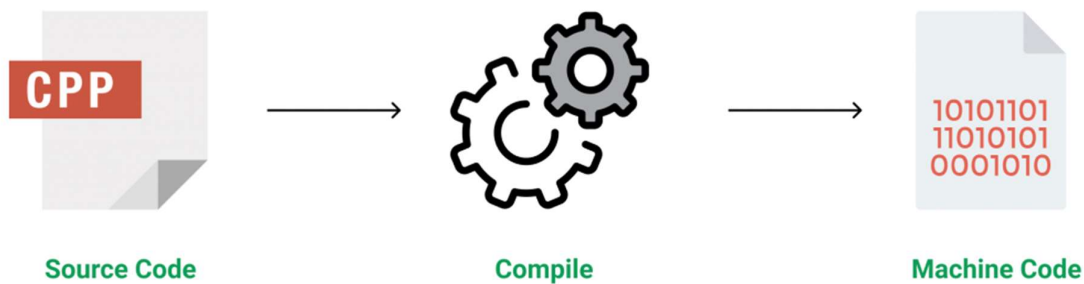


Introduction to C++ Programming Language

C++ is a general-purpose programming language that was developed as an enhancement of the C language to include object-oriented paradigm. It is an imperative and a **compiled** language.



Some of the **features & key-points** to note about the programming language are as follows:

- **SIMPLE**
- **Machine Independent but Platform Dependent:**
- **Mid-level language:** It is a mid-level language as we can do both systems-programming (drivers, kernels, networking etc.) and build large-scale user applications (Media Players, Photoshop, Game Engines etc.)
- **Speed of execution**
- **Object-Oriented**
- **STL (MOST IMPORTANT)**

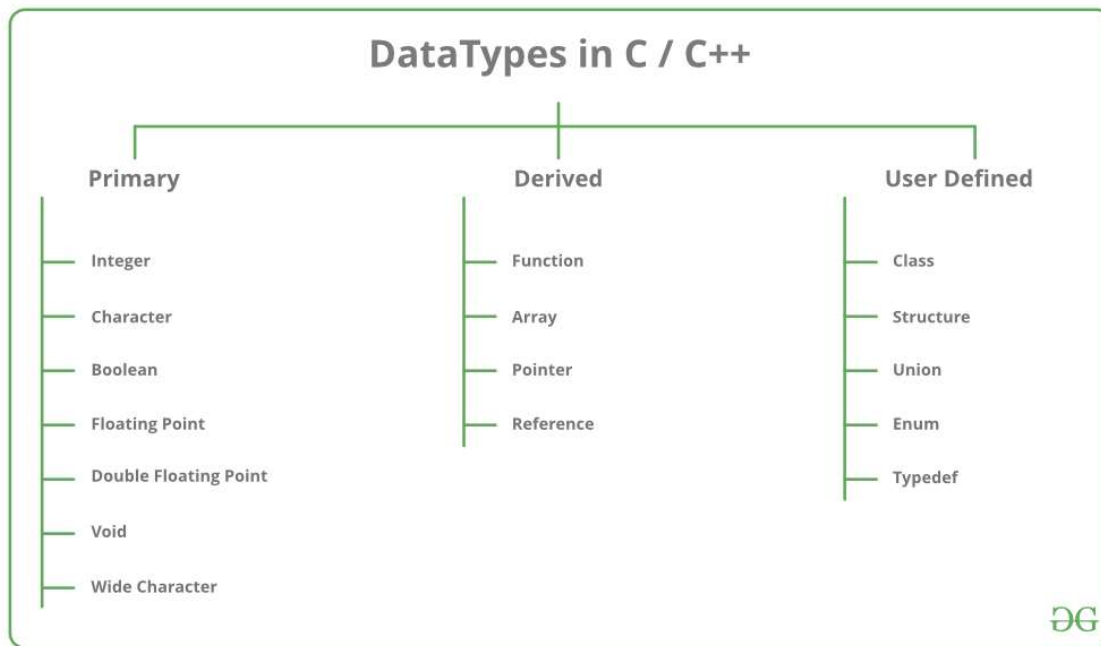
Applications of C++:

C++ finds varied usage in applications such as:

- Operating Systems & Systems Programming. e.g. *Linux-based OS (Ubuntu etc.)*
- Browsers (*Chrome & Firefox*)
- Graphics & Game engines (*Photoshop, Blender, Unreal-Engine*)
- Database Engines (*MySQL, MongoDB, Redis etc.*)
- Cloud/Distributed Systems

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.



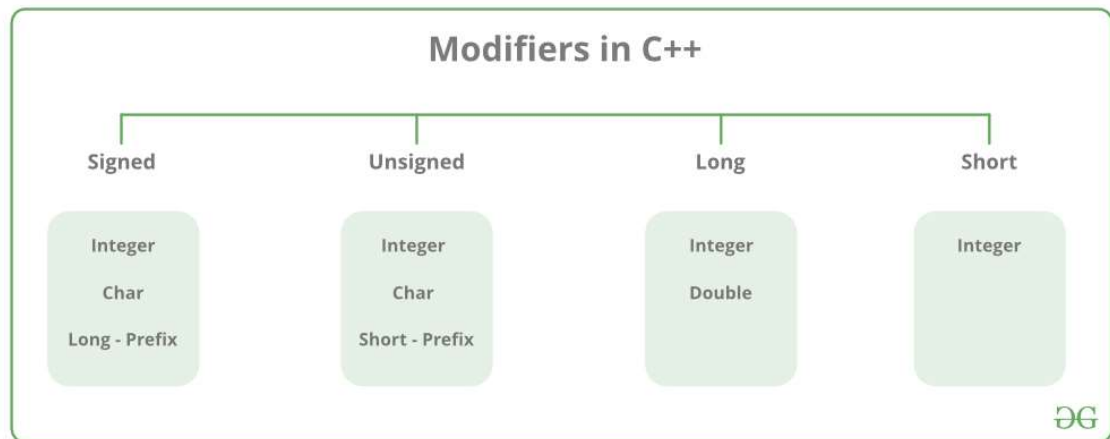
Primitive data types

- **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.

Datatype Modifiers

As the name implies, datatype modifiers are used with the built-in data types to modify the length of data that a particular data type can hold.

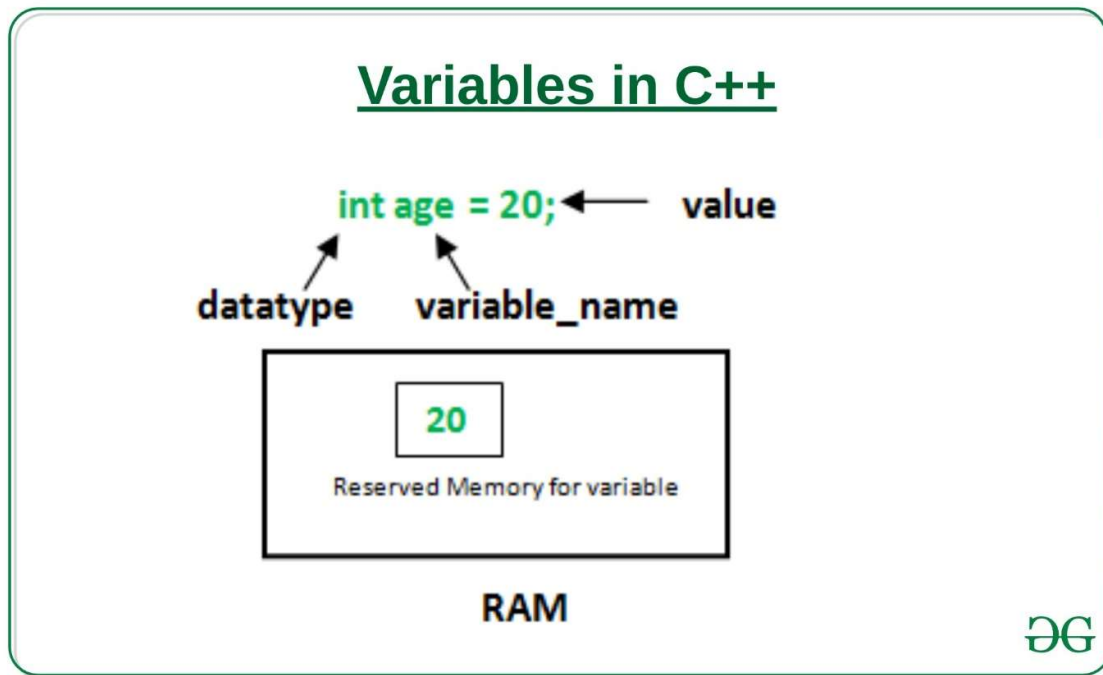


Data Type	Size (in bytes)	Range
short int	2	-32,768 to 32,767
unsigned short int	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
int	4	-2,147,483,648 to 2,147,483,647
long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	8	0 to 4,294,967,295
long long int	8	-(2 ⁶³) to (2 ⁶³)-1
unsigned long long 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	

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.



Difference between variable declaration and definition

The **variable declaration** refers to the part where a variable is first declared or introduced before its first use. A **variable definition** is a part where the variable is assigned a memory location and a value. Most of the times, variable declaration and definition are done together.

CPP

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

int main()
{
    // declaration and definition
    // of variable 'a123'
    char a123 = 'a';

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

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

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

    return 0;
}
```

Basic Input / Output in C++

The two keywords **cout** in **C++** and **cin** in **C++** are used very often for printing outputs and taking inputs respectively. These two are the most basic methods of taking input and printing output in C++. To use cin and cout in C++ one must include the header file *iostream* in the program.

- **Standard output stream (cout):** Usually the standard output device is the display screen. The C++ **cout** statement is the instance of the ostream class. It is used to produce output on the standard output device which is usually the display screen. The data needed to be displayed on the screen is inserted in the standard output stream (cout) using the insertion operator (<<).

C++

```
#include <iostream>

using namespace std;

int main()
{
    char sample[] = "GeeksforGeeks";

    cout << sample << " - A computer science portal for geeks";

    return 0;
}
```

Output:

```
GeeksforGeeks - A computer science portal for geeks
```

- **Standard input stream (cin):** Usually the input device in a computer is the keyboard. C++ cin statement is the instance of the class **istream** and is used to read input from the standard input device which is usually a keyboard. The extraction operator (>>) is used along with the object **cin** for reading inputs. The extraction operator extracts the data from the object **cin** which is entered using the keyboard.

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

int main()
{
    int age;

    cout << "Enter your age:";
    cin >> age;
    cout << "\nYour age is: " << age;

    return 0;
}
```

Input:

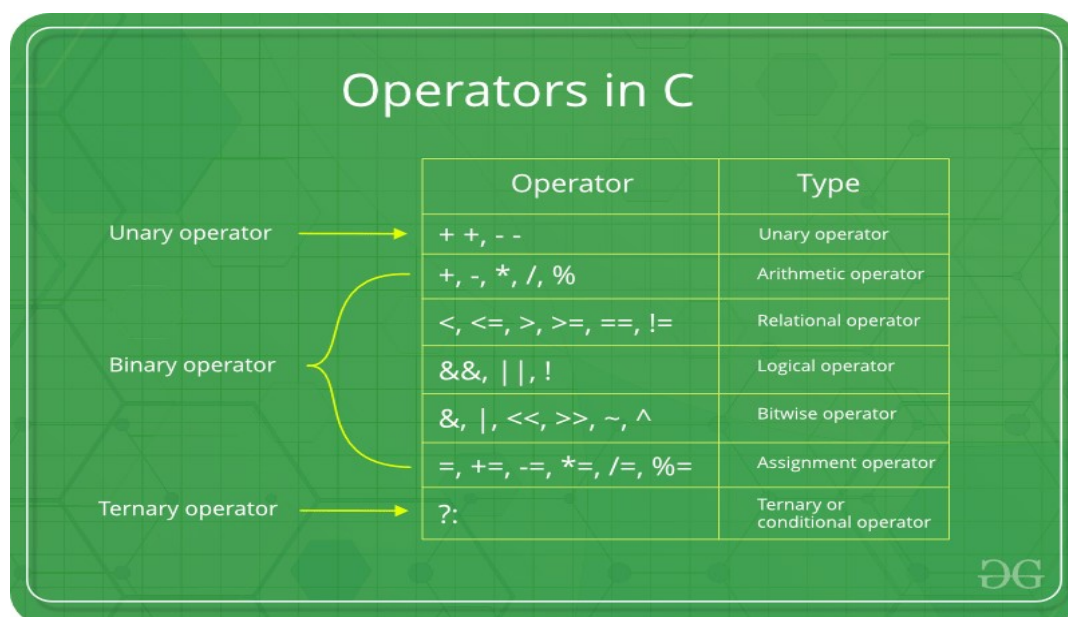
18

Output:

Enter your age:
Your age is: 18

Operators in C / C++

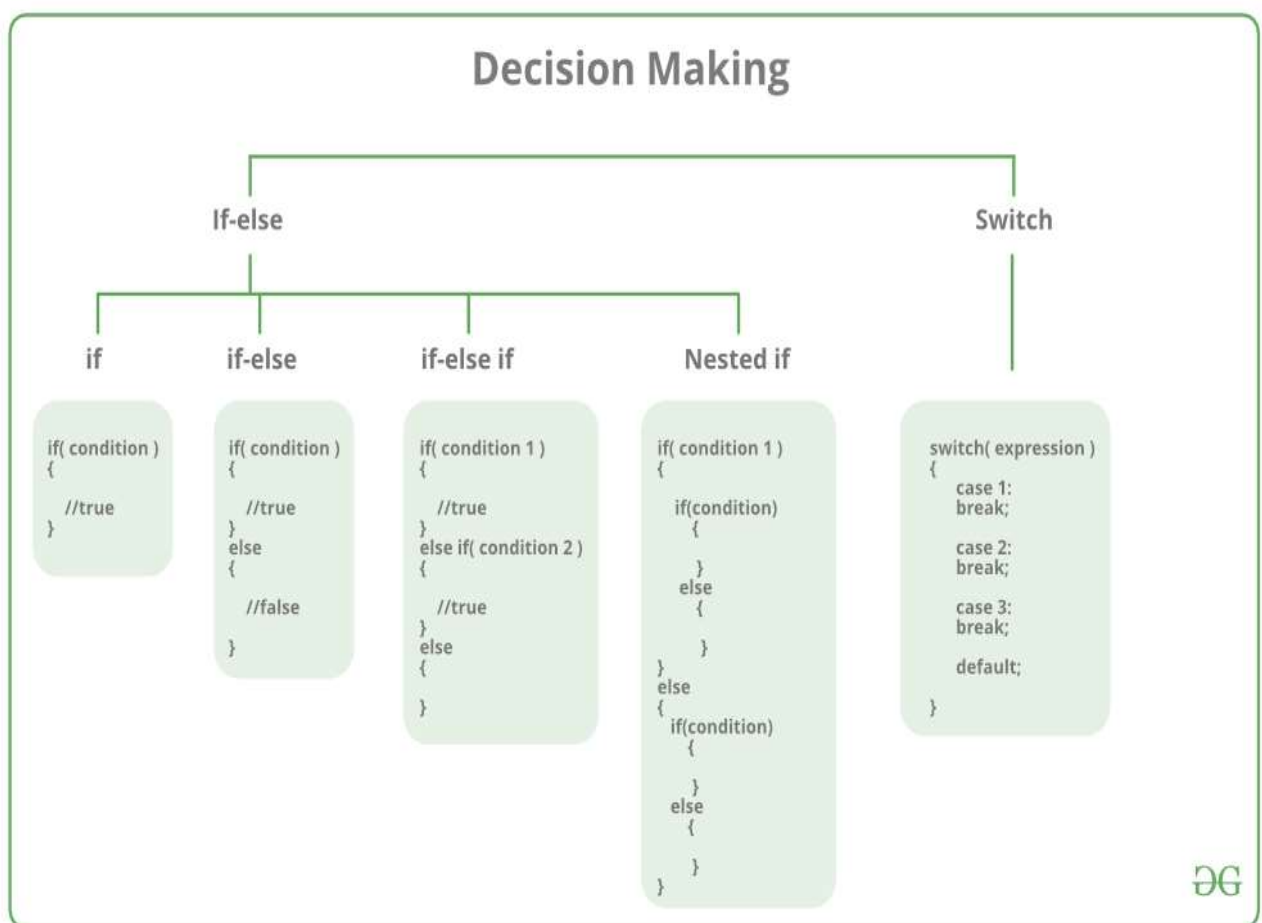
We can define operators as symbols that help us to perform specific mathematical and logical computations on operands.



Other Operators:

- **SIZEOF** operator
- **Comma** operator
- **Conditional** operator (ternary operator)

Decision Making in C / C++ (if, if. Else, Nested if, if-else-if)



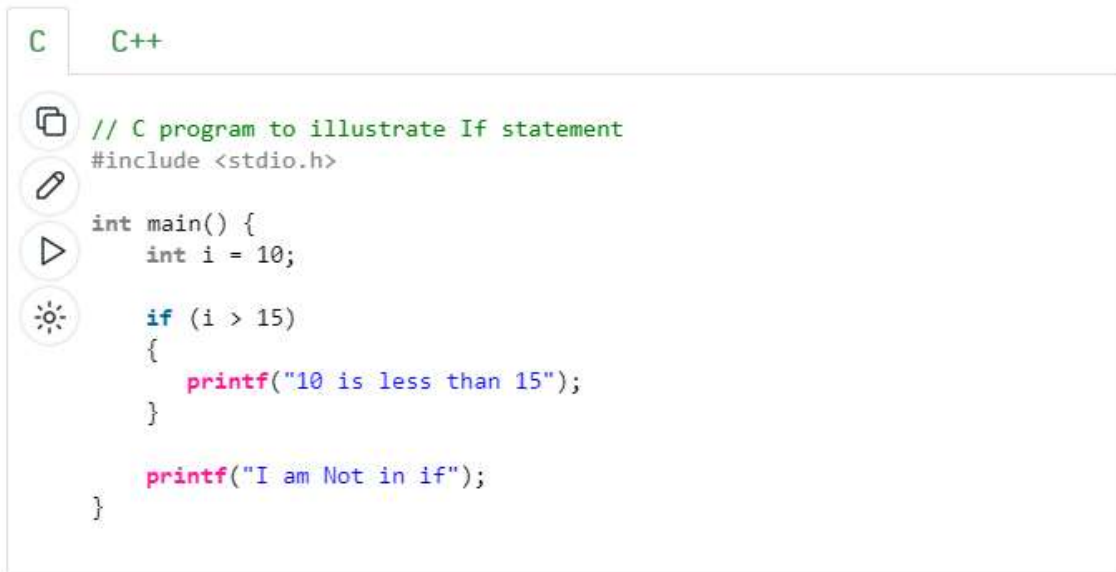
If statement in C/C++

If statement is the most simple decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e. if a certain condition is true then a block of statement is executed otherwise not.

Syntax:

If (condition)

```
{  
    // Statements to execute if  
    // condition is true  
}
```



```
C    C++  
  
// C program to illustrate If statement  
#include <stdio.h>  
  
int main() {  
    int i = 10;  
  
    if (i > 15)  
    {  
        printf("10 is less than 15");  
    }  
  
    printf("I am Not in if");  
}
```

If-else in C/C++

If (condition)

```
{  
    // executes this block if  
    // condition is true  
}
```

Else

```
{  
    // executes this block if  
    // condition is false  
}
```

C C++

```
// C program to illustrate If statement
#include <stdio.h>

int main() {
    int i = 20;

    if (i < 15){
        printf("i is smaller than 15");
    }
    else{
        printf("i is greater than 15");
    }
    return 0;
}
```

Syntax:

```
If (condition1)
{
    // executes when condition1 is true
    If (condition2)
    {
        // executes when condition2 is true
    }
}
```

```
// C++ program to illustrate nested-if statement
#include <iostream>
using namespace std;

int main()
{
    int i = 10;

    if (i == 10)
    {
        // First if statement
        if (i < 15)
            cout<<"i is smaller than 15\n";

        // Nested - if statement
        // Will only be executed if statement above
        // is true
        if (i < 12)
            cout<<"i is smaller than 12 too\n";
        else
            cout<<"i is greater than 15";
    }

    return 0;
}
```

