C++ Day1

25 August 2024 07:40

```
// C++ program to display "Hello World"

// Header file for input output functions
#include <iostream>

using namespace std;

// Main() function: where the execution of
// program begins

int main()
{
    // Prints hello world
    cout << "Hello World";

    return 0;
}
```

This line is a comment line. A comment is used to display additional information about the program. A comment does not contain any programming logic.

Single Line Comment: // commented line
Multi Line Comment: /* All commented line */

This is a pre-processor directive. The **#include** directive tells the compiler to include the content of a file in the source code.

For example, #include<iostream> tells the compiler to include the standard iostream file which contains declarations of all the standard input/output library functions.

This is used to import the entity of the std namespace into the current namespace of the program.

cout is part of the standard library, and all the elements in the standard C++ library are declared within what is called a *namespace*: the namespace `std`.

A function is a group of statements that are designed to perform a specific task. The main() function is the entry point of every C++ program, no matter where the function is located in the program.

The opening braces '{' indicates the beginning of the main function and the closing braces '}' indicates the ending of the main function.

std::cout is an instance of the std::iostream class, that is used to display output on the screen. Everything followed by the character << in double quotes " " is displayed on the output device. The semi-colon character at the end of the statement is used to indicate that the statement is ending there

This statement is used to return a value from a function and indicates the finishing of a function. This statement is basically used in functions to return the results of the operations performed by a function.

C++ Syntax:

	1	`#include <iostream>`	Header File
	2	`using namespace std;`	Standard Namespace
	3	`int main()`	Main Function
FUNCTION BODY	4	`{`	
	5	`int num1 = 24;` `int num2 = 34;`	Declaration of Variable
	6	`int result = num1 + num2;`	Expressions
	7	`cout << result << endl;`	Output
	8	`return 0;`	Return Statement
	9	`}`	

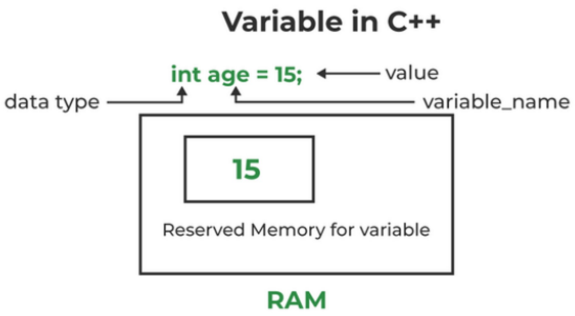
C++ Variables

Variables in C++ 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.

A typical variable declaration is of the form:

```
// 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.



Rules for declaring a variable:

1. The name of the variable contains letters, digits, and underscores.
2. The name of the variable is case sensitive (ex Arr and arr both are different variables).
3. The name of the variable does not contain any whitespace and special characters (ex #,$,%,*, etc).
4. All the variable names must begin with a letter of the alphabet or an underscore(_).
5. We cannot used C++ keyword(ex float,double,class)as a variable name.

Valid variable names:

```
int x;   //can be letters
int _yz; //can be underscores
int z40;//can be letters
```

Invalid variable names:

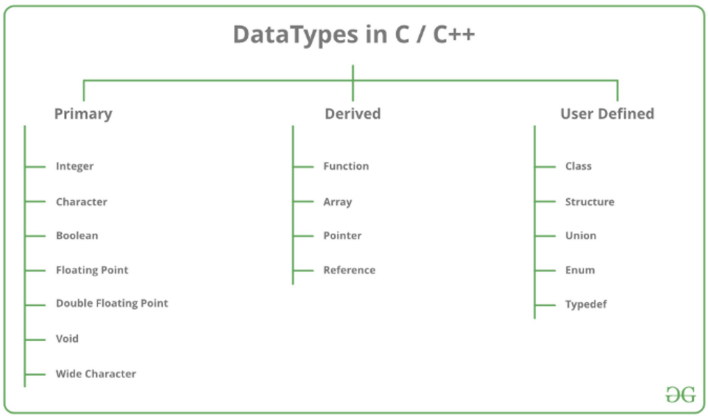
```
int 89; Should not be a number
int a b; //Should not contain any whitespace
int double;// C++ keyword CAN NOT BE USED
```

```cpp
// C++ program to show difference between
// definition and declaration of a
// variable
#include <iostream>
using namespace std;

int main()
{
    // this is declaration of variable a
    int a;

    // this is initialisation of a
    a = 10;

    // this is definition = declaration + initialisation
    int b = 20;

    // declaration and definition
    // of variable 'a123'
    char a123 = 'a';

    // This is also both declaration and definition
    // as 'c' is allocated memory and
    // assigned some garbage value.
    float c;

    // multiple declarations and definitions
    int _c, _d45, e;

    // Let us print a variable
    cout << a123 << endl;

    return 0;
}
```

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 they can store.

C++ supports the following data types:

1. **Primary** or **Built-in** or **Fundamental data type**
2. **Derived data types**
3. **User-defined data types**
4. 

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	-3.4×10^38 to 3.4×10^38
double	8	-1.7×10^308 to1.7×10^308
long double	12	-1.1×10^4932 to1.1×10^4932
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.*

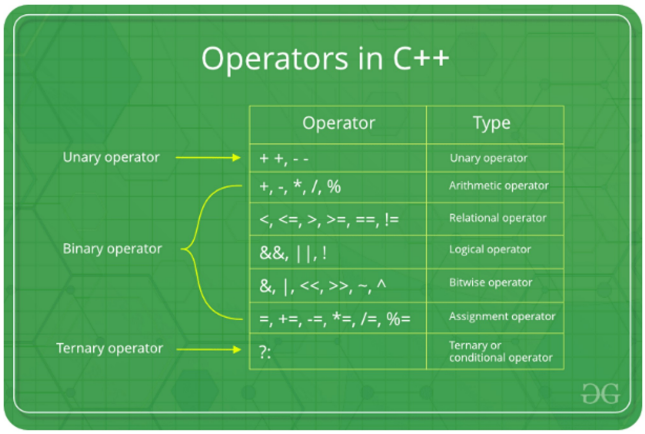
```
// C++ Program to Demonstrate the correct size
// of various data types on your computer.
#include <iostream>
using namespace std;

int main()
{
    cout << "Size of char : " << sizeof(char) << endl;
    cout << "Size of int : " << sizeof(int) << endl;

    cout << "Size of long : " << sizeof(long) << endl;
    cout << "Size of float : " << sizeof(float) << endl;

    cout << "Size of double : " << sizeof(double) << endl;

    return 0;
}
```

Operators in C++:

An **operator** is a symbol that operates on a value to perform specific mathematical or logical computations.

Operators in C++ can be classified into 6 types:

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Ternary or Conditional Operators



Next Chapter:

C++ Control Statements

- C++ if Statement
- C++ if-else Statement
- C++ if-else-if Ladder
- C++ Nested if-else Statement
- C++ Switch Statement
- C++ Loops
- C++ for Loop
- C++ Range-Based for Loop
- C++ while Loop
- C++ do...while Loop