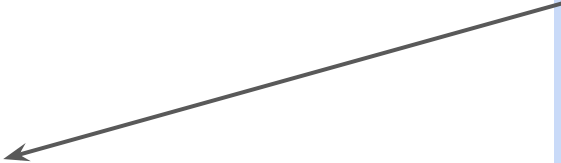


Junior C++ developer

Лекция 1

- Hello/world
- Основные типы данных
- Переменные
- Операторы
- Ввод и вывод

Отсюда начинается
выполнение
программы.
Функция “main”.



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


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

Параметры, которые передаются в функцию.

void - означает, что в функцию не передаются никакие параметры


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

Возвращаемое значение.
В данном случае значение **0** возвращается операционной системе после выполнения функции ***main***.



Тип возвращаемого значения.

int - целочисленный.



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

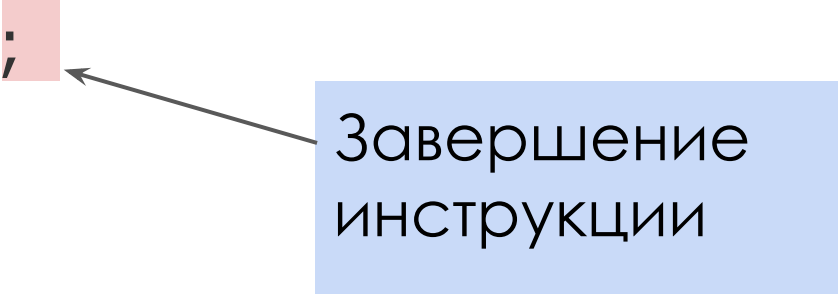
Тип возвращаемого
значения

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

Возвращаемое значение
ДОЛЖНО СООТВЕТСТВОВАТЬ
типу возвращаемого
значения.

Возвращаемое
значение

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



Завершение
инструкции

The diagram consists of a light blue rectangular box containing the Russian text 'Завершение инструкции' (End of instruction). A black arrow originates from the left side of this box and points towards the semicolon at the end of the 'return 0;' statement in the code block above.

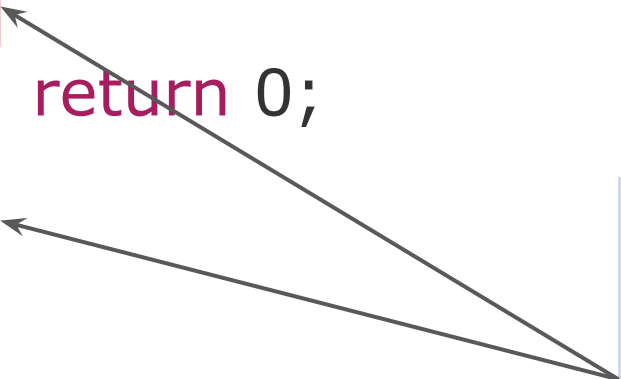
```
int main(void)
```

```
{
```

```
    return 0;
```

```
}
```

Начало и конец блока с кодом. В данном случае скобки определяют начало и конец тела функции.




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

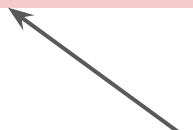
```
int main()
```

```
{
```

```
    cout<<"Hello World" << endl;
```

```
    return 0;
```

```
}
```



Инструкция. Вывести "Hello world" в стандартный поток вывода

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

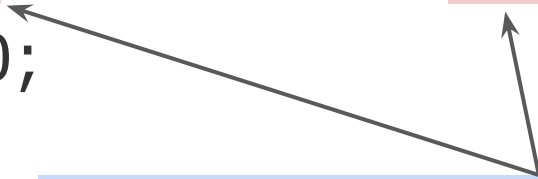
```
int main()  
{  
    cout << "Hello World" << endl;  
    return 0;  
}
```



Стандартный поток вывода

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

```
int main()
{
    cout << "Hello World" << endl;
    return 0;
}
```



Операторы << означают
“поместить данные в
ВЫХОДНОЙ ПОТОК”

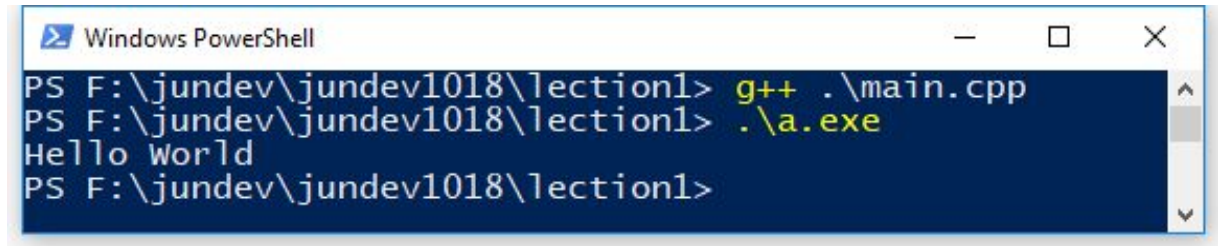
```
#include <iostream>  
using namespace std;
```

Обозначает перенос
на следующую строку

```
int main()  
{  
    cout<<"Hello World" << endl;  
    return 0;  
}
```

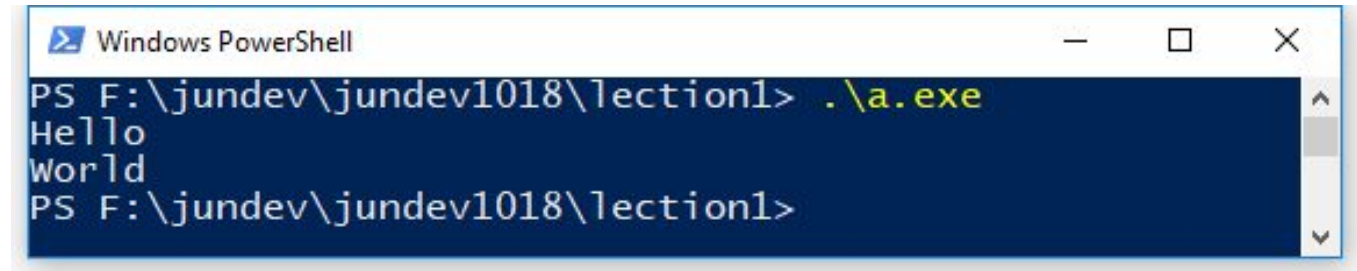
```
#include <iostream>
using namespace std;
```

```
int main()
{
    cout<<"Hello World" << endl;
    return 0;
}
```

A screenshot of a Windows PowerShell terminal window. The title bar reads "Windows PowerShell". The terminal has a dark blue background with white text. It shows the following commands and output:
1. Command: `g++ .\main.cpp`
2. Command: `.\a.exe`
3. Output: `Hello World`
4. Command: (blank line)
The window includes standard Windows window controls (minimize, maximize, close) in the top right corner and a vertical scrollbar on the right side.

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

```
int main()
{
    cout<<"Hello" << endl << "World" << endl;
    return 0;
}
```

A screenshot of a Windows PowerShell terminal window. The title bar reads "Windows PowerShell". The command prompt shows the path "F:\jundev\jundev1018\1ection1" and the command ".\a.exe" being executed. The output of the program is displayed as two lines: "Hello" followed by a newline, and "World" followed by a newline. The prompt then returns to "PS F:\jundev\jundev1018\1ection1>".

```
Windows PowerShell
PS F:\jundev\jundev1018\1ection1> .\a.exe
Hello
World
PS F:\jundev\jundev1018\1ection1>
```

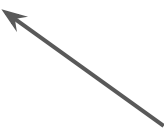
```
#include <iostream>
using namespace std;
```

```
int main()
{
```

```
    cout<<"Hello World" << endl;
    return 0;
```

```
}
```

Подключение библиотеки для работы с потоками ввода/вывода. В данном случае необходима для работы с **cout**



```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```


```
{
```

```
    cout<<"Hello World" << endl;
```

```
    return 0;
```

```
}
```

Все файлы стандартной библиотеки C++ определены в пространстве имен ***std***.




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

```
int main(void) {
```


```
    int numZombies = 20;
```

```
    cout << "Number of Zombies you want to kill? " << endl;
```

```
    int numKilled;
```

```
    cin >> numKilled;
```

Стандартный поток ввода
данных.



```
    cout << "Number of zombies left: " << endl;
```

```
    cout << numZombies - numKilled << endl;
```

```
    return 0;
```

```
}
```

```
int value;
cin >> value;
```

Из входного потока читается последовательность символов до пробела (табуляции, переноса на следующую строку), затем эта последовательность преобразуется к типу переменной **value**, и получаемое значение помещается в **value**

С++ сильно-типизированный язык (strong typing).

Языки с сильной типизацией:

- не позволяют смешивать сущности разных типов в выражениях
- не выполняют никаких автоматических преобразований



ОСНОВНЫЕ ТИПЫ ДАННЫХ

Целые (integers)

short, long, int

Действительные (real)

float, double

Символы (character)

char, unsigned char

Булевы (bool)

bool

Инициализация переменных

```
int i1 = 1;  
cout << "i1 = " << i1 << endl;  
int i2;  
i2 = 2;  
cout << "i2 = " << i2 << endl;  
int i3(3);  
cout << "i3 = " << i3 << endl;  
int i4{4};  
cout << "i4 = " << i4 << endl;
```

Auto

- говорит компилятору чтобы он сам определил тип переменной
- переменные все равно строго типизированы

```
auto a1 = 1;  
auto a2 = 2.2;  
auto a3 = 'c';  
auto a4 = "s"; //c-style string  
auto a5 = true;  
auto a6 = 3L;  
auto a7 = 1'000'000'000'000; //c++14  
auto a8 = 0xFF; //255  
auto a9 = 0b111; //7
```

Приведение типов

Компилятор
автоматически приводит,
если возможно

Типы можно приводить
самостоятельно

Никогда! не используйте
приведение типов в стиле
Си. Это не безопасно

Как делать нельзя

```
double x = 3.7;  
int i = (int) (x);  
i = (int) (x + 0.5);  
cout << i << endl;
```

```
double d1 = 2.2;  
// int i5 = d1;  
int i5 = static_cast<int>(d1);
```

`static_cast` - приводит типы во время компиляции, в случае ошибки приложение не скомпилируется.

В угловых скобках `<>` указывается тип к которому будет приведено выражение в круглых скобках `()`

`(d1)` типа `double` будет приведено к типу `<int>`



Дополнительный код

Десятичное представление	Двоичное представление в доп. коде
127	0111 1111
1	0000 0001
0	0000 0000
-1	1111 1111
-127	1000 0001

```
#include <iostream>
using namespace std;
int main()
{
    int salary = 20;
    cout << "Salary is ";
    cout << salary * 40 << endl;
    return 0;
}
```

Что такое salary?

- a) переменная
- b) ячейка в памяти
- c) число 20
- d) все вместе

Переменная

Ячейка в оперативной памяти, в которой хранится определенная информация.

Переменная:

- имеет название
- имеет определенный тип данных
- хранит определенное значение

Правила именования переменных

1. Имя должно начинаться с буквы или символа “_”
2. После первого символа могут использоваться любые цифры, буквы, символ “_”
3. Имя не должно совпадать с ключевым словом
4. Имена переменных чувствительны к регистру
5. Системные переменные обычно начинаются с символа “_”

/ - Оператор деления

- Результат деления зависит от типа данных
- При делении двух переменных типа `int` остаток откидывается, то есть происходит округление в меньшую сторону

$$5 / 2 = ?$$

$$5.0 / 2.0 = ?$$

$$5.0 / 2 = ?$$

% - оператор деления по модулю

Оператор вычисляет остаток от деления двух чисел

$$5 \% 2 = ?$$

$$2 \% 5 = ?$$

Операторы

Сколько часов и минут в 142 минутах?

// в часе 60 минут

```
int hours = 142 / 60;
```

// сколько минут в остатке

```
int minutes = 142 % 60;
```

' / ' - оператор деления

' % ' - оператор остаток от деления

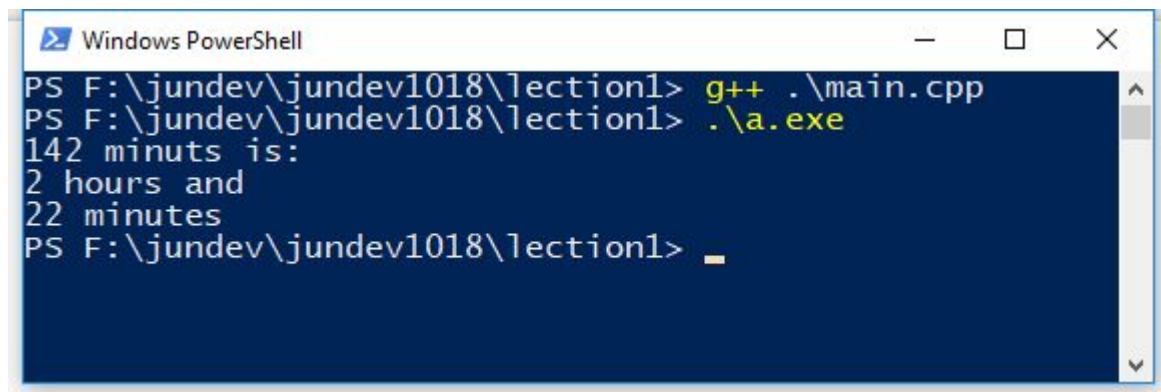

```
#include <iostream>
using namespace std;
```

```
int main(void) {
```

```
    cout << "142 minuts is:" << endl
         << (142 / 60) << " hours"
         << " and" << endl
         << (142 % 60) << " minutes"
         << endl;
```

```
    return 0;
```

```
}
```



```
Windows PowerShell
PS F:\jundev\jundev1018\lection1> g++ .\main.cpp
PS F:\jundev\jundev1018\lection1> .\a.exe
142 minuts is:
2 hours and
22 minutes
PS F:\jundev\jundev1018\lection1> _
```

Целочисленное переполнение

Ситуация в компьютерной арифметике, при которой вычисленное в результате операции значение не может быть помещено в n-битный целочисленный тип данных.

```
unsigned char a = 100;
```

```
unsigned char b = 158;
```

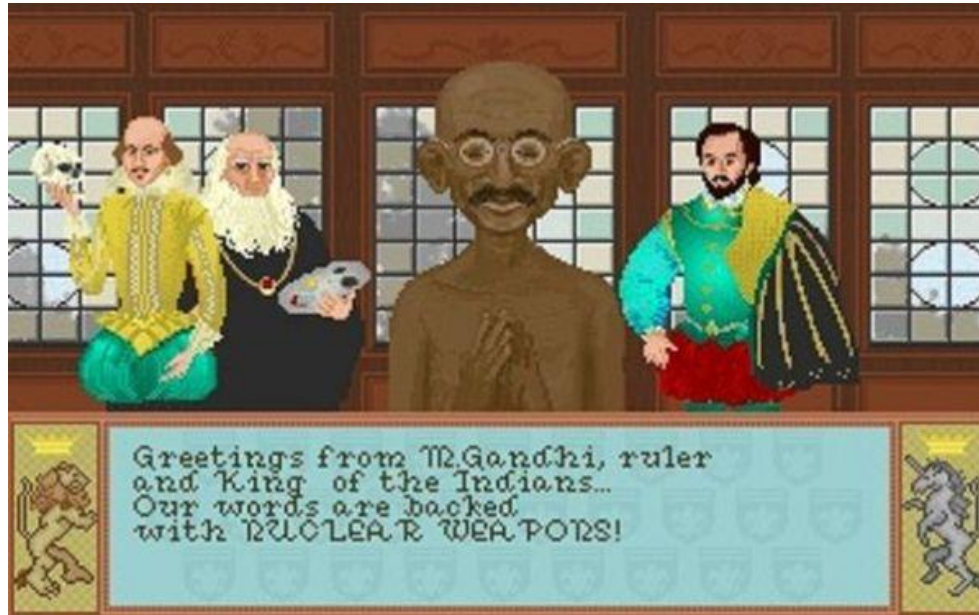
```
unsigned int z = a + b;
```

z = ?

$$100 + 158 = 258$$

$$\begin{array}{r} 01100100 \\ + \\ 10011110 \\ \hline 100000010 \end{array}$$

Почему Ганди развязал ядерную войну?



Целочисленное переполнение

В первой части игры Civilization у Ганди по умолчанию уровень агрессивности был равен 1, а диапазон допустимых значений был от 0 до 10.

После принятия Демократии уровень агрессивности уменьшался на 2 единицы.

`unsigned char a = 1;`

`a - 2 = ?`

$$1 - 2 = -1$$

-1 в дополнительном коде
это 1111 1111.

А так как тип данных
`unsigned char`, то получаем
255