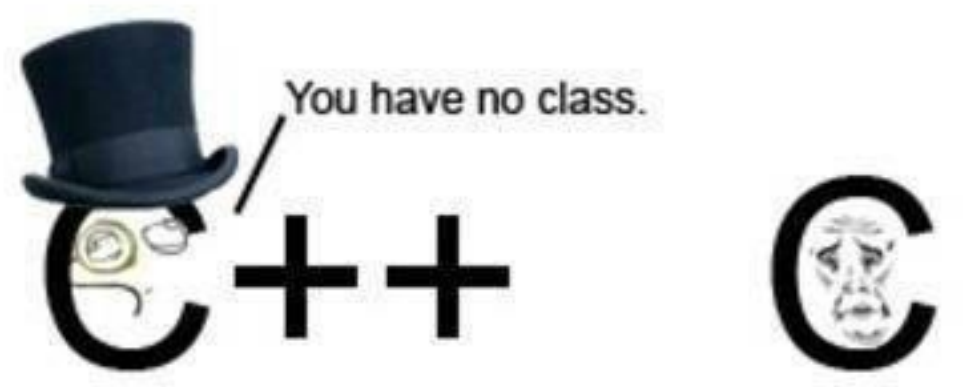
Основы программированя C++

Лекция 7. Классы и ключевые слова ООП

Мемчик



МЫ ПРИШЛИ К ООП!

Что такое ООП и с чем его едят?

Объектно-ориентированное программирование (ООП) - подход к программированию, при котором основными концепциями являются понятия объектов и классов.

Класс — это определяемый разработчиком абстрактный тип данных.

Объект — конкретное представление абстракции, имеющее свои свойства и методы. Созданные объекты на основе одного класса называются экземплярами этого класса.

Основные понятия ООП:

- **Инкапсуляция** — свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе.
- **Наследование** - свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью.
- **Полиморфизм** - свойство системы, позволяющее использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

На примере: машина
Класс «ВАЗ-2106»

Класс в C++

```
enum rultype { //Перечисление
    left, right
};

class VAZ2106 //Класс
{
    int Volume;
    char* Kuzov;
    rultype Rule;
    int YearOfProd;
};

struct VAZ2106 //Структура
{
    int Volume;
    char* Kuzov;
    rultype Rule;
    int YearOfProd;
};
```

Конструкция класса

```
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*/
};
```

Private, public, protected – метки доступа

Компоненты класса

- **Поле** - параметры объекта (конечно, не все, а только необходимые в программе), задающие его состояние. Иногда поля данных объекта называют свойствами объекта, из-за чего возможна путаница. Физически поля представляют собой значения (переменные, константы), объявленные как принадлежащие классу.
- **Метод - п**роцедуры и функции, связанные с классом. Они определяют действия, которые можно выполнять над объектом такого типа, и которые сам объект может выполнять.

Конструктор, деструктор, методы-свойства

- **Конструктор** - специальная функция, которая выполняет начальную инициализацию элементов данных, причём имя конструктора обязательно должно совпадать с именем класса.

Есть 3 вида конструкторов:

  - По умолчанию – class().
  - С параметрами – class(int i, …)
  - Копирования – class(class &obj)

- **Деструктор** - особый метод класса, который срабатывает во время уничтожения объектов класса. Чаще всего его роль заключается в том, чтобы освободить динамическую память, которую выделял конструктор для объекта.

Объявлен деструктор в классе как ~class()

Некоторые из полей потребуют сокрытия и правильной настройки поведения при изменении объекта. То есть прямой доступ к ним запрещён.

Если необходимо изменить объект или получить его, нужно реализовать методы-свойства, иначе называемые **get** и **set** (геттеры и сеттеры). С помощью них можно задать поведение для изменения или получения значений объекта.

Статические поля

Для того, чтобы обратиться к полям класса, нужен созданный объект этого класса. Но есть случаи, когда у класса есть константные поля или поля, не зависящие от объекта напрямую. Такие поля организуются как **статичные** и перед ними стоит атрибут **static.**

Необходимы такие поля бывают когда есть некие константы (как пример, число Pi, или число литров водки для создания ВАЗ-2106).

Задание

Организовать класс автомобиля ВАЗ-2109 малинового цвета. Реализовать все необходимые поля, конструкторы, деструкторы, геттеры, сеттеры, методы. Поля должны быть приватными.

Мемчик в конце