C++

Meaning

**C++** is an object-oriented **programming** (OOP) **language** that is viewed by many as the best **language** for creating large-scale applications. **C++** is a superset of the C **language**. A related **programming language**, Java, is based on **C++** but optimized for the distribution of program objects in a network such as the Internet.

Java is somewhat simpler and easier to learn than C++ and has characteristics that give it other advantages over C++. However, both languages require a considerable amount of study.

Syntax

When we consider a C++ program, it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what a class, object, methods, and instant variables mean.

* **Object** − Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors - wagging, barking, eating. An object is an instance of a class.
* **Class** − A class can be defined as a template/blueprint that describes the behaviors/states that object of its type support.
* **Methods** − A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
* **Instance Variables** − Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

C++ Program Structure

* The C++ language defines several headers, which contain information that is either necessary or useful to your program. For this program, the header **<iostream>** is needed.
* The line **using namespace std;** tells the compiler to use the std namespace. Namespaces are a relatively recent addition to C++.
* The next line '**// main() is where program execution begins.**' is a single-line comment available in C++. Single-line comments begin with // and stop at the end of the line.
* The line **int main()** is the main function where program execution begins.
* The next line **cout << "Hello World";** causes the message "Hello World" to be displayed on the screen.
* The next line **return 0;** terminates main( )function and causes it to return the value 0 to the calling process.

## Compile and Execute C++ Program

* Open a text editor and add the code as above.
* Save the file as: hello.cpp
* Open a command prompt and go to the directory where you saved the file.
* Type 'g++ hello.cpp' and press enter to compile your code. If there are no errors in your code the command prompt will take you to the next line and would generate a.out executable file.
* Now, type 'a.out' to run your program.
* You will be able to see ' Hello World ' printed on the window.

## C++ Keywords

|  |  |  |  |
| --- | --- | --- | --- |
| asm | else | new | this |
| auto | enum | operator | throw |
| bool | explicit | private | true |
| break | export | protected | try |
| case | extern | public | typedef |
| catch | false | register | typeid |
| char | float | reinterpret\_cast | typename |
| class | for | return | union |
| const | friend | short | unsigned |
| const\_cast | goto | signed | using |
| continue | if | sizeof | virtual |
| default | inline | static | void |
| delete | int | static\_cast | volatile |
| do | long | struct | wchar\_t |
| double | mutable | switch | while |
| dynamic\_cast | namespace | template |  |

## Arithmetic Operators

There are following arithmetic operators supported by C++ language −

Assume variable A holds 10 and variable B holds 20, then −

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Adds two operands | A + B will give 30 |
| - | Subtracts second operand from the first | A - B will give -10 |
| \* | Multiplies both operands | A \* B will give 200 |
| / | Divides numerator by de-numerator | B / A will give 2 |
| % | Modulus Operator and remainder of after an integer division | B % A will give 0 |
| ++ | [Increment operator](https://www.tutorialspoint.com/cplusplus/cpp_increment_decrement_operators.htm), increases integer value by one | A++ will give 11 |
| -- | [Decrement operator](https://www.tutorialspoint.com/cplusplus/cpp_increment_decrement_operators.htm), decreases integer value by one | A-- will give 9 |

## Relational Operators

There are following relational operators supported by C++ language

Assume variable A holds 10 and variable B holds 20, then.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | Checks if the values of two operands are equal or not, if yes then condition becomes true. | (A == B) is not true. |
| != | Checks if the values of two operands are equal or not, if values are not equal then condition becomes true. | (A != B) is true. |
| > | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. | (A > B) is not true. |
| < | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. | (A < B) is true. |
| >= | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. | (A >= B) is not true. |
| <= | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. | (A <= B) is true. |

## Logical Operators

There are following logical operators supported by C++ language.

Assume variable A holds 1 and variable B holds 0, then.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| && | Called Logical AND operator. If both the operands are non-zero, then condition becomes true. | (A && B) is false. |
| || | Called Logical OR Operator. If any of the two operands is non-zero, then condition becomes true. | (A || B) is true. |
| ! | Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true, then Logical NOT operator will make false. | !(A && B) is true. |

Conditional Statement

C++ Conditions and If Statements

C++ supports the usual logical conditions from mathematics:

* Less than: a < b
* Less than or equal to: a <= b
* Greater than: a > b
* Greater than or equal to: a >= b
* Equal to a == b
* Not Equal to: a != b

You can use these conditions to perform different actions for different decisions.

C++ has the following conditional statements:

* Use if to specify a block of code to be executed, if a specified condition is true
* Use else to specify a block of code to be executed, if the same condition is false
* Use else if to specify a new condition to test, if the first condition is false
* Use switch to specify many alternative blocks of code to be executed

Use the if statement to specify a block of C++ code to be executed if a condition is true.

Syntax:

if (condition) {  
  *// block of code to be executed if the condition is true*  
}

## The else Statement

Use the else statement to specify a block of code to be executed if the condition is false.

Syntax

if (condition) {  
  *// block of code to be executed if the condition is true*  
} else {  
  *// block of code to be executed if the condition is false*  
}

## The else if Statement

Use the else if statement to specify a new condition if the first condition is false.

Syntax

if (condition1) {  
  *// block of code to be executed if condition1 is true*  
} else if (condition2) {  
  *// block of code to be executed if the condition1 is false and condition2 is true*  
} else {  
  *// block of code to be executed if the condition1 is false and condition2 is false*  
}

## Short Hand If...Else (Ternary Operator)

There is also a short-hand if else, which is known as the **ternary operator** because it consists of three operands. It can be used to replace multiple lines of code with a single line. It is often used to replace simple if else statements:

Syntax

variable = (condition) ? expressionTrue : expressionFalse;