

Лабораторна робота №12

Класи як основа ООП. Об'єкти типу клас

Мета роботи:

- порівняти об'єктно-орієнтований та функціональний підходи у програмуванні;
- ознайомитись на практиці з класами, об'єктами та головними елементами об'єктного підходу;
- навчитись створювати і використовувати об'єкти типу клас;

Теоретичні відомості

Класи та об'єкти в C++

Класи та об'єкти в C++ є основними концепціями об'єктно-орієнтованого програмування (ООП). Об'єктно-орієнтоване програмування – розширення структурного програмування, в якому основними концепціями є поняття класів і об'єктів.

Класи в C++ – це абстракція, що описує методи і властивості ще не існуючих об'єктів. **Об'єкти** – конкретне уявлення абстракції, що має свої властивості та методи. Створені об'єкти на основі одного класу називаються **екземплярами** цього класу. Ці об'єкти можуть мати різну поведінку, властивості, але все одно будуть об'єктами одного класу. В ООП існує три основних принципи побудови класів:

1. Інкапсуляція – це властивість, що дозволяє об'єднати в класі і дані, і методи, які працюють з ними і приховати деталі реалізації від користувача.

2. Успадкування – це властивість, що дозволяє створити новий клас-нащадок на основі вже існуючого, при цьому всі характеристики класу батька присвоюються класу-нащадку.

3. Поліморфізм – властивість класів, що дозволяє використовувати об'єкти класів з однаковим інтерфейсом без інформації про тип і внутрішній структурі об'єкта.

Оголошення класів в C++

```

class // ім'я класу
{
    private:
        // Список властивостей і методів для використання всередині класу
    public:
        // Список методів доступних іншим функціям та об'єктам програми
protected:
        // список засобів, доступних при спадкуванні
};

```

Приклад. Програма, в якій оголошено найпростіший клас, в якому оголошена одна функція, яка друкує повідомлення.

```

#include <iostream>
using namespace std

        // оголошення класу
class CppClass
{
    public: // специфікатор доступу
        void message () // функція виводить повідомлення на екран
        {
            cout << "Classes and Objects in C++ \n\n\n";
        }
}; // Кінець оголошення класу CppStudio

int main (int argc, char * argv [])
{
    CppClass obj;           // Оголошення об'єкта
    obj.message();          // Виклик функції класу message
    system("pause");
}

```

Set-функції та Get-функції класів

Кожен об'єкт має якісь свої властивості або атрибути, які характеризують його впродовж усього життя. Атрибути об'єкта зберігаються в змінних, оголошених всередині класу, якому належить даний об'єкт. Причому, оголошення змінних повинно виконуватися зі специфікатором доступу `private`. Такі змінні називаються елементами даних або полями класу. Оскільки елементи даних оголошені в `private`, то і доступ до них можуть отримати тільки методи класу, зовнішній доступ до елементів даних заборонений. Тому прийнято оголошувати в класах спеціальні методи – так звані Set і Get функції, за допомогою яких можна маніпулювати елементами даних. Set-функції ініціалізують елементи даних, Get-функції дозволяють переглянути значення елементів даних. Доопрацюємо клас `CppClass` так, щоб у ньому можна було зберігати дату в

форматі дд.мм.гг. Для зміни і перегляду дати реалізуємо Set- і Get-функції.

```
#include <iostream>
using namespace std;

class CppClass // ім'я класу
{
private:
    int day, month, year;

public:
    void message () // функція виводить повідомлення на екран
    {
        cout << "\n\tClasses and Objects in C++ \n\n";
    }
    void setDate (int date_day, int date_month, int date_year)
    {
        day = date_day;           // Ініціалізація дня
        month = date_month;       // Ініціалізація місяця
        year = date_year;         // Ініціалізація року
    }
    void getDate () // відобразити поточну дату
    {
        cout<<"\n\tDate:"<<day<<"."<<month<<"."<<year<<endl<<endl;
    }
}; // Кінець оголошення класу CppStudio

int main ()
{
    setlocale (LC_ALL, "rus");
    int day, month, year;
    cout << "\n\tВведіть поточний день, місяць і рік: \n\n";
    cout << "\tдень : ";
    cin>>day;
    cout << "\tмісяць: ";
    cin >>month;
    cout << "\tрік : ";
    cin>>year;
    CppClass obj; // Оголошення об'єкта
    obj.message (); // Виклик функції класу message
    obj.setDate (day, month, year); // Ініціалізація дати
    obj.getDate (); // Відобразити дату
    system ("pause"); return 0;
}
```

Введіть поточний день, місяць і рік:

день : 30
місяць: 10
рік : 2019

Classes and Objects in C++

Date:30.10.2019

У визначенні класу специфікатор доступу `private` обмежує доступ до змінних, які оголошені після нього і до початку специфікатора доступу `public`. Таким чином, до змінних `day`, `month`, `year`, можуть отримати доступ тільки методи класу. Функції, що не належать класу, не можуть звертатися до цих змінних. Дані або методи класу, оголошені після специфікатора доступу `private`, але до початку наступного специфікатора доступу називаються закритими елементами даних і закритими методами класу.

Доцільно оголошувати елементи даних після специфікатора доступу `private`, а методи класу – після специфікатора `public`. Тоді, для маніпулювання елементами даних, оголошуються спеціальні функції – `get` і `set`.

В клас `CppClass` ми додали два методи `setDate()` і `getDate()`. Метод `setDate()` (set-функція) ініціалізує елементи даних. Тобто метод `setDate()` ініціалізує змінні `day`, `month`, `year`. Щоб переглянути, значення закритих елементів даних, оголошена функція `getDate()` (get-функція), яка повертає значення з змінних `day`, `month`, `year` у вигляді дати.

Конструктори і деструктори

Коли створюються елементи (змінні) класу, то не можливо присвоїти їм значення у самому визначенні класу. Компілятор видасть помилку. Тому необхідно створювати окремий метод (так звану set-функцію) класу, за допомогою якого і буде відбуватися ініціалізація елементів. При цьому, якщо необхідно створити, наприклад, 20 об'єктів класу, то прийдеться 20 разів викликати set-функції. Тут якраз зможе допомогти конструктор класу.

Конструктор (`construct` – створювати) – це спеціальний метод класу, призначений для ініціалізації елементів класу деякими початковими значеннями.

Деструктор (`destruct` – руйнувати) – спеціальний метод класу, який служить для знищення елементів класу. Найчастіше його використовують тоді, коли в конструкторі, при створенні об'єкта класу, динамічно була виділена ділянка пам'яті і необхідно цю пам'ять очистити, якщо ці значення вже не потрібні для подальшої роботи програми.

Важливо пам'ятати:

- 1) конструктор і деструктор завжди оголошуємо в розділі public;
- 2) при оголошенні конструктора тип повернення не вказується, в тому числі – void;
- 3) у деструктора так само немає типу повернення, деструктору не можна передавати ніяких параметрів;
- 4) ім'я класу і конструктора повинні бути ідентичними;
- 5) ім'я деструктора ідентично імені конструктора, але з приставкою ~;
- 6) у класі допустимо створювати декілька конструкторів, якщо це необхідно. Імена будуть однаковими, а компілятор буде їх розрізняти по переданим параметрами (перевантаження функцій). Якщо ми не передаємо в конструктор параметри, він вважається конструктором за замовчуванням;
- 7) у класі може бути оголошений лише один деструктор.

Приклад.

```
#include <iostream>
using namespace std;

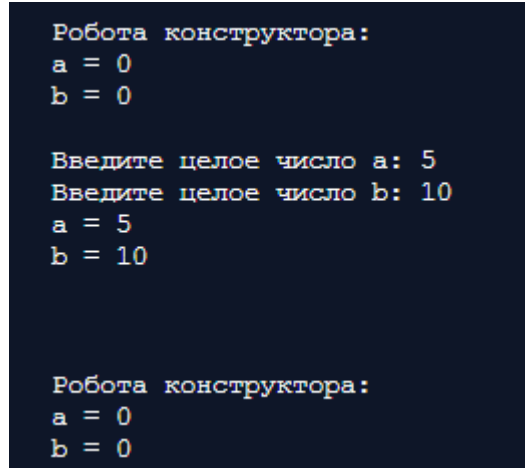
class AB
{
    private:
        int a; int b;
    public:
        AB()    //це конструктор
        {
            a = 0; b = 0;    //початкові значення полів
            cout << "\n\n\tРобота конструктора: " << endl;
            cout << "\ta = " << a << endl;
            cout << "\tb = " << b << endl << endl;
        }
        void setAB() // змінюємо початкові значення
        {
            cout << "\tВведіть ціле число a: "; cin >> a;
            cout << "\tВведіть ціле число b: "; cin >> b;
        }
        void getAB()
        {
            cout << "\ta = " << a << endl;
            cout << "\tb = " << b << endl << endl;
        }
};

int main()
{
    setlocale(LC_ALL, "rus");
```

```

AB obj1;           //спрацює конструктор для I об'єкта
obj1.setAB();      //задаємо нові значення
obj1.getAB();      //виводимо їх на екран
AB obj2;           //спрацює конструктор для II об'єкта
system("pause");
return 0;
}

```



```

Робота конструктора:
a = 0
b = 0

Введите целое число a: 5
Введите целое число b: 10
a = 5
b = 10

Робота конструктора:
a = 0
b = 0

```

Успадкування класів

Успадкування класів дозволяє створювати похідні класи (класи спадкоємці), взявши за основу всі методи й елементи базового класу (класу батька). Таким чином, економиться час на написання і налагодження коду нової програми. Об'єкти похідного класу вільно можуть використовувати все, що створено і налагоджено в базовому класі. При цьому можна в похідний клас дописати необхідний код для удосконалення програми: додати нові поля, методи і т. д. Базовий клас залишиться недоторканим. Нижче наведено код програми, в якій створено два класи: базовий – FirstClass і похідний від нього SecondClass.

Приклад

```

#include <iostream>
using namespace std;

class FirstClass    // базовий клас
{
protected:        // специфікатор доступу до елементу value
    int value;
public:
    FirstClass() {value = 0;}
    FirstClass(int x) {value = x;}
    void show_value() {cout << value << endl;}
};
class SecondClass : public FirstClass    // похідний клас
{
    public:

```

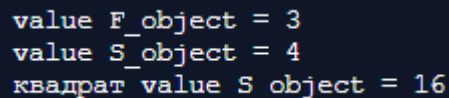
```

//конструктор класу SecondClass викликає конструктори класу
// FirstClass
SecondClass() : FirstClass() {}
SecondClass(int inputS) : FirstClass(inputS) {}
void ValueSqr() // Без специфікатора protected не змогли б змінити value
{
    value *= value;
}
};

int main()
{
    setlocale(LC_ALL, "rus");
    FirstClass F_object(3);    // об'єкт базового класу
    cout << "\n\tvalue F_object = ";
    F_object.show_value();
    SecondClass S_object(4);   // об'єкт похідного класу
    cout << "\tvalue S_object = ";
    S_object.show_value();     // виклик методу базового класу
    S_object.ValueSqr();       // підносимо value до квадрату
    cout << "\tkвадрат value S_object = ";
    S_object.show_value();
    // F_object.ValueSqr();     // ПОМИЛКА: немає доступу
    cout << endl;
    system("pause");
    return 0;
}

```

Результат:



```

value F_object = 3
value S_object = 4
квадрат value S_object = 16

```

Основна інформація про успадкування класів:

- 1) Успадкування – це визначення похідного класу, який може звертатися до всіх елементів і методів базового класу, за винятком тих, що перебувають у розділі private.
- 2) Похідний клас ще називають нащадком або підкласом, а базовий – батьківським, або надкласом, або суперкласом.
- 3) Синтаксис визначення похідного класу:

```
class Імя_Похідн_Класу: специфікатор_доступу Імя_Баз_Класу {...};
```
- 4) Похідний клас має доступ до всіх полів і методів базового класу, а базовий клас може використовувати тільки свої власні поля і методи.
- 5) У похідному класі необхідно явно визначати свої конструктори, деструктори

і перевантажені оператори присвоювання через те, що вони не успадковуються від базового класу. Але їх можна викликати явним чином при визначенні конструктора, деструктора або перевантаження оператора присвоєння похідного класу, наприклад, таким чином (для конструктора):

```
Конструктор_Похідн_Класу (...): Конструктор_Баз_Класу (...) {...}.
```

Порядок виконання роботи

1. Засвоїти теоретичний матеріал. Використовуючи лекційний матеріал та інші літературні джерела, навчитись описувати класи, засвоїти способи створення об'єктів.
2. Підготувати звіт з лабораторної роботи, у якому представити такі матеріали:
 - зміст завдання і варіант;
 - лістинг програми;
 - схему ієрархії класів програмного результати виконання;
 - вигляд екрану при виконанні програми;
 - результати роботи розробленого програмного засобу;
 - висновки.

Варіанти індивідуальних завдань

Варіант 1.

1. Створити клас STUDENT, який містить поля:
 - Name – Прізвище та ініціали;
 - Year – рік народження;
 - Bal – оцінки з 4 предметів (масив з 4 елементів).
2. Написати програму, що використовує даний клас і виконує дії:
 - вводить з клавіатури масив даних Group, що складається з N змінних типу STUDENT;
 - виводить на екран прізвища і рік народження студентів середній бал яких більше 4.0;

Варіант 2.

1. Створити клас SKLAD, який містить поля:
 - Name – Назва товару;
 - Type – одиниця вимірювання;
 - Quantity – кількість одиниць товару;
 - Cost – ціна одиниці товару.
2. Написати програму, що використовує даний клас і виконує дії:

- вводить з клавіатури масив даних SHOP, що складається з N змінних типу SKLAD;
- виводить на екран ціну та кількість товару, назва якого вводится з клавіатури або виводить повідомлення про його відсутність.

Варіант 3.

1. Створити клас TRAIN, який містить поля:

- Nazv – назва;
- Numer – номер поїзда;
- Date – дата відправлення;
- Time – час відправлення поїзда.

2. Написати програму, що використовує даний клас і виконує дії:

- вводить з клавіатури масив даних AVTOPARK що складається з N змінних типу TRAIN;
- виводить на екран всі рейси, які час відправлення яких після 15.00 по введеній даті.

Варіант 4.

1. Створити клас ABONENT, який містить поля:

- Name – прізвище абонента;
- Init – ініціали абонента;
- Nomer – номер телефону;
- Adress –домашня адреса.

2. Написати програму, що використовує даний клас і виконує дії:

- вводить з клавіатури масив даних TELEFON, що складається з N змінних типу ABONENT;
- виводить на екран прізвище, ініціали та домашню адресу за введеним номером телефону, або виводить повідомлення про його відсутність.

Варіант 5.

1. Написати клас DETAL, який містить поля:

- Name – назва деталі;
- Sort – сорт виробу;
- Date –дата виготовлення
- Quant – кількість;
- Cost - ціна деталі.

2. Написати програму, що використовує даний клас і виконує дії:

- вводить з клавіатури масив даних ZAKAZ, що складається з N змінних типу DETAL;

- виводить на екран всі деталі I сорту які виготовлені пізніше заданої дати, яка введена з клавіатури.

Варіант 6.

1. Написати клас BOOK, який містить поля:

- Name – назва книги;
- Avtor – автори книги;
- Data – дата друку;
- Cost - ціна книги.

2. Написати програму, що використовує даний клас і виконує дії:

- вводить з клавіатури масив даних SHOP, що складається з N змінних типу BOOK;
- виводить на екран всі книги які були надруковані в заданому році.

Варіант 7.

1. Написати клас TOVAR, який містить поля:

- Name – назва товару;
- Cost_Z – ціна закупки товару;
- Cost_P - ціна продажу товари.
- Quantity – кількість одиниць товару;
- Pributok – прибуток.

2. Написати програму, що використовує даний клас і виконує дії:

- вводить з клавіатури масив даних SHOP, що складається з N змінних типу TOVAR і обчислює прибуток по кожному товару;
- виводить на екран всі товари, які мають найбільший прибуток.

Варіант 9.

1. Написати клас VISTAVA, який містить і поля:

- Nazva – назва вистави;
- Date – дата вистави;
- Cost - ціна квитка.
- Day_week – день неділі;

2. Написати програму, що використовує даний клас і виконує дії:

- вводить з клавіатури масив даних THEATRE, що складається з N змінних типу VISTAVA;
- виводить на екран всі вистави і дати їх проходження, які відбудуться в заданий день тижня.

Варіант 10.

1. Написати клас VIDEO, який містить поля:

- Name – назва відео фільму;
- Year – рік зйомки фільму;
- Genre – жанр фільму;
- Rezhiser – режисер;

2. Написати програму, що використовує даний клас і виконує дії:

- вводить з клавіатури масив даних VIDEO_COLLECTION, що складається з N змінних типу VIDEO;
- виводить на екран список фільмів, по введеному з клавіатури режисеру.

Варіант 11

Користувач вводить масив трикутників.

Властивість: три сторони

Операції:

- збільшення / зменшення розміру сторін в задану кількість разів;
- обчислення периметра;
- обчислення площі;
- визначення значень кутів.

Користувач вибирає елемент масиву і виконувану операцію.

Контрольні питання

1. Що таке об'єктно-орієнтоване програмування?
2. Яка різниця між об'єктно-орієнтованим та функціональним програмуванням?
3. Що таке класи? Який синтаксис їх опису?
4. Що таке об'єкти та екземпляри класу?
5. Які основні принципи в об'єктно-орієнтованому програмуванні? Назвіть їх і дайте коротку характеристику сутностей цих принципів.
6. Наведіть приклад опису будь-якого класу та приклад створення екземпляру цього класу.
7. Для чого потрібні set- і get-функції?
8. Які специфікатори доступу використовуються в описах класів? Поясніть їх значення.
9. Дайте поняття закритих полів і методів класу.
10. Що таке конструктори і деструктори класу? Для чого їх використовують?
11. Які особливості конструкторів у класах?
12. Які особливості деструкторів у класах?