***C++ Program:***

1. Basic program:

// Online C++ compiler to run C++ program online

#include <iostream>

using namespace std;

int main() {

// Write C++ code here

std::cout << "Hello world";

return 0;

}

Variable:

Local:

2.

// CPP program to illustrate

// usage of local variables

#include<iostream>

using namespace std;

void func()

{

// this variable is local to the

// function func() and cannot be

// accessed outside this function

int age=18;

cout<<age;

}

int main()

{

cout<<"Age is: ";

func();

return 0;

}

Global:

3. // CPP program to illustrate

// usage of global variables

#include<iostream>

using namespace std;

// global variable

int global = 5;

// global variable accessed from

// within a function

void display()

{

cout<<global<<endl;

}

// main function

int main()

{

display();

// changing value of global

// variable from main function

global = 10;

display();

}

4.

// 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;

cout << "size of long double:"<< sizeof(long double) << endl;

return 0;

}

Function:

1.

#include <iostream>

using namespace std;

// declaring a function

void greet() {

cout << "Hello there!";

}

int main() {

// calling the function

greet();

return 0;

}

2.

// program to print a text

#include <iostream>

using namespace std;

// display a number

void displayNum(int n1, float n2) {

cout << "The int number is " << n1 << endl;

cout << "The double number is " << n2;

}

int main() {

int num1 = 5;

double num2 = 5.5;

// calling the function

displayNum(num1, num2);

return 0;

}

Functions:

#include <iostream>

using namespace std;

// declaring a function

void name() {

cout << "Hello there!";

}

int main() {

// calling the function

name();

return 0;

}

// program to print a text

2.

#include <iostream>

using namespace std;

// display a number

void displayNum(int n1, float n2) {

cout << "The int number is " << n1;

cout << "The double number is " << n2;

}

int main() {

int num1 = 5;

double num2 = 5.5;

// calling the function

displayNum(num1, num2);

return 0;

}

Function return type:

// program to add two numbers using a function

#include <iostream>

using namespace std;

// declaring a function

int add(int a, int b) {

return (a + b);

}

int main() {

int sum;

// calling the function and storing

// the returned value in sum

sum = add(100, 78);

cout << "100 + 78 = " << sum << endl;

return 0;

}

Constant:

// C++ Program to illustrate the use of #define to create a

// constant

#include <iostream>

using namespace std;

// using #define to create a macro

#define Side 5

int main()

{

// using constant

const double area = Side \* Side;

cout << "The area of square with side 5 is ";

cout << area;

return 0;

}

2.

// C++ Program to illustrate the use of #define to create a

// constant

#include <iostream>

using namespace std;

// using #define to create a macro

#define Side 5

int main()

{

// using constant

const double area = Side \* Side;

int a=21;

area=area+a;

cout << "The area of square with side 5 is ";

cout << area << endl;

cout << "a=" << a << endl;

cout << area << endl;

return 0;

}

Arithmetic operator:

These operators operate or work with two operands. For example: Addition(+), Subtraction(-), etc.

1.

// Online C++ compiler to run C++ program online

#include <iostream>

using namespace std;

int main()

{

int a, b;

a = 10;

b = 5;

cout << "a + b = " << (a + b) << endl ;

cout << "a - b = " << (a - b) << endl ;

cout << "a \* b = " << (a \* b) << endl ;

cout << "a / b = " << (a / b) << endl ;

}

2.

// CPP Program to demonstrate the Arithmetic Operators

#include <iostream>

using namespace std;

int main()

{

int a = 8, b = 3;

// Addition operator

cout << "a + b = " << (a + b) << endl;

// Subtraction operator

cout << "a - b = " << (a - b) << endl;

// Multiplication operator

cout << "a \* b = " << (a \* b) << endl;

// Division operator

cout << "a / b = " << (a / b) << endl;

// Modulo operator

cout << "a % b = " << (a % b) << endl;

return 0;

}

2. Relational Operators:

These operators are used for the comparison of the values of two operands. For example, ‘>’ checks if one operand is greater than the other operand or not, etc. The result returns a Boolean value, i.e., true or false.

// CPP Program to demonstrate the Relational Operators

#include <iostream>

using namespace std;

int main()

{

int a = 6, b = 4;

// Equal to operator

cout << "a == b is " << (a == b) << endl;

// Greater than operator

cout << "a > b is " << (a > b) << endl;

// Greater than or Equal to operator

cout << "a >= b is " << (a >= b) << endl;

// Lesser than operator

cout << "a < b is " << (a < b) << endl;

// Lesser than or Equal to operator

cout << "a <= b is " << (a <= b) << endl;

// true

cout << "a != b is " << (a != b) << endl;

return 0;

}

3. Logical Operators:

These operators are used to combine two or more conditions or constraints or to complement the evaluation of the original condition in consideration. The result returns a Boolean value, i.e., true or false.

// CPP Program to demonstrate the Logical Operators

#include <iostream>

using namespace std;

int main()

{

int a = 6, b = 4;

// Logical AND operator

cout << "a && b is " << (a && b) << endl;

// Logical OR operator

cout << "a ! b is " << (a > b) << endl;

// Logical NOT operator

cout << "!b is " << (!b) << endl;

return 0;

}

4. Bitwise Operators:

These operators are used to perform bit-level operations on the operands. The operators are first converted to bit-level and then the calculation is performed on the operands. Mathematical operations such as addition, subtraction, multiplication, etc. can be performed at the bit level for faster processing.

// CPP Program to demonstrate the Bitwise Operators

#include <iostream>

using namespace std;

int main()

{

int a = 6, b = 4;

// Binary AND operator

cout << "a & b is " << (a & b) << endl;

// Binary OR operator

cout << "a | b is " << (a | b) << endl;

// Binary XOR operator

cout << "a ^ b is " << (a ^ b) << endl;

// Left Shift operator

cout << "a<<1 is " << (a << 1) << endl;

// Right Shift operator

cout << "a>>1 is " << (a >> 1) << endl;

// One’s Complement operator

cout << "~(a) is " << ~(a) << endl;

return 0;

}

5. Assignment Operators

These operators are used to assign value to a variable. The left side operand of the assignment operator is a variable and the right side operand of the assignment operator is a value. The value on the right side must be of the same data type as the variable on the left side otherwise the compiler will raise an error.

#include <iostream>

using namespace std;

int main()

{

int a = 10, b = 2;

// Assignment Operator

cout << "a = " << a << endl;

// Add and Assignment Operator

cout << "a += b is " << (a += b) << endl;

// Subtract and Assignment Operator\\

cout << "a -= b is " << (a -= b) << endl;

// Multiply and Assignment Operator

cout << "a \*= b is " << (a \*= b) << endl;

// Divide and Assignment Operator

cout << "a /= b is " << (a /= b) << endl;

return 0;

}

Namespace:

What does :: represent in C++?

Two colons (::) are used in C++ as a scope resolution operator.

//< C++ compiler to run C++ program online

#include <iostream>

using namespace std;

namespace dog{

void eat(){

cout << "dog eat rice in the table" << endl;

}

}

namespace cow{

void color(){

cout << "cow is yellow" << endl;

}

}

int main() {

dog::eat();

cow::color();

return 0;

}

post increment operator (++) pre increment operator (++)

// Online C++ compiler to run C++ program online

#include <iostream>

using namespace std;

int main()

{

int val;

val=5;

val=val+1; //val++ incre decer val--

cout << val<< endl;

return 0;

}

post increment operator++ pre increment operator++

// Online C++ compiler to run C++ program online

#include <iostream>

using namespace std;

int main()

{

int val;

val=5;

cout << val++<< endl; //post increment operator

cout << ++val<< endl; //pre increment operator

return 0;

}

post increment operator (++)

// Online C++ compiler to run C++ program online

#include <iostream>

using namespace std;

int main()

{

int val;

val=5;

cout << val++<< endl; //post increment operator

cout <<val<<endl;

return 0;

}

(++) pre increment operator

// Online C++ compiler to run C++ program online

#include <iostream>

using namespace std;

int main()

{

int val;

val=5;

cout << ++val<< endl; //pre increment operator

cout <<val<<endl;

return 0;

}

pre increment operator(--)

// Online C++ compiler to run C++ program online

#include <iostream>

using namespace std;

int main()

{

int val;

val=5;

cout << --val<< endl; //pre increment operator

cout <<val<<endl;

return 0;

}

Logical operator (and)

// C++ Program to illustrate the logical AND Operator

#include <iostream>

using namespace std;

int main()

{

// initialize variables

int age = 25;

bool isStudent = true;

// Using AND operator in if condition

if (age > 18 && isStudent) {

cout << "You are eligible for a student discount."<< endl;

} else {

cout << "You are not eligible for a student discount."<<endl;

return 0;

}

}

or

// C++ program to demonstrate the logical or operator

#include <iostream>

using namespace std;

int main()

{

int num = 7;

// using logical or for conditional statement

if (num <= 0 || num >= 10) {

cout << "The number is outside the range of 0 to 10." << endl;

}

else {

cout << "The number is between 0 to 10." << endl;

}

return 0;

}

if:

// Program to print positive number entered by the user

// If the user enters a negative number, it is skipped

#include <iostream>

using namespace std;

int main() {

int number;

cout << "Enter an integer: ";

cin >> number;

// checks if the number is positive

if (number > 0) {

cout << "You entered a positive integer: " << number << endl;

}

cout << "This statement is always executed.";

return 0;

}

if else:

// Program to check whether an integer is positive or negative

// This program considers 0 as a positive number

#include <iostream>

using namespace std;

int main() {

int number;

cout << "Enter an integer: ";

cin >> number;

if (number >= 0) {

cout << "You entered a positive integer: " << number << endl;

}

else {

cout << "You entered a negative integer: " << number << endl;

}

cout << "This line is always printed.";

return 0;

}

if.. else .. elseif:

// Program to check whether an integer is positive, negative or zero

#include <iostream>

using namespace std;

int main() {

int number;

cout << "Enter an integer: ";

cin >> number;

if (number > 0) {

cout << "You entered a positive integer: " << number << endl;

}

else if (number < 0) {

cout << "You entered a negative integer: " << number << endl;

}

else {

cout << "You entered 0." << endl;

}

cout << "This line is always printed.";

return 0;

}

Nested if:

// C++ program to find if an integer is positive, negative or zero

// using nested if statements

#include <iostream>

using namespace std;

int main() {

int num;

cout << "Enter an integer: ";

cin >> num;

// outer if condition

if (num != 0) {

// inner if condition

if (num > 0) {

cout << "The number is positive." << endl;

}

// inner else condition

else {

cout << "The number is negative." << endl;

}

}

// outer else condition

else {

cout << "The number is 0 and it is neither positive nor negative." << endl;

}

cout << "This line is always printed." << endl;

return 0;

}

Switch case

// Program to build a simple calculator using switch Statement

#include <iostream>

using namespace std;

int main() {

char oper;

float num1, num2;

cout << "Enter an operator (+, -, \*, /): ";

cin >> oper;

cout << "Enter two numbers: " << endl;

cin >> num1 >> num2;

switch (oper) {

case '+':

cout << num1 << " + " << num2 << " = " << num1 + num2;

break;

case '-':

cout << num1 << " - " << num2 << " = " << num1 - num2;

break;

case '\*':

cout << num1 << " \* " << num2 << " = " << num1 \* num2;

break;

case '/':

cout << num1 << " / " << num2 << " = " << num1 / num2;

break;

default:

// operator is doesn't match any case constant (+, -, \*, /)

cout << "Error! The operator is not correct";

break;

}

return 0;

}

2.

#include <iostream>

using namespace std;

// Driver Code

int main()

{

// switch variable

char x = 'A';

// switch statements

switch (x) {

case 'A':

cout << "Choice is A";

break;

case 'B':

cout << "Choice is B";

break;

case 'C':

cout << "Choice is C";

break;

default:

cout << "Choice other than A, B and C";

break;

}

return 0;

}

3.

// C++ program to demonstrate the placement of default

// anywhere

#include <iostream>

using namespace std;

int main()

{

int day;

cout << "Enter a day number (1-7): ";

cin >> day;

switch (day) {

default: // Default placed first for demonstration

cout << "Not a valid weekday." << endl;

break;

case 1:

cout << "It's Monday!" << endl;

break;

case 2:

cout << "It's Tuesday!" << endl;

break;

case 3:

cout << "It's Wednesday!" << endl;

break;

case 4:

cout << "It's Thursday!" << endl;

break;

case 5:

cout << "It's Friday!" << endl;

break;

case 6:

cout << "It's Saturday!" << endl;

break;

case 7:

cout << "It's Sunday!" << endl;

break;

}

return 0;

}

While loop:

// C++ program to illustrate while loop

#include <iostream>

using namespace std;

int main()

{

// initialization expression

int i = 1;

// test expression

while (i < 6) {

cout << "Hello World\n";

// update expression

i++;

}

return 0;

}

2.

// C++ program to illustrate while loop

#include <iostream>

using namespace std;

int main()

{

// initialization expression

int i = 1;

// test expression

while (i > -5) {

cout << i << "\n";

// update expression

i--;

}

return 0;

}

Do… While:

// C++ program to illustrate do-while loop

#include <iostream>

using namespace std;

int main()

{

// Initialization expression

int i = 1;

do {

// Loop body

cout << i << endl;

// Update expression

i++;

}

// Test expression

while (i <= 5);

return 0;

}

2.

#include <iostream>

using namespace std;

int main() {

int i = 1;

do{

cout<<i<<"\n";

i++;

} while (i <= 10) ;

}

(Nested if) Do.. while:

#include <iostream>

using namespace std;

int main() {

int i = 1;

do{

int j = 1;

do{

cout<<i<<"\n";

j++;

} while (j <= 3) ;

i++;

} while (i <= 3) ;

}

For:

// C++ program to illustrate for loop to print numbers from

// 1 to n

#include <iostream>

using namespace std;

int main()

{

// initializing n (value upto which you want to print

// numbers

int n = 5;

int i; // initialization of loop variable

for (i = 1; i <= n; i++) {

cout << i << " "; }

return 0;

}

2.

// C++ program to illustrate for loop to print numbers from

// n to 1 (reverse counting).

#include <iostream>

using namespace std;

int main()

{

// initializing n (value upto which you want to print

// numbers

int n = 5;

int i; // initialization of loop variable

for (i = n; i >= 1; i--) {

cout << i << " ";

}

return 0;

}

Break:

// program

#include <iostream>

using namespace std;

int main() {

for (int i = 1; i <= 5; i++) {

// break condition

if (i == 3) {

break;

}

cout << i << endl;

}

return 0;

}

2.

// using break statement inside

// nested for loop

#include <iostream>

using namespace std;

int main() {

int number;

int sum = 0;

// nested for loops

// first loop

for (int i = 1; i <= 3; i++) {

// second loop

for (int j = 1; j <= 3; j++) {

if (i == 2) {

break;

}

cout << "i = " << i << ", j = " <\< j << endl;

}

}

return 0;

}

Continue:

// C++ program to explain the use

// of continue statement with for loop

#include <iostream>

using namespace std;

int main()

{

// loop from 1 to 10

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

// If i is equals to 4,

// continue to next iteration

// without printing

if (i == 4)

continue;

else

// otherwise print the value of

cout << i << " ";

}

return 0;

}

(While continue:)

// C++ program to explain the use

// of continue statement with while loop

#include <iostream>

using namespace std;

int main()

{

int i = 0;

// loop from 1 to 10

while (i < 10) {

i++;

// If i is equals to 4,

// continue to next iteration

// without printing

if (i == 4) {

continue;

}

else {

// otherwise print the value of

cout << i << " ";

}

}

return 0;

}

For continue:

// C++ program to explain the use

// of continue statement with nested loops

#include <iostream>

using namespace std;

int main()

{

for (int i = 1; i <= 2; i++) {

for (int j = 0; j <= 4; j++) {

if (j == 2) {

continue;

}

cout << j << " ";

}

cout << endl;

}

return 0;

}

Goto:

#include <iostream>

using namespace std;

int main () {

// Local variable declaration:

int a = 10;

// do loop execution

LOOP:do {

if( a == 15) {

// skip the iteration.

a = a + 1;

goto LOOP;

}

cout << "value of a: " << a << endl;

a = a + 1;

}

while( a < 20 );

return 0;

}

2.

#include <iostream>

using namespace std;

int main(){

int num; cout<<"Enter a number: "; cin>>num;

if (num % 2==0){

goto print;

}

else {

cout<<"Odd Number";

}

print:

cout<<"Even Number";

return 0;

}

Array:

1.#include <iostream>

using namespace std;

int main() {

int numbers[5] = {7, 5, 6, 12, 35};

cout << "\n The numbers are: ";

// Printing array elements

// using traditional for loop

for (int i = 0; i < 5; ++i) {

cout << numbers[i] << " ";

}

return 0;

}

2. #include <iostream>

using namespace std;

int main() {

int numbers[5];

cout << "Enter 5 numbers: " << endl;

// store input from user to array

for (int i = 0; i < 5; ++i) {

cin >> numbers[i];

}

cout << "The numbers are: ";

// print array elements

for (int n = 0; n < 5; ++n) {

cout << numbers[n] << " ";

}

return 0;

}

3. // C++ Program to Illustrate How to Access Array Elements

#include <iostream>

using namespace std;

int main()

{

int arr[3];

// Inserting elements in an array

arr[0] = 10;

arr[1] = 20;

arr[2] = 30;

// Accessing and printing elements of the array

cout << "arr[0]: " << arr[0] << endl;

cout << "arr[1]: " << arr[1] << endl;

cout << "arr[2]: " << arr[2] << endl;

return 0;

}

String:

1.

// C++ Program to demonstrate use of string keyword

#include <iostream>

using namespace std;

int main()

{

string s = "Any one";

string str("CPP program");

cout << "s = " << s << endl;

cout << "str = " << str << endl;

return 0;

}

2.

// C++ Program to demonstrate C-style string declaration

#include <iostream>

using namespace std;

int main()

{

char s1[] = { 'g', 'f', 'g', '\0' };

char s2[4] = { 'g', 'f', 'g', '\0' };

char s3[4] = "gfg";

char s4[] = "gfg";

cout << "s1 = " << s1 << endl;

cout << "s2 = " << s2 << endl;

cout << "s3 = " << s3 << endl;

cout << "s4 = " << s4 << endl;

return 0;

}

3.

#include <iostream>

#include<string.h>

using namespace std;

int main()

{

string name; // variable declaration.

std::cout << "Enter your name :" << std::endl;

getline(cin,name); // implementing a getline() function

cout<<"\n Hello "<<name;

return 0;

}

2.

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char name1[10] = "Guru99";

char name2[10] = "John";

char name3[10];

int len;

strcpy(name3, name1);

cout << "strcpy( name3, name1) : " << name3 << endl;

strcat(name1, name2);

cout << "strcat( name1, name2): " << name1 << endl;

len = strlen(name1);

cout << "strlen(name1) : " << len << endl;

return 0;

}

Strlen():

#include <iostream>

#include <cstring>

using namespace std;

int main() {

// initialize C-string

char song[] = "We Will Rock You!";

// print the length of the song string

cout << strlen(song);

return 0;

}

pointer variable:

1.

#include <iostream>

using namespace std;

int main() {

int var = 5;

// store address of var

int\* point\_var = &var;

// print value of var

cout << "var = " << var << endl;

// print address of var

cout << "Address of var (&var) = " << &var << endl << endl;

// print pointer point\_var

cout << "point\_var = " << point\_var << endl;

// print the content of the address point\_var points to

cout << "Content of the address pointed to by point\_var (\*point\_var) = " << \*point\_var << endl;

return 0;

}

2.

// C++ program to illustrate Array Name as Pointers

#include <bits/stdc++.h>

using namespace std;

void func()

{

// Declare an array

int val[3] = { 5, 10, 20 };

// declare pointer variable

int\* ptr;

// Assign the address of val[0] to ptr

// We can use ptr=&val[0];(both are same)

ptr = val;

cout << "Elements of the array are: ";

cout << ptr[0] << endl << ptr[1] << endl << ptr[2];

}

// Driver program

int main() { func(); }

Conditional operator:

#include <iostream>

#include <string>

using namespace std;

int main() {

double marks;

// take input from users

cout << "Enter your marks: ";

cin >> marks;

// ternary operator checks if

// marks is greater than 40

string result = (marks >= 40) ? "passed" : "failed";

cout << "You " << result << " the exam.";

return 0;

}

2.

#include <iostream>

#include <string>

using namespace std;

int main() {

int number = 0;

string result;

// nested ternary operator to find whether

// number is positive, negative, or zero

result = (number == 0) ? "Zero" : ((number > 0) ? "Positive" : "Negative");

cout << "Number is " << result;

return 0;

}

New dynamic

// C++ program to demonstrate how to create dynamic variable

// using new

#include <iostream>

#include <memory>

using namespace std;

int main()

{

// pointer to store the address returned by the new

int\* ptr;

// allocating memory for integer

ptr = new int;

// assigning value using dereference operator

\*ptr = 10;

// printing value and address

cout << "Address: " << ptr << endl;

cout << "Value: " << \*ptr;

return 0;

}

2.

// C++ program to illustrate how to initialize a dynamic

// variable with allocation

#include <iostream>

#include <memory>

using namespace std;

// Custom data type with constructor to take initial value

struct cust {

int p;

cust(int q)

: p(q)

{

}

cust() = default;

};

int main()

{

// creating inbuit data types with initial value

int\* p = new int(25);

float\* q = new float(75.25);

// Works fine, doesn’t require constructor

cust\* var1 = new cust;

// OR

// Works fine, doesn’t require constructor

var1 = new cust();

// Notice error if you comment this line

cust\* var = new cust(25);

cout << \*p << " " << \*q << " " << var->p;

return 0;

}

Delete dynamic:

// C++ program to illustrate dynamic allocation

// and deallocation of memory using new and delete

#include <iostream>

using namespace std;

int main()

{

// Pointer initialization to null

int\* p = NULL;

// Request memory for the variable

// using new operator

p = new (nothrow) int;

if (!p)

cout << "allocation of memory failed\n";

else {

// Store value at allocated address

\*p = 29;

cout << "Value of p: " << \*p << endl;

}

// Request block of memory

// using new operator

float\* r = new float(75.25);

cout << "Value of r: " << \*r << endl;

// Request block of memory of size n

int n = 5;

int\* q = new (nothrow) int[n];

if (!q)

cout << "allocation of memory failed\n";

else {

for (int i = 0; i <= n; i++)

q[i] = i + 1;

cout << "Value store in block of memory: ";

for (int i = 0; i <n; i++)

cout << q[i] << " ";

}

// freed the allocated memory

delete p;

delete r;

// freed the block of allocated memory

delete[] q;

return 0;

}

Reading file:

// C++ program to read from a file

#include <fstream>

#include <iostream>

#include <string>

using namespace std;

int main()

{

// Open the input file named "input.txt"

ifstream inputFile("input.txt");

// Check if the file is successfully opened

if (!inputFile.is\_open()) {

cerr << "Error opening the file!" << endl;

return 1;

}

string line; // Declare a string variable to store each

// line of the file

// Read each line of the file and print it to the

// standard output stream

cout << "File Content: " << endl;

while (getline(inputFile, line)) {

cout << line << endl; // Print the current line

}

// Close the file

inputFile.close();

return 0;

}

Header files: (#include <cmath>)

#include<iostream>

#include <cmath>

using namespace std;

int main()

{

int num1 = 5, num2 = 10;

cout << "Minimum of the two numbers is: ";

cout << min(num1, num2) << "\n";

cout << "Maximum of the two numbers is: ";

cout << max(num1, num2) << "\n";

return 0;

}

Multiple file compilation:

firstFile.h

// firstFile.h

// C++ Program to implement

// using gcc commands

int add(int a, int b) { return a + b; }

secondFile.cpp

// secondFile.cpp

// C++ Program to implement

// Using gcc commands

#include <iostream>

#include "firstFile.h"

using namespace std;

// just declaring the function

int add(int a, int b);

int main()

{

// In this there is no add function definition. add

// function definition is in another file.

cout << add(14, 16) << endl;

cout << add(2, 3) << endl;

return 0;

}