1.5 C++ VARIABLES & CONSTANTS

Any program processes data. They are located in the computer's RAM, and can be **variable** (the value of the date can change) or **constant** (the value cannot change).

Variables

A **variable** represents a memory location where a value of a particular **type** resides. Any variable is characterized by:

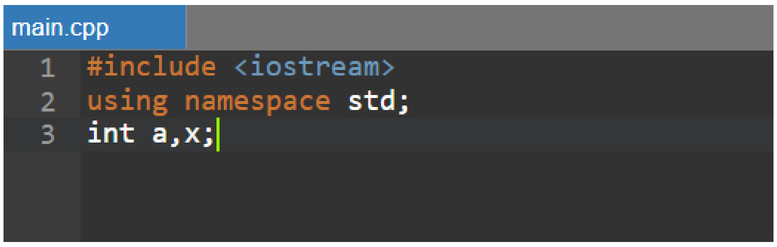
- the address of the variable. Computer RAM is addressable – each byte in memory has a sequence number associated with it, starting at 0. This number represents the address of that byte and is displayed in base 16 by default.
- variable identifier - represents a name for the variable - the connection between the variable and its address. The identifier complies with the following rules:
    - contains uppercase, lowercase letters of the English alphabet, numbers and the underline character '_'. Uppercase letters are considered different from lowercase letters, so Answer, Answer, and ANSWER are different identifiers.
    - the first character cannot be a number. Although it is possible for an identifier to start with '_', it is not recommended to avoid certain conflicts with system identifiers.
    - identifier cannot be part of the list of reserved words - available at the end of this article.
    - there are no limits on the length of an identifier, but only the first 31 characters are significant.
- the type of the variable – determines what kind of values the variable can take, between which limits they are, as well as what operations can be performed with the variable.
- visibility area – represents the area in the program where the variable exists and can be used. Variables can be global or local.
- local variables are declared within a block (in curly braces {...}) and are only visible within that block. They have random initial values.
- global variables are declared outside any block and are visible in all blocks following the declaration. They are initialized to 0.

In C/C++, variables must be declared, specifying the type and identifier. The syntax is:

Data_type List_identifiers;

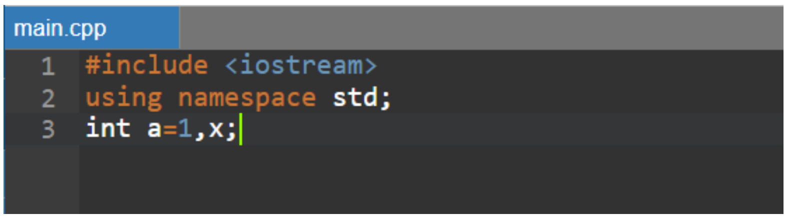
where *Data_type* can be any valid C++ type (read here about data types), and List*_identifier* consists of at least one identifier. If there are more, they will be separated by the comma character (,).

e.g.:


main.cpp

```cpp
#include <iostream>
using namespace std;
int a,x;
```

Two variables have been declared, named a and x, which will be able to store integer values from a range that we will study later. When declaring, variables can be initialized with a value corresponding to the data type used:


main.cpp

```cpp
#include <iostream>
using namespace std;
int a=1,x;
```

The following C++ identifiers are valid: *a, number, Number, other_number, a2b, _amount* - not recommended*, a very_long_variable_name*.

The following C++ identifiers are incorrect:

- *2a* – starts with number. Identifiers can start with letters or '_'
- *other number* – contains prohibited character: space
- *one-number* - contains forbidden character: minus
- *number* – contains the letter ă. Identifiers can only contain ASCII letters – from the English alphabet.

Constants

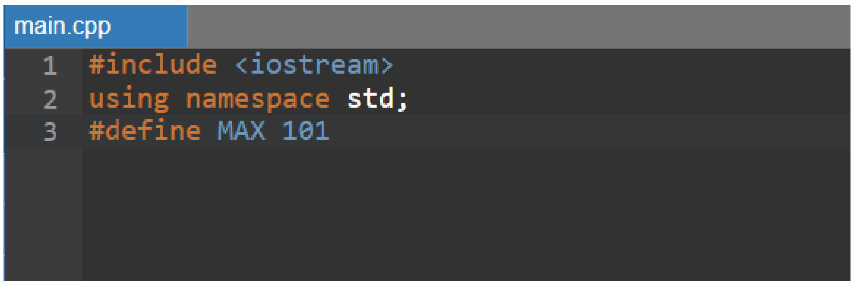
Constants are data that **do not change** their value during program execution. They can be **constants with name**, or **literal constants**, given directly by their value.

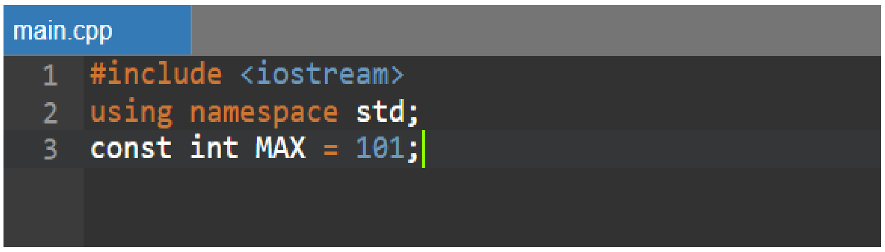
Symbolic constants

Symbolic (named) constants can be specified in two ways:

- by the *define* directive
- variables can be declared with the *const* modifier; they become read-only, and their value **can no longer be modified**

e.g.:


main.cpp
```cpp
#include <iostream>
using namespace std;
#define MAX 101
```


main.cpp
```cpp
#include <iostream>
using namespace std;
const int MAX = 101;
```

Literals

Constant values can appear in a program, whether they are numbers, characters, strings, or otherwise. These are also called **literal constants** or **literals**.

Integer constants

Represent whole numbers – with no fractional part. May be:

- Decimal constants – in base 10:
  - examples: 176, -54, 0;
  - can contain the digits: 0 1 2 3 4 5 6 7 8 9;
- Octal constants – in base 8
  - always start with 0;
  - examples: 015, 062;
  - can contain the digits: 0 1 2 3 4 5 6 7;
- Hexadecimal constants – in base 16:
  - always start with 0x;
  - examples: 0x15, 0x6f, 0xff;
  - may contain the digits: 0 1 2 3 4 5 6 7 8 9 A B C D E F.

Real constant

They represent **real numbers** and are also called **floating point**. The decimal separator is the period character (.) and can appear in two forms:

1. **standard writing** (fixed): -1.5 14,974

2. **scientific notation**, with mantissa and exponent. The number -0.567E+2 means -0.567*10+2, i.e. -56.7:
   - -0.567 represents the mantissa;
   - +2 represents the exponent.

Character constant – char

They are made up of a single character, delimited by apostrophes: (').

e.g.: *'a', 'B', '~', '?'*

A separate category of characters consists of **ESCAPE sequences**. An escape sequence consists of **two characters**, the first of which is the backslash: \. Represents characters that cannot be written as such or have a special meaning. From a syntactic point of view, they are characters, being delimited by an apostrophe. Probably the most well-known escape sequence is **'\n'** – new line (enter).

Among the escape sequences we mention:

- '\n' – Newline
- '\b' – Backspace
- '\f' – Form feed
- '\r' – Return
- '\t' – horizontal TAB
- '\\' – Backslash
- ''' – Apostrophe
- '\"' – Quotation marks
- '\?' – Question mark
- '\0' - Null character

String constants

They are delimited by quotation marks ("). May contain **escape sequences**.

e.g.: *"number", "n = ", "I'm done.\n"*

Pay attention to!

- Initialization is mandatory when declaring *read-only* variables!
- An octal constant cannot contain the digit 9. The value 0295 is not correct and will produce a compile error!
- A char constant contains exactly one character. We can't have multiple characters between apostrophes, except for escape sequences, but an escape sequence is only one character!
- 'A' and "A" are not the same thing: 'A' is a character, and "A" is a string, consisting of a single character!

Reserved words

Not every word can be used as an identifier. There is **a list of words** in C++ that have a well-defined meaning and cannot be used for any other purpose. They are called reserved words (keywords) and are as follows:

alignas	**else**	**requires**
alignof	**enum**	**return**
and	**explicit**	**short**
and_eq	**export**	**signed**
asm	**extern**	**sizeof**
auto	**false**	**static**
bitand	**float**	**static_assert**
bitor	**for**	**static_cast**
bool	**friend**	**struct**
break	**goto**	**switch**
case	**if**	**template**
catch	**inline**	**this**
char	**int**	**thread_local**
char16_t	**long**	**throw**
char32_t	**mutable**	**true**
class	**namespace**	**try**
compl	**new**	**typedef**
concept	**noexcept**	**typeid**
const	**not**	**typename**
constexpr	**not_eq**	**union**
const_cast	**nullptr**	**unsigned**
continue	**operator**	**using**
decltype	**or**	**virtual**
default	**or_eq**	**void**
delete	**private**	**volatile**
do	**protected**	**wchar_t**
double	**public**	**while**
dynamic_cast	**register**	**xor**
	reinterpret_cast	**xor_eq**