# Занятие №1. **Основы C++**

```
1: //Тривиальная программа C++, которая выводит строку приветствия
2:
3: #include <iostream>
4: using namespace std;
5: int main ()
6: {
7:    cout << "Hello Programmer!";
8:    return 0;
9:}
```

C++ использует символы // для комментария, который продолжается до конца строки. C++ поддерживает комментарии языка C, которые начинаются с символов /* и заканчиваются символами */. Строка 1 содержит комментарий, который кратко описывает программу.

*Комментариями* называются пояснения, помещаемые в тексте программы для того, чтобы объяснить или описать некоторые ее части.

Транслятор игнорирует комментарии, но программист использует их, чтобы знать, что делает программа, особенно если она не использовалась длительное время, и о ее особенностях забыли.

Программа C++ не имеет никаких зарезервированных ключевых слов, которые обозначают ее конец. C++ использует довольно простую схему организации программы. Эта схема поддерживает два уровня кода: *глобальный* и *уровень функций*.

Кроме того, функция **main**, определяемая со строки 5, играет очень специфическую роль, потому что выполнение программы C++ всегда начинается с этой функции.

Следовательно, в программе **может быть только одна** функция **main**. Вы можете располагать функцию main в любом месте программы.

Строки и символы C++ заключаются соответственно в двойные и одиночные кавычки. Таким образом, 'A' является одиночным символом, в то время как "A" — строка, состоящая всего из одного символа.
Смешивание в C++ односимвольных строк и символов запрещено.

Строки могут содержать любое число символов, в том числе ни одного. Строка, не имеющая символов, называется **пустой строкой**.

C++ определяет операторные блоки, ограниченные символами { и }. См. строки с 6-ой по 9-ю соответственно.

Каждый оператор в программе C++ должен заканчиваться точкой с запятой (;).

Программы на C++ содержат директиву препроцессора `#include`.
Пример этому можно найти в строке 3, в которой компилятору C++ дается указание включить файл заголовка IOSTREAM в текст программы.

IOSTREAM обеспечивает операции, которые поддерживают базовый потоковый ввод и вывод.

C++ не имеет встроенных операций ввода/вывода.
Вместо этого язык полагается на библиотеки, специализирующиеся в различных типах ввода/вывода.

Программы на C++ содержат директиву препроцессора `#include`.
Пример этому можно найти в строке 3, в которой компилятору C++ дается указание включить файл заголовка IOSTREAM в текст программы.

IOSTREAM обеспечивает операции, которые поддерживают базовый потоковый ввод и вывод.

C++ не имеет встроенных операций ввода/вывода.
Вместо этого язык полагается на библиотеки, специализирующиеся в различных типах ввода/вывода.

Программа выводит строку "Hello Programmer!" в стандартный поток вывода `cout`, который является окном MSDOS. При этом программа использует операцию вывода <<, направляющую выводимую строку в выходной поток.

Функция `main` должна возвращать значение, которое отражает состояние программы C++. Возвращаемое значения *0* сообщает операционной системе о том, что программа завершилась без ошибок.

## Предопределенные типы данных в C++

Для представления *логических значений*, *целых чисел*, *символов*, *чисел с плавающей точкой обычной точности*, *чисел с плавающей точкой двойной точности* и *незначимых данных* C++ предлагает соответственно типы данных **bool**, **int**, **char**, **float**, **double** и **void**.

В языке C++ тип **void** для возвращаемого функцией значения используется для указания на то, что функция не вырабатывает значимого результата, то есть функция действует как процедура.

В языке C++ гибкость в отношении типов данных увеличивается благодаря возможности применения модификаторов типов данных. Модификаторами типа являются: **signed**, **unsigned**, **short** и **long**.

## Правила именования идентификаторов:

- Первый символ должен быть буквой или подчеркиванием ( _ ).
- Последующие символы могут быть *буквами*, *цифрами* или *подчеркиваниями*.
- *Максимальная длина* идентификатора составляет по умолчанию *32 символа* (это может быть изменено в опциях компилятора).
- В идентификаторах C++ **имеет значение регистр букв**. Таким образом, имена *rate*, *RATE* и *Rate* относятся к трем различным идентификаторам.
- Идентификаторами *не могут* быть зарезервированные слова, например, **int**, **double** или **static**.

примеры **допустимых** идентификаторов:

```
X
x
aString
DAYS_IN_WEEK
BinNumber0
bin_number_0
bin0Number2
_length
```

некоторые из **недопустимых**:

```
123aNumber
const
NoSpaces Allowed
NorAre*Most+Symbols
```

# Директива #include

Программа C++ содержит директиву #include. Эта директива предписывает компилятору включить в программу текст указанного файла, так, как если бы вы сами набрали этот текст.
Таким образом, директива #include является лучшей альтернативой, чем вырезка текста из одного файла и вставка его в другой файл. Можно создать заголовочный файл, включающий в себя общий код, и затем просто включать его во все программы, где это требуется.

# Директива #include

Общий синтаксис для директивы #include

```
#include <имя_файла>
#include "имя_файла«
```

Формы директивы #include различаются способом поиска указанного файла.
*Первая* форма ищет файл в специальном каталоге для включаемых файлов.
*Вторая* форма расширяет диапазон поиска, проводя поиск в текущем каталоге перед поиском в каталоге включаемых файлов.

## Директива #define

С помощью директивы #define создаются макроопределения (макросы). Наиболее распространенным из них является простой макрос подстановки: вы предписываете препроцессору заменять каждое вхождение определенного текстового шаблона на другой текстовый шаблон.

Общий синтаксис для директивы # define

```
#define имя_константы значение_константы
```

**Примеры:**

#define ASCII_A 65

#define DAYS_IN_WEEK 7

# Объявление переменных

Общий синтаксис для объявления переменных

```
тип имяПеременной;
тип имяПеременной = начальноеЗначение;
тип перем1 [= нач_знач1], перем2 [= нач_знач2];
```

**Примеры**

```
int j ;
double z = 32.314;
long fileSize, diskSize, totalFileSize = 0;
```

# Арифметические операции языка C++

| Оператор C++ | Назначение | Тип данных | Пример |
|---|---|---|---|
| + | Унарный плюс | Числа | x =+ y + 3; |
| - | Унарный минус | Числа | x =- y; |
| + | Сложение | Числа | z = y + x; |
| - | Вычитание | Числа | z = y - x; |
| * | Умножение | Числа | z = y * x; |
| / | Деление | Числа | z = y / x; |
| % | Делениепо модулю | Целые числа | z = y % x; |

## Операции инкремента и декремента (изменения на 1)

Язык C++ поддерживает специальные операции инкремента (увеличения на 1) и декремента (уменьшения на 1).

Операции инкремента (++) и декремента (--) дают вам возможность соответственно увеличивать или уменьшать на 1 хранимое в переменной значение.

**Примеры**

```
lineNumver++;
++index;
```

# Арифметические операции присваивания

| Операция присваивания | Длинная форма | Пример |
|---|---|---|
| x+=y | x = x + y | x += 12; |
| x -= y | x = x - y | x -= 34 + y; |
| x *= y | x = x * y | scale *= 10; |
| x /= y | x = x / y | z /= 34 * y; |
| x %= y | x = x % y | z %= 2; |

## Приведение типа

Одной из обязанностей компилятора является автоматическое преобразование значения из одного типа данных в другой, совместимый с ним.

Приведение типа является свойством языка, которое дает вам возможность явно определять, каким образом некоторое значение будет преобразовано из первоначального типа данных в совместимый с ним тип.

Таким образом, приведение типа дает компилятору указание, чтобы он выполнял именно то преобразование, которое желаете вы, а не то, которое он считает необходимым.

# Приведение типа

Язык C++ поддерживает следующие формы приведения типа:

```
приведение_к_типу(выражение)
```

и

```
(приведение_к_типу) выражение
```

**Примеры**

```
int i = 2; float a, b;
a = float(i);
b = (float)i;
```

Операции *отношений* (меньше чем, больше чем и равно) и *логические операции* (И, ИЛИ и НЕ) являются базовыми строительными блоками в конструкциях принятия решений в любом языке программирования.

| Операция C++ | Значение | Пример |
|---|---|---|
| && | Логическое AND | if (i > 1 && i < 10) |
| \|\| | Логическое OR | if (c==0 \|\| c==9) |
| ! | Логическое NOT | if (!(c>1 && c<9)) |
| < | Меньше чем | if (i < 0) |
| <= | Меньше или равно | if (i <= 0) |
| > | Больше чем | if (j > 10) |
| >= | Больше или равно | if (x >= 8.2) |
| == | Равно | if (c == '\0') |
| != | Не равно | if (c != '\n') |
| ?: | Условное присваивание | k = (i<1) ? 1 : i; |