9-11 классы

Программирование
на C++

Презентация занятия

Стандартный поток вывода. Переменные и типы данных.

1 занятие

КОД БУДУЩЕГО

Минцифры России

инжиниринум
МГТУ им. Н.Э. Баумана

НАЦИОНАЛЬНЫЕ ПРОЕКТЫ РОССИИ
ЦИФРОВАЯ ЭКОНОМИКА

20.35 УНИВЕРСИТЕТ

2020

Программирование
на C++

Теоретическая часть

Стандартный поток вывода.
Переменные и типы данных.

1 занятие

инжиниум®
МГТУ им. Н.Э. Баумана

2020

Тема: Стандартный поток вывода. Переменные и типы данных.

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.

Тема: Стандартный поток вывода. Переменные и типы данных.

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

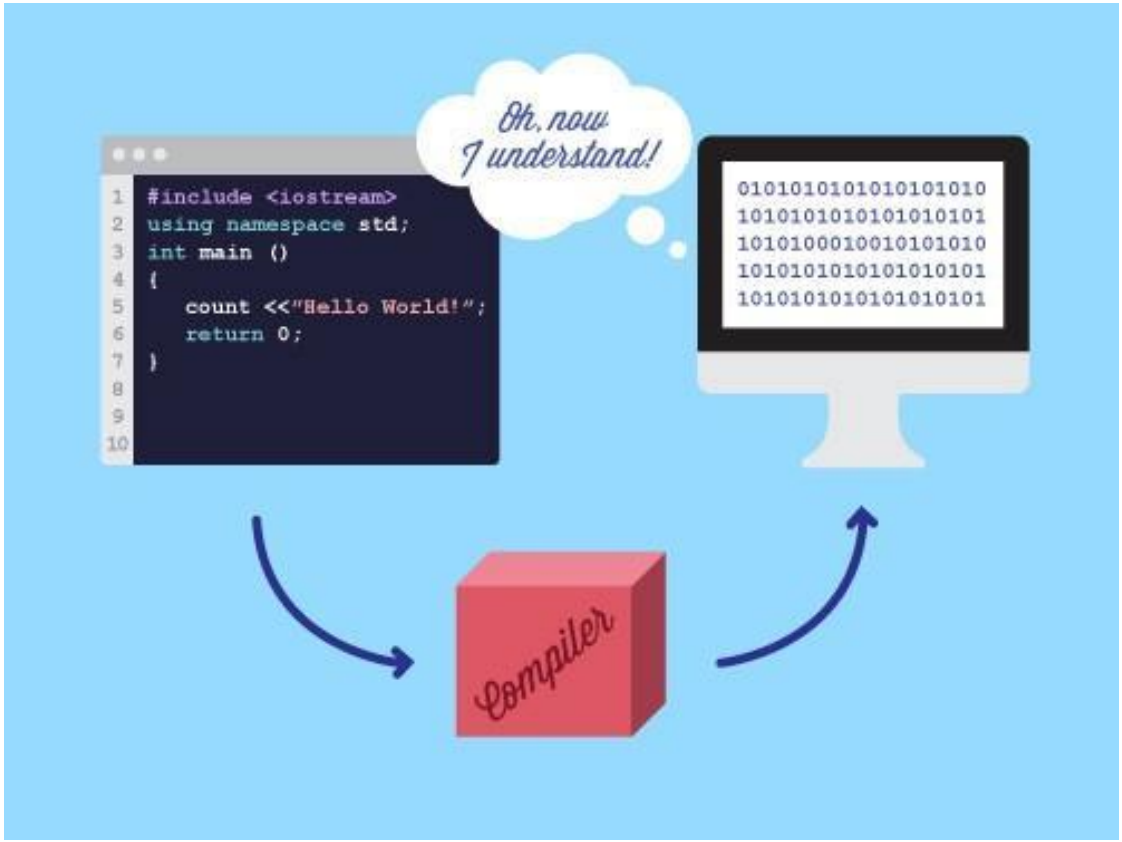
В Windows наиболее распространены бинарные (binary -> 0 и 1) исполняемые файлы. Самый часто встречающийся их вид - это приложение. Приложения имеют расширения EXE и могут запускаться самостоятельно.

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

*.exe

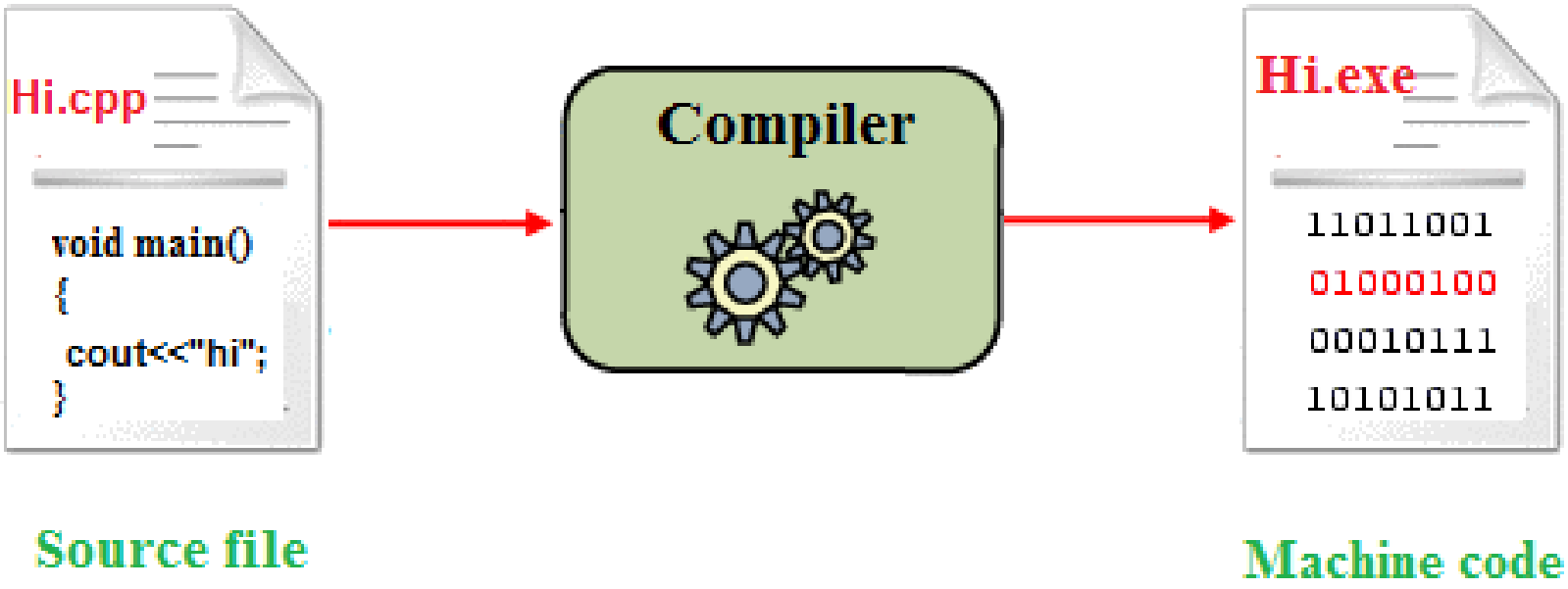
Тема: Стандартный поток вывода. Переменные и типы данных.

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



Тема: Стандартный поток вывода. Переменные и типы данных.

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



Тема: Стандартный поток вывода. Переменные и типы данных.

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6
7       return (0);
8   }
```

Тема: Стандартный поток вывода. Переменные и типы данных.

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.

Тема: Стандартный поток вывода. Переменные и типы данных.

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.

Тема: Стандартный поток вывода. Переменные и типы данных.

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       cout << "Hello, world!";
7       return (0);
```

Тема: Стандартный поток вывода. Переменные и типы данных.

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       cout << "Hello, world!";
7       cout << "Hello, world!";
8       cout << "Hello, world!";
9       return (0);
10  }
```

```
Hello, world!Hello, world!Hello, world!
```

Тема: Стандартный поток вывода. Переменные и типы данных.

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       cout << "Hello, world!" << endl;
7       cout << "Hello, world!" << endl;
8       cout << "Hello, world!" << endl;
9       return (0);
10  }
```

```
Hello, world!
Hello, world!
Hello, world!
```

Тема: Стандартный поток вывода. Переменные и типы 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.

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       cout << "Hello world!"; // prints "Hello world"
7       return (0);
8   }
```

Тема: Стандартный поток вывода. Переменные и типы данных.

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.

```
1   #include <iostream>
2   using namespace std;
3
4   /*
5   Эта программа печатает в стандартный вывод
6   "Hello world"
7   */
8
9   int main()
10  {
11      cout << "Hello world!"; // prints "Hello world"
12      return (0);
13  }
```

Тема: Стандартный поток вывода. Переменные и типы данных.

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).

Тема: Стандартный поток вывода. Переменные и типы данных.

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

int myVariable = 10;

Язык программирования C++ чувствителен к регистру, так что myVariable и myvariable это два разных идентификатора (имени).

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       cout << "Output:" << endl;
7       int myVariable = 10;
8       cout << myVariable;
9       return (0);
10  }
```

```
Output:
10
```

Тема: Стандартный поток вывода. Переменные и типы данных.

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.

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       cout << "Ouput:" << endl;
7       int a = 30;
8       int b = 12;
9       int sum = a + b;
10
11      cout << sum;
12
13      return (0);
14  }
```

```
Ouput:
42
```

Тема: Стандартный поток вывода. Переменные и типы данных.

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;
```

Тема: Стандартный поток вывода. Переменные и типы данных.

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;
```

Тема: Стандартный поток вывода. Переменные и типы данных.

Получение введенной информации

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

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int a;
7       cout << "Please enter a number" << endl;
8       cin >> a;
9
10      cout << "There is " << a << " in your variable" << endl;
11
12      return (0);
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.

Тема: Стандартный поток вывода. Переменные и типы данных.

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

```
Please enter a number
123
There is 123 in your variable
```

Получение введенной информации

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

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int a, b;
7       cout << "Enter a number" << endl;
8       cin >> a;
9       cout << "Enter another number" << endl;
10      cin >> b;
11
12      cout << a << " " << b;
13
14      return (0);
15  }
```

```
Enter a number
1
Enter another number
2
1 2
```

Тема: Стандартный поток вывода. Переменные и типы данных.

Получение введенной информации

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

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int a, b;
7       int sum;
8       cout << "Enter a number" << endl;
9       cin >> a;
10      cout << "Enter another number" << endl;
11      cin >> b;
12      sum = a + b;
13      cout << "Sum is: " << sum << endl;
14      return (0);
15  }
```

```
Enter a number
10
Enter another number
100
Sum is: 110
```

Тема: Стандартный поток вывода. Переменные и типы данных.

Арифметические операторы
С++ поддерживает следующие арифметические операторы.

Оператор	Символ	Вид
Сложение	+	x + y
Вычитание	-	x - y
Умножение	*	x * y
Деление	/	x / y
Деление по модулю	%	x % y

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int a = 10;
7       int b = 3;
8       cout << "Output:" << endl;
9       cout << "a + b == " << a + b << endl;
10      cout << "a - b == " << a - b << endl;
11      cout << "a * b == " << a * b << endl;
12      cout << "a / b == " << a / b << endl;
13      cout << "a % b == " << a % b << endl;
14      return (0);
15  }
```

```
Output:
a + b == 13
a - b == 7
a * b == 30
a / b == 3
a % b == 1
```

Тема: Стандартный поток вывода. Переменные и типы данных.

Деление по модулю

Оператор деления по модулю (%) неофициально известен как оператор остатка, потому что он возвращает остаток после деления целочисленных переменных.

Приоритет операторов

Приоритет операторов определяет порядок вычисления, который влияет на то, как выражения будут вычислены. Определенные операторы имеют приоритет выше других; например, оператор умножения имеет приоритет выше, чем у оператора сложения.

Как и в математике, использование скобок изменяет приоритет операторов.

Тема: Стандартный поток вывода. Переменные и типы данных.

Приоритет операторов

Скобки присваивают операциям высокий приоритет. Если выражение в скобках находится в другом выражении, также закрытом скобками, то сперва вычисляется выражение, лежащее внутри.

Если никакие выражения не заключены в скобки, то мультипликативные (умножение, деление, деление по модулю) операторы будут вычислены до аддитивных (сложение, вычитание) операторов.

Операторы присваивания

Простой оператор присваивания (=) присваивает правую часть выражения к левой части.

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

Тема: Стандартный поток вывода. Переменные и типы данных.

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
```

Тема: Стандартный поток вывода. Переменные и типы данных.

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;
```

Программирование
на C++

Практическая часть

Исполняемые файлы.
Стандартный поток вывода.

1 занятие

инжиниум®
МГТУ им. Н.Э. Баумана

2020

Тема: Стандартный поток вывода. Переменные и типы данных.

Задание 1

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

Тема: Стандартный поток вывода. Переменные и типы данных.

Задание 2

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.