**Программирование на С++**

**9-11 классы**

**Занятие 10**

**Тема: Шаблоны в С++.**

**Цели и задачи:**

* Рассказать о принципе полиморфизма в С++ и ООП в целом

**По результатам занятия слушатель будет знать:**

* Принцип полиморфизма (в функциях и классах

**Теоретическая часть**

**Шаблоны функций**

Функции и классы помогают писать программы легко, безопасно и делать их более удобными для управления.

Однако пока у функций и классов есть все эти преимущества, в некоторых случаях они могут быть отчасти лимитированы требованиями к языку C++.

Например, возможно вы захотите написать функцию, которая подсчитывает сумму двух чисел, подобно следующей (пример на слайде 6)

Становится необходимым писать новую функцию для каждого типа, например для double.

Не будет ли лучше написать одну версию функции sum() для работы с параметрами любого типа?

Шаблоны функций дают нам возможность сделать это!

Основная идея использования шаблонов функций заключается в отсутствии необходимости определения точного типа для каждой переменной. Кроме того, C++ нас обеспечивает способностью определения функций, с помощью типов, которые называются параметрами типа шаблона.

**Определение шаблонной функции**

Для определения шаблона функции, используйте ключевое слово template, после которого следует определение типа шаблона:

**template** <class T>

Компилятор автоматически вызывает функцию для соответствующего типа.

Шаблоны функций могут сохранить много времени, потому что они создаются один раз и работают с разными типами.

Шаблоны функций улучшают управление кодом, потому что сокращается количество дублированного кода.

Повышение безопасности является другим преимуществом в использовании шаблонов функций, в связи с тем, что нет необходимости вручную копировать функции и менять типы.

Шаблоны функций также позволяют работать с множеством типов. Определение происходит с помощью разделения запятыми.

**Шаблоны классов**

Так же как и с шаблонами функций, мы можем определить шаблоны классов, позволяя классам иметь элементы, которые используют параметры шаблонов в качестве типов.

Такой же синтаксис используется для определения шаблона класса:

**template <class T>**  
class MyClass

{  
 //code  
};

При определении ваших элементов функций вне класса, например, в другом исходном файле, будет использован специфический синтаксис.

Вам необходимо указать общий тип в угловых скобках после имени класса.

**Специализация Шаблона**

В случае регулярных шаблонов классов, способ хранения различных типов данных идентичен; для всех типов данных запускается один и тот же код.

Специализация шаблона позволяет определять различные исполнения шаблона, при передаче специфического типа в качестве аргумента.

Например, вам необходимо хранить символьный тип данных другим образом, не так, как мы делали с числовыми типами данных.

**Практическая часть**

**Задание 1**

Необходимо написать функцию, которая примет два числа, определит максимальное из них и вернет его в программу. Будем иметь ввиду, что в функцию мы можем передать числа разных типов. Возможен и случай, что одно число будет целым, а второе – вещественным.

\*

Проверить работоспособность программы с типом данных char и string

(Файл с кодом лежит в папке code/)