Your first C++ program

```
#include <iostream>
using namespace std;
int main ()
{
    cout<<"Hello World!";
    return 0;
}
```

The program prints `Hello World!` in the output screen.

How the program works?

Now, let’s dissect the above code. The code is divided into six major parts:

#include <iostream> $\longrightarrow$ This statement includes the header file into the application so that you are able to use the operations included in them. iostream contain mechanisms to get the input from the user and print some information to a file or screen.

using namespace std; → It's a region where your code resides. It limits or expands the scope of your code to one or more files.

int main () → As the name suggests, it is the main function of the program. The code inside { } is called the body and is executed first when you run your C++ program.
{
cout << "Hello World!"; → This statement prints "Hello World!" onto the output screen. The cout is an object of standard output stream. What this means is, it outputs/prints the data after << , i.e. Hello World! into a stream (in this case, the output screen).
What is <<?

<< is the insertion operator used to write formatted data into the stream.

return 0; → This is called a return statement. It isn't mandatory to return anything from the main() function but is rather a convention. If not return, the compiler returns a status automatically.

}

Semicolon ";" The semicolon is a terminal. It terminates a statement. When missed or incorrectly used, it will cause a lot of issues.

C++ Variables

What are Variables

Variable are used in C++, where we need storage for any value, which will change in program. Variable can be declared in multiple ways each with different memory requirements and functioning. Variable is the name of memory location allocated by the compiler depending upon the datatype of the variable.

Basic types of Variables

Each variable while declaration must be given a datatype, on which the memory assigned to the variable depends. Following are the basic types of variables,

`bool`	For variable to store boolean values(True or False)
`char`	For variables to store character types.
`int`	for variables with integral values
`float` and `double` are also types for variables with large and floating point values	

So, how do we define, declare and use various types of variables?

Declaration and Initialization

Variables must be declared before they are used. Usually it is preferred to declare them at the starting of the program, but in C++ they can be declared in the middle of program too, but must be done before using them.

Example :

```
int i;      // declared but not initialised

char c;

int i, j, k;  // Multiple declaration
```

Initialization means assigning value to an already declared variable,

```
int i;   // declaration

i = 10;  // initialization
```

Initialization and declaration can be done in one single step also,

```
int i=10;         //initialization and declaration in same step

int i=10, j=11;
```

If a variable is declared and not initialized by default it will hold a garbage value. Also, if a variable was once declared and you try to declare it again, we will get a compile time error.

```
int i, j;

i=10;

j=20;

int j=i+j;  //compile time error, cannot redeclare a variable in same scope
```

Scope of Variables

All the variables have their area of functioning, and out of that boundary they don't hold their value, this boundary is called scope of the variable. For most of the cases its between the curly braces,in which variable is declared that a variable exists, not outside it. We will study the storage classes later, but as of now, we can broadly divide variables into two main types,

- Global Variables
- Local variables

Global variables

Global variables are those, which are once declared and can be used throughout the lifetime of the program by any class or any function. They must be declared outside the main() function. If only declared, they can be assigned different values at different time in program lifetime. But even if they are declared and initialized at the same time outside the main() function, then also they can be assigned any value at any point in the program.

Example: Only declared, not initialized

```
include <iostream>

using namespace std;

int x;                // Global variable declared

int main()

{

 x=10;                 // Initialized once

 cout <<"first value of x = "<< x;

 x=20;                 // Initialized again

 cout <<"Initialized again with value = "<< x;

}
```

Local Variables

Local variables are the variables which exist only between the curly braces, in which its declared. Outside that they are unavailable and leads to compile time error.

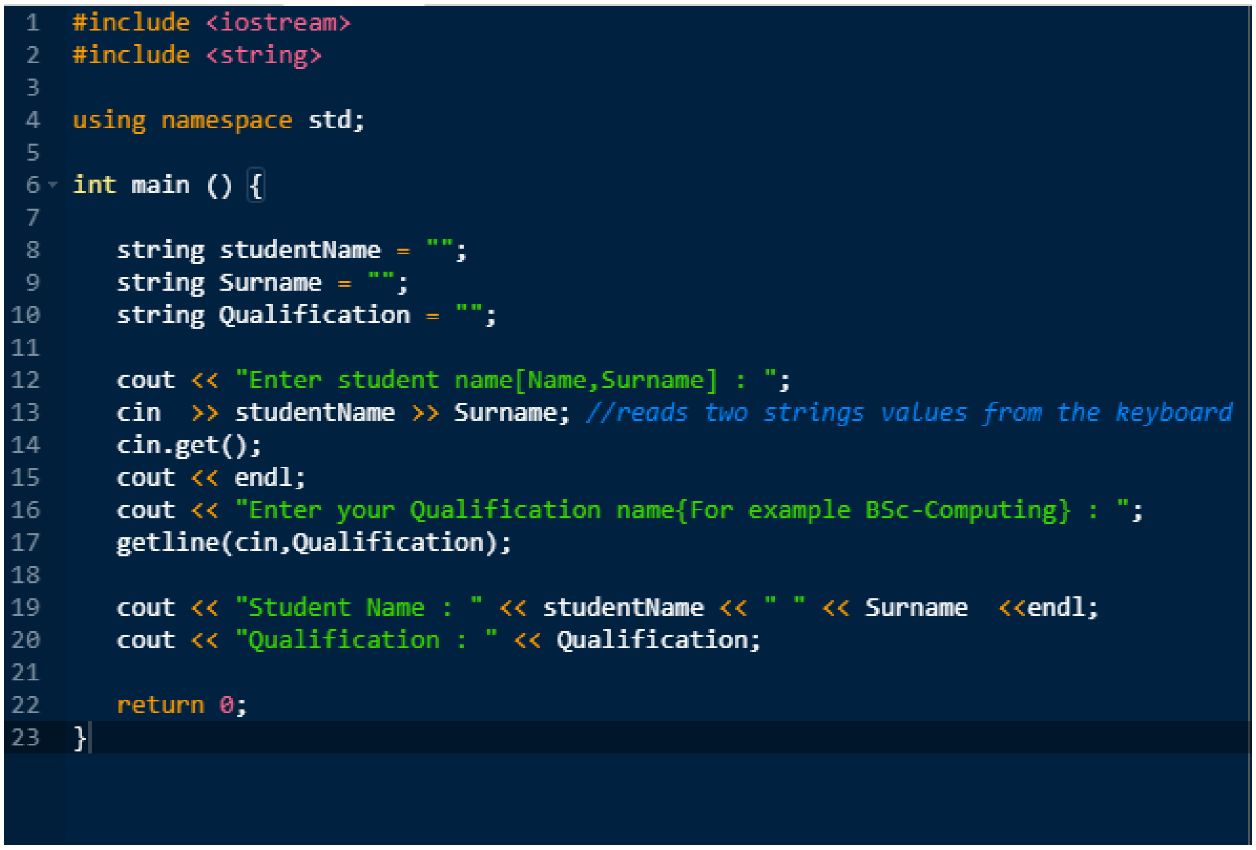
Example:

```
include <iostream>

using namespace std;
```

```
int main()
{
 int i=10;
 if(i<20)        // if condition scope starts
  {
    int n=100;   // Local variable declared and initialized
  }              // if condition scope ends
 cout << n;      // Compile time error, n not available here
}
```

C++ strings

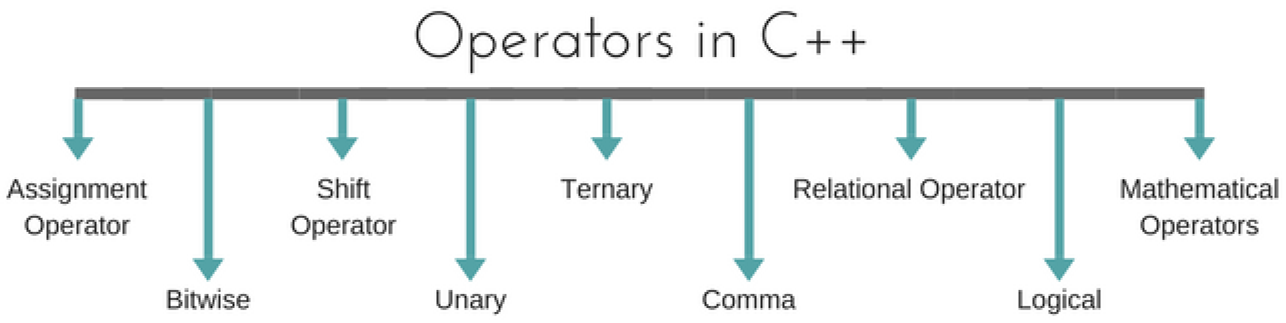
In computer **programming**, a **string** is traditionally a sequence of characters, either as a literal constant or as some kind of variable. In order to use the strings class in your program import the package #include <string>



```cpp
#include <iostream>
#include <string>

using namespace std;

int main () {

   string studentName = "";
   string Surname = "";
   string Qualification = "";

   cout << "Enter student name[Name,Surname] : ";
   cin  >> studentName >> Surname; //reads two strings values from the keyboard
   cin.get();
   cout << endl;
   cout << "Enter your Qualification name{For example BSc-Computing} : ";
   getline(cin,Qualification);

   cout << "Student Name : " << studentName << " " << Surname  <<endl;
   cout << "Qualification : " << Qualification;

   return 0;
}
```

Operators in C++

Operators are special type of functions, that takes one or more arguments and produces a new value. For example : addition (+), substraction (-), multiplication (*) etc, are all operators. Operators are used to perform various operations on variables and constants.



Compiled By: Kgosi S.E

084 461 6270

Types of operators

Assignment Operator (=)

Operates '=' is used for assignment, it takes the right-hand side (called rvalue) and copy it into the left-hand side (called lvalue). Assignment operator is the only operator which can be overloaded but cannot be inherited.

Mathematical Operators

There are operators used to perform basic mathematical operations. Addition (+) , subtraction (-) , diversion (/) multiplication (*) and modulus (%) are the basic mathematical operators. Modulus operator cannot be used with floating-point numbers.

C++ and C also use a shorthand notation to perform an operation and assignment at same type. *Example*,

```
int x=10;

x += 4 // will add 4 to 10, and hence assign 14 to X.

x -= 5 // will subtract 5 from 10 and assign 5 to x.
```

Relational Operators

These operators establish a relationship between operands. The relational operators are: less than (<), greater than (>), less than or equal to (<=), greater than equal to (>=), equivalent (==) and not equivalent (! =).

You must notice that assignment operator is (=) and there is a relational operator, for equivalent (==). These two are different from each other, the assignment operator assigns the value to any variable, whereas equivalent operator is used to compare values, like in if-else conditions,

Example

```
int x = 10;  //assignment operator

x=5;         // again assignment operator

if(x == 5)   // here we have used equivalent relational operator, for comparison

{

 cout <<"Successfully compared";}
```

Logical Operators

The logical operators are AND (&&) and OR (||). They are used to combine two different expressions together.

If two statements are connected using AND operator, the validity of both statements will be considered, but if they are connected using OR operator, then either one of them must be valid. These operators are mostly used in loops (especially `while` loop) and in Decision making.