TUTORIAL 1: BASIC OF C++

C++ is a **middle-level programming** language developed by Bjarne Stroustrup starting in 1979 at Bell Labs.

- **Requirements** : A text editor, like Notepad, to write C++ code
- A compiler, like GCC, to translate the C++ code into a language that the computer will understand.

CODE:

```
#include <iostream>

using namespace std;

int main() {

  cout << "Hello World!";

  return 0;

}
```

Line 1: #include <**iostream**> is a header file library that lets us work with input and output objects, such as **cout** (used in line 5). Header files add functionality to C++ programs.

Line 2: using **namespace std** means that we can use names for objects and variables from the standard library.

Omitting Namespace

You might see some C++ programs that runs without the standard namespace library. The using namespace std line can be omitted and replaced with the std keyword, followed by the **::** operator for some objects:

CODE:

```
#include <iostream>

int main() {

  std::cout << "Hello World!";

  return 0;

}
```

SCOPE RESOLUTION OPERATOR

:: here it is scope resolution operator.

USE:

1) To access a global variable when there is a local variable with same name:

```
#include<iostream>

using namespace std;

int x; // Global x

int main()
```

```
{
int x = 10; // Local x
cout << "Value of global x is " << ::x;
cout << "\nValue of local x is " << x;
return 0;
}
```

OUTPUT:

Value of global x is 0

Value of local x is 10

Some other use:

2) To define a function outside a class.
3) In case of multiple Inheritance

In C++;

The **cout** object, together with the << operator, is used to output values/print text.

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

NEW LINE:

For new line we can use **“\n” or endl. See below example**

cout << "Hello World!" << endl;

cout << "Hello World! \n\n";

COMMENTS:

For single line comment use : //

For Multiline comments use : /* and */

Some interesting code for expanding view of C++

```
#include<iostream>

using namespace std;


int main()

{

   int x,y,z;

   x=y=z=10;

   cout<<x+y+z;

   return 0;

}
```

Rules for Identifiers: Identity of variable (name of variable)

Names can contain letters, digits and underscores

Names must begin with a letter or an underscore (_)

Names are case sensitive (myVar and myvar are different variables)

Names cannot contain whitespaces or special characters like !, #, %, etc.

Reserved words (like C++ keywords, such as int) cannot be used as names

C++ Constants

When you do not want others (or yourself) to change existing variable values, use the const keyword (this will declare the variable as "constant", which means unchangeable and read-only)

```
const int myNum = 15;  // myNum will always be 15

myNum = 10;  // error: assignment of read-only variable 'myNum'
```

C++ User Input

You have already learned that **cout is used to output (print)** values. Now we will use **cin to get user input**.

cin is a predefined variable that reads data from the keyboard with the extraction operator (>>).

Example:

```
int x;

cout << "Type a number: "; // Type a number and press enter
```

```
cin >> x; // Get user input from the keyboard
cout << "Your number is: " << x; // Display the input value
```

Good To Know

cout is pronounced "see-out". Used for output, and uses the insertion operator (<<)

cin is pronounced "see-in". Used for input, and uses the extraction operator (>>)

Lets create simple calculator

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

int main()
{

//Create simple calculator
int x,y;
cout << "Enter First digit : ";
cin >> x;
cout << "\nEnter Second digit : ";
cin >> y;
cout << "\nSum = " << x+y;
   return 0;
}
```

Basic Data Types

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

Data Type	Size	Description
Bool	1 bit	Stores true or false values (0/1)
char	1 byte	Stores a single character/letter/number, or ASCII values
int	2 or 4 bytes	Stores whole numbers, without decimals
float	4 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 6-7 decimal digits

double	8 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits

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.

One example with character datatype

```
#include<iostream>

using namespace std;


int main()

{

//Create simple calculator

char a = 65, b = 66, c = 67;

cout << a;

cout << b;

cout << c;


   return 0;

}
```

OUTPUT: ABC

String Types

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;
```