**Классы и объекты в С++ 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 С++ от С состоит в том, что в С нет классов, а следовательно язык С не поддерживает ООП, в отличие от С++.**

**Классы в С++** — это абстракция описывающая методы, свойства, ещё не существующих объектов.

**Объекты** — конкретное представление абстракции, имеющее свои свойства и методы.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

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:

1. **Инкапсуляция** — это свойство, позволяющее объединить в классе и данные, и методы, работающие с ними и скрыть детали реализации от пользователя.
2. **Наследование** — это свойство, позволяющее создать новый класс-потомок на основе уже существующего, при этом все характеристики класса родителя присваиваются классу-потомку.
3. **Полиморфизм** — свойство классов, позволяющее использовать объекты классов с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | // classes.cpp: определяет точку входа для консольного приложения.    #include "stdafx.h"  #include <iostream>  **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 << "\nwebsite: cppstudio.comntheme: Classes and Objects in C + +\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    **int** main(**int** argc, **char**\* argv[])  {  **setlocale**(LC\_ALL, "rus"); // установка локали  **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;  } |

**Объектно-ориентированная декомпозиция** — разделение системы, выбрав в качестве критерия декомпозиции принадлежность ее элементов к различным абстракциям данной проблемной области. Абстракции описываются в виде объектов. Тогда каждый объект обладает своим собственным поведением, и каждый из них моделирует некоторый объект реального мира. С этой точки зрения объект является вполне осязаемой вещью, которая демонстрирует вполне определенное поведение. Объекты что-то делают, и мы можем, послав им сообщение, попросить их выполнить то-то и то-то.

**Принципиальное различие между структурным и объектно-ориентированным подходом заключается в способе декомпозиции системы. Объектно-ориентированный подход использует объектную декомпозицию, при этом статическая структура системы описывается в терминах объектов и связей между ними, а поведение системы описывается в терминах обмена сообщениями между объектами. Каждый объект системы обладает своим собственным поведением, моделирующим поведение объекта реального мира.**

**Также в структурным программировании нет классов, объектов, вместо классов используются структуры, функции. В ООП есть классы, объекты, методы, конструкторы, деструкторы. Но также можно использовать структуры, и функции.**