Лекция 2. Классы и объекты.

Классы и объекты в C++ являются основными концепциями объектно-ориентированного программирования — ООП. Объектно-ориентированное программирование — расширение структурного программирования, в котором основными концепциями являются понятия классов и объектов. Основное отличие языка программирования C++ от C состоит в том, что в C нет классов, а следовательно язык C не поддерживает ООП, в отличие от C++.

Чтобы понять, для чего же в действительности нужны классы, проведём аналогию с каким-нибудь объектом из повседневной жизни, например, с велосипедом. Велосипед — это объект, который был построен согласно чертежам. Так вот, эти самые чертежи играют роль классов в ООП. Таким образом классы — это некоторые описания, схемы, чертежи по которым создаются объекты. Теперь ясно, что для создания объекта в ООП необходимо сначала составить чертежи, то есть классы. Классы имеют свои функции, которые называются методами класса. Передвижение велосипеда осуществляется за счёт вращения педалей, если рассматривать велосипед с точки зрения ООП, то механизм вращения педалей — это метод класса. Каждый велосипед имеет свой цвет, вес, различные составляющие — всё это свойства. Причём у каждого созданного объекта свойства могут различаться. Имея один класс, можно создать неограниченно количество объектов (велосипедов), каждый из которых будет обладать одинаковым набором методов, при этом можно не задумываться о внутренней реализации механизма вращения педалей, колёс, срабатывания системы торможения, так как всё это уже будет определено в классе. Разобравшись с назначением класса, дадим ему грамотное определение.

Классы в C++ — это абстракция описывающая методы, свойства, ещё не существующих объектов. **Объекты** — конкретное представление абстракции, имеющее свои свойства и методы. Созданные объекты на основе одного класса называются экземплярами этого класса. Эти объекты могут иметь различное поведение, свойства, но все равно будут являться объектами одного класса. В ООП существует три основных принципа построения классов:

Каждое свойство построения классов мы рассмотрим подробно по мере необходимости, а пока просто запомните эти три. А теперь вернёмся к классам, для начала рассмотрим структуру объявления классов.

```
1       // объявление классов в C++
2       class /*имя класса*/
3       {
4         private:
5         /* список свойств и методов для использования внутри класса */
6         public:
7         /* список методов доступных другим функциям и объектам программы */
8         protected:
9         /*список средств, доступных при наследовании*/
10      };
```

Объявление класса начинается с зарезервированного ключевого слова **class**, после которого пишется имя класса. В фигурных скобочках, **строки 3 — 10** объявляется тело класса, причём после закрывающейся скобочки обязательно нужно ставить точку с запятой, **строка 10**. В теле класса объявляются три метки спецификации доступа, **строки 4, 6, 8,** после каждой метки нужно обязательно ставить двоеточие. В **строке 4** объявлена метка спецификатора доступа private. Все методы и свойства класса, объявленные после спецификатор доступа private будут доступны только внутри класса. В **строке 6** объявлен спецификатор доступа public, все методы и свойства класса, объявленные после спецификатора доступа public будут доступны другим функциям и объектам в программе. Пока на этом остановимся, спецификатор доступа protected разбирать сейчас не будем, просто запомните, что он есть. При объявлении класса, не обязательно объявлять три спецификатора доступа, и не обязательно их объявлять в таком порядке. Но лучше сразу определиться с порядком объявления спецификаторов доступа, и стараться его придерживаться. Разработаем программу, в которой объявим простейший класс, в котором будет объявлена одна функция, печатающая сообщение.

```
1       // classes.cpp: определяет точку входа для консольного приложения.
```

```
2
3       #include "stdafx.h"
4       #include <iostream>
5       using namespace std;
6       // начало объявления класса
7       class CppStudio // имя класса
8       {
9       public: // спецификатор доступа
10          void message() // функция (метод класса) выводящая сообщение на экран
11          {
12              cout << "website: cppstudio.com\ntheme: Classes and Objects in C + +\n";
13          }
14      }; // конецобъявленияклассаCppStudio
15
16      int main(int argc, char* argv[])
17      {
18          CppStudio objMessage; // объявлениеобъекта
19          objMessage.message(); // вызовфункцииклассаmessage
20          system("pause");
21          return 0;
22      }
```

В **строках 7 — 14** мы определили класс с именем **CppStudio**. Имя класса принято начинать с большой буквы, последующие слова в имени также должны начинаться с большой буквы. Такое сочетание букв называют верблюжьим регистром, так как чередование больших и маленьких букв напоминает силуэт верблюда. В теле класса объявлен спецификатор доступа public, который позволяет вызывать другим функциям методы класса, объявленные после public. Вот именно поэтому в главной функции, в **строке 19** мы смогли вызвать функцию message(). В классе **CppStudio** объявлена всего одна функция, которая не имеет параметров и выводит сообщение на экран, строка **12**. 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 отдельном файле или в главном файле, позже посмотрим как это делается. В **строке 18** объявлена переменная objMessage типа CppStudio, так вот, переменная objMessage — это объект класса CppStudio. Таким образом, класс является сложным типом данных. После того как объект класса объявлен, можно воспользоваться его методами. Метод всего один — функция message(). Для этого обращаемся к методу объекта objMessage через точку, как показано в **строке 19**, в результате программа выдаст текстовое сообщение set — функции и get — функции классов

Каждый объект имеет какие-то свои свойства или атрибуты, которые характеризуют его на протяжении всей жизни. Атрибуты объекта хранятся в переменных, объявленных внутри класса, которому принадлежит данный объект. Причём, объявление переменных должно выполняться со спецификатором доступа private. Такие переменные называются элементами данных. Так как элементы данных объявлены в private, то и 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. Доработаем 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.

```
1       // classes.cpp: определяет точку входа для консольного приложения.
2
3       #include "stdafx.h"
4       #include <iostream>
5       using namespace std;
6
7       class CppStudio // имякласса
8       {
```

```
9       private: // спецификатордоступа private
10         int day, // день
11            month, // месяц
12            year; // год
13      public: // спецификатордоступа public
14         void message() // функция (метод класса) выводящая сообщение на экран
15         {
16            cout << "\nwebsite: cppstudio.comntheme: Classes and Objects in C + +\n";
17         }
18         void setDate(int date_day, int date_month, int date_year) // установкадатывформатедд.мм.гг
19         {
20            day   = date_day; // инициализация день
21            month = date_month; // инициализация месяц
22            year  = date_year; // инициализация год
23         }
24         void getDate() // отобразить текущую дату
25         {
26            cout<< "Date: " << day << "." <<month<< "." << year << endl;
27         }
28      }; // конецобъявленияклассa CppStudio
29
30      int main(int argc, char* argv[])
31      {
32         setlocale(LC_ALL, "rus"); // установкалокали
33         int day, month, year;
34         cout << "Введите текущий день месяц и год!\n";
35         cout << "день: ";     cin >> day;
36         cout << "месяц: ";    cin >> month;
37         cout << "год: ";  cin >> year;
38         CppStudio objCppstudio; // объявлениеобъекта
39         objCppstudio.message(); // вызовфункцииклассa message
40         objCppstudio.setDate(day, month, year); // инициализациядаты
41         objCppstudio.getDate(); // отобразитьдату
42         system("pause");
43         return 0;
44      }
```

В определении класса появился новый спецификатор доступа private, **строка 9**. Данный спецификатор доступа ограничивает доступ к переменным, которые объявлены после него и до начала спецификатора доступа public, **строки 9 — 12**. 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 мы добавили два метода setDate() и getDate(), подробно рассмотрим каждый метод. Метод setDate() определён с **18 по 23 строки**. Как уже ранее упоминалось, **set — функции** инициализируют элементы данных, поэтому метод setDate() выполняет именно такую функцию. То есть 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 main(), как и всегда, создаем объект класса, и через объект вызываем его методы,**строки 39 — 41**. Если бы элементы данных были объявлены после спецификатора public мы бы смогли к ним обратиться точно также, как и к методам класса.