Лекція 3. Основи поняття класу

Об'єктно-орієнтоване програмування побудовано на понятті класу.

Почнемо з визначення термінів класу і об'єкту. Клас визначає новий тип даних, який задає формат об'єкта. Клас включає як дані, так і код, призначений для виконання дій над цими даними. Отже, клас пов'язує дані з кодом. У C++ специфікація класу використовується для побудови об'єктів. Об'єкти – це екземпляри класу. По суті, клас є набором планів, які визначають, як будувати об'єкт. Важливо розуміти, що клас – це логічна абстракція, яка реально не існує до тих пір, поки не буде створено об'єкт цього класу, тобто то, що стане фізичним представленням цього класу в пам'яті комп'ютера.

Визначаючи клас, ви оголошуєте дані, які він містить, і код, який виконується над цими даними. Хоча дуже прості класи можуть містити тільки код або тільки дані, більшість реальних класів містять обидва компоненти. У класі дані оголошуються у вигляді змінних, а код оформляється у вигляді функцій. Функції та змінні, складові класу, називаються його членами. Таким чином, змінна, яка оголошена в класі, називається членом даних, а функція, яка оголошена в класі, називається функцією-членом. Іноді замість терміна член даних використовується термін змінна екземпляра (або змінна реалізації).

Оголошення класу починається з ключового слова class.

Оголошення класу синтаксично подібно оголошенню структури. Розглянемо приклад.

Наступний клас визначає тип Cat, який описує представника сімейства котячих. Кожен кіт має ім’я та хвіст певної довжини.

```
// Створення класу Cat
class Cat
{
 string name; // ім'я кота
 int tail; // довжина хвоста кота
public:
 void init();
 void SetName(string name);
 string GetName();
 void SetTail(int tail);
 int GetTail();
};
```

Розглянемо детально оголошення цього класу.

Всі члени класу Cat оголошені в тілі інструкції class. Членами даних класу Cat є змінні name і tail. Крім того, тут визначено кілька функцій-членів: init(), SetName(string name), GetName(), SetTail(int tail) і GetTail().

Будь-який клас може містити як закриті, так і відкриті члени. За замовчуванням всі елементи, які визначені в класі, є закритими (private-члени). Наприклад, змінні name і tail є закритими. Це означає, що до них можуть отримати доступ тільки інші члени класу Cat, ніякі інші частини програми цього зробити не можуть. У цьому полягає один із проявів інкапсуляції: програміст в повній мірі може керувати доступом до певних елементів даних. Закритими можна оголосити і функції (в цьому прикладі таких немає), і тоді їх зможуть викликати тільки інші члени цього класу.

Ключове слово public використовується для оголошення відкритих членів класу.

Щоб зробити частини класу відкритими (тобто доступними для інших частин програми), необхідно оголосити їх після ключового слова public. Всі змінні або функції, визначені після модифікатора public, доступні для всіх інших функцій програми. Отже, в класі Cat функції init(), SetName(string name), GetName(), SetTail(int tail) і GetTail() є відкритими. Зазвичай в програмі організується доступ до закритих членів класу через його відкриті функції. Зверніть увагу на те, що після ключового слова public слід двокрапка.

Слід мати на увазі, що об'єкт утворює свого роду зв'язок між кодом і даними. Так, будь-яка функція-член має доступ до закритих елементів класу. Це означає, що функції init(), SetName(string name), GetName(), SetTail(int tail) і GetTail() мають доступ до змінних name і tail. Щоб додати функцію-член в клас, треба визначити її прототип в оголошенні цього класу.

Визначивши клас, можна створити об'єкт цього «класового» типу, використовуючи ім'я класу. Таким чином, ім'я класу стає модифікатором нового типу. Наприклад, при виконанні наступної інструкції створюється два об'єкти Tomas і Barsik типу Cat:

```
Cat Tomas, Barsik;
```

Після створення об'єкт класу матиме власну копію членів даних, які складають клас. Це означає, що кожен з об'єктів Tomas і Barsik буде мати власні окремі копії змінних name і tail. Отже, дані, пов'язані з об'єктом Tomas, відокремлені (ізольовані) від даних, пов'язаних з об'єктом Barsik.

Щоб отримати доступ до відкритого члену класу через об'єкт цього класу, використовується оператор «крапка» (саме так це робиться і при роботі зі структурами). Наприклад, щоб вивести на екран значення name, що належить об'єкту Tomas, використовується наступна інструкція:

```
cout << Tomas.GetName();
```

В C++ клас створює новий тип даних, який можна використовувати для створення об'єктів. Зокрема, клас створює логічну конструкцію, яка визначає відносини між її членами. Оголошуючи змінну класу, ми створюємо об'єкт. Об'єкт характеризується фізичним існуванням і є конкретним екземпляром класу. Кожен об'єкт займає певну область пам'яті. Кожен об'єкт класу має власну копію даних, визначених у цьому класі.

В оголошенні класу Cat містяться прототипи функцій-членів. Оскільки функції-члени забезпечені своїми прототипами у визначенні класу, їх не потрібно поміщати більше ні в яке інше місце програми.

Щоб реалізувати функцію, яка є членом класу, необхідно повідомити компілятору, до якого класу вона належить, назвавши ім'я цієї функції разом з ім'ям класу. Наприклад, ось як можна записати код функції SetName(string name):

```
void Cat::SetName(string name)
{
 this->name = name;
}
```

Оператор дозволу області видимості кваліфікує ім'я члена разом з ім'ям його класу.

Оператор "::" називається оператором дозволу області видимості. По суті, він повідомляє компілятору, що дана версія функції SetName(string name)

належить класу Cat. Іншими словами, оператор "::" заявляє про те, що функція SetName (string name) знаходиться в області видимості класу Cat.

Різні класи можуть використовувати однакові імена функцій. Компілятор ж визначить, до якого класу належить функція, за допомогою оператора дозволу області видимості і імені класу.

Функції-члени можна викликати тільки із заданого об'єкта. Щоб викликати функцію-член з частини програми, яка знаходиться поза класом, необхідно використовувати ім'я об'єкта і оператор "крапка". Наприклад, при виконанні цього коду буде викликана функція init () для об'єкта obj.

```
Cat obj;
obj.init();
```

Якщо одна функція-член викликає іншу функцію-член того ж класу, не потрібно вказувати ім'я об'єкта і використовувати оператор "крапка". В цьому випадку компілятор вже точно знає, який об'єкт піддається обробці. Ім'я об'єкта і оператор "крапка" необхідні тільки тоді, коли функція-член викликається кодом, розташованим поза класом. З тих же причин функція-член може безпосередньо звертатися до будь-якого члену даних свого класу, але код, розташований поза класом, повинен звертатися до змінної класу, використовуючи ім'я об'єкта і оператор "крапка".

У наведеній нижче програмі клас Cat ілюструється повністю (для цього об'єднані всі вже знайомі вам частини коду і додані відсутні деталі).

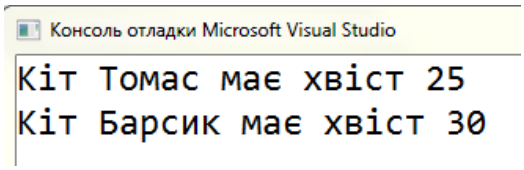
```
// Створення класу Cat
class Cat
{
 string name; // ім'я кота
 int tail; // довжина хвоста кота
public:
 void init();
 void SetName(string name);
 string GetName();
 void SetTail(int tail);
 int GetTail();
};

void Cat::init()
{
 name = "Томас";
 tail = 25;
}
```

```
void Cat::SetName(string name)
{
 this->name = name;
}
string Cat::GetName()
{
 return name;
}
void Cat::SetTail(int tail)
{
 this->tail = tail;
}
int Cat::GetTail()
{
 return tail;
}

int main()
{
 setlocale(0, "ukr");
 Cat Tomas, Barsik;
 Tomas.init();
 cout << "Кіт " << Tomas.GetName() << " має хвіст " << Tomas.GetTail() <<
endl;
 Barsik.SetName("Барсик");
 Barsik.SetTail(30);
 cout << "Кіт " << Barsik.GetName() << " має хвіст " << Barsik.GetTail()
<< endl;
}
```

При виконанні ця програма генерує такі результати:



Доступ до членів класу

Тепер розглянемо, як здійснюється доступ до членів класу Cat. Члени класу змінні name і tail оголошені без ключового слова private, а це означає, що ці змінні є прихованими, бо доступ private визначається за замовчуванням. Це і є реалізація принципу інкапсуляції. Закриті члени класу name і tail доступні тільки іншим членам того ж класу. До них не може отримати доступ та частина програми, яка існує поза цим класом. Яким чином можна отримати доступ до закритих членів? Принцип інкапсуляції передбачає для цього використання відкритого інтерфейсу, тобто відкритих методів (функцій), які реалізують свою власну внутрішню логіку взаємодії із закритими членами.

Для доступу до змінних name і tail створені функції, що дозволяють встановлювати і отримувати значення змінних name: SetName, GetName, SetTail(int tail) і GetTail(). Прийнято називати їх, використовуючи імена, які починаються відповідно на «Set» і «Get». На програмістському жаргоні їх ще називають ***сеттери і ґеттери*** (setters і getters). Як видно з приведеного коду, ці функції досить прості, але зверніть увагу на те, як описується функція SetName:

```
void Student::SetName(string name)
{
 this->name = name;
}
```

Тут в якості параметра використовується змінна на ім’я name, але таке ж саме ім'я має і відповідний член даних. Цю невизначеність, коли і ліворуч і праворуч від оператора присвоювання стоять різні змінні з однаковими іменами (ліворуч – назва члена даних, а праворуч – назва параметра) розв’язують за допомогою ключового слова this, це неявно визначений вказівник на поточний об'єкт класу. Зверніть увагу, що для доступу до членів класу через вказівник використовується оператор "->".

Коли доступ до деякого члену класу відбувається ззовні цього класу, його необхідно кваліфікувати (уточнити) за допомогою імені конкретного об'єкта.

Створюються об’єкти (екземпляри класу Cat) на ім’я Tomas і Barsik:

```
Cat Tomas, Barsik;
```

Виклик відкритої функції SetName ззовні класу Cat здійснюється за допомогою імені об'єкта Barsik:

```
Barsik.SetName("Барсик");
```