**C++ Data Types**

All variables 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 a different amount of memory.

Data types in C++ is mainly divided into three 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
* Boolean
* Floating Point
* Double Floating Point
* Valueless or Void
* Wide Character

**2.Derived Data Types:** The data-types that are derived from the primitive or built-in datatypes are referred to as Derived Data Types. These can be of four types namely:

* Function
* Array
* Pointer
* Reference

**3.Abstract or User-Defined Data Types:** These data types are defined by user itself. Like, defining a class in C++ or a structure. C++ provides the following user-defined datatypes:

* Class
* Structure
* Union
* Enumeration
* Typedef defined DataType

This article discusses **primitive data types** available in C++.

* **Integer**: Keyword used for integer data types is **int**. Integers typically requires 4 bytes of memory space and ranges from -2147483648 to 2147483647.
* **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 and ranges from -128 to 127 or 0 to 255.
* **Boolean:** Boolean data type is used for storing boolean or logical values. A boolean variable can store either true or false. Keyword used for boolean data type is **bool**.
* **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.
* **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.
* **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.
* **Wide Character:** 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.

**Datatype Modifiers**

**Data Type Size (in bytes) Range**

short int 2 -32,768 to 32,767

unsigned short int 2 0 to 65,535

unsigned int 4 0 to 4,294,967,295

int 4 -2,147,483,648 to 2,147,483,647

long int 4 -2,147,483,648 to 2,147,483,647

unsigned long int 4 0 to 4,294,967,295

long long int 8 -(2^63) to (2^63)-1

unsigned long long int 8 0 to 18,446,744,073,709,551,615

signed char 1 -128 to 127

unsigned char 1 0 to 255

float 4

double 8

long double 12

wchar\_t 2 or 4 1 wide character

**Note** : Above values may vary from compiler to compiler. In the above example, we have considered GCC 32 bit.

We can display the size of all the data types by using the sizeof() operator and passing the keyword of the datatype as argument to this function as shown below:

#include<iostream>

using namespace std;

int main()

{

    cout << "Size of char : " << sizeof(char)

        << " byte" << endl;

    cout << "Size of int : " << sizeof(int)

        << " bytes" << endl;

    return 0;

}

**Variables in C++**

A variable is a name given to a memory location. It is the basic unit of storage in a program.

* The value stored in a variable can be changed during program execution.
* A variable is only a name given to a memory location, all the operations done on the variable effects that memory location.
* In C++, all the variables must be declared before use.

**How to declare variables?**

// Declaring a single variable:

type variable\_name;

// Declaring multiple variables:

type variable1\_name, variable2\_name, variable3\_name;

A variable name can consist of alphabets (both upper and lower case), numbers and the underscore ‘\_’ character. However, the name must not start with a number.

**datatype**: Type of data that can be stored in this variable.

**variable\_name**: Name given to the variable.

**value**: It is the initial value stored in the variable.

**Loops in C and C++**

Loops in programming come into use when we need to repeatedly execute a block of statements. For example: Suppose we want to print “Hello World” 10 times. This can be done in two ways as shown below:

**For Loop:**

A for loop is a repetition control structure which allows us to write a loop that is executed a specific number of times. The loop enables us to perform n number of steps together in one line.

*Syntax:*

*for (initialization expr; test expr; update expr)*

*{*

*// body of the loop*

*// statements we want to execute*

*}*

**C++ program to illustrate for loop**

#include <iostream>

using namespace std;

int main()

{

for (int i = 1; i <= 10; i++)

{

cout << "Hello World\n";

}

return 0;

}

Output:

Hello World

Hello World

Hello World

Hello World

Hello World

Hello World

Hello World

Hello World

Hello World

Hello World

**While Loop:**

While studying for loop we have seen that the number of iterations is known beforehand, i.e. the number of times the loop body is needed to be executed is known to us. while loops are used in situations where we do not know the exact number of iterations of loop beforehand. The loop execution is terminated on the basis of test condition.

**Syntax**:

**initialization expression;**

while (**test\_expression**)

{

// statements

**update\_expression;**

}

**C++ program to illustrate while loop**

#include <iostream>

using namespace std;

 int main()

{

    int i = 1;

    // test expression

    while(i < 6)

    {    cout << "Hello World\n";

    // update expression

        i++;

    }

    return 0;

}

Output:

Hello World

Hello World

Hello World

Hello World

Hello World

**do while loop:**

In do while loops also the loop execution is terminated on the basis of test condition. The main difference between do while loop and while loop is in do while loop the condition is tested at the end of loop body, i.e do while loop is exit controlled whereas the other two loops are entry controlled loops.

**Note**: In do while loop the loop body will execute at least once irrespective of test condition.

**Syntax**:

**initialization expression;**

do

// statements

{

**update\_expression;**

} while (**test\_expression**);

Note: Notice the semi – colon(“;”) in the end of loop.

**program to illustrate do-while loop**

#include <iostream>

using namespace std;

int main()

{

    int i = 2; // Initialization expression

    do{

        cout << "Hello World\n"; // loop body

        i++; // update expression

    }

    while (i < 1); // test expression

    return 0;

}// codition not true but one time print Hello World.

Output:

Hello World

In the above program the test condition (i<1) evaluates to false. But still as the loop is exit – controlled the loop body will execute once.

**What about an Infinite Loop?**

An infinite loop (sometimes called an endless loop ) is a piece of coding that lacks a functional exit so that it repeats indefinitely. An infinite loop occurs when a condition always evaluates to true. Usually, this is an error.

#include <iostream>

using namespace std;

int main ()

{

    int i;

        // This is an infinite for loop as the condition

        // expression is blank

    for ( ; ; ){

        cout << "This loop will run forever.\n";

    }

}

Output:

This loop will run forever.

This loop will run forever.

.................. .

**Important Points:**

* Use for loop when number of iterations is known beforehand, i.e. the number of times the loop body is needed to be executed is known.
* Use while loops where exact number of iterations is not known but the loop termination condition is known.
* Use do while loop if the code needs to be executed at least once like in Menu driven programs

