

# Об'єктно-орієнтоване програмування

Лекція №6. Успадкування

к.п.н. Стяглик Н.І.

# Мета та задачі заняття

Опанувати значення понять:

- ієрархічність;
- спадковість;
- агрегація;
- композиція;
- асоціація.

Ознайомитися з основами створення та використання технологій одиночного та множинного успадкування класів мовою програмування C++.

# Ієрархічність та її різновиди


Для подолання складності системи будують як ієрархічні.

- ☐ Ієрархія – це впорядкування абстракцій, розташування їх по рівнях.
- ☐ Розрізняють два різновиди ієрархій:
  - ієрархія класів або спадковість,
  - ієрархія об'єктів або агрегація.

# Спадковість (успадкування)

Спадковість – це механізм отримання нового класу із існуючих шляхом запозичення структурної або функціональної частини одного або декількох інших класів.

Відповідно розрізняють одиночне та множинне успадкування.



# Одиночне успадкування

# Одиночне успадкування

Синтаксис оголошення наступний:

```
class Student  
{  
char* name, *group;  
...  
};  
class Starosta: public Student  
{ int level;  
...  
};
```

Клас Starosta – похідний від класу Student, а Student – базовий для Starosta.

Клас Starosta крім своїх власних членів (level) містить і члени класу Student

# Підтипи

```
void f(Student *p, Starosta *q)
{ Student *KN [15];
  KN[0] = p;
  KN[7] = q;
  ... }
```

# Конструктори похідних класів

В оголошенні будь-якого конструктора похідного класу потрібно враховувати наступне:

- Конструювання об'єкту похідного класу обов'язково відбувається з викликом конструктора базового класу.
- Якщо явний виклик конструктора базового класу не передбачено у конструкторі похідного, за умовчання буде викликано конструктор без параметрів базового класу.
- Для організації виклику потрібної версії конструктора базового класу, можна скористатись списком ініціалізації, наприклад:

```
Student :: Student (char*a, char* b): name (a), group (b) {...}
```

```
Starosta :: Starosta (char*a, char* b, int c): Student (a,b), level  
(c) {...}
```




# Перевизначення функцій в похідних класах

Якщо в похідному класі створена функція з таким самим ім'ям, що і в базовому класі, то має місце заміщення методу. Кажуть, що в цьому випадку метод похідного класу приховує метод або методи (якщо їх декілька з однаковими іменами і різними сигнатурами) базового класу.

# Доступ при успадкуванні

Доступ в базовому класі	Модифікатор наслідування	Доступ у створеному класі
<b>Private</b> <b>Protected</b> <b>Public</b>	Public	----- Protected Public
<b>Private</b> <b>Protected</b> <b>Public</b>	Protected	----- Protected Protected
<b>Private</b> <b>Protected</b> <b>Public</b>	Private	----- Private Private



# Множинне успадкування

# Множинне успадкування

Допустиме у C++ множинне успадкування (multiple inheritance) дає можливість отримати похідний клас від декількох базових.

Для опису ієрархії множинного успадкування можна використати орієнтований ациклічний граф.

Синтаксис заголовка класу розширюється, щоб можна було використати список базових класів з атрибутами.

Наприклад:

```
class A: public B, public C{...}
```

# Віртуальні базові класи

Якщо похідний клас включає 2 або більше копій деякого базового класу, то вирішити цю проблему можна застосувавши ключове слово `virtual` перед визначенням спадкування класу.

# Приклад:

## Базовий клас:

```
class Unit
```

```
{ protected:
```

```
    int health;
```

```
    public:
```

```
    void setHealth(int a)
```

```
        { health = a; }
```

```
    void showHealth()
```

```
        { cout << "Unit health : " << health << endl; };
```

```
    Unit(): health(10) { };
```

```
    Unit(int a): health(a) { };
```

```
};
```

# Приклад:

## Похідний клас:

```
class Soldier : virtual public Unit
```

```
{ protected:
```

```
    int damage;
```

```
    int speed;
```

```
    public:
```

```
    void showDamage()
```

```
        { cout << "Soldier dama ge: " << damage << endl; }
```

```
    void showSpeed()
```

```
        { cout << "Soldier speed: " << speed << endl; }
```

```
    Soldier():damage(20), speed(3) { } };
```

# Приклад:

## Похідний клас:

```
class Horse : virtual public Unit
{ protected:
    int speed;
    public:
    void showSpeed()
        {cout << "Horse speed: " << speed << endl;}
    Horse():speed(20) { };
};
```



# Приклад:

Похідний клас (множинне успадкування):

```
class Horseman : public Horse, public Soldier
{ private:
int speed;
public:
void showSpeed()
    {cout << "Horseman speed: " << speed << endl;}
Horseman():speed(Horse::speed){};
};
```

# Контрольні запитання

- Що таке успадкування?
- Описати структуру об'єкта похідного класу.
- Описати правила доступу для різних типів успадкування.
- Чим відрізняються два види ієрархії?
- Як відбувається конструювання об'єктів при множинному успадкуванні?