Содержание

[ОБЩАЯ ИНФОРМАЦИЯ 2](#_heading=h.gjdgxs)

[Тема занятия](#_heading=h.30j0zll) 2

[Цели и задачи 2](#_heading=h.1fob9te)

[Ожидаемый результат 2](#_heading=h.3znysh7)

[Структура занятия 3](#_heading=h.tyjcwt)

[ОПИСАНИЕ ЗАНЯТИЯ 4](#_heading=h.3dy6vkm)

# ОБЩАЯ ИНФОРМАЦИЯ

### Тема занятия

### Исполняемые файлы. Стандартный поток вывода.

Переменные и типы данных.

### Цели и задачи

* Научиться запускать программы на С++
* Понять, что такое компилятор
* Написать первую программу
* Познакомиться со стандартным вводом и выводом в С++
* Познакомиться с переменными и типами данных
* Научиться использовать арифметические операции с переменными

### Ожидаемый результат

По результатам занятия Слушатель должен знать:

* Что такое язык программирования
* Как запускать программный код, написанный на языке С++
* Что такое компилятор и зачем он нужен
* Что такое переменная
* Что такое стандартный ввод и вывод
* Что такое тип данных
* Что такое комментарий

По результатам занятия Слушатель должен уметь:

* Работать в текстовом редакторе
* Запускать код, написанный на языке С++
* Работать с компилятором
* Создавать переменные в С++
* Работать со стандартным вводом и выводом
* Использовать арифметические операции с переменными

### Структура занятия

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Тайминг занятия** | |  |  |  |  |
| № | Этапы | Что делает преподаватель | Что делает Слушатель | Время | Общее время |
| 1 | Приветственное слово преподавателя |  |  | 5 мин. | 5 мин. |
| 2 | Повторение пройденного |  |  | 5 мин. | 40 мин. |
| 3 | Теоретическая часть |  |  | 30 мин. |
| 4 | Вопросы по теоретической части |  |  | 5 мин. |
|  | *Перерыв* |  |  | 15 мин. | 15 мин. |
| 5 | Практическая часть |  |  | 40 мин. | 45 мин. |
| 6 | Рефлексия и вопросы |  |  | 5 мин. |

# ОПИСАНИЕ ЗАНЯТИЯ

**Теоретическая часть**

**Преамбула**

Компьютер понимает только определённые команды, а код – это всего лишь несколько команд для компьютера, расположенных в определённом порядке.

Код на определённом языке программирования – это ни что иное, как обычный текст со строгими правилами (синтаксисом)

Вспомните, что и в русском языке есть свой синтаксис!

Программисты пишут код, код – это просто текст, сохранённый в файле с определённым расширением. Каким образом нам «исполнить» или «выполнить» код, который мы будем писать?

Чтобы ответить на этот вопрос нам надо понять, а что можем «запустить» на своём компьютере.

**Исполняемые файлы (\*.exe)**

Любой файл, который вы можете запустить на своём компьютере является исполняемым.

В Windows наиболее распространены бинарные (binary -> 0 и 1) исполняемые файлы. Самый часто встречающийся их вид - это приложение. Приложения имеют расширения EXE и могут запускаться самостоятельно.

Теперь можно переформулировать наш вопрос: как нам перевести наш код в бинарный файл, который мы можем запустить?

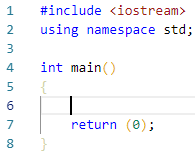
**Как превратить код в язык, понятный компьютеру**

Чтобы превратить код в исполняемый файл создали специальную программу, которую назвали компилятором.

Если забывать освобождать выделенную (динамически) память, то образуются утечки памяти, потому что эта память остаётся выделенной, пока программа не будет закрыта.

Написанный нами код должен быть сохранён в файле с специальным расширением - .cpp (С Plus Plus – C++)

**«Точка входа» в программу на С++**



На самом деле, любая программа чаще всего существует не сама по себе. Она может общаться с другими программами, системами, интернетом и т.д.

Под словом “общаться” мы в первую очередь подразуемеваем “обмениваться данными”. То есть, принимать какие-то данные извне, а собственные данные — наоборот, куда-то отправлять.

Примеров обмена данными между программами много даже в повседневной жизни.

Так, на многих сайтах ты можешь вместо регистрации авторизоваться при помощи своего аккаунта в Facebook или Twitter. В этой ситуации две программы, скажем, Twitter и сайт, на котором ты пытаешься зарегистрироваться, обмениваются необходимыми данными между собой, после чего ты видишь конечный результат — успешную авторизацию.

**Потоки в программирования**

Для описания процесса обмена данными в программировании часто используется термин “поток”.

Откуда вообще взялось такое название? “Поток” больше ассоциируется с рекой или ручьем, чем с программированием. Поток — это, по сути, перемещающийся кусок данных. То есть в программировании по потоку “течет” не вода, а данные в виде байтов и символов.

Из потока данных мы можем получать данные частями и что-то с ними делать.

При помощи потоков ты можешь работать с любыми источниками данных: интернет, файловая система твоего компьютера или что-то еще — без разницы. Потоки — инструмент универсальный. Они позволяют программе получать данные отовсюду (входящие потоки) и отправлять их куда угодно (исходящие). Их задача одна — брать данные в одном месте и отправлять в другое.

Потоки делятся на два вида:

Входящий поток (Input) — используется для приема данных

Исходящий поток (Output) — для отправки данных.

**Комментарии**

Комментариями называются пояснительные выражения, которые вы можете включать в ваш код на языке C++, чтобы объяснить, что именно выполняет программа.

Компилятор игнорирует все, что находится в комментариях. Это значит, что их не будет видно в результате выполнения программы**.**

Комментарий, который начинается с двух слэшей (//), называется однострочным комментарием. Комбинация двух слэш символов указывает компилятору игнорировать все, что следует за ними, вплоть до окончания строки.

**Многострочные комментарии**

Комментарии, в которых необходимо использование множества строк начинаются с /\* и заканчиваются \*/

Вы можете поместить их на одной строке, или же поместить одну и более строк между ними.

**Переменные**

Создание переменной резервирует место, или пространство в памяти для хранения значений. Компилятору необходимо, чтобы вы указали тип данных для каждой объявляемой переменной.

C++ предлагает большой ассортимент встроенных типов данных.

Целочисленный тип, встроенный тип, представляет собой целое число. Для определения переменной целочисленного типа используется ключевое слово int (integer).

C++ требует чтобы вы указали тип и идентификатор (имя) для каждой переменной.

Идентификатор это имя для переменной, функции, класса, модуля, или чего-либо другого определенного пользователем. Идентификатор начинается с буквы (A-Z или a-z) или нижнего подчеркивания (\_), с последующими дополнительно буквами, нижними подчеркиваниями, и цифрами (от 0 до 9).

Например, определим переменную под названием myVariable которая может хранить целочисленные значения

**Переменные**

Определяйте все переменные с именем и типом данных до их использования в программе. В случае, если у вас есть несколько переменных одинакового типа, можно определять их в одном объявлении, разделяя их запятыми.

int a, b;

Переменным могут быть присвоены значения и они могут использоваться для выполнения операций.

Например, мы можем дополнительно создать переменную sum, и сложить две переменные.

**Объявляем переменные**

У вас есть возможность присвоить значение переменной во время ее объявления или объявить переменную и присвоить ей значение позже.

Вы также можете изменить значение переменной.

Несколько примеров:

**int a;   
int b = 42;  
  
a = 10;  
b = 3;**

**Ввод пользователем (стандартный поток ввода)**

Чтобы позволить пользователю ввести значение используйте cin вместе с оператором извлечения (>>). Переменная содержащая извлекаемую информацию следует за оператором.

Следующий пример показывает, как принимать введенную пользователем информацию и сохранять ее в переменной num:

**int num;**

**cin >> num;**

**Получение введенной информации**

Следующая программа подсказывает пользователю ввести число и сохраняет его в переменной a:

(пример на слайде 13)

После запуска программы выводится сообщение "Please enter a number", затем ожидается ввод пользователем числа и нажатие кнопки Enter.

Введенное число сохраняется в переменной a.

Программа будет ждать столько времени, сколько необходимо пользователю чтобы ввести число.

**Получение введенной информации**

Вы можете выполнить ввод пользователем информации множество раз, как сделано в следующей программе:

(пример на слайде 14)

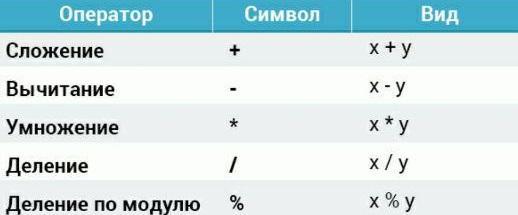
**Получение введенной информации**

Давайте создадим программу, которая позволяет ввести два числа и выводит на экран их сумму.

(пример на слайде 15)

**Арифметические операторы**

C++ поддерживает следующие арифметические операторы.



C++ имеет короткие операторы одновременного выполнения операции и присваивания.

int x = 10;  
**x += 4;** // equivalent to x = x + 4  
**x -= 5;** // equivalent to x = x – 5

**x \*= 3;** // equivalent to x = x \* 3  
**x /= 2;** // equivalent to x = x / 2  
**x %= 4;** // equivalent to x = x % 4

**Оператор инкремента**

Оператор инкремента используется для увеличения целочисленного значения на единицу.

Следующие выражения эквивалентны:

**x++; x +=1; x = x + 1;**

**Оператор декремента**

Оператор декремента (--) работает почти таким же образом, как и оператор инкремента, но вместо увеличения значения, он уменьшает его на единицу.

**x--; x -=1; x = x - 1;**

**Практическая часть**

**Задание1**

Выведите на экран (отправьте в стандартный поток вывода) приветственное предложение с пользователем. Затем представьтесь (с новой строки)

**Задание2**

Пользователь вводит 2 числа. Ваша программа в стандартный вывод (на экран) посылает сумму этих чисел, и разницу и произведение.

\*

Также выведите среднее арифметическое двух чисел (среднее арифметическое нескольких чисел – это их сумма заданных чисел, делённая на их количество)

\*\*

Выведите на экран поясняющие предложение, например, “Hello!” , “Enter first number” и т.д.

\*\*\*

Не используйте промежуточные переменные для хранения результатов арифметических операций.