принцип спадкування?
2. Як називається клас, який успадковується?
3. Як називається клас, який успадковує?
4. Наведіть приклади багаторівневої ієрархії класів
5. Для чого використовується специфікатор protected?
6. Якщо базовий клас успадковується з використанням ключового слова private, якими стають його public і protected-члени для похідного класу?
7. Що таке множинне спадкування?