OOPS & C ++

C++ is a cross-platform language that can be used to create high-performance applications.

C++ was developed by Bjarne Stroustrup, as an extension to the [C language](https://www.w3schools.com/c/index.php).

## Use C++

C++ is one of the world's most popular programming languages.

C++ can be found in today's operating systems, Graphical User Interfaces, and embedded systems.

C++ is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.

C++ is portable and can be used to develop applications that can be adapted to multiple platforms.

C++ is fun and easy to learn!

As C++ is close to [C](https://www.w3schools.com/c/index.php), [C#](https://www.w3schools.com/cs/index.php) and [Java](https://www.w3schools.com/java/default.asp), it makes it easy for programmers to switch to C++ or vice versa.

EX:

#include <iostream>  
  
int main()

{  
  cout << "Hello World!";  
 return 0;  
}

## Single-line Comments

Single-line comments start with two forward slashes (//).

This example uses a single-line comment before a line of code:

### Example

// This is a comment  
cout << "Hello World!";

## C++ Multi-line Comments

Multi-line comments start with /\* and ends with \*/.

Any text between /\* and \*/ will be ignored by the compiler:

### Example

/\* The code below will print the words Hello World!  
to the screen, and it is amazing \*/

cout << "Hello World!";

## C++ Variables

Variables are containers for storing data values.

In C++, there are different **types** of variables (defined with different keywords), for example:

* int - stores integers (whole numbers), without decimals, such as 123 or -123
* double - stores floating point numbers, with decimals, such as 19.99 or -19.99
* char - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
* string - stores text, such as "Hello World". String values are surrounded by double quotes
* bool - stores values with two states: true or false

## Declaring (Creating) Variables

To create a variable, specify the type and assign it a value:

### Syntax

type variableName = value;

Ex;

Int a=10;

## C++ Identifiers

The general rules for naming variables are:

* Names can contain letters, digits and underscores
* Names must begin with a letter or an underscore (\_)
* Names are case sensitive (myVar and myvar are different variables)
* Names cannot contain whitespaces or special characters like !, #, %, etc.
* Reserved words (like C++ keywords, such as int) cannot be used as names

CONSTANT:

When you do not want others (or yourself) to change existing variable values, use the const keyword (this will declare the variable as "constant", which means **unchangeable**

### Example

**const** int myNum = 15;  // myNum will always be 15  
myNum = 10;

## C++ User Input

cin is a predefined variable that reads data from the keyboard with the extraction operator (>>).

In the following example, the user can input a number, which is stored in the variable x. Then we print the value of x:

### Example

int x;   
cout << "Type a number: "; // Type a number and press enter  
cin >> x; // Get user input from the keyboard  
cout << "Your number is: " << x; // Display the input value

### Example

int x, y;  
int sum;  
cout << "Type a number: ";  
cin >> x;  
cout << "Type another number: ";  
cin >> y;  
sum = x + y;  
cout << "Sum is: " << sum;

## C++ Data Types

As explained in the [Variables](https://www.w3schools.com/cpp/cpp_variables.asp) chapter, a variable in C++ must be a specified data type:

### Example

int myNum = 5;               // Integer (whole number)  
float myFloatNum = 5.99;     // Floating point number  
double myDoubleNum = 9.98;   // Floating point number  
char myLetter = 'D';         // Character  
bool myBoolean = true;       // Boolean  
string myText = "Hello";     // String

# Control Structures in C++

More

## Control Statements, Looping :

* Conditional structure: if and else
* For Loop
* While Loop
* Do While
* Goto, Break and Continue
* Switch Statement and Break

## Conditional structure: if and else

* The if statement executes based test expression inside the braces.
* If statement expression is to true, If body statements are executed and Else body statements are skipped.
* If statement expression is to false If body statements are skipped and Else body statements are executed.
* Simply, Block will execute based on If the condition is true or not.
* IF conditional statement is a feature of this programming language which performs different computations or actions depending on whether a programmer-specified boolean condition evaluates to true or false. Apart from the case of branch prediction, this is always achieved by selectively altering the control flow based on some condition.

### if …. else

### Syntax

if (expression )

{

//If Body statements

}

else

{

//Else Body statements

}

### if …. else

### Syntax

### Example

if (i == 3)

{

St1;

}

else

{

St 2

}

### Example Program

/\* Example Program For If..else In C++ Programming Language \*/

// Header Files

#include<iostream>

#include<conio.h>

//Main Function

void main()

{

// Variable Declaration

int a;

//Get Input Value

cout<<"Enter the Number :";

cin>>a;

//If Condition Check

if(a > 10)

{

// Block For Condition Success

cout<<a <<" Is Greater than 10";

}

else

{

// Block For Condition Fail

cout<<a<<" Is Less than/Equal to 10";

}

// Wait For Output Screen

getch();

//Main Function return Statement

}

### Sample Output:

Enter the Number :8

8 Is Less than/Equal to 10

Enter the Number :10

10 Is Less than/Equal to 10

## Looping

This chapter will look at C's mechanisms for controlling looping and iteration. Even though some of these mechanisms may look familiar and indeed will operate in a standard fashion most of the time.

## The for statement

The C++ for statement has the following form:

### Syntax:

for  (expression1;Condition;expression2)

statement;

for  (expression1;Condition;expression2) {

block of statements

}

expression1 initialises; expression2 is the terminate test; expression3 is the modifier (which may be more than just simple increment);

NOTE: C/C++ basically treats for statements as while type loops

### For loop example program:

/\* Example Program For for Loop In C++ Programming Language \*/

// Header Files

#include<iostream.h>

#include<conio.h>

//Main Function

int main()

{

// Variable Declaration

int x=3;

//for loop

for (x=3;x>0;x--)

{

cout<<"x="<<x<<endl;

}

// Wait For Output Screen

getch();

//Main Function return Statement

return 0;

}

### Output:

x=3

x=2

x=1

## The while statement

The while statement is similar to those used in other The while has the form:

### Syntax:

while (expression)

statement;

while (expression)

{

block of statements

}

### While statement example program

/\* Example Program For while Loop In C++ Programming Language \*/

// Header Files

#include<iostream>

#include<conio.h>

//Main Function

int main()

{

// Variable Declaration

int x=3;

//while loop

while (x>0)

{

cout<<"x="<<x<<endl;

x--;

}

// Wait For Output Screen

getch();

//Main Function return Statement

return 0;

}

### Output:

x=3

x=2

x=1

## The do-while statement

### Syntax:

do

{

statement;

} while (expression);

### do while Loop example:

/\* Example Program For Do While Loop In C++ Programming Language \*/

// Header Files

#include<iostream>

#include<conio.h>

//Main Function

int main()

{

// Variable Declaration

int x=3;

//do while loop

do {

cout<<"x="<<x<<endl;

x--;

}while (x>0);

// Wait For Output Screen

getch();

//Main Function return Statement

return 0;

}

### outputs:

x=3

x=2

x=1

## C/C++ provides two commands to control how we loop: break and continue

break -- exit form loop or switch.

continue -- skip 1 iteration of loop.

Consider the following example where we read in integer values and process them according to the following conditions. If the value we have read is negative, we wish to print an error message and abandon the loop. If the value read is great than 100, we wish to ignore it and continue to the next value in the data. If the value is zero, we wish to terminate the loop.

## Switch Case Statement

In C/C++ programming language, the switch statement is a type of selection mechanism used to allow block code among many alternatives.Simply, It changes the control flow of program execution via multiple blocks.

### Switch Statement Rules

* A switch works with the char and int data types.
* It also works with enum types
* Switch expression/variable datatype and case datatype are should be matched.
* A switch block has many numbers of case statements, Each case ends with a colon.
* Each case ends with a break statement. Else all case blocks will execute until a break statement is reached.
* The switch exists When a break statement is reached,
* A switch block has only one number of default case statements, It should end of the switch.
* The default case block executed when none of the cases is true.
* No break is needed in the default case.

### Switch Statement Usage

* We can use switch statements alternative for an if..else ladder.
* The switch statement is often faster than nested if...else Ladder.
* Switch statement syntax is well structured and easy to understand.

### Switch Statement Syntax:

switch ( <expression>

{

case value1:

  //Block 1 Code Here

  break;

case value2:

  //Block 1 Code Here

  break;

...

default:

  Code to execute for not match case

  break;

}

### Example Program For Switch

/\*  Example Program For Switch Case In C++ Programming Language \*/

// Header Files

#include<iostream>

#include<conio.h>

//Main Function

int main() {

// Variable Declaration

char ch;

//Get Input Value

cout << "Enter the Vowel (In Capital Letter):";

cin>>ch;

//Switch Case Check

switch (ch) {

case 'A': cout << "Your Character Is A\n";

break;

case 'E': cout << "Your Character Is E\n";

break;

case 'I': cout << "Your Character Is I\n";

break;

case 'O': cout << "Your Character Is O\n";

break;

case 'U': cout << "Your Character Is U\n";

break;

default: cout << "Your Character is Not Vowel.Otherwise Not a Capital Letter\n";

break;

}

// Wait For Output Screen

getch();

//Main Function return Statement

return 0;

}

### Sample Output:

Enter the Vowel (In Capital Letter):A

Your Character Is A

Enter the Vowel (In Capital Letter):O

Your Character Is O

Enter the Vowel (In Capital Letter):h

Your Character is Not Vowel.Or Not a Capital Letter

## The goto statement

goto allows making an absolute jump to another point in the program. You should use this feature with caution since its execution causes an unconditional jump ignoring any type of nesting limitations.  
The destination point is identified by a label, which is then used as an argument for the goto statement. A label is made of a valid identifier followed by a colon (:).  
Generally speaking, this instruction has no concrete use in structured or object oriented programming aside from those that low-level programming fans may find for it. For example, here is our countdown loop using goto:

// goto loop example

/\*  Example Program For goto In C++ Programming Language \*/

// Header Files

#include<iostream>

#include<conio.h>

using namespace std;

//Main Function

int main() {

// Variable Declaration

int num = 10;

//goto statement declaration

loop: