

Name:Muhammad Sahi

Roll no:CSU-F17-107

Class:BSCS-3-B

Subject:OOP

Assignment no:1

Submitted to:Sir Usman

### **What is Syntax**

### Ans: Syntax is the grammar, structure, or order of the elements in a language statement

### Syntax and Structure of C++ program

#### First C++ program

#include <iostream>

using namespace std;

int main()

{

cout << "Hello this is C++";

}

**Header files** are included at the beginning just like in C program. Here iostream is a header file which provides us with input & output streams. Header files contained predeclared function libraries, which can be used by users for their ease.

**Using namespace std**, tells the compiler to use standard namespace. Namespace collects identifiers used for class, object and variables. NameSpace can be used by two ways in a program, either by the use of using statement at the beginning, like we did in above mentioned program or by using name of namespace as prefix before the identifier with scope resolution (::) operator.

*Example :* std::cout << "A";

**main()**, is the function which holds the executing part of program its return type is int.

**cout <<**, is used to print anything on screen, same as printf in C language. **cin** and **cout** are same as scanf and printf, only difference is that you do not need to mention format specifiers like, %d for int etc, in cout & cin

1. **Data types**

# C++ Data Types

All [variables](https://www.geeksforgeeks.org/variables-and-keywords-in-c/) use data-type during declaration to restrict the type of data to be stored. Therefore, we can say that data types are used to tell the variables the type of data it can store.  
Whenever a variable is defined in C++, the compiler allocates some memory for that variable based on the data-type with which it is declared. Every data type requires different amount of memory.

Data types in C++ is mainly divided into two types:

1. **Primitive Data Types**: These data types are built-in or predefined data types and can be used directly by the user to declare variables. example: int, char , float, bool etc. Primitive data types available in C++ are:
   * Integer
   * Character
   * Floating Point
   * Double Floating Point
   * Valueless or Void
   * String

**Abstract or user defined data type**: These data types are defined by user itself. Like, defining a class in C++ or a structure.

* **Integer**: An **integer** is a whole number (not a fraction) that can be positive, negative, or zero. Therefore, the numbers 10, 0, -25, and 5,148 are all **integers**. Unlike floating point numbers, **integers** cannot have decimal places. **Integers** are a commonly used**data type** in computer programming.Keyword used for integer data types is **int**. Integers typically requires 4 bytes of memory space.

Integers are whole numbers that can have both positive and negative values but no decimal values. Example: 0, -5, 10

In C programming, keyword int is used for declaring integer variable. For example: int id;

Here, id is a variable of type integer.

You can declare multiple variable at once in C programming. For example:

int id, age;

## **float - Floating types**

* **Floating Point**: Floating Point data type is used for storing single precision floating point values or decimal values. Keyword used for floating point data type is **float**. Float variables typically requires 4 byte of memory space.

Floating type variables can hold real numbers such as: 2.34, -9.382, 5.0 etc. You can declare a floating point variable in C by using either float or double keyword. For example:

float accountBalance;

double bookPrice;

* **Character**: Character data type is used for storing characters. Keyword used for character data type is **char**. Characters typically requires 1 byte of memory space.

 For example:

char test = 'h';

Here, test is a character variable. The value of test is 'h'.

The size of character variable is 1 byte.

* **[Wide Character](https://www.geeksforgeeks.org/wide-char-and-library-functions-in-c/):** Wide character data type is also a character data type but this data type has size greater than the normal 8-bit datatype. Represented by **wchar\_t**. It is generally 2 or 4 bytes long
* **Double Floating Point**: Double Floating Point data type is used for storing double precision floating point values or decimal values. Keyword used for double floating point data type is **double**. Double variables typically requires 8 byte of memory space.
* The size of float (single precision float data type) is 4 bytes. And the size of double (double precision float data type) is 8 bytes. Floating point variables has a precision of 6 digits whereas the precision of double is 14 digits
* **void**: Void means without any value. void datatype represents a valueless entity. Void data type is used for those function which does not returns a value.

1) For a function that does not return anything. Basically all the computation are done within the function and you've nothing to return the caller. Usually you don't find such a function  
  
E.g.  
  
void DoSomething(int a, int b, int \*c)  
{  
    \*c = a + b  
}

**. String**

The term **string** generally means an ordered sequence of characters, with a first character, a second character, and so on, and in most programming languages such**strings** are enclosed in either single or double quotes. In **C++** the enclosing delimiters are double quotes

#include <iostream>

#include <string>

using namespace std;

int main ()

{

string mystring = "This is a string";

cout << mystring;

return 0;

}

## **User-defined data types**

We have three types of user-defined data types in C++  
1. struct  
2. union  
3. enum

## **Derived data types in C++**

We have three types of derived-defined data types in C++  
1. Array  
2. Function  
3. Pointer

1. **Array**

**Arrays in C++** An **array** is a collection of elements of the same type placed in contiguous memory locations that can be individually referenced by using an index to a unique identifier. Five values of type int can be declared as an **array** without having to declare five different variables (each with its own identifier)

It may be one dimensional and two dimensional.

For example

#include<iostream>

using namespace std;

int main()

{

int i,j;

int a[3][5];

for(int j=0;j<3;j++)

{

for(int i=0;i<5;i++)

{

cout<<"enter the value of index"<<j<<i<<endl;

cin>>a[j][i];

}

}

cout<<a[j][i]<<" ";

cout<<endl;

}

1. **Loop Operators**

**Loops are used in programming to repeat a specific block of code.**

**. While loop**

* The while loop evaluates the test expression.
* If the test expression is true, codes inside the body of while loop is evaluated.
* Then, the test expression is evaluated again. This process goes on until the test expression is false.
* When the test expression is false, while loop is terminated

#include <iostream>using namespace std;

int main() {

int number, i = 1, factorial = 1;

cout << "Enter a positive integer: ";

cin >> number;

while ( i <= number) {

factorial \*= i; //factorial = factorial \* i;

++i;

}

cout<<"Factorial of "<< number <<" = "<< factorial;

return 0;}

**. Do While**

The do while loop is a variant of the while loop with one important difference. The body of do while loop is executed once before the test expression is checked

* The codes inside the body of loop is executed at least once. Then, only the test expression is checked.
* If the test expression is true, the body of loop is executed. This process continues until the test expression becomes false.
* When the test expression is false, do...while loop is terminated.

#include <iostream>using namespace std;

int main() {

float number, sum = 0.0;

do {

cout<<"Enter a number: ";

cin>>number;

sum += number;

}

while(number != 0.0);

cout<<"Total sum = "<<sum;

return 0;}

**. For loop**

A **for** loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times

**.** The **init** step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.

**.** Next, the **condition** is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and flow of control jumps to the next statement just after the for loop.

**.** After the body of the for loop executes, the flow of control jumps back up to the **increment** statement. This statement can be left blank, as long as a semicolon appears after the condition.

**.** The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the for loop terminates.

#include <iostream>using namespace std;

int main () {

// for loop execution

for( int a = 10; a < 20; a = a + 1 ) {

cout << "value of a: " << a << endl;

}

return 0;}

**. Switch**

A **switch** statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

For example

#include<iostream>

using namespace std;

int main()

{

char ch;

cout<<"enter a number";

cin>>ch;

switch(ch)

{

case'1':

cout<<"my first assignment is to write a program to find sum of two numbers";

break;

case'2':

cout<<"my second assignment is to write a program to find radius of a circle";

break;

case'3':

cout<<"my third assignment is to write a program of quadratic formula";

break;

case'4':

cout<<"my fourth assignment is to write a program to find distance of a car";

break;

case'5':

cout<<"my fifth assignment is to write a program using switch statement";

break;

default:

cout<<"invalid";

}

}

1. **Functions**

A **function** is a group of statements that together perform a task. Every **C++** program has at least one**function**, which is main(), and all the most trivial programs can define additional **functions**. You can divide up your code into separate **functions**.

It also includes built in and user defined functions.

For example

#include<iostream>

using namespace std;

int addition(int a, int b);

int subtraction( int a, int b);

int main()

{

int a,b;

cout<<"enter a number";

cin>>a;

cout<<"enter a number";

cin>>b;

cout<<"the addition is"<<addition(a,b)<<endl;

cout<<"the subtraction is"<<subtraction(a,b)<<endl;

return 0;

}

int addition( int a, int b)

{

return a+b;

}

int subtraction(int a , int b)

{

if (a>b)

int c=a-b;

else

{

int c=b-a;

return c;

}

}

1. **Pointers**

A **pointer** is a variable that stores the address of another variable. Unlike other variables that hold values of a certain type, **pointer** holds the address of a variable.

For example

#include<iostream>

using namespace std;

int main()

{

int a=50;

cout<<"the value of a"<<a<<endl;

int \*b;

b=&a;

\*b=100;

cout<<"the value of a using \*b"<<a<<endl;;

int\*\*c;

c=&b;

\*\*c=200;

cout<<"value of a using \*\*c;"<<a<<endl;

}