Лекція 2

КЛАСИ ТА ОБ'ЄКТИ

План

1. Члени класу і керування доступом
2. Створення об’єктів класу
3. Конструктори і деструктори

1. Члени класу і керування доступом

Дані (поля) класу – змінні, які оголошені безпосередньо в класі.

Функції-члени (методи, операції) – функції, які описані всередині класу.

1. Члени класу і керування доступом

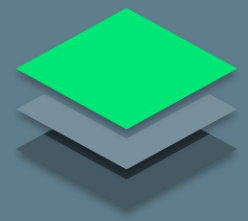
Модифікатори доступу

```
class Circle
{
private:
   double radius;

public:
   void setRadius(double r)
    {
      radius = r;
    }

   double getRadius()
   {
      return radius;
   }

   double getArea()
   {
      return radius*radius*3.1416;
   }
};
```

Ми не можемо **ініціалізувати поля** безпосередньо при оголошенні класу, а в int main() ми не маємо до них доступу, тому використовуємо **set-функцію** (сеттер)

get-функція (геттер) повертає значення закритих змінних-членів класу

1. Члени класу і керування доступом



Модифікатори доступу:

public (відкритий член класу)	член класу може використовуватися будь-якою функцією, яка є членом даного або похідного класу
private (закритий, внутрішній член класу)	член класу може використовуватися тільки функціями-членами і "дружніми" функціями даного класу, у похідному класі він недоступний
protected (захищений, внутрішній член ієрархії класів)	член класу може використовуватися функціями-членами і "дружніми" функціями, похідними від даного класу

2. Створення об’єктів класу

```
int main()
{
    Circle c1;
    c1.setRadius(2.5);
    cout<<"Radius="<<c1.getRadius()<<" Area="<<c1.getArea();
    return 0;
}
```

```
Radius=2.5 Area=19.635
--------------------------------
Process exited after 0.04658 seconds with return value 0
Press any key to continue . . .
```

2. Створення об’єктів класу

```
int main()
{
    Circle c1;
    c1.setRadius(2.5);
    cout<<"Circle1: Radius="<<c1.getRadius()<<" Area="<<c1.getArea()<<endl;

    Circle c2;
    c2.setRadius(4);
    cout<<"Circle2: Radius="<<c2.getRadius()<<" Area="<<c2.getArea()<<endl;
    return 0;
}
```

```
Circle1: Radius=2.5 Area=19.635
Circle2: Radius=4 Area=50.2656

--------------------------------
Process exited after 0.05193 seconds with return value 0
Press any key to continue . . .
```

2. Створення об’єктів класу

```
int main()
{
    Circle c1;
    c1.setRadius(2.5);
    cout<<"Circle1: Radius="<<c1.getRadius()<<" Area="<<c1.getArea()<<endl;

    Circle c2;
    c2.setRadius(4);
    cout<<"Circle2: Radius="<<c2.getRadius()<<" Area="<<c2.getArea()<<endl;

    Circle c3;
    cout<<"Circle3: Radius="<<c3.getRadius()<<" Area="<<c3.getArea()<<endl;
    return 0;
}
```

```
Circle1: Radius=2.5 Area=19.635
Circle2: Radius=4 Area=50.2656
Circle3: Radius=9.41436e-318 Area=0
```

3. Конструктори і деструктори

Конструктор – це метод класу, який призначений для **ініціалізації полів** класу (надання початкових значень).

Конструктор викликається сам автоматично під час **створення** об‘єкту класу (сеттер потрібно викликати).

```
class Circle
{
private:
    double radius;

public:
    Circle()
    {

    }
```

3. Конструктори і деструктори

```
class Circle
{
private:
   double radius;

public:
    Circle(double r)
    {
      radius = r;
    }

    double getArea()
    {
      double area = radius*radius*3.1416;
      cout<<"Area = "<<area<<endl;
    }
};
```

```
int main()
{
   Circle c1(2.5);
   c1.getArea();
   return 0;
}
```

```
Area = 19.635

--------------------------------
```

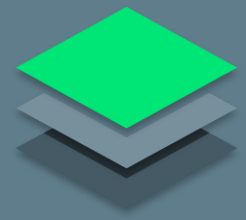
3. Конструктори і деструктори

```
class Circle
{
private:
   double radius;

public:
    Circle()
    {
      radius = 1;
    }

    Circle(double r)
    {
      radius = r;
    }

    double getArea()
    {
      double area = radius*radius*3.1416;
      cout<<"Area = "<<area<<endl;
    }
};
```

```
int main()
{
    Circle c1(2.5);
    c1.getArea();
    Circle c2;
    c2.getArea();
    return 0;
}
```

```
Area = 19.635
Area = 3.1416

--------------------------------
```

3. Конструктори і деструктори

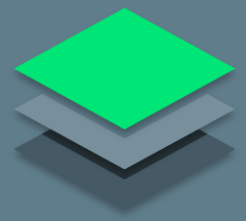


Якщо класі **не створено жодного конструктора**, то для класу створюється ***конструктор за замовчуванням*** без параметрів.

Якщо ви створюєте хоча б один **власний конструктор**, то конструктор за замовчуваний вже не створюється.

Якщо потрібен і конструктор ***за замовчуванням*** без параметрів, то його потрібно описати явно.

3. Конструктори і деструктори

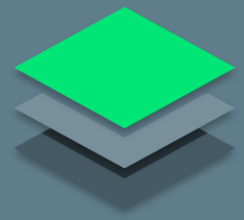


Деструктор – метод класу, який призначений для знищення об‘єкту класу.

Найчастіше його роль полягає в тому, щоб **звільнити динамічну пам'ять**, яку виділяв конструктор для об'єкта.

```
public:
    Circle(double r)
    {
      radius = r;
    }

    ~Circle()
    {

    }
```

3. Конструктори і деструктори



Властивості конструкторів та деструкторів:

- конструктори та деструктор завжди оголошуємо в розділі **public**;
- конструктор та деструктор **немає типу**, навіть void;
- **ім'я конструктора** співпадає з іменем **класу**;
- **ім'я деструктора** співпадає з іменем класу з **приставкою ~**;
- в класі можна створити **декілька конструкторів**, але їх синтаксис повинен відповідати правилам **перевантаження функцій**;
- в класі можна створити лише **один деструктор**.

Структура класу

```
class MyClass
{
 public:
 // елементи в цій секції доступні з будь-якої частини програми
 MyClass(); // конструктор
 ~MyClass(); // деструктор
 protected:
 // елементи в цій секції доступні з класу та його нащадків
 private:
 // елементи в цій секції доступні лише з класу;
 // це область доступу за замовчуванням
};
```

Домашнє завдання

Розробити **три варіанта** програми: створити клас **Circle** з полями **radius** та **color** (колір – це текст, наприклад “green”) для обчислення площі круга. На екран виводити радіус, колір та площу. Перевірити роботу програм, створивши декілька об'єктів.

1. Для встановлення та отримання значень кожного з полів використовувати **set-функції** та **get-функції**.
2. Замість **set-функцій** створити **конструктор з параметрами** для отримання значень відповідних полів та **конструктор за замовчуванням** (**radius = 1** та **color=“red”**).
3. ! Написати функції **setData()** для встановлення значень полів, значеннями, які ввів користувач з клавіатури та **getData()** для виведення радіуса та кольора на екран. Ці значення мають передаватися в конструктор з параметрами.

Корисні посилання

Модифікатори доступу

Геттери та сеттери

Конструктор класу

Перевантаження конструкторів

Деструктор класу