

CPP

C++ Programming Basics

Example (01): Hello world

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

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

Example (02): Read x and y as integer then print result of

$x+y$, $x-y$, $x*y$, x/y , $x\%y$

```
#include <iostream>
using namespace std;
int main()
{
    int x,y;
    cout << "x= ";
    cin >> x;
    cout << "y= ";
    cin >> y;

    cout << x << " + " << y << " = " << x+y << endl;
    cout << x << " - " << y << " = " << x-y << endl;
    cout << x << " * " << y << " = " << x*y << endl;
    cout << x << " / " << y << " = " << x/y << endl;
    cout << x << " \% " << y << " = " << x%y << endl;
    return 0;
}
```

Example (03): Find result of: $f(x) = \sqrt{x - 4^x}$

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    double x,result;
    cout << "x= ";
    cin >> x;
    result = sqrt(x- pow(4,x));
    cout << "sqrt(x- pow(4,x))= " << result << endl;
    return 0;
}
```

Example (04): swap x and y using third variable

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

int main()
{
    double x,y, temp;
    cout << "x= ";
    cin >> x;
    cout << "y= ";
    cin >> y;
    temp = x;
    x = y;
    y = temp;
    cout << "-----\n";
    cout << "x = " << x << "ny = " << y << endl;
    return 0;
}
```

Example (05): swap x and y without using third variable

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

int main()
{
    double x,y;
    cout << "x= ";
    cin >> x;
    cout << "y= ";
    cin >> y;
    x = x + y;
    y = x - y;
    x = x - y;
    cout << "-----\n";
    cout << "x = " << x << "ny = " << y << endl;
    return 0;
}
```

Exercises

1. Write cpp program to convert INCH to CM where: 1 INCH = 2.54 CM.
2. Write cpp program to find perimeter (p) and area of triangle x y z Where:

$$p = x + y + z$$

$$\text{area} = \sqrt{p(p - x)(p - y)(p - z)}$$

Loops and Decisions

+ Example (01): Find maximum value of two variables

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

int main()
{
    cout << "Insert two numbers to find the maximum one\n";
    cout << "-----\n";

    double FNum, SNum;
    cout << "First Number= ";
    cin >> FNum;
    cout << "Second Number= ";
    cin >> SNum;
    cout << "-----\n";

    if (FNum > SNum)
        cout << "First Number= " << FNum << " Is the maximum Number\n";
    else if(FNum < SNum)
        cout << "Second Number= " << SNum << " Is the maximum Number\n";
    else
        cout << "First Number = Second Number";

    return 0;
}
```

+ Example (02): Find maximum value of three variables.

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

int main()
{
    double x, y, z;
    cin >> x >> y >> z;
    cout << "-----\n";

    if (x>=y && x>=z)
        cout << x;
    else if (y>=x && y>=z)
        cout << y;
    else
        cout << z;
    return 0;
}
```

Example (03): Write program that find product of n:

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

int main()
{
    int n, pro=1;
    cout << "n = ";
    cin >> n;
    for (int i=2; i<=n; i++)
        pro *=i;
    cout << pro;
    return 0;
}
```

Example (04): Write program that count summation of n numbers:

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

int main()
{
    double n, sum=0, x;
    cout << "Insert Number of the Numbers = ";
    cin >> n;
    for (int i=1; i<n; i++)
    {
        cout << "number " << i << " = ";
        cin >> x;
        sum +=x;
    }
    cout << "\n-----\n" << sum;
    return 0;
}
```

Example (05): Write program that read n of numbers then count and print:

Number of positive numbers,

The summation of positive numbers,

Number of negative numbers,

The summation of negative numbers,

Number of numbers that equal zero.

as the following:

```
Insert n of numbers to insert it:5
-----
Number1 = -2
Number2 = 0
Number3 = 0
Number4 = 2
Number5 = 1
-----
Number of Positive = 2
Summasion of Positive = 3
Number of zeros = 2
Number of negative = 1
Summasion of negative = -2
```

•••

```

#include <iostream>
using namespace std;

int main()
{
    int n;                      //number of numbers you want to insert
    double x;                   //the number you will insert
    int np =0;                  //Number of Positive (np)
    double sp =0;                //Summasion of Positive (sp)
    int n0 = 0;                  //Number of zero (n0)
    int nn = 0;                  //Number of negative (nn)
    double sn = 0;                //Summasion of negative (sn)

    cout << "Insert n of numbers to insert it:";
    cin >> n;

    cout << "-----\n";

    for (int i=0; i<n; i++)
    {
        cout << "Number " << i << " = ";
        cin >> x;

        if (x > 0)
        {
            np = np + 1;
            sp = sp + x;
        }

        else if (x < 0)
        {
            nn = nn + 1;
            sn = sn + x;
        }
        else
            n0 = n0 + 1;
    }

    cout << "-----\n";

    cout << "Number of Positive = " << np << "\n";
    cout << "Summasion of Positive = " << sp << "\n";
    cout << "Number of zeros = " << n0 << "\n";
    cout << "Number of negative = " << nn << "\n";
    cout << "Summasion of negative = " << sn << "\n";
    return 0;
}

```

Example (06): Test if number is prim:

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

int main()
{
    int i,x;
    cout << "Insert integer number: ";
    cin >> x;
    for (i=2; i<x/2; i++)
    {
        if (x%i ==0)
            break;
    }
    cout << "-----\n";
    if (x%i ==0)
        cout << x << " is not prim number";
    else
        cout << x << " is prim number";
    return 0;
}
```

OR

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

int main()
{
    int i,x;
    cout << "Insert integer number: ";
    cin >> x;
    for (i=2; i<x/i; i++)
    {
        if (x%i ==0)
            break;
    }
    cout << "-----\n";
    if (x%i ==0)
        cout << x << " is not prim number";
    else
        cout << x << " is prim number";
    return 0;
}
```

Example (07): Write program that prints prime numbers from 1 to 100;

```
#include <iostream>
using namespace std;
int main ()
{
    int i, j;
    for(i=2; i<100; i++)
    {
        for(j=2; j <= (i/j); j++)
            if(!(i%j)) break; // if factor found, not prime
        if(j > (i/j)) cout << i << " is prime\n";
    }
    return 0;
}
```

Example (08): Demonstrates SWITCH statement.

```
#include <iostream>
using namespace std;
int main ()
{
    int speed;      //turntable speed
    cout << "\nEnter 33, 45, or 78: ";
    cin >> speed; //user enters speed
    switch(speed) //selection based on speed
    {
        case 33:   //user entered 33
            cout << "LP album\n";
            break;
        case 45:   //user entered 45
            cout << "Single selection\n";
            break;
        case 78:   //user entered 78
            cout << "Obsolete format\n";
            break;
        default:
            cout << "Not Supported";
    }
    return 0;
}
```

Example (08): Demonstrates While with SWITCH statements.

```
#include <iostream>
using namespace std;
#include <process.h> //for exit()
#include <conio.h> //for getche()
int main()
{
    char dir='a';
    int x=10, y=10;
    while( dir != '\r' ) //quit on Enter key
    {
        cout << "\n\nYour location is " << x << ", " << y;
        if( x<5 || x>15 ) //if x west of 5 OR east of 15
            cout << "\nBeware: dragons lurk here";
        cout << "\nEnter direction (n, s, e, w): ";
        dir = getche(); //get direction
        switch(dir)
        {
            case 'n': y--; break; //update coordinates
            case 's': y++; break;
            case 'e': x++; break;
            case 'w': x--; break;
        } //end switch
    } //end while
    return 0;
} //end main()
```

Exercises

- 1) Write program that read "n" to find summation of:

$$1 + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{9} + \dots + \frac{1}{n}$$

- 2) Write program that read "n" to find summation of:

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \frac{1}{8} + \dots + \frac{1}{n}$$

- 3) Write program that read n of numbers then print the multiplication table of n until n*12 as the following.

```
Insert integer number to print the multiplication table of n :6
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
6 * 10 = 60
6 * 11 = 66
6 * 12 = 72
```

- 4) Write program that read n of numbers then prints the mean of numbers.

- 5) Write program that read n of numbers then prints the maximum value.

- 6) Write program that prints numbers divided by 7 from 1 to 100.

- 7) Write program that prints the following output using nested for.

1	2	3
2	4	6
3	6	9

- 8) Write program that read n as integer and find:

$$\sum_{i=1}^n (2i - 1)^7$$

- 9) Write program that read "n" as integer then prints Fibonacci series of "n" as:

0 1 1 2 3 5 8 13 21 34

- 10) Use for loops to construct a program that displays a pyramid of Xs on the screen.

The pyramid should look like this:

X
XXX
XXXXX
XXXXXX
XXXXXXXX

Except that it should be 20 lines high, instead of the 5 lines shown here. One way to do this is to nest two inner loops, one to print spaces and one to print Xs, inside an outer loop that steps down the screen from line to line.

- 11) Modify the product program in this examples so that it repeatedly asks for a number and calculates its product, until the user enters 0, at which point it terminates. You can enclose the relevant statements in product in a while loop or a do loop to achieve this effect.

Arrays

1) Averaging Array Elements.

```
#include <iostream>
using namespace std;
int main()
{
    const int SIZE = 6; //size of array
    int j;
    double sales[SIZE]; //array of 6 variables
    cout << "Enter widget sales for 6 days\n";

    for(j=0; j<SIZE; j++) //put figures in array
        cin >> sales[j];

    double total = 0;
    for(j=0; j<SIZE; j++) //read figures from array
        total += sales[j]; //to find total

    double average = total / SIZE; // find average
    cout << "Average = " << average << endl;
    return 0;
}
```

2) Averaging Array Elements.

```
#include <iostream>
using namespace std;
int main()
{
    const int SIZE = 6; //size of array
    int j;
    double sales[SIZE]; //array of 6 variables
    cout << "Enter widget sales for 6 days\n";

    for(j=0; j<SIZE; j++) //put figures in array
        cin >> sales[j];

    double total = 0;
    for(j=0; j<SIZE; j++) //read figures from array
        total += sales[j]; //to find total

    double average = total / SIZE; // find average
    cout << "Average = " << average << endl;
    return 0;
}
```

•••

3) Sorting array.

```
#include <iostream>
using namespace std;
int main()
{
    char a[] = {'d', 'c', 'f', 'b', 'a', 'e'};
    int n = sizeof(a)/ sizeof(char);
    for (int i=0; i<n; i++)
        for (int j=i+1; j<n; j++)
            if (int(a[j]) < int(a[i]))
            {
                a[j] = a[j] + a[i];
                a[i] = a[j] - a[i];
                a[j] = a[j] - a[i];
            }
    for (int i=0; i<n; i++)
        cout << a[i] << "\t";
    return 0;
}
```

4) Check array sorted or not.

```
#include <iostream>
using namespace std;
int main()
{
    char a[] = {'d', 'c', 'f', 'b', 'a', 'e'};
    int n = sizeof(a)/ sizeof(char);
    for (int i=0; i<n-1; i++)
        if (a[i+1]<a[i])
        {
            cout << "Not sorted array\n";
            break;
        }
    return 0;
}
```

5) Check array is unique elements or not.

```
#include <iostream>
#include <cstdlib> //For exit()
using namespace std;
int main()
{
    char a[] = {'d', 'c', 'f', 'b', 'e', 'e'};
    int n = sizeof(a)/ sizeof(char);
    for (int i = 0; i <n-1; i++)
        for (int j = i+1; j < n; j++)
            if (a[i] == a[j])
            {
                cout << "Is Not unique";
                exit(1);
            }
    return 0;
}
```

6) Three-Way Set Disjointness

Suppose we are given three sets, A , B , and C , with these sets stored in three different integer arrays, a , b , and c , respectively. The *three-way set disjointness* problem is to determine if these three sets are disjoint, that is, whether there is no element x such that $x \in A$, $x \in B$, and $x \in C$.

```
#include <iostream>
#include <cstdlib> //For exit()
using namespace std;

int main()
{
    char a[] = {'d', 'c', 'f', 'b', 'c', 'f'};
    char b[] = {'c', 'h', 'j', 'i', 'f'};
    char c[] = {'m', 'n', 'f'};
    int na = sizeof(a)/ sizeof(char);
    int nb = sizeof(b)/ sizeof(char);
    int nc = sizeof(c)/ sizeof(char);
    for (int i = 0; i < na; i++)
        for (int j = 0; j < nb; j++)
            for (int k = 0; k < nc; k++)
                if ((a[i] == b[j]) && (b[j] == c[k]))
                {
                    cout << "Are not Disjointness";
                    exit(1);
                }
    return 0;
}
```

Exercises

- 1) Write CPP program that read matrix of $n*n$ of double numbers then prints the summation of diameter of the matrix.
- 2) Write CPP program that read matrix of $n*n$ of double numbers then prints the summation of upper triangle of the matrix.
- 3) Write CPP program that read matrix of n char elements then print elements that are duplicated without duplicate at printing.
- 4) Write CPP program that read two matrixes of $n*n$ then print the summation.
- 5) Write CPP program that read two matrixes of $n*n$ then print the multiplication of it.

•••

Functions

1) Demonstrates simple function.

```
// table.cpp
// demonstrates simple function
#include <iostream>
using namespace std;
void starline(); //function declaration

int main()
{
    starline(); //call to function
    return 0;
}
//-----
// starline()
// function definition
void starline() //function declarator
{
    for(int j=0; j<45; j++) //function body
        cout << '*';
    cout << endl;
}
```

2) Demonstrates simple function Passing Constants.

```
// table.cpp
// demonstrates simple function
#include <iostream>
using namespace std;
void starline(char, int); //function declaration

int main()
{
    starline('-', 9); //call to function
    return 0;
}
//-----
// starline()
// function definition
void starline(char c, int n) //function declarator
{
    for(int j=0; j<n; j++) //function body
        cout << '*';
    cout << endl;
}
```

3) Demonstrates simple function Passing Constants and variable.

```
#include <iostream>
using namespace std;
void starline(char, int); //function declaration

int main()
{
    starline('-', 9); //call to function
    return 0;
}
void starline(char c, int n) //function declarator
{
    for(int j=0; j<n; j++) //function body
        cout << c;
    cout << endl;
}
```

4) Demonstrates function return value using factorial example.\

```
#include <iostream>
using namespace std;
int factorial(int);
int main()
{
    cout << factorial(5);
    return 0;
}
int factorial(int n)
{
    int pro = 1;
    for(int i=2; i<=n; i++)
        pro *= i;
    return pro;
}
```

5) Demonstrates Pass by reference swap tow variable.

```
#include <iostream>
using namespace std;
void swap(int&, int&);
int main()
{
    int x=7, y=10;
    cout << "x= " << x << "\t\ty= " << y << "\n";
    swap(x, y);
    cout << "x= " << x << "\t\ty= " << y << "\n";
    return 0;
}
void swap(int& x, int& y)
{
    x = x + y;
    y = x - y;
    x = x - y;
}
```

6) Demonstrates Pass array by reference resetting array elements to zero.

```
#include <iostream>
using namespace std;
const int SIZE = 5;
void resetToZero(int (&a)[SIZE]);int main()
{
    int a[SIZE] = {1,2,3,4,5};
    cout << "Printing array before reset\n";
    for (int i=0; i<SIZE; i++)
        cout << a[i] << "\t";
    resetToZero(a);
    cout << "\nPrinting array after reset\n";
    for (int i=0; i<SIZE; i++)
        cout << a[i] << "\t";
    return 0;
}
void resetToZero(int (&a)[SIZE])
{
    for (int i=0; i<SIZE; i++)
        a[i] = 0;
}
```

7) Demonstrates multifunction and Pass array by reference by reverse array and swap functions.

```
#include <iostream>
using namespace std;
void swap(int &x, int &y);
void reverse (int (&a)[5]);

const int SIZE = 5;
int main()
{
    int x[SIZE] = {1,2,3,4,5};
    reverse (x);
    for (int i=0; i<SIZE; i++)
        cout << x[i] << endl;
    return 0;
}

void reverse (int (&a)[SIZE])
{
    int n = sizeof(a)/ sizeof(int);
    for (int i=0; i<n/2; i++)
        swap(a[i], a[n-i-1]);
}

void swap (int &x, int &y)
{
    x = x+y;
    y = x-y;
    x = x-y;
}
```

8) Demonstrates Default arguments by repeateChar function.

```
#include <iostream>
using namespace std;
void repeateChar(int, char = 'c');

int main()
{
    repeateChar(9);
    return 0;
}

void repeateChar(int n, char c)
{
    for(int j=0; j<n; j++)
        cout << c;
    cout << endl;
}
```

9) Demonstrates overloading function by using abs function to find absolute value for deferent arguments.

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

int abs(int x);
double abs(double x);

int main()
{
    int a = -5;
    double b = -9;
    std::cout << std::abs(a) << abs(b);
    return 0;
}
int abs(int x)
{
    return sqrt(x*x);
}
double abs(double x)
{
    return sqrt(x*x);
}
```

10) Demonstrates overloading function for deferent number of arguments.

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

void printName(string fName);
void printName(string fName, string sName);

int main()
{
    string fName = "Emadeldeen",
           sName = "Abdallah";
    printName(fName);
    printName(fName, sName);
    return 0;
}
void printName(string fName)
{
    cout << fName << "\n";
}
void printName(string fName, string sName)
{
    cout << fName << " " << sName << "\n";
}
```

11) Demonstrates passing const by reference arguments to a function.

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

void printArr(const char (&a)[5]);
int main()
{
    char a[5] = {'a', 'b', 'c', 'd', 'e'};
    printArr(a);
    return 0;
}
void printArr(const char (&a)[5])
{
    int n = sizeof(a)/sizeof(char);
    for (int i=0; i<n; i++)
        cout << a[i] << "\t";
}
```

12) Static Local Variables

A static local variable has the visibility of an automatic local variable (that is, inside the function containing it). However, its lifetime is the same as that of a global variable, except that it doesn't come into existence until the first call to the function containing it. Thereafter it remains in existence for the life of the program.

Static local variables are used when it's necessary for a function to remember a value when it is not being executed; that is, between calls to the function. In the next example, a function, getavg(), calculates a running average. It remembers the total of the numbers it has averaged before, and how many there were. Each time it receives a new number, sent as an argument

from the calling program, it adds this number to the total, adds 1 to a count, and returns the new average by dividing the total by the count. Here's the listing for STATIC:

```
#include <iostream>
using namespace std;
float getavg(float); //declaration
int main()
{
    float data=1, avg;
    while( data != 0 )
    {
        cout << "Enter a number: ";
        cin >> data;
        avg = getavg(data);
        cout << "New average is " << avg << endl;
    }
    return 0;
}

// finds average of old plus new data
float getavg(float newdata)
{
    static float total = 0; //static variables are initialized
    static int count = 0; // only once per program
    count++; //increment count
    total += newdata; //add new data to total
    return total / count; //return the new average
}
```

Exercises

- 1) Raising a number n to a power p is the same as multiplying n by itself p times. Write a function called power() that takes a double value for n and an int value for p, and returns the result as a double value. Use a default argument of 2 for p, so that if this argument is omitted, the number n will be squared. Write a main() function that gets values from the user to test this function.
- 2) Start with the power() function of Exercise 1, which works only with type double. Create a series of overloaded functions with the same name that, in addition to double, also work with types char, int, long, and float. Write a main() program that exercises these overloaded functions with all argument types.
- 3) Write a function called zeroSmaller() that is passed two int arguments by reference and then sets the smaller of the two numbers to 0. Write a main() program to exercise this function.
- 4) Write a function called hms_to_secs() that takes three int values—for hours, minutes, and seconds—as arguments, and returns the equivalent time in seconds (type long). Create a program that exercises this function by repeatedly obtaining a time value in hours, minutes, and seconds from the user (format 12:59:59), calling the function, and displaying the value of seconds it returns.
- 5) Write a CPP function called secs_to_hms() that takes four parameters (int Sec, int& h, int& m, int& s) to convert secs to hours, minutes, seconds. Create a program that exercises this function, calling the function, and displaying the value in format hh:mm:ss.
- 6) Write a function that, when you call it, displays a message telling how many times it has been called: “I have been called 3 times”, for instance. Write a main() program that calls this function at least 10 times. Try implementing this function in two different ways. First, use a global variable to store the count. Second, use a local static variable. Which is more appropriate? Why can’t you use a local variable?
- 7) Write CPP function to sort an array of 10 double elements;
- 8) Write CPP Function that return true if array is sorted else return false.
- 9) Write CPP Function that return true if array is unique elements else return false.
- 10) Write CPP Function that take matrix of 10 char elements as a parameter then prints elements that are duplicated without duplicates at printing.
- 11) Write CPP Function that take matrix of 10 char elements and c as char to search if c is an element in the array return true else return false.
- 12) Write CPP function named arrayMul that return array c where: array c = array b * array c.