**Программирование на С++**

**9-11 классы**

**Занятие 7**

**Тема: Объектно-ориентированное программирование.**

**Классы в С++.**

**Цели и задачи:**

* Рассказать о разнице между процедурным и структурным программированием
* Рассказать о месте ООП в современных задачах
* Рассказать об основных характеристиках ООП
* Создать класса с конструктором и методами
* Создать объект класса и указатели на этот объект
* Получить доступ к методам и полям объекта класса

**По результатам занятия слушатель будет знать:**

* Что такое ООП
* Разницу между процедурным и объектно-ориентированном программированием

**По результатам занятия слушатель будет уметь:**

* Создавать классы в С++
* Работать с объектами класса и указателями на этот объект
* Создавать поля и методы класса

**Теоретическая часть**

**Введение в ООП**

Оглянитесь вокруг — везде находятся объекты: книги, здания, еда и даже Вы. **Объекты имеют два основных компонента**:

* **Свойства** (например: вес, цвет, размер, прочность, форма и т. д.)
* **Поведение**, которое они могут проявить (например: открывать что-либо, делать какие-то действия и т. д.)

Свойства и поведение неотделимы друг от друга.

**Объектно-ориентированное программирование** (сокр. **«ООП»**) предоставляет возможность создавать объекты, которые объединяют свойства и поведение в самостоятельный союз, который затем можно использовать.

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++.

**Аналогия**

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.

**Пример объекта класса и реализации 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.

**Классы в С++** — 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.

**Основные принципы ООП (Три кита ООП)**

В ООП существует три основных принципа:

* **Инкапсуляция** — это свойство, позволяющее объединить в классе и данные, и методы, работающие с ними и скрыть детали реализации от пользователя.
* **Наследование** — это свойство, позволяющее создать новый класс-потомок на основе уже существующего.
* **Полиморфизм** — свойство классов, позволяющее использовать объекты классов с одинаковым интерфейсом.

**Объявление классов**

Объявление класса начинается с зарезервированного ключевого слова class, после которого пишется имя класса. В фигурных скобочках объявляется тело класса, причём после закрывающейся скобочки обязательно нужно ставить точку с запятой. В теле класса объявляются две метки спецификации доступа, после каждой метки нужно обязательно ставить двоеточие. Все методы и свойства класса, объявленные после спецификатора доступа private будут доступны только внутри класса. Методы и свойства класса, объявленные после спецификатора доступа public будут доступны другим функциям и объектам в программе (т е вне определения класса).

**Понятие и реализация конструктора**

**Конструктором** класса называется специальная функция-член. Он выполняется (вызывается) при создании новых объектов класса.

Имя конструктора совпадает с именем класса. У конструктора нет возвращаемого значения (даже типа void)

**Определение класса в другом файле**

Важно грамотно объяснить ученикам, что залог хорошего кода- его структурированность, так как такой код легче поддерживать и читать.

**Вопросы для размышления**

* Что если названия скрытых полей класса совпадут с аргументами, передаваемыми в конструктор?
* Взаимодействие двух объектов одного класса (например, сложение)

(Данные вопросы нацелены на критическое мышление учеников, и в рамках данного занятия они являются риторическими. И первый, и второй вопросы будут разобраны в течение следующих занятий)

**Практическая часть**

**Задание 1**

Определить класс Child, который содержит такие поля (члены класса): закрытые – имя ребенка, фамилию и возраст , публичные – методы ввода данных и отображения их на экран. Объявить два объекта класса, внести данные и показать их.

\*

Сделать конструктор по умолчанию и конструктор

\*\*

Выполнить задание, используя указатели на объекты класса

(Файл с кодом лежит в папке code/)