ЛАБОРАТОРНА РОБОТА № 5

СПАДКУВАННЯ І ВІРТУАЛЬНІ ФУНКЦІЇ

Мета. Отримати практичні навички створення ієрархії класів і використання статичних компонентів класу.

Основний зміст роботи.

Написати програму, в якій створюється ієрархія класів. Включити поліморфні об'єкти в зв'язаний список, використовуючи статичні компоненти класу. Показати використування віртуальних функцій.

Короткі теоретичні відомості.

Спадкування це механізм отримання нового класу на основі вже існуючого. Існуючий клас може бути доповнений або змінений для створення нового класу.

Існуючі класи називаються **базовими,** а нові **похідними.** Похідний клас успадковує опис базового класу; потім він може бути змінений додаванням нових членів, зміною існуючих функцій-членів і зміною прав доступу. За допомогою спадкування може бути створена ієрархія класів, які спільно використовують код і інтерфейси.

Успадковані компоненти не переміщаються в похідний клас, а залишаються в базових класах.

В ієрархії похідний об'єкт успадковує дозволені для наслідування компоненти всіх базових об'єктів **(public, protected).**

В ієрархії класів угоди щодо доступності компонентів класу наступні:

Private член класу може використовуватися тільки функціями членами даного класу і фунціями – «друзями» свого класу. У похідному класі він недоступний.

Protected те ж, що і private, але додатково член класу з даними атрибутом доступа може використовуватися функціями-членами і функціями-"друзями" класів, похідних від даного.

Public член класу може використовуватися будь-якою функцією, яка є членом даного чи похідного класу, а також до **public** - членам можливий доступ ззовні через ім'я об'єкта.

Слід мати на увазі, що оголошення **friend** не є атрибутом доступу і не наслідується.

Синтаксис визначення похідного класу:

```
class ім'я: [private | protected | public] базовий_класс
{Тіло класу};
```

Якщо базових класів кілька, вони перелічуються через кому.

Ключ доступу може стояти перед кожним класом, наприклад:

```
class A {... };
class B {... };
class C {... };
class D: A, protected B, public C {...};
```

Приклад простого наслідування:

```
class Coord // базовий клас
{
int x, y;
```

```
public:
Coord (int _x, int _y) {x = _x; y = _y ;)
Coord () {x = 0; y = 0;}
int GetX () {return x;}
int GetY () {return y;}
void SetX (int _x) {x = _x;}
void SetY (int _y) {y = _y;}
}

class OutCoord: public Coord // похідний клас
{
public:
OutCoord (int _x, int _y): Coord (_x, _y) {}
void ShowX () {cout << GetX () << '';}
void ShowY () {cout << GetY () << '';}
};

main ()
{
OutCoord * ptr;
ptr = new OutCoord (10, 20); // Створення об'єкта в кучі
ptr-> ShowX ();
ptr-> ShowY ();
cout << "\ n";
delete ptr; // Видалення об'єкта з кучі
return 0;
}
```

Побудова конструктора:

```
<Констр_похід_класу> (<список_параметрів>):
<Констр_базов_класу> (<список_аргументів>)
{<Тіло_конструктора>}
```

Віртуальні функції.

До механізму віртуальних функцій звертаються в тих випадках, коли в кожному похідному класі потрібен свій варіант деякої компонентної функції. Класи, які включають такі функції, називаються **поліморфними** і відіграють особливу роль в ООП.

Віртуальні функції надають механізм пізнього (відкладеного) або динамічного звязування. Будь-яка нестатичні функція базового класу може бути зроблена віртуальною, для чого і використується ключове слово **virtual.**

Приклад:

```
class base
{
public:
```

```
virtual void print () {cout << "\ nbase";}
};
class dir: public base
{
public:
void print () {cout << "\ ndir";}
};
void main ()
{
base B, * bp = &B;
dir D, * dp = &D;
base * p = &D;
bp -> print (); // base
dp -> print (); // dir
p -> print (); // dir
}
```

Таким чином, інтерпретація кожного виклику віртуальної функції через покажчик на базовий клас залежить від значення цього покажчика, тобто від типу об'єкта, для якого виконується виклик.

Вибір того, яку віртуальну функцію викликати, буде залежати від типу об'єкта, на який фактично (у момент виконання програми) спрямований покажчик, а не від типу покажчика.

Віртуальними можуть бути тільки нестатичні функції-члени. Віртуальність успадковується. Після того як функція визначена як віртуальна, її повторне визначення в похідному класі (з тим же самим прототипом) створює в цьому класі нову віртуальную функцію, причому специфікатор **virtual** може не використовуватися.

Конструктори не можуть бути віртуальними, на відміну від деструкторів. Практично кожен клас, що має віртуальну функцію, повинен мати віртуальний деструктор.

Абстрактні класи.

Абстрактним називається клас, в якому є хоча б одна чисто (пуста) віртуальна фунція.

Чистої віртуальною функцією називається компонентна функція, яка має наступне визначення:

```
virtual тип імя_функциї (список_формальних_параметрів) = 0;
```

Чисто віртуальна функція нічого не робить і недоступна для викликів. Її призначення - служити основою для підміни нею функцій в похідних класах. Абстрактний клас може використовуватися тільки як **базовий** для похідних класів.

Механізм абстрактних класів розроблений для представлення спільних понять, які в подальшому передбачається конкретизувати. При цьому побудова ієрархії класів виконується по наступній схемі. В основі ієрархії стоїть абстрактний базовий клас. Він використовується для наслідування інтерфейсу. Похідні класи будуть конкретизовувати і реалізувувати цей інтерфейс.В абстрактному класі оголошені чисті віртуальні функції, які по суті є абстрактні методи.

Приклад абстрактного класу:

```
    class Base {
    public:
    Base (); // конструктор за замовчуванням
    Base (const Base &); // конструктор копіювання
    virtual ~ Base (); // віртуальний деструктор
    virtual void Show () = 0; // чисто віртуальна функція
    // Інші чисті віртуальні функції
    protected: // захищені члени класу
    private:
    // Часто залишається порожнім, інакше буде заважати майбутнім
розробкам
    };
    class Derived: virtual public Base {
    public:
    Derived (); // конструктор за замовчуванням
    Derived (const Derived &); // конструктор копіювання
    Derived (параметри); // конструктор з параметрами
    virtual ~ Derived (); // віртуальний деструктор
    void Show (); // перевизначена віртуальна функція
    // Інші перевизначені віртуальні функції
    // Інші перевантажені операції
    protected:
    //  Використовується замість private, якщо очікується
спадкування
    private:
    // Використовується для деталей реалізації
    };
```

Порядок виконання роботи.

1. Визначити ієрархію класів (відповідно до варіанта).

2. Визначити в класі статичну компоненту покажчик на початок зв'язаного списку об'єктів та статичну функцію для перегляду списку.

3. Реалізувати класи.

4. Написати демонстраційну програму, в якій створюються об'єкти різних класів і поміщаються в список, після чого список проглядається.

5. Зробити відповідні методи не віртуальними і подивитися, що буде.

6. Реалізувати варіант, коли об'єкт додається до списку при створенні, тобто в конструкторі.

Методичні вказівки.

1. Для визначення ієрархії класів зв'язати відношенням спадкування класи, наведені в додатку (для заданого варіанта). З перерахованих класів вибрати один, який буде стояти на чолі ієрархії. Це абстрактний клас.

2. Визначити у класах всі необхідні конструктори і деструктор.

3. Компонентні дані класу специфікувати як **protected.**

4. Приклад визначення статичних компонентів:

static person * begin; // покажчик на початок списку

static void print (void); // перегляд списку

5. Статичну компоненту-дане ініціалізувати поза визначення класу, в глобальній області.

6. Для додавання об'єкта в список передбачити метод класу, тобто об'єкт сам додає себе в список. Наприклад, **a.Add ()** об'єкт **a** додає себе в список.

Включення об'єкта в список можна виконувати при створенні об'єкта, тобто помістити оператори включення в конструктор. У разі ієрархії класів, включення об'єкта в список повинен виконувати тільки конструктор базового класу. Ви повинні продемонструвати обидва ці способи.

7. Список переглядати шляхом виклику віртуального методу **Show** кожного об'єкта.

8. Статичний метод перегляду списку викликати не через об'єкт, а через клас.

9. Визначення класів, їх реалізацію, демонстраційну програму помістити в окремі файли.

Варіанти завдань.

Перелік класів:

1) студент, викладач, персона, завкафедрою;
2) службовець, персона, робочий, інженер;
3) робітник, кадри, інженер, адміністрація;
4) деталь, механізм, виріб, вузол;
5) організація, страхова компанія, суднобудівна компанія, завод;
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, експрес;
12) двигун, двигун внутрішнього згоряння, дизель, турбореактивний двигун;
13) республіка, монархія, королівство, держава;
14) ссавці, парнокопитні, птиці, тварина;
15) корабель, пароплав, парусник, корвет.