Школа:		
Дата:	**ФИО учителя:**	
Класс:	**Участвовали:**	**Не участвовали:**
Тема урока: Инструктаж по ТБ. Синтаксис языка программирования С++. Отличия его от других языков.		
Цели обучения, которые достигаются на данном уроке	Знать и использовать синтаксисы языка программирования С++ Знать отличия от других языков	
Цели урока	**Все учащиеся смогут**: • назвать и определить синтаксисы языка **Большинство учащихся смогут**: • назвать назначение и записывать синтаксисы языка **Некоторые учащиеся смогут:** • применять операторы языка программирования С++ в решении задач	
Критерии оценивания	Учащийся: - поясняет принцип работы; - исправляет ошибки в ПРОГРАММЕ с оператором;	
Воспитание ценностей	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	Синтаксис языка программирования С++. Отличия его от других языков.	
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. Какие языки програмирования вы знаете? На каких из них вы можете написать программу? Какие средства нужны, что бы создать исполняемый файл из вашего кода?	Презентация

Середина урока _25__ мин	C++ — язык общего назначения и задуман для того, чтобы настоящие программисты получили удовольствие от самого процесса программирования. Язык программирования C++ задумывался как язык, который будет: лучше и современней языка C; поддерживать абстракцию данных; поддерживать объектно-ориентированное программирование. содержать большую и расширяемую стандартную библиотеку. За исключением второстепенных деталей, он практически содержит язык C как подмножество (хотя есть пример программы, которая является программой на языке C, но не может быть скомпилирована на языке C++). Язык C расширяется введением гибких и эффективных средств, предназначенных для построения новых типов. Программист структурирует свою задачу, определив новые типы, которые точно соответствуют понятиям предметной области задачи. Такой метод построения программы обычно называют абстракцией данных. Информация о типах содержится в некоторых объектах типов, определенных пользователем. С такими объектами можно работать надежно и просто даже в тех случаях, когда их тип нельзя установить на стадии трансляции. Программирование с использованием таких объектов обычно называют объектно-ориентированным. Если этот метод применяется правильно, то программы становятся короче и понятнее, а сопровождение их упрощается. Ключевым понятием C++ является класс. Класс — это определяемый пользователем тип. Классы обеспечивают инкапсуляцию ("упрятывание") данных, их инициализацию, неявное преобразование пользовательских типов, динамическое задание типов, контролируемое пользователем управление памятью и средства для перегрузки операций.	

	В языке C++ концепции контроля типов и модульного построения программ реализованы более полно, чем в C. Кроме того, C++ содержит усовершенствования, прямо с классами не связанные: символические константы, функции-подстановки, стандартные значения параметров функций, перегрузка имен функций, операции управления свободной памятью и ссылочный тип. В C++ сохранены все возможности C эффективной работы с основными объектами, отражающими аппаратную "реальность" (разряды, байты, слова, адреса и т.д.). Это позволяет достаточно эффективно реализовывать пользовательские типы. Как язык, так и стандартные библиотеки C++ проектировались в расчете на переносимость. Имеющиеся реализации языка будут работать в большинстве систем, поддерживающих C. В программах на C++ можно использовать библиотеки C. Большинство служебных программ, рассчитанных на C, можно использовать и в C++. Можно сказать, что Си и C++ сосуществуют между собой. Когда в 2011 году вышел новый стандарт языка C++ — C++11, вместе с ним вышел и стандарт языка Си — C11. Все заголовочные файлы стандартной библиотеки языка C++ не содержат расширения .h. Например: #include<iostream> #include<vector> #include<algorithm> Заголовочные файлы из стандартной библиотеки языка C можно использовать в языке C++, но их имена изменились - в начало файла добавилась буква "c", а расширение ".h" исчезло. То есть при желании использовать функции, которые в языке C определены в заголовочных файлах stdio.h или math.h, их требуется подключать следующим образом: #include<cstdio> #include<cmath>	

Язык C++ содержит обширную стандартную библиотеку.
Основные части библиотеки следующие:

- Ввод-вывод описан в заголовочных файлах iostream, fstream и других.
- Работа со строками описана в файле string и других.
- Контейнеры (структуры данных) описаны в большом числе заголовочных файлов в соответствии с типом контейнера. Например, vector - динамический массив, set - множество с возможностью быстрого добавления, удаления, поиска элементов, map - ассоциативный массив (словарь), list - двусвязный список, stack - стек, queue - очередь.
- Алгоритмы (линейный и двоичный поиск, нахождение следующей перестановки, случайная перестановка) описаны в заголовочном файле algorithm.

Первоначально часть стандартной библиотеки называлась STL - Standard Template Library и развивалась независимо от языка C++ компаниями HP и SGI. Затем она была добавлена в стандарт языка, но название STL сохранилось и часто употребляется применительно к той части библиотеки, которая относится к контейнерам и алгоритмам.
Имена (функций, переменных) в языке C++ можно разделять на "пространства имен" для удобства - чтобы могли существовать функции и переменные с одинаковыми именами в разных "пространствах имен".
Пространство имен объявляется так:

```
namespace my_namespace
{
   // Описание функций, переменных, классов
   int var;
};
```

Для доступа к переменной var из пространства имен my_namespace нужно

	писать my_namespace::var. Можно также использовать инструкцию: using namespace my_namespace; тогда все имена из пространства имен my_namespace можно использовать без указания имени пространства имен (просто писать var). Вся стандартная библиотека находится в пространстве имен std, поэтому нужно либо писать std::cout << a << std::endl; для вывода переменной a. Или написать в начале программы using namespace std; и тогда можно будет просто писать cout << a << endl;	
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**