Об’єктно-орієнтоване програмування

Лекція №1. Введення в ООП

Визначення ООП

Об’єктно-орієнтоване програмування – це особливий концептуальний підхід до проектування програм. Кожна складова стає самостійним об’єктом, який має власні коди та дані, що відносяться до цього об’єкту.

Основні засоби ООП

- Абстракція
- Інкапсуляція
- Поліморфізм
- Успадкування

Абстракція

- В комп'ютерних вирахуваннях абстракція – це ключовий крок у представленні інформації у термінах його інтерфейсу з користувачем.

Інтерфейс – це сумісно використовувана частина, що призначена для взаємодії двох систем (приклад: комп'ютер – принтер; користувач – програмне забезпечення)

Інкапсуляція

Інкапсуляція – це механізм, що об'єднує дані та код, що маніпулює цими даними, а також захищає і те і інше від зовнішнього втручання або невірного використання. Коли коди та дані об'єднуються таким чином, створюється об'єкт (Object).

Поліморфізм

- Поліморфізм – це можливість використовувати однакові імена для рішення двох чи більше схожих, але технічно різних задач. Метою поліморфізму в ООП є використання одного імені для завдання загальних для класу дій.

Успадкування

- Успадкування – це процес, засобами якого один об'єкт може отримувати властивості іншого. Тобто, об'єкт може успадковувати основі властивості іншого об'єкту та додавати до них особливості, характерні тільки для нього

Об'єкт

- Об'єкт – це інкапсульована абстракція, що включає в себе інформацію про стан та чітко визначену множину протоколу доступу (поведінка).

  Поведінка об'єкту визначається складом операцій, що можна виконувати над будь-яким екземпляром множини.

Клас

- Клас – множина об'єктів, об'єднаних загальністю структури та поведінки.

  Ідея класу полягає в об’єднанні даних і алгоритмів їх опрацювання.

  Дані називають **полями класу**, алгоритми – **методами**, а власне об’єднання – **інкапсуляцією.**

Визначення класу:

- Визначення ім'я класу (визначає новий тип);
- Визначення стану класу (склад, типи та імена полів у класі, для зберігання інформації, а також рівні їх захисту);

  Дані, що визначають стан класу, отримали назву членів-даних класу
- Визначення методів класу (визначення прототипів функцій, що забезпечать необхідну обробку інформації)

Створення нового класу

```
class <назва класу>
  {
    <специфікатор доступу>:
    <тип поля 1> <назви полів 1>;
    …
    <тип поля N> <назви полів N>;
    <декларації чи описи методів класу>;
  };
```

Специфікатори доступу

Специфікатор доступу	Опис
private	Доступність лише для методів класу
protected	Доступність лише для методів класу та методів похідних класів
public	Доступність для будь-якої зовнішньої функції

В описі класу специфікатор доступу може бути відсутній. Тоді за замовчуванням активним є специфікатор private, поки явно не задано інше.

Методи класу

- По функціональному призначенню методи класу поділяються на:
  - **Конструктори** – призначені для ініціалізації стану екземплярів класу при їх створенні;
  - **Деструктори** – призначені для виконання яких-то додаткових дій в момент знищення екземплярів класу;
  - **Селектори** – призначені для обробки стану класу без його зміни;
  - **Модифікатори** – призначені для зміни стану класу;
  - **Ітератори** – призначені для організації послідовного доступу до елементів даних, що визначають стан деякого екземпляру класу.

Методи класу

- По відношенню до класу методи поділяються на:
  - **Функція-член класу** – функція, що належить самому класу і не існує поза класом; прототипи функцій-членів класу включені до визначення класу
  - **Функція-друг класу** – зовнішня по відношенню до класу функція, що може існувати поза класом, але має доступ до закритої (та захищеної) частини класу. Прототип функції-друга класу також включається у визначення класу, але починається спеціальним ключовим словом **Friend**

Методи класу

Конструктори і деструктори класу можуть бути реалізовані тільки функціями-членами класу та мають спеціальний синтаксис. Інші методи класу мають загальний синтаксис функцій язика C++ та можуть бути реалізовані як функціями-членами та і функціями-друзями класу.

Приклад, клас TPoint, який містить координати точки і такі методи: засвічування, гасіння й переміщення точки

```
Class TPoint
  {
    protected:
        int x,y;     //Координати
    public:
        TPoint(int a, int b); //Ініціалізує поля координат числами a і в
        void On()          //Рисує точку поточним кольором
            {Draw(getcolor());}
        void Off()         //Витирає точку – малює її кольором фону
            {Draw(getbkcolor());}
        virtual void Draw(int color)    //Рисує точку кольором color
            {putpixel(x,y,color);}
        void Move(int dx, int dy);
  };
```

Методи класу TPoint

- Поза описом класу заголовок методи має вигляд:

<назва класу>::<назва методу>(<список формальних параметрів>)

Приклад:

```
TPoint::TPoint(int a, int b)
  {
    x=a; y=b;
  }
```

Методи класу TPoint

- Методи класу викликають так:

<назва об'єкту>.<назва методу>(<список фактичних параметрів>);

Оголосити й використати екземпляр Point класу TPoint можна так:

```
TPoint Point(50,50);
    Point.On();
    Point.Move(35,70);
    Point.Off();
    …
```

Або за допомогою динамічних змінних

```
TPoint* PointPtr=new Point(100,100);
    PointPtr->On();
    PointPtr->Move(35,70);
    PointPtr->Off();
```

Метод **operator**

Для класів визначений спеціальний метод operator, а саме:

<тип> operator<символ>(<формальні параметри>) {<тіло методу>}

У цьому разі як символ можна використовувати усі арифметичні операції, команду присвоєння, команди присвоєння, суміщені з арифметичними операціями та різні пари дужок, наприклад: operator+, operator=, operator() тощо.

Правила опису власних оператор-методів аналогічні до правил створення звичайних функцій чи методів,

Використовуючи клас TPoint та operator() нарисуйте 1000 точок, випадково розміщених на екрані.

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
Class TPoint
    {
    protected: int x, y;
    public:
            TPoint(int a=o, int b=o)
            {x=a; y=b;}
            void On()
                {Draw(gercolor());}
            void Off()
                {Draw(getbkcolor());}
      virtual   void Draw(int color)
                {putpixel(x,y,color);}
            TPoint& operator()(int i, int j)
                {x=i; y=j;
                return *this;}
};
```

Використовуючи клас TPoint та operator() нарисуйте 1000 точок, випадково розміщених на екрані.

```
void main()
{
    intgdriver=DETECT,gmode,errorcode;
    initgraph(&gdriver, &gmode, “”);
    TPoint P;
    randomize();
    for(int i=0; i<1000; i++)  P(random(i),random(j)).On();
    getch();
    closegraph();
}
```