Школа:		
Дата:	**ФИО учителя:**	
Класс:	**Участвовали:**	**Не участвовали:**
Тема урока: Реализация функций на С++.		
Цели обучения, которые достигаются на данном уроке	Знать и использовать реализацию функции языка программирования С++	
Цели урока	**Все учащиеся смогут**: назвать и определить реализацию функции языка программирования С++ **Большинство учащихся смогут**: назвать назначение и записывать реализацию функции языка программирования С++ **Некоторые учащиеся смогут:** • применять операторы языка программирования С++ в решении задач	
Критерии оценивания	Учащийся: - поясняет принцип работы операторов; - исправляет ошибки в программе с оператором; - самостоятельно разрабатывает алгоритм и программу для решения задачи	
Воспитание ностей	Данный урок направлен на развитие ценностей академической честности, сплоченности и умения работать в команде, ответственности и лидерства. Привитие ценностей осуществляется посредством установления правил работы в группе, оказания поддержки менее способным учащимся.	
Предварительные знания	Основные операторы языка программирования С++	
Межпредметные связи	Математика	
Запланированные этапы урока	**Запланированная деятельность на уроке**	**Ресурсы**
Начало урока _7__ мин	**1. Организационный момент.** Мотивация к учебной деятельности. Целеполагание. Совместно с учащимися определяются цели урока **2. Повторение материала** с целью актуализации знаний.	Презентация

Середина урока _25__ мин	Очень часто в программировании необходимо выполнять одни и те же действия. Например, мы хотим выводить пользователю сообщения об ошибке в разных местах программы, если он ввел неверное значение. без функций это выглядело бы так:
	#include <iostream>
	#include <string>
	
	using namespace std;
	
	int main()
	{
	string valid_pass = "qwerty123";
	string user_pass;
	cout << "Введите пароль: ";
	getline(cin, user_pass);	

```
    if (user_pass == valid_pass) {
        cout << "Доступ разрешен." << endl;
    } else {
        cout << "Неверный пароль!" << endl;
    }
    return 0;
}
```

А вот аналогичный пример с функцией:

```
#include <iostream>
#include <string>

using namespace std;

void check_pass (string password)
{
    string valid_pass = "qwerty123";
    if (password == valid_pass) {
        cout << "Доступ разрешен." << endl;
    } else {
        cout << "Неверный пароль!" << endl;
    }
}

int main()
{
    string user_pass;
    cout << "Введите пароль: ";
    getline (cin, user_pass);
    check_pass (user_pass);
    return 0;
}
```

После компиляции не будет никакой разницы для процессора, как для первого кода, так и для второго. Но ведь такую проверку пароля мы можем делать в нашей программе довольно много раз. И тогда получается копипаста и код становится нечитаемым. Функции — один из самых важных компонентов языка C++.

Любая функция имеет тип, также, как и любая переменная.

- Функция может возвращать значение, тип которого в большинстве случаев аналогично типу самой функции.
- Если функция не возвращает никакого значения, то она должна иметь тип **void** (такие функции иногда называют процедурами)
- При объявлении функции, после ее типа должно находиться имя функции и две круглые скобки — открывающая и закрывающая, внутри которых

могут находиться один или несколько аргументов функции, которых также может не быть вообще.

- после списка аргументов функции ставится открывающая фигурная скобка, после которой находится само тело функции.
- В конце тела функции обязательно ставится закрывающая фигурная скобка.

Пример построения функции

```
#include <iostream>
using namespace std;

void function_name ()
{
   cout << "Hello, world" << endl;
}

int main()
{
   function_name(); // Вызов функции
   return 0;
}
```

Рассмотрим пример функции, возвращающей значение на примере проверки пароля.

```
#include <iostream>
#include <string>

using namespace std;

string check_pass (string password)
{
   string valid_pass = "qwerty123";
   string error_message;
   if (password == valid_pass) {
      error_message = "Доступ разрешен.";
   } else {
      error_message = "Неверный пароль!";
   }
   return error_message;
}

int main()
{
   string user_pass;
   cout << "Введите пароль: ";
   getline (cin, user_pass);
   string error_msg = check_pass (user_pass);
   cout << error_msg << endl;
   return 0;
```

}

В данном случае функция **check_pass** имеет тип **string**, следовательно она будет возвращать только значение типа string, иными словами говоря строку. Давайте рассмотрим алгоритм работы этой программы.

Самой первой выполняется функция **main()**, которая должна присутствовать в каждой программе. Теперь мы объявляем переменную **user_pass** типа string, затем выводим пользователю сообщение «Введите пароль», который после ввода попадает в строку user_pass. А вот дальше начинает работать наша собственная функция check_pass().

Также, можно организовать повторный ввод пароля с помощью **рекурсии** (о ней мы еще поговорим). Если объяснять вкратце, рекурсия — это когда функция вызывает сама себя. Смотрите еще один пример:

```
#include <iostream>
#include <string>

using namespace std;

bool password_is_valid (string password)
{
	string valid_pass = "qwerty123";
	if (valid_pass == password)
		return true;
	else
		return false;
}

void get_pass ()
{
	string user_pass;
	cout << "Введите пароль: ";
	getline(cin, user_pass);
	if (!password_is_valid(user_pass)) {
		cout << "Неверный пароль!" << endl;
		get_pass (); // Здесь делаем рекурсию
	} else {
		cout << "Доступ разрешен." << endl;
	}
}

int main()
{
	get_pass ();
	return 0;
}
```

Функции очень сильно облегчают работу программисту и

	намного повышают читаемость и понятность кода, в том числе и для самого разработчика (не удивляйтесь этому, т. к. если вы откроете код, написанный вами полгода назад,не сразу поймете соль, поверьте на слово). Не расстраивайтесь, если не сразу поймете все аспекты функций в C++, т. к. это довольно сложная тема и мы еще будем разбирать примеры с функциями в следующих уроках. Совет: не бойтесь экспериментировать, это очень хорошая практика, а после прочтения данной статьи порешайте элементарные задачи, но с использованием функций. Это будет очень полезно для вас. Если Вы найдете какие-либо ошибки в моем коде, обязательно напишите об этом в комментариях. здесь же можно задавать все вопросы.	
Конец урока **_8__ мин**	**4. Рефлексия** Учитель возвращается к целям урока, обсуждая уровень их достижения. Для дальнейшего планирования уроков учащимся задаются вопросы: - что узнал, чему научился; - что осталось непонятным; - над чем необходимо работать. Вопросы обсуждаются устно.	Стикеры

Дифференциация – каким образом Вы планируете оказать больше поддержки? Какие задачи Вы планируете поставить перед более способными учащимися?	**Оценивание – как Вы планируете проверить уровень усвоения материала учащимися?**	**Охрана здоровья и соблюдение техники безопасности**