Практическое занятие № 19

Тема: Классы в С++

Цель работы. Изучить принципы создания и использования классов

Теоретическая часть

Классы и объекты в С++ являются основными концепциями объектно-ориентированного программирования (ООП).

Классы — это некоторые описания, по которым создаются объекты. Для создания объекта в ООП необходимо сначала составить классы. Классы имеют свои функции, которые называются методами класса. Каждый объект обладает свойствами. У каждого созданного объекта свойства могут различаться. Имея один класс, можно создать такое количество объектов, которое необходимо в задаче. При этом, каждый из объектов будет обладать одинаковым набором методов, внутренняя реализация которых определяется в классе.

Классы - это абстракция описывающая методы, свойства, ещё не существующих объектов. **Объекты** - конкретное представление абстракции, имеющее свои свойства и методы. Созданные объекты на основе одного класса также называются экземплярами этого класса. Эти объекты могут иметь различное поведение, свойства, но все равно являются объектами одного класса. Объектно-ориентированное программирование основывается на трех принципах, согласно которым строятся классы:

1. Принцип **инкапсуляции** позволяет объединить в классе данные и методы, работающие с ними, при этом, детали реализации скрываются от пользователя.
2. Принцип **наследования** служит для создания нового класса-потомка на основе уже существующего класса-родителя. При наследовании все характеристики (свойства, методы) класса-родителя присваиваются классу-потомку.
3. **Полиморфизм** - принцип ООП, позволяющий использовать объекты классов с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Классы в С++ объявляются следующим образом:

```
class /*имя класса*/
{
  private:
  /* список свойств и методов для использования внутри класса */
  public:
  /* список методов доступных другим функциям и объектам программы */
  protected:
  /*список средств, доступных при наследовании*/
};
```

Объявление класса начинается с зарезервированного ключевого слова **class**, после которого пишется имя класса. В фигурных скобках объявляется тело класса, причём после закрывающейся скобки обязательно ставится точку с запятой. В теле класса объявляются три метки спецификации доступа, после каждой метки ставится двоеточие. Все методы и свойства класса, объявленные после спецификатора **private** доступны только внутри класса. Все методы и свойства класса, объявленные после спецификатора **public** доступны другим функциям и объектам в программе. Все методы и свойства класса, объявленные после спецификатора **protected** доступны как внутри объекта класса, так и объектам, созданным на основе классов-потомков. При объявлении класса, не обязательно объявлять три спецификатора доступа, и не обязательно их объявлять в таком порядке.

Ниже представлена программа, в которой объявлен простейший класс с одной функцией, выводящей сообщение.

Пример 1

```
#include <iostream>
#include <cstdlib>
using namespace std;
// начало объявления класса
class CppStudio // имя класса
{
public: // спецификатор доступа
   void message() // функция (метод класса) выводящая сообщение на экран
   {
      cout << "Пример класса,\nв котором есть только один метод\n";
   }
}; // конец объявления класса CppStudio

int main(int argc, char* argv[])
{
   CppStudio objMessage; // объявление объекта
   objMessage.message(); // вызов метода message
   system("pause");
   return 0;
}
```

В **строках 4 - 11** определяется класс с именем **CppStudio**. Имя класса принято начинать с большой буквы, последующие слова в имени также начинают с большой буквы. В теле класса объявлен спецификатор доступа public, который позволяет вызывать другим функциям методы класса.
В классе **CppStudio** объявлена всего одна функция, которая не имеет параметров и выводит сообщение на экран. Методы класса — это те же функции, только объявлены они внутри класса, поэтому всё что относится к функциям актуально и для методов классов. Объявление классов выполняется аналогично объявлению функций, то есть класс можно объявлять в главном или в отдельном файле.
В **строке 15** объявлен объект **objMessage** класса **CppStudio**. Таким образом, класс является сложным типом данных. После того как объект класса объявлен, можно воспользоваться его методами, например, objMessage.message().

set-функции и get-функции классов

Каждый объект имеет какие-то свои свойства или атрибуты, которые характеризуют его. Атрибуты объекта хранятся в переменных, объявленных внутри класса, которому принадлежит данный объект. Как правило объявление переменных выполняется со спецификатором доступа private. Такие переменные называются элементами данных. Так как элементы данных объявлены в private, то доступ к ним могут получить только методы класса, внешний доступ к элементам данных запрещён. Поэтому принято объявлять в классах специальные методы — так называемые **set** и **get** функции, с помощью которых можно манипулировать элементами данных. **set**-функции инициализируют элементы данных, **get**-функции позволяют просмотреть значения элементов данных. Изменим класс CppStudio так, чтобы в нём можно было хранить дату в формате дд.мм.гг.

Для изменения и просмотра даты реализуем соответственно set и get функции.

Пример 2

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

class CppStudio // имя класса
{
private: // спецификатор доступа private
   int day, // день
      month, // месяц
      year; // год
public: // спецификатор доступа public
   void message() // функция (метод класса) выводящая сообщение на экран
   {
      cout << "\nПример класса с введением даты\n";
   }
   void setDate(int date_day, int date_month, int date_year) // установка даты в формате дд.мм.гг
   {
      day   = date_day; // инициализация день
      month = date_month; // инициализация месяц
      year  = date_year; // инициализация год
   }
   void getDate() // отобразить текущую дату
   {
      cout << "Date: " << day << "." << month << "." << year << endl;
   }
}; // конец объявления класса CppStudio

int main(int argc, char* argv[])
{
   int day, month, year;
   cout << "Введите текущий день месяц и год!\n";
   cout << "день: ";     cin >> day;
   cout << "месяц: ";    cin >> month;
   cout << "год: ";  cin >> year;
   CppStudio objCppstudio; // объявление объекта
   objCppstudio.message(); // вызов функции класса message
   objCppstudio.setDate(day, month, year); // инициализация даты
   objCppstudio.getDate(); // отобразить дату
   system("pause");
   return 0;
}
```

В определении класса был добавлен новый спецификатор доступа private, который ограничивает доступ к переменным, которые объявлены после него и до начала спецификатора доступа public. Таким образом, к переменным day, month, year, могут получить доступ только методы класса. Функции не принадлежащие классу, не могут обращаться к этим переменным. Элементы данных или методы класса, объявленные после спецификатора доступа private, но до начала следующего спецификатора доступа называются закрытыми элементами данных и закрытыми методами класса. Следуя принципу наименьших привилегий и принципу хорошего программирования, целесообразно объявлять элементы данных после спецификатора доступа private, а методы класса — после спецификатора доступа public. Тогда, для манипулирования элементами данных, объявляются специальные функции - get и set. В класс CppStudio добавили два метода setDate() и getDate(). Метод setDate() инициализирует переменные day, month, year. Чтобы просмотреть, значения в закрытых элементах данных объявлена функция getDate(), которая возвращает значения из переменных day, month,

year в виде даты. На этом определение класса закончено. В главном методе main(), создается объект класса, и через объект вызываются его методы. Если бы элементы данных были объявлены после спецификатора public, то к ним можно было бы обращаться точно также, как и к методам класса.

Конструкторы

В предыдущей программе, у класса CppStudio были объявлены элементы данных, которые могут хранить информацию о дате. После того, как был создан объект класса, была вызвана set-функция, для того, чтобы задать текущую дату (тем самым проинициализировать элементы данных), а потом была вызвана get-функция. Вызов get—функции до инициализации данных становится безсмысленным.

При создании объектов, можно сразу выполнять инициализацию элементов данных класса. **Конструктор** - специальная функция, участвующая в создании нового объекта класса, при помощи которой можно выполнить начальную инициализацию элементов данных. Имя конструктора обязательно должно совпадать с именем класса. Важным отличием конструктора от остальных функций является то, что он не возвращает никаких значений. В любом классе должен быть конструктор. Если конструктор явным образом не объявлен (как в предыдущих примерах), то компилятор предоставляет конструктор по умолчанию, без параметров.

Добавим в класс CppStudio конструктор.

Пример 3

```
#include <iostream>
#include <cstdlib>
using namespace std;
class CppStudio // имя класса
{
private: // спецификатор доступа private
   int day, // день
       month, // месяц
       year; // год
public: // спецификатор доступа public
   CppStudio(int date_day, int date_month, int date_year ) // конструктор класса
   {
      setDate(date_day, date_month, date_year); // вызов функции установки даты
   }
   void message() // функция (метод класса) выводящая сообщение на экран
   {
      cout << "\nПример 3\nОбъявление конструкторов\n";
   }
   void setDate(int date_day, int date_month, int date_year) // установка даты в формате дд.мм.гг
   {
      day   = date_day; // инициализация день
      month = date_month; // инициализация месяц
      year  = date_year; // инициализация год
   }
   void getDate() // отобразить текущую дату
   {
      cout << "date: " << day << "." << month << "." << year << endl;
   }
}; // конец объявления класса CppStudio
```

```
int main(int argc, char* argv[])
{
   CppStudio objCppstudio(11,11,2011); // объявление объекта и инициализация элементов данных
   objCppstudio.message(); // вызов функции message
   objCppstudio.getDate(); // отобразить дату
   system("pause");
   return 0;
}
```

В приведенном примере конструктор имеет три параметра, через которые в теле конструктора вызывается set-функция для установки даты. Можно реализовать начальную инициализацию элементов данных класса и без применения set — функции. При объявлении объекта класса, после имени объекта в круглых скобках заданы три аргумента.

Задания к работе

1. Изучите примеры, приведенные в теоретической части. В отчет включите код программ и скриншоты результатов их работы.
2. Напишите класс Triangle, который должен описывать прямоугольный треугольник. Предусмотрите в классе конструктор, через который треугольник задается по двум его катетам. Предусмотрите в классе set- и get-функции, а также функции вычисления площади треугольника и длины его гипотенузы.
3. **Напишите функцию сравнения двух треугольников с результатами: больше, меньше, равны.
4. Напишите программу, которая демонстрирует использование написанного класса.
5. В отчете приведите созданный код программы, программный код класса Triangle, скрин-шоты ее работы, а также пояснения к классу и программе.