Basic element and tools of C++ language • Language symbols • Definitions name • keywords • Constant represent • Variables represent Data types in C++, and the represent methods in memory • char type • integer type • real type • Boolean (logical) type	الثاني

When we consider a C++ program, it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what a class, object, methods, and instant variables mean.

- **Object** − Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors - wagging, barking, eating. An object is an instance of a class.
- **Class** − A class can be defined as a template/blueprint that describes the behaviors/states that object of its type support.
- **Methods** − A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
- **Instance Variables** − Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

C++ Program Structure

Let us look at a simple code that would print the words *Hello World*.

Live Demo

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

// main() is where program execution begins.
int main() {
   cout << "Hello World"; // prints Hello World
   return 0;
}
```

Let us look at the various parts of the above program −

- The C++ language defines several headers, which contain information that is either necessary or useful to your program. For this program, the header **<iostream>** is needed.
- The line **using namespace std;** tells the compiler to use the std namespace. Namespaces are a relatively recent addition to C++.
- The next line '**// main() is where program execution begins.**' is a single-line comment available in C++. Single-line comments begin with // and stop at the end of the line.
- The line **int main()** is the main function where program execution begins.
- The next line **cout << "Hello World";** causes the message "Hello World" to be displayed on the screen.
- The next line **return 0;** terminates main( )function and causes it to return the value 0 to the calling process.

Compile and Execute C++ Program

Let's look at how to save the file, compile and run the program. Please follow the steps given below −

- Open a text editor and add the code as above.
- Save the file as: hello.cpp
- Open a command prompt and go to the directory where you saved the file.
- Type 'g++ hello.cpp' and press enter to compile your code. If there are no errors in your code the command prompt will take you to the next line and would generate a.out executable file.
- Now, type 'a.out' to run your program.
- You will be able to see ' Hello World ' printed on the window.

```
$ g++ hello.cpp
$ ./a.out
Hello World
```

Make sure that g++ is in your path and that you are running it in the directory containing file hello.cpp.

You can compile C/C++ programs using makefile. For more details, you can check our 'Makefile Tutorial'.

Semicolons and Blocks in C++

In C++, the semicolon is a statement terminator. That is, each individual statement must be ended with a semicolon. It indicates the end of one logical entity.

For example, following are three different statements −

```
x = y;
y = y + 1;
add(x, y);
```

A block is a set of logically connected statements that are surrounded by opening and closing braces. For example −

```
{
   cout << "Hello World"; // prints Hello World
   return 0;
}
```

C++ does not recognize the end of the line as a terminator. For this reason, it does not matter where you put a statement in a line. For example −

```
x = y;
y = y + 1;
add(x, y);
```

is the same as

```
x = y; y = y + 1; add(x, y);
```

C++ Identifiers

A C++ identifier is a name used to identify a variable, function, class, module, or any other user-defined item. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores, and digits (0 to 9).

C++ does not allow punctuation characters such as @, $, and % within identifiers. C++ is a case-sensitive programming language. Thus, **Manpower** and **manpower** are two different identifiers in C++.

Here are some examples of acceptable identifiers −

```
mohd       zara    abc   move_name  a_123
myname50   _temp   j     a23b9      retVal
```

C++ Keywords

The following list shows the reserved words in C++. These reserved words may not be used as constant or variable or any other identifier names.

asm	else	new	this
auto	enum	operator	throw
bool	explicit	private	true
break	export	protected	try
case	extern	public	typedef
catch	false	register	typeid
char	float	reinterpret_cast	typename
class	for	return	union
const	friend	short	unsigned
const_cast	goto	signed	using
continue	if	sizeof	virtual
default	inline	static	void

Whitespace in C++

A line containing only whitespace, possibly with a comment, is known as a blank line, and C++ compiler totally ignores it.

Whitespace is the term used in C++ to describe blanks, tabs, newline characters and comments. Whitespace separates one part of a statement from another and enables the compiler to identify where one element in a statement, such as int, ends and the next element begins.

Statement 1

```
int age;
```

In the above statement there must be at least one whitespace character (usually a space) between int and age for the compiler to be able to distinguish them.

Statement 2

```
fruit = apples + oranges;   // Get the total fruit
```

In the above statement 2, no whitespace characters are necessary between fruit and =, or between = and apples, although you are free to include some if you wish for readability purpose.

C++ Data Types

As explained in the Variables chapter, a variable in C++ must be a specified data type:

Example

```
int myNum = 5;               // Integer (whole number)
float myFloatNum = 5.99;     // Floating point number
double myDoubleNum = 9.98;   // Floating point number
char myLetter = 'D';         // Character
bool myBoolean = true;       // Boolean
string myText = "Hello";     // String
```

Basic Data Types

The data type specifies the size and type of information the variable will store:

Data Type	Size	Description
int	4 bytes	Stores whole numbers, without decimals
float	4 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 7 decimal digits
double	8 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits
boolean	1 byte	Stores true or false values
char	1 byte	Stores a single character/letter/number, or ASCII values

Use `int` when you need to store a whole number without decimals, like 35 or 1000, and `float` or `double` when you need a floating point number (with decimals), like 9.99 or 3.14515.

int

```
int myNum = 1000;
cout << myNum;
```

Run example »

float

```
float myNum = 5.75;
cout << myNum;
```

Run example »

double

```
double myNum = 19.99;
cout << myNum;
```

Run example »

`float` vs. `double`

The **precision** of a floating point value indicates how many digits the value can have after the decimal point. The precision of `float` is only six or seven decimal digits, while `double` variables have a precision of about 15 digits. Therefore it is safer to use `double` for most calculations.

Scientific Numbers

A floating point number can also be a scientific number with an "e" to indicate the power of 10:

Example

```
float f1 = 35e3;
double d1 = 12E4;
cout << f1;
cout << d1;
```

Booleans

A boolean data type is declared with the `bool` keyword and can only take the values `true` or `false`. When the value is returned, `true` = `1` and `false` = `0`.

Example

```
bool isCodingFun = true;
bool isFishTasty = false;
```

```
  cout << isCodingFun;  // Outputs 1 (true)
cout << isFishTasty;  // Outputs 0 (false)
```

Boolean values are mostly used for conditional testing, which you will learn more about in a later chapter.

Characters

The `char` data type is used to store a **single** character. The character must be surrounded by single quotes, like 'A' or 'c':

Example

```
char myGrade = 'B';
cout << myGrade;
```

Alternatively, you can use ASCII values to display certain characters:

Example

```
char a = 65, b = 66, c = 67;
cout << a;
cout << b;
cout << c;
```

Tip: A list of all ASCII values can be found in our ASCII Table Reference.

Strings

The `string` type is used to store a sequence of characters (text). This is not a built-in type, but it behaves like one in its most basic usage. String values must be surrounded by double quotes:

Example

```
string greeting = "Hello";
cout << greeting;
```

To use strings, you must include an additional header file in the source code, the `<string>` library:

Example

```
// Include the string library
#include <string>

// Create a string variable
string greeting = "Hello";

// Output string value
cout << greeting;
```