

Practice C++ Programming Statements after Foundation Training For Students of Complete Placement Preparatory Masterclass

Join Our Placement Preparation Masterclass on link below

<https://talentbattle.in/prepare/placement-preparation>



contact
n | www.TalentBattle.in

C++ "Hello World!" Program
// Your First C++ Program

```
#include <iostream>
```

```
int main() {  
    std::cout << "Hello World!";  
    return 0;  
}
```

Example: Print Number Entered by User

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

```
int main() { int  
    number;  
  
    cout << "Enter an integer: ";  
    cin >> number;  
    cout << "You entered " << number;  
    return 0;  
}
```

Example: Program to Add Two Integers

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

```
int main()  
{  
    int firstNumber, secondNumber, sumOfTwoNumbers;  
  
    cout << "Enter two integers: ";  
    cin >> firstNumber >> secondNumber;  
  
    // sum of two numbers is stored in variable sumOfTwoNumbers  
    sumOfTwoNumbers = firstNumber + secondNumber;  
  
    // Prints sum  
    cout << firstNumber << " + " << secondNumber << " = " << sumOfTwoNumbers;  
  
    return 0;  
}
```

Example: Compute quotient and remainder

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

```
int main()  
{
```

```

int divisor, dividend, quotient, remainder;

cout << "Enter dividend: ";
cin >> dividend;

cout << "Enter divisor: ";
cin >> divisor;

quotient = dividend / divisor;
remainder = dividend % divisor;

cout << "Quotient = " << quotient << endl;
cout << "Remainder = " << remainder;

return 0;
}
*****

```

Example: Find Size of a Variable

```

#include <iostream>
using namespace std;

int main()
{
    cout << "Size of char: " << sizeof(char) << " byte" << endl;
    cout << "Size of int: " << sizeof(int) << " bytes" << endl;
    cout << "Size of float: " << sizeof(float) << " bytes" << endl;
    cout << "Size of double: " << sizeof(double) << " bytes" << endl;

    return 0;
}
*****

```

Example 1: Swap Numbers (Using Temporary Variable)

```

#include <iostream>
using namespace std;

int main()
{
    int a = 5, b = 10, temp;

    cout << "Before swapping." << endl;
    cout << "a = " << a << ", b = " << b << endl;

    temp = a;
    a = b;
    b = temp;

    cout << "\nAfter swapping." << endl;
    cout << "a = " << a << ", b = " << b << endl;

    return 0;
}
*****

```

Example 2: Swap Numbers Without Using Temporary Variables

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

int main()
{
    int a = 5, b = 10;

    cout << "Before swapping." << endl;
    cout << "a = " << a << ", b = " << b << endl;

    a = a + b;
    b = a - b;
    a = a - b;

    cout << "\nAfter swapping." << endl;
    cout << "a = " << a << ", b = " << b << endl;

    return 0;
}
*****
```

Example 1: Check Whether Number is Even or Odd using if else

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

int main()
{
    int n;

    cout << "Enter an integer: ";
    cin >> n;

    if ( n % 2 == 0)
        cout << n << " is even.";
    else
        cout << n << " is odd.";

    return 0;
}
*****
```

Example 2: Check Whether Number is Even or Odd using ternary operators

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

int main()
{
    int n;

    cout << "Enter an integer: ";
    cin >> n;
```

```

    (n % 2 == 0) ? cout << n << " is even." : cout << n << " is odd.";

    return 0;
}
*****

Example: Check Vowel or a Consonant Manually
#include <iostream>
using namespace std;

int main() {
    char c;
    bool isLowercaseVowel, isUppercaseVowel;

    cout << "Enter an alphabet: ";
    cin >> c;

    // evaluates to 1 (true) if c is a lowercase vowel
    isLowercaseVowel = (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u');

    // evaluates to 1 (true) if c is an uppercase vowel
    isUppercaseVowel = (c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U');

    // show error message if c is not an alphabet
    if (!isalpha(c))
        printf("Error! Non-alphabetic character.");
    else if (isLowercaseVowel || isUppercaseVowel)
        cout << c << " is a vowel.";
    else
        cout << c << " is a consonant.";

    return 0;
}
*****

```

Example 1: Find Largest Number Using if Statement

```

#include <iostream>
using namespace std;

int main() {
    float n1, n2, n3;

    cout << "Enter three numbers: ";
    cin >> n1 >> n2 >> n3;

    if (n1 >= n2 && n1 >= n3)
        cout << "Largest number: " << n1;

    if (n2 >= n1 && n2 >= n3)
        cout << "Largest number: " << n2;

    if (n3 >= n1 && n3 >= n2)
        cout << "Largest number: " << n3;
}

```

```

    return 0;
}
*****

```

Example 2: Find Largest Number Using if...else Statement

```

#include <iostream>
using namespace std;

int main() {
    float n1, n2, n3;

    cout << "Enter three numbers: ";
    cin >> n1 >> n2 >> n3;

    if((n1 >= n2) && (n1 >= n3))
        cout << "Largest number: " << n1;
    else if((n2 >= n1) && (n2 >= n3))
        cout << "Largest number: " << n2;
    else
        cout << "Largest number: " << n3;

    return 0;
}
*****

```

Example 3: Find Largest Number Using Nested if...else statement

```

#include <iostream>
using namespace std;

int main() {
    float n1, n2, n3;

    cout << "Enter three numbers: ";
    cin >> n1 >> n2 >> n3;

    if (n1 >= n2) {
        if (n1 >= n3)
            cout << "Largest number: " << n1;
        else
            cout << "Largest number: " << n3;
    }
    else {
        if (n2 >= n3)
            cout << "Largest number: " << n2;
        else
            cout << "Largest number: " << n3;
    }

    return 0;
}
*****

```

Example: Roots of a Quadratic Equation

```

#include <iostream>
#include <cmath>

```

```

using namespace std;

int main() {

    float a, b, c, x1, x2, discriminant, realPart, imaginaryPart;
    cout << "Enter coefficients a, b and c: ";
    cin >> a >> b >> c;
    discriminant = b*b - 4*a*c;

    if (discriminant > 0) {
        x1 = (-b + sqrt(discriminant)) / (2*a);
        x2 = (-b - sqrt(discriminant)) / (2*a);
        cout << "Roots are real and different." << endl;
        cout << "x1 = " << x1 << endl;
        cout << "x2 = " << x2 << endl;
    }

    else if (discriminant == 0) {
        cout << "Roots are real and same." << endl;
        x1 = -b/(2*a);
        cout << "x1 = x2 = " << x1 << endl;
    }

    else {
        realPart = -b/(2*a);
        imaginaryPart = sqrt(-discriminant)/(2*a);
        cout << "Roots are complex and different." << endl;
        cout << "x1 = " << realPart << "+" << imaginaryPart << "i" << endl;
        cout << "x2 = " << realPart << "-" << imaginaryPart << "i" << endl;
    }

    return 0;
}
*****

```

Example: Sum of Natural Numbers using loop

```

#include <iostream>
using namespace std;

int main() {
    int n, sum = 0;

    cout << "Enter a positive integer: ";
    cin >> n;

    for (int i = 1; i <= n; ++i) {
        sum += i;
    }

    cout << "Sum = " << sum;
    return 0;
}
*****

```

Example: Check if a year is leap year or not

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

```
int main() {
    int year;

    cout << "Enter a year: ";
    cin >> year;

    if (year % 4 == 0) {
        if (year % 100 == 0) {
            if (year % 400 == 0)
                cout << year << " is a leap year.";
            else
                cout << year << " is not a leap year.";
        }
        else
            cout << year << " is a leap year.";
    }
    else
        cout << year << " is not a leap year.";
}
```

```
return 0;
```

```
}
```

```
*****
```

Example: Find Factorial of a given number

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

```
int main() {
    int n;
    long double factorial = 1.0;

    cout << "Enter a positive integer: ";
    cin >> n;

    if (n < 0)
        cout << "Error! Factorial of a negative number doesn't exist.";
    else {
        for (int i = 1; i <= n; ++i) {
            factorial *= i;
        }
        cout << "Factorial of " << n << " = " << factorial;
    }
}
```

```
return 0;
```

```
}
```

```
*****
```

Example 1: Display Multiplication table up to 10

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



```

int main()
{
    int n;

    cout << "Enter a positive integer: ";
    cin >> n;

    for (int i = 1; i <= 10; ++i) {
        cout << n << " * " << i << " = " << n * i << endl;
    }

    return 0;
}
*****

```

Example 2: Display multiplication table up to a given range

```

#include <iostream>
using namespace std;

int main()
{
    int n, range;

    cout << "Enter an integer: ";
    cin >> n;

    cout << "Enter range: ";
    cin >> range;

    for (int i = 1; i <= range; ++i) {
        cout << n << " * " << i << " = " << n * i << endl;
    }

    return 0;
}
*****

```

Example 1: Fibonacci Series up to n number of terms

```

#include <iostream>
using namespace std;

int main() {
    int n, t1 = 0, t2 = 1, nextTerm = 0;

    cout << "Enter the number of terms: ";
    cin >> n;

    cout << "Fibonacci Series: ";

    for (int i = 1; i <= n; ++i) {
        // Prints the first two terms.
        if (i == 1) {
            cout << t1 << ", ";

```

```

        continue;
    }
    if(i == 2) {
        cout << t2 << ", ";
        continue;
    }
    nextTerm = t1 + t2;
    t1 = t2;
    t2 = nextTerm;

    cout << nextTerm << ", ";
}
return 0;
}
*****

```

Example 2: Program to Generate Fibonacci Sequence Up to a Certain Number

```

#include <iostream>
using namespace std;

int main() {
    int t1 = 0, t2 = 1, nextTerm = 0, n;

    cout << "Enter a positive number: ";
    cin >> n;

    // displays the first two terms which is always 0 and 1
    cout << "Fibonacci Series: " << t1 << ", " << t2 << ", ";

    nextTerm = t1 + t2;

    while(nextTerm <= n) {
        cout << nextTerm << ", ";
        t1 = t2;
        t2 = nextTerm;
        nextTerm = t1 + t2;
    }
    return 0;
}
*****

```

Example 1: Find GCD using while loop

```

#include <iostream>
using namespace std;

int main()
{
    int n1, n2;

    cout << "Enter two numbers: ";
    cin >> n1 >> n2;

    while(n1 != n2)
    {

```

```

        if(n1 > n2)
            n1 -= n2;
        else
            n2 -= n1;
    }

    cout << "HCF = " << n1;
    return 0;
}
*****

```

Example: 2. Find HCF/GCD using for loop

```

#include <iostream>
using namespace std;

int main() {
    int n1, n2, hcf;
    cout << "Enter two numbers: ";
    cin >> n1 >> n2;

    // Swapping variables n1 and n2 if n2 is greater than n1.
    if (n2 > n1) {
        int temp = n2;
        n2 = n1;
        n1 = temp;
    }

    for (int i = 1; i <= n2; ++i) {
        if (n1 % i == 0 && n2 % i == 0) {
            hcf = i;
        }
    }

    cout << "HCF = " << hcf;
    return 0;
}
*****

```

Example 1: Find LCM

```

#include <iostream>
using namespace std;

int main()
{
    int n1, n2, max;

    cout << "Enter two numbers: ";
    cin >> n1 >> n2;

    // maximum value between n1 and n2 is stored in max
    max = (n1 > n2) ? n1 : n2;

    do
    {

```

```

        if (max % n1 == 0 && max % n2 == 0)
        {
            cout << "LCM = " << max;
            break;
        }
        else
            ++max;
    } while (true);

    return 0;
}
*****

```

Example: C++ Program to Reverse an Integer

```

#include <iostream>
using namespace std;

int main() {
    int n, reversedNumber = 0, remainder;

    cout << "Enter an integer: ";
    cin >> n;

    while(n != 0) {
        remainder = n%10;
        reversedNumber = reversedNumber*10 + remainder;
        n /= 10;
    }

    cout << "Reversed Number = " << reversedNumber;

    return 0;
}
*****

```

Example 1: Compute Power Manually

```

#include <iostream>
using namespace std;

int main()
{
    int exponent;
    float base, result = 1;

    cout << "Enter base and exponent respectively: ";
    cin >> base >> exponent;

    cout << base << "^" << exponent << " = ";

    while (exponent != 0) {
        result *= base;
        --exponent;
    }
}

```

```

    cout << result;

    return 0;
}
*****

```

Example 1: Prefix ++ Increment Operator Overloading with no return type

```

#include <iostream>
using namespace std;

class Check
{
private:
    int i;
public:
    Check(): i(0) { }
    void operator ++()
        { ++i; }
    void Display()
        { cout << "i=" << i << endl; }
};

int main()
{
    Check obj;

    // Displays the value of data member i for object obj
    obj.Display();

    // Invokes operator function void operator ++( )
    ++obj;

    // Displays the value of data member i for object obj
    obj.Display();

    return 0;
}
*****

```

Example 2: Prefix Increment ++ operator overloading with return type

```

#include <iostream>
using namespace std;

class Check
{
private:
    int i;
public:
    Check(): i(0) { }

    // Return type is Check
    Check operator ++()
    {
        Check temp;

```

```

        ++i;
        temp.i = i;

        return temp;
    }

    void Display()
    { cout << "i = " << i << endl; }
};

int main()
{
    Check obj, obj1;
    obj.Display();
    obj1.Display();

    obj1 = ++obj;

    obj.Display();
    obj1.Display();

    return 0;
}

```

Example 3: Postfix Increment ++ Operator Overloading

Overloading of increment operator up to this point is only true if it is used in prefix form.

This is the modification of above program to make this work both for prefix form and postfix form.

```

#include <iostream>
using namespace std;

class Check
{
private:
    int i;
public:
    Check(): i(0) { }
    Check operator ++ ()
    {
        Check temp;
        temp.i = ++i;
        return temp;
    }

    // Notice int inside barcket which indicates postfix increment.
    Check operator ++ (int)
    {
        Check temp;
        temp.i = i++;
        return temp;
    }
}

```

```

    void Display()
    { cout << "i = "<< i << endl; }
};

int main()
{
    Check obj, obj1;
    obj.Display();
    obj1.Display();

    // Operator function is called, only then value of obj is assigned to obj1
    obj1 = ++obj;
    obj.Display();
    obj1.Display();

    // Assigns value of obj to obj1, only then operator function is called.
    obj1 = obj++;
    obj.Display();
    obj1.Display();

    return 0;
}
Output

```

```

i = 0
i = 0
i = 1
i = 1
i = 2
i = 1

```

When increment operator is overloaded in prefix form; Check operator ++() is called but, when increment operator is overloaded in postfix form; Check operator ++ (int) is invoked.

Notice, the int inside bracket. This int gives information to the compiler that it is the postfix version of operator.

Don't confuse this int doesn't indicate integer.

Example 4: Operator Overloading of Decrement -- Operator

Decrement operator can be overloaded in similar way as increment operator.

```

#include <iostream>
using namespace std;

class Check
{
private:
    int i;
public:
    Check(): i(3) { }

```

```

Check operator -- ()
{
    Check temp;
    temp.i = --i;
    return temp;
}

// Notice int inside barcket which indicates postfix decrement.
Check operator -- (int)
{
    Check temp;
    temp.i = i--;
    return temp;
}

void Display()
{ cout << "i = " << i << endl; }
};

int main()
{
    Check obj, obj1;
    obj.Display();
    obj1.Display();

    // Operator function is called, only then value of obj is assigned to obj1
    obj1 = --obj;
    obj.Display();
    obj1.Display();

    // Assigns value of obj to obj1, only then operator function is called.
    obj1 = obj--;
    obj.Display();
    obj1.Display();

    return 0;
}
*****

```

Example: Binary Operator Overloading to Subtract Complex Number

```

#include <iostream>
using namespace std;

```

```

class Complex
{
private:
    float real;
    float imag;
public:
    Complex(): real(0), imag(0){}
    void input()
    {

```



```

        cout << "Enter real and imaginary parts respectively: ";
        cin >> real;
        cin >> imag;
    }

    // Operator overloading
    Complex operator - (Complex c2)
    {
        Complex temp;
        temp.real = real - c2.real;
        temp.imag = imag - c2.imag;

        return temp;
    }

    void output()
    {
        if(imag < 0)
            cout << "Output Complex number: " << real << imag << "i";
        else
            cout << "Output Complex number: " << real << "+" << imag << "i";
    }
};

int main()
{
    Complex c1, c2, result;

    cout << "Enter first complex number:\n";
    c1.input();

    cout << "Enter second complex number:\n";
    c2.input();

    // In case of operator overloading of binary operators in C++ programming,
    // the object on right hand side of operator is always assumed as argument by compiler.
    result = c1 - c2;
    result.output();

    return 0;
}
*****
Example: Print ASCII Value in C++
#include <iostream>
using namespace std;

int main() {
    char c;
    cout << "Enter a character: ";
    cin >> c;
    cout << "ASCII Value of " << c << " is " << int(c);
    return 0;
}

```

```

}
*****
C++Program to Multiply Two Numbers
#include <iostream>
using namespace std;

int main()
{
    double firstNumber, secondNumber, productOfTwoNumbers;
    cout << "Enter two numbers: ";

    // Stores two floating point numbers in variable firstNumber and secondNumber respectively
    cin >> firstNumber >> secondNumber;

    // Performs multiplication and stores the result in variable productOfTwoNumbers
    productOfTwoNumbers = firstNumber * secondNumber;

    cout << "Product = " << productOfTwoNumbers;

    return 0;
}

```

```

*****
Example: Check Palindrome Number
#include <iostream>
using namespace std;

int main()
{
    int n, num, digit, rev = 0;

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

    n = num;

    do
    {
        digit = num % 10;
        rev = (rev * 10) + digit;
        num = num / 10;
    } while (num != 0);

    cout << " The reverse of the number is: " << rev << endl;

    if (n == rev)
        cout << " The number is a palindrome.";
    else
        cout << " The number is not a palindrome.";

    return 0;
}
*****

```

Example: Check Prime Number

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

int main() {
    int i, n;
    bool isPrime = true;

    cout << "Enter a positive integer: ";
    cin >> n;

    // 0 and 1 are not prime numbers
    if (n == 0 || n == 1) {
        isPrime = false;
    }
    else {
        for (i = 2; i <= n / 2; ++i) {
            if (n % i == 0) {
                isPrime = false;
                break;
            }
        }
    }
    if (isPrime)
        cout << n << " is a prime number";
    else
        cout << n << " is not a prime number";

    return 0;
}
*****
```

Example #1: Display Prime Numbers Between two Intervals

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

int main() {
    int low, high, i; bool
    isPrime = true;

    cout << "Enter two numbers (intervals): ";
    cin >> low >> high;

    cout << "\nPrime numbers between " << low << " and " << high << " are: " << endl;

    while (low < high) {
        isPrime = true;
        if (low == 0 || low == 1) {
            isPrime = false;
        }
        else {
            for (i = 2; i <= low / 2; ++i) {
                if (low % i == 0) {
```

```

        isPrime = false;
        break;
    }
}

if (isPrime)
    cout << low << " ";

    ++low;
}

return 0;
}
*****

```

Example: Check Armstrong Number of 3 Digits

```

#include <iostream>
using namespace std;

int main() {
    int num, originalNum, remainder, result = 0;
    cout << "Enter a three-digit integer: ";
    cin >> num;
    originalNum = num;

    while (originalNum != 0) {
        // remainder contains the last digit
        remainder = originalNum % 10;

        result += remainder * remainder * remainder;

        // removing last digit from the original number
        originalNum /= 10;
    }

    if (result == num)
        cout << num << " is an Armstrong number.";
    else
        cout << num << " is not an Armstrong number.";

    return 0;
}
*****

```

Example: Check Armstrong Number of n Digits

```

#include <cmath>
#include <iostream>

using namespace std;

int main() {
    int num, originalNum, remainder, n = 0, result = 0, power;
    cout << "Enter an integer: ";

```

```

cin >> num;

originalNum = num;

while (originalNum != 0) {
    originalNum /= 10;
    ++n;
}
originalNum = num;

while (originalNum != 0) {
    remainder = originalNum % 10;

    // pow() returns a double value
    // round() returns the equivalent int
    power = round(pow(remainder, n));
    result += power;
    originalNum /= 10;
}

if (result == num)
    cout << num << " is an Armstrong number.";
else
    cout << num << " is not an Armstrong number.";
return 0;
}
*****

```

Example: Display Armstrong Number Between Intervals

```

#include <iostream>
using namespace std;

int main()
{
    int num1, num2, i, num, digit, sum;

    cout << "Enter first number: ";
    cin >> num1;

    cout << "Enter second number: ";
    cin >> num2;

    cout << "Armstrong numbers between " << num1 << " and " << num2 << " are: " << endl;
    for(i = num1; i <= num2; i++)
    {
        sum = 0;
        num = i;

        for(; num > 0; num /= 10)
        {
            digit = num % 10;
            sum = sum + digit * digit * digit;
        }
    }
}

```

```

        if(sum == i)
        {
            cout << i << endl;
        }
    }

    return 0;
}
*****

```

Example: Display all Factors of a Number

```
#include <iostream>
```

```
using namespace std;
```

```

int main() {
    int n, i;

    cout << "Enter a positive integer: ";
    cin >> n;

    cout << "Factors of " << n << " are: ";
    for(i = 1; i <= n; ++i) {
        if(n % i == 0)
            cout << i << " ";
    }

    return 0;
}
*****

```

Example 1: Program to print half pyramid using *

```

*
* *
* * *
* * * *
* * * * *

```

Source Code

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

```

int main()
{
    int rows;

    cout << "Enter number of rows: ";
    cin >> rows;

    for(int i = 1; i <= rows; ++i)
    {
        for(int j = 1; j <= i; ++j)
        {
            cout << "* ";

```

```

    }
    cout << "\n";
}
return 0;
}

```

Example 2: Program to print half pyramid using numbers

```

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
Source Code

```

```

#include <iostream>
using namespace std;

int main()
{
    int rows;

    cout << "Enter number of rows: ";
    cin >> rows;

    for(int i = 1; i <= rows; ++i)
    {
        for(int j = 1; j <= i; ++j)
        {
            cout << j << " ";
        }
        cout << "\n";
    }
    return 0;
}

```

Example 3: Program to print half pyramid using alphabets

```

A
B B
C C C
D D D D
E E E E E
Source Code

```

```

#include <iostream>
using namespace std;

int main()
{
    char input, alphabet = 'A';

    cout << "Enter the uppercase character you want to print in the last row: ";
    cin >> input;

    for(int i = 1; i <= (input - 'A' + 1); ++i)

```

```

{
    for(int j=1; j <= i; ++j)
    {
        cout << alphabet << " ";
    }
    ++alphabet;

    cout << endl;
}
return 0;
}

```

Programs to print inverted half pyramid using * and numbers

Example 4: Inverted half pyramid using *

```

* * * * *
* * * *
* * *
* *
*

```

Source Code

```

#include <iostream>
using namespace std;

int main()
{
    int rows;

    cout << "Enter number of rows: ";
    cin >> rows;

    for(int i = rows; i >= 1; --i)
    {
        for(int j = 1; j <= i; ++j)
        {
            cout << "* ";
        }
        cout << endl;
    }

    return 0;
}

```

Example 5: Inverted half pyramid using numbers

```

1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

```

Source Code

```

#include <iostream>
using namespace std;

```



```

int main()
{
    int rows;

    cout << "Enter number of rows: ";
    cin >> rows;

    for(int i = rows; i >= 1; --i)
    {
        for(int j = 1; j <= i; ++j)
        {
            cout << j << " ";
        }
        cout << endl;
    }

    return 0;
}

```

Programs to display pyramid and inverted pyramid using * and digits

Example 6: Program to print full pyramid using *

```

*
***
*****
*****
*****

```

Source Code

```

#include <iostream>
using namespace std;

int main()
{
    int space, rows;

    cout << "Enter number of rows: ";
    cin >> rows;

    for(int i = 1, k = 0; i <= rows; ++i, k = 0)
    {
        for(space = 1; space <= rows-i; ++space)
        {
            cout << " ";
        }

        while(k != 2*i-1)
        {
            cout << "* ";
            ++k;
        }
        cout << endl;
    }
    return 0;
}

```

```
}
```

Example 7: Program to print pyramid using numbers

```
1
2 3 2
3 4 5 4 3
4 5 6 7 6 5 4
5 6 7 8 9 8 7 6 5
Source Code
```

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

```
int main()
{
    int rows, count = 0, count1 = 0, k = 0;

    cout << "Enter number of rows: ";
    cin >> rows;

    for(int i = 1; i <= rows; ++i)
    {
        for(int space = 1; space <= rows-i; ++space)
        {
            cout << " ";
            ++count;
        }

        while(k != 2*i-1)
        {
            if (count <= rows-1)
            {
                cout << i+k << " ";
                ++count;
            }
            else
            {
                ++count1;
                cout << i+k-2*count1 << " ";
            }
            ++k;
        }
        count1 = count = k = 0;

        cout << endl;
    }
    return 0;
}
```

Example 8: Inverted full pyramid using *

```
*****
*****
*****
***
```

*

Source Code

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

int main()
{
    int rows;

    cout << "Enter number of rows: ";
    cin >> rows;

    for(int i = rows; i >= 1; --i)
    {
        for(int space = 0; space < rows-i; ++space)
            cout << " ";

        for(int j = i; j <= 2*i-1; ++j)
            cout << "* ";

        for(int j = 0; j < i-1; ++j)
            cout << "* ";

        cout << endl;
    }

    return 0;
}
```

Example 9: Print Pascal's triangle

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

Source Code

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

int main()
{
    int rows, coef = 1;

    cout << "Enter number of rows: ";
    cin >> rows;

    for(int i = 0; i < rows; i++)
    {
        for(int space = 1; space <= rows-i; space++)
            cout << " ";
    }
}
```

```

    for(int j=0; j <= i; j++)
    {
        if (j == 0 || i == 0)
            coef = 1;
        else
            coef = coef*(i-j+1)/j;

        cout << coef << " ";
    }
    cout << endl;
}

return 0;
}

```

Example 10: Print Floyd's Triangle.

```

1
2 3
4 5 6
7 8 9 10

```

Source Code

```

#include <iostream>
using namespace std;

int main()
{
    int rows, number = 1;

    cout << "Enter number of rows: ";
    cin >> rows;

    for(int i = 1; i <= rows; i++)
    {
        for(int j = 1; j <= i; ++j)
        {
            cout << number << " ";
            ++number;
        }

        cout << endl;
    }

    return 0;
}
*****

```

Example: Simple Calculator using switch statement

```

# include <iostream>
using namespace std;

```

```

int main() {

```

```

char op;
float num1, num2;

cout << "Enter operator: +, -, *, /: ";
cin >> op;

cout << "Enter two operands: ";
cin >> num1 >> num2;

switch(op) {
    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:
        // If the operator is other than +, -, * or /, error message is shown
        cout << "Error! operator is not correct";
        break;
}

return 0;
}
*****

```

Example: Prime Numbers Between two Intervals

```

#include <iostream>
using namespace std;

int checkPrimeNumber(int);

int main() {
    int n1, n2;
    bool flag;

    cout << "Enter two positive integers: ";
    cin >> n1 >> n2;

    // swapping n1 and n2 if n1 is greater than n2
    if (n1 > n2) {
        n2 = n1 + n2;
        n1 = n2 - n1;
    }
}

```

```

    n2 = n2 - n1;
}

cout << "Prime numbers between " << n1 << " and " << n2 << " are: ";

for(int i = n1+1; i < n2; ++i) {
    // If i is a prime number, flag will be equal to 1
    flag = checkPrimeNumber(i);

    if(flag)
        cout << i << " ";
}

return 0;
}

// user-defined function to check prime number
int checkPrimeNumber(int n) {
    bool isPrime = true;

    // 0 and 1 are not prime numbers
    if (n == 0 || n == 1) {
        isPrime = false;
    }
    else {
        for(int j = 2; j <= n/2; ++j) {
            if (n%j == 0) {
                isPrime = false;
                break;
            }
        }
    }

    return isPrime;
}
*****

```

Example: Check Whether a Number can be Expressed as a Sum of Two Prime Numbers

```

#include <iostream>
using namespace std;

bool checkPrime(int n);

int main() {
    int n, i;
    bool flag = false;

    cout << "Enter a positive integer: ";
    cin >> n;

    for(i = 2; i <= n/2; ++i) {
        if (checkPrime(i)) {
            if (checkPrime(n - i)) {

```

```

        cout << n << " = " << i << " + " << n-i << endl;
        flag = true;
    }
}

if (!flag)
    cout << n << " can't be expressed as sum of two prime numbers.";

return 0;
}

// Check prime number
bool checkPrime(int n)
{
    int i;
    bool isPrime = true;

    // 0 and 1 are not prime numbers
    if (n == 0 || n == 1) {
        isPrime = false;
    }
    else {
        for (i = 2; i <= n/2; ++i) {
            if (n % i == 0) {
                isPrime = false;
                break;
            }
        }
    }

    return isPrime;
}
*****

```

Example: Calculate Sum of Natural numbers using Recursion

```

#include<iostream>
using namespace std;

int add(int n);

int main() {
    int n;

    cout << "Enter a positive integer: ";
    cin >> n;

    cout << "Sum = " << add(n);

    return 0;
}

int add(int n) {

```

```

    if(n !=0)
        return n + add(n - 1);
    return 0;
}
*****

```

Example: Calculate Factorial Using Recursion

```

#include<iostream>
using namespace std;

int factorial(int n);

int main()
{
    int n;

    cout << "Enter a positive integer: ";
    cin >> n;

    cout << "Factorial of " << n << " = " << factorial(n);

    return 0;
}

int factorial(int n)
{
    if(n > 1)
        return n * factorial(n - 1);
    else
        return 1;
}

```

Example: Calculate H.C.F using recursion

```

#include <iostream>
using namespace std;

int hcf(int n1, int n2);

int main()
{
    int n1, n2;

    cout << "Enter two positive integers: ";
    cin >> n1 >> n2;

    cout << "H.C.F of " << n1 << " & " << n2 << " is: " << hcf(n1, n2);

    return 0;
}

int hcf(int n1, int n2)
{
    if (n2 != 0)

```



```

        return hcf(n2, n1 % n2);
    else
        return n1;
}
*****

```

Example 1: C++ Program to convert binary number to decimal

```

#include <iostream>
#include <cmath>

using namespace std;

int convertBinaryToDecimal(long long);

int main()
{
    long long n;

    cout << "Enter a binary number: ";
    cin >> n;

    cout << n << " in binary = " << convertBinaryToDecimal(n) << " in decimal";
    return 0;
}

int convertBinaryToDecimal(long long n)
{
    int decimalNumber = 0, i = 0, remainder;
    while (n != 0)
    {
        remainder = n % 10;
        n /= 10;
        decimalNumber += remainder * pow(2, i);
        ++i;
    }
    return decimalNumber;
}
*****

```

Example 1: Convert Octal Number to Decimal

```

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

int octalToDecimal(int octalNumber);

int main()
{
    int octalNumber;
    cout << "Enter an octal number: ";
    cin >> octalNumber;
    cout << octalNumber << " in octal = " << octalToDecimal(octalNumber) << " in decimal";

    return 0;
}

```

```

}

// Function to convert octal number to decimal
int octalToDecimal(int octalNumber)
{
    int decimalNumber = 0, i = 0, rem;
    while (octalNumber != 0)
    {
        rem = octalNumber % 10;
        octalNumber /= 10;
        decimalNumber += rem * pow(8, i);
        ++i;
    }
    return decimalNumber;
}

```

Output

Enter an octal number: 2341

2341 in octal = 1249 in decimal

In the program, the octal number is stored in the variable octalNumber and passed to function octalToDecimal().

This function converts the octal number passed by user to its equivalent decimal number and returns it to main() function.

Example 2: Convert Decimal Number to Octal

```

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

int decimalToOctal(int decimalNumber);

int main()
{
    int decimalNumber;
    cout << "Enter a decimal number: ";
    cin >> decimalNumber;
    cout << decimalNumber << " in decimal = " << decimalToOctal(decimalNumber) << " in octal";

    return 0;
}

// Function to convert decimal number to octal
int decimalToOctal(int decimalNumber)
{
    int rem, i = 1, octalNumber = 0;
    while (decimalNumber != 0)
    {
        rem = decimalNumber % 8;
        decimalNumber /= 8;
        octalNumber += rem * i;
        i *= 10;
    }
}

```

```

    }
    return octalNumber;
}
Output

```

Enter an decimal number: 78

78 in decimal= 116 in octal

In the program, the decimal number is stored in the variable decimalNumber and passed to function decimalToOctal().

This function converts the decimal number passed by user to its equivalent octal number and returns it to main() function.

Example 1: Program to Convert Binary to Octal

In this program, we will first convert the binary number to decimal. Then, the decimal number is converted to octal.

```

#include <iostream>

```

```

#include <cmath>

```

```

using namespace std;

```

```

int convertBinarytoOctal(long long);

```

```

int main()

```

```

{

```

```

    long long binaryNumber;

```

```

    cout << "Enter a binary number: ";

```

```

    cin >> binaryNumber;

```

```

    cout << binaryNumber << " in binary = " << convertBinarytoOctal(binaryNumber) << " in octal ";

```

```

    return 0;

```

```

}

```

```

int convertBinarytoOctal(long long binaryNumber)

```

```

{

```

```

    int octalNumber = 0, decimalNumber = 0, i = 0;

```

```

    while(binaryNumber != 0)

```

```

    {

```

```

        decimalNumber += (binaryNumber%10) * pow(2,i);

```

```

        ++i;

```

```

        binaryNumber/=10;

```

```

    }

```

```

    i = 1;

```

```

    while (decimalNumber != 0)

```

```

    {

```

```

        octalNumber += (decimalNumber % 8) * i;

```

```

        decimalNumber /= 8;

```

contact@talentbattle.in |

www.talentbattle.in

```

        i *= 10;
    }

    return octalNumber;
}
*****

```

Example 2: Program to Convert Octal to Binary

In this program, the octal number is converted to decimal at first. Then, the decimal number is converted to binary number.

```

#include <iostream>
#include <cmath>

using namespace std;

long long convertOctalToBinary(int);
int main()
{
    int octalNumber;

    cout << "Enter an octal number: ";
    cin >> octalNumber;

    cout << octalNumber << " in octal = " << convertOctalToBinary(octalNumber) << " in binary";

    return 0;
}

long long convertOctalToBinary(int octalNumber)
{
    int decimalNumber = 0, i = 0;
    long long binaryNumber = 0;

    while(octalNumber != 0)
    {
        decimalNumber += (octalNumber%10) * pow(8,i);
        ++i;
        octalNumber/=10;
    }

    i = 1;

    while (decimalNumber != 0)
    {
        binaryNumber += (decimalNumber % 2) * i;
        decimalNumber /= 2;
        i *= 10;
    }

    return binaryNumber;
}
*****

```

Example: Reverse a sentence using recursion.

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

// function prototype
void reverse(const string& a);

int main() {
    string str;

    cout << " Please enter a string " << endl;
    getline(cin, str);

    // function call
    reverse(str);

    return 0;
}

// function definition
void reverse(const string& str) {

    // store the size of the string
    size_t numOfChars = str.size();

    if(numOfChars == 1) {
        cout << str << endl;
    }
    else {
        cout << str[numOfChars - 1];

        // function recursion
        reverse(str.substr(0, numOfChars - 1));
    }
}
```

Example: Program to Computer Power Using Recursion

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

int calculatePower(int, int);

int main()
{
    int base, powerRaised, result;

    cout << "Enter base number: ";
    cin >> base;

    cout << "Enter power number(positive integer): ";
    cin >> powerRaised;
```

```

    result = calculatePower(base, powerRaised);
    cout << base << "^" << powerRaised << " = " << result;

    return 0;
}

int calculatePower(int base, int powerRaised)
{
    if (powerRaised != 0)
        return (base * calculatePower(base, powerRaised - 1));
    else
        return 1;
}
*****

```

Example: Calculate Average of Numbers Using Arrays

```

#include <iostream>
using namespace std;

int main()
{
    int n, i;
    float num[100], sum=0.0, average;

    cout << "Enter the numbers of data: ";
    cin >> n;

    while (n > 100 || n <= 0)
    {
        cout << "Error! number should in range of (1 to 100)." << endl;
        cout << "Enter the number again: ";
        cin >> n;
    }

    for (i = 0; i < n; ++i)
    {
        cout << i + 1 << ". Enter number: ";
        cin >> num[i];
        sum += num[i];
    }

    average = sum / n;
    cout << "Average = " << average;

    return 0;
}
*****

```

Example: Display Largest Element of an array

```

#include <iostream>
using namespace std;

int main()
{

```

```

int i, n;
float arr[100];

cout << "Enter total number of elements(1 to 100): ";
cin >> n;
cout << endl;

// Store number entered by the user
for(i = 0; i < n; ++i)
{
    cout << "Enter Number " << i + 1 << " : ";
    cin >> arr[i];
}

// Loop to store largest number to arr[0]
for(i = 1; i < n; ++i)
{
    // Change < to > if you want to find the smallest element
    if(arr[0] < arr[i])
        arr[0] = arr[i];
}
cout << "Largest element = " << arr[0];

return 0;
}
*****
Example: Calculate Standard Deviation by Passing it to Function
#include <iostream>
#include <cmath>
using namespace std;

float calculateSD(float data[]);

int main()
{
    int i;
    float data[10];

    cout << "Enter 10 elements: ";
    for(i = 0; i < 10; ++i)
        cin >> data[i];

    cout << endl << "Standard Deviation = " << calculateSD(data);

    return 0;
}

float calculateSD(float data[])
{
    float sum = 0.0, mean, standardDeviation = 0.0;

    int i;

```

```

for(i=0; i < 10; ++i)
{
    sum += data[i];
}

mean = sum/10;

for(i=0; i < 10; ++i)
    standardDeviation += pow(data[i] - mean, 2);

return sqrt(standardDeviation / 10);
}
*****

```

Example: Add Two Matrices using Multi-dimensional Arrays

```

#include <iostream>
using namespace std;

int main()
{
    int r, c, a[100][100], b[100][100], sum[100][100], i, j;

    cout << "Enter number of rows (between 1 and 100): ";
    cin >> r;

    cout << "Enter number of columns (between 1 and 100): ";
    cin >> c;

    cout << endl << "Enter elements of 1st matrix: " << endl;

    // Storing elements of first matrix entered by user.
    for(i=0; i < r; ++i)
        for(j=0; j < c; ++j)
        {
            cout << "Enter element a" << i+1 << j+1 << " : ";
            cin >> a[i][j];
        }

    // Storing elements of second matrix entered by user.
    cout << endl << "Enter elements of 2nd matrix: " << endl;
    for(i=0; i < r; ++i)
        for(j=0; j < c; ++j)
        {
            cout << "Enter element b" << i+1 << j+1 << " : ";
            cin >> b[i][j];
        }

    // Adding Two matrices
    for(i=0; i < r; ++i)
        for(j=0; j < c; ++j)
            sum[i][j] = a[i][j] + b[i][j];
}

```



```

// Displaying the resultant sum matrix.
cout << endl << "Sum of two matrix is: " << endl;
for(i = 0; i < r; ++i)
    for(j = 0; j < c; ++j)
    {
        cout << sum[i][j] << " ";
        if(j == c - 1)
            cout << endl;
    }

return 0;
}
*****

```

Example: Multiply two matrices without using functions

```

#include <iostream>
using namespace std;

int main()
{
    int a[10][10], b[10][10], mult[10][10], r1, c1, r2, c2, i, j, k;

    cout << "Enter rows and columns for first matrix: ";
    cin >> r1 >> c1;
    cout << "Enter rows and columns for second matrix: ";
    cin >> r2 >> c2;

    // If column of first matrix is not equal to row of second matrix,
    // ask the user to enter the size of matrix again.
    while (c1 != r2)
    {
        cout << "Error! column of first matrix not equal to row of second.";

        cout << "Enter rows and columns for first matrix: ";
        cin >> r1 >> c1;

        cout << "Enter rows and columns for second matrix: ";
        cin >> r2 >> c2;
    }

    // Storing elements of first matrix.
    cout << endl << "Enter elements of matrix 1:" << endl;
    for(i = 0; i < r1; ++i)
        for(j = 0; j < c1; ++j)
        {
            cout << "Enter element a" << i + 1 << j + 1 << " : ";
            cin >> a[i][j];
        }

    // Storing elements of second matrix.
    cout << endl << "Enter elements of matrix 2:" << endl;
    for(i = 0; i < r2; ++i)
        for(j = 0; j < c2; ++j)

```

```

    {
        cout << "Enter element b" << i + 1 << j + 1 << " : ";
        cin >> b[i][j];
    }

// Initializing elements of matrix mult to 0.
for(i=0; i < r1; ++i)
    for(j=0; j < c2; ++j)
    {
        mult[i][j]=0;
    }

// Multiplying matrix a and b and storing in array mult.
for(i=0; i < r1; ++i)
    for(j=0; j < c2; ++j)
        for(k=0; k < c1; ++k)
        {
            mult[i][j] += a[i][k] * b[k][j];
        }

// Displaying the multiplication of two matrix.
cout << endl << "Output Matrix: " << endl;
for(i=0; i < r1; ++i)
    for(j=0; j < c2; ++j)
    {
        cout << " " << mult[i][j];
        if(j == c2-1)
            cout << endl;
    }

return 0;
}
*****

```

Example: Find Transpose of a Matrix

```

#include <iostream>
using namespace std;

int main() {
    int a[10][10], transpose[10][10], row, column, i, j;

    cout << "Enter rows and columns of matrix: ";
    cin >> row >> column;

    cout << "\nEnter elements of matrix: " << endl;

// Storing matrix elements
for (int i = 0; i < row; ++i) {
    for (int j = 0; j < column; ++j) {
        cout << "Enter element a" << i + 1 << j + 1 << " : ";
        cin >> a[i][j];
    }
}

```

```

// Printing the a matrix
cout << "\nEntered Matrix: " << endl;
for (int i = 0; i < row; ++i) {
    for (int j = 0; j < column; ++j) {
        cout << " " << a[i][j];
        if (j == column - 1)
            cout << endl << endl;
    }
}

// Computing transpose of the matrix
for (int i = 0; i < row; ++i)
    for (int j = 0; j < column; ++j) {
        transpose[j][i] = a[i][j];
    }

// Printing the transpose
cout << "\nTranspose of Matrix: " << endl;
for (int i = 0; i < column; ++i)
    for (int j = 0; j < row; ++j) {
        cout << " " << transpose[i][j];
        if (j == row - 1)
            cout << endl << endl;
    }

return 0;
}
*****

```

Example: Access Array Elements Using Pointer

```

#include <iostream>
using namespace std;

int main()
{
    int data[5];
    cout << "Enter elements: ";

    for (int i = 0; i < 5; ++i)
        cin >> data[i];

    cout << "You entered: ";
    for (int i = 0; i < 5; ++i)
        cout << endl << *(data + i);

    return 0;
}
*****

```

Example: Program to Swap Elements Using Call by Reference

```

#include <iostream>
using namespace std;

```

```

void cyclicSwap(int *a, int *b, int *c);

int main()
{
    int a, b, c;

    cout << "Enter value of a, b and c respectively: ";
    cin >> a >> b >> c;

    cout << "Value before swapping: " << endl;
    cout << "a, b and c respectively are: " << a << ", " << b << ", " << c << endl;

    cyclicSwap(&a, &b, &c);

    cout << "Value after swapping numbers in cycle: " << endl;
    cout << "a, b and c respectively are: " << a << ", " << b << ", " << c << endl;

    return 0;
}

void cyclicSwap(int *a, int *b, int *c)
{
    int temp;
    temp = *b;
    *b = *a;
    *a = *c;
    *c = temp;
}
*****

```

Example 1: Find Frequency of Characters of a String Object

```

#include <iostream>
using namespace std;

int main()
{
    string str = "C++ Programming is awesome";
    char checkCharacter = 'a';
    int count = 0;

    for (int i = 0; i < str.size(); i++)
    {
        if (str[i] == checkCharacter)
        {
            ++count;
        }
    }

    cout << "Number of " << checkCharacter << " = " << count;

    return 0;
}
*****

```

Example 1: From a C-style string

This program takes a C-style string from the user and calculates the number of vowels, consonants, digits and white-spaces.

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

int main()
{
    char line[150];
    int vowels, consonants, digits, spaces;

    vowels = consonants = digits = spaces = 0;

    cout << "Enter a line of string: ";
    cin.getline(line, 150);
    for(int i = 0; line[i] != '\0'; ++i)
    {
        if(line[i] == 'a' || line[i] == 'e' || line[i] == 'i' ||
           line[i] == 'o' || line[i] == 'u' || line[i] == 'A' ||
           line[i] == 'E' || line[i] == 'I' || line[i] == 'O' ||
           line[i] == 'U')
        {
            ++vowels;
        }
        else if((line[i] >= 'a' && line[i] <= 'z') || (line[i] >= 'A' && line[i] <= 'Z'))
        {
            ++consonants;
        }
        else if(line[i] >= '0' && line[i] <= '9')
        {
            ++digits;
        }
        else if(line[i] == ' ')
        {
            ++spaces;
        }
    }

    cout << "Vowels: " << vowels << endl;
    cout << "Consonants: " << consonants << endl;
    cout << "Digits: " << digits << endl;
    cout << "White spaces: " << spaces << endl;

    return 0;
}
```

Example 2: From a String Object

This program takes a string object from the user and calculates the number of vowels, consonants, digits and white-spaces.

```
#include <iostream>
```

```

using namespace std;

int main()
{
    string line;
    int vowels, consonants, digits, spaces;

    vowels = consonants = digits = spaces = 0;

    cout << "Enter a line of string: ";
    getline(cin, line);

    for(int i = 0; i < line.length(); ++i)
    {
        if((line[i]=='a' || line[i]=='e' || line[i]=='i' ||
            line[i]=='o' || line[i]=='u' || line[i]=='A' ||
            line[i]=='E' || line[i]=='I' || line[i]=='O' ||
            line[i]=='U'))
        {
            ++vowels;
        }
        else if((line[i]>='a'&& line[i]<='z') || (line[i]>='A'&& line[i]<='Z'))
        {
            ++consonants;
        }
        else if(line[i]>='0' && line[i]<='9')
        {
            ++digits;
        }
        else if(line[i]==' ')
        {
            ++spaces;
        }
    }

    cout << "Vowels: " << vowels << endl;
    cout << "Consonants: " << consonants << endl;
    cout << "Digits: " << digits << endl;
    cout << "White spaces: " << spaces << endl;

    return 0;
}

```

Example 1: Remove all characters except alphabets

This program takes a string (object) input from the user and removes all characters except alphabets.

```

#include <iostream>
using namespace std;

int main() {
    string line;
    string temp = "";

```

```

cout << "Enter a string: ";
getline(cin, line);

for (int i = 0; i < line.size(); ++i) {
    if ((line[i] >= 'a' && line[i] <= 'z') || (line[i] >= 'A' && line[i] <= 'Z')) {
        temp = temp + line[i];
    }
}
line = temp;
cout << "Output String: " << line;
return 0;
}
*****

```

Example 2: Remove all characters except alphabets

This program below takes a string (C-style string) input from the user and removes all characters except alphabets.

```

#include <iostream>
using namespace std;

int main() {
    char line[100], alphabetString[100];
    int j = 0;
    cout << "Enter a string: ";
    cin.getline(line, 100);

    for (int i = 0; line[i] != '\0'; ++i)
    {
        if ((line[i] >= 'a' && line[i] <= 'z') || (line[i] >= 'A' && line[i] <= 'Z'))
        {
            alphabetString[j++] = line[i];
        }
    }
    alphabetString[j] = '\0';

    cout << "Output String: " << alphabetString;
    return 0;
}
*****

```

Example 1: Concatenate String Objects

```

#include <iostream>
using namespace std;

int main()
{
    string s1, s2, result;

    cout << "Enter string s1: ";
    getline (cin, s1);

```

```

    cout << "Enter string s2: ";
    getline (cin, s2);

    result = s1 + s2;

    cout << "Resultant String = "<< result;

    return 0;
}
*****

```

Example 2: Concatenate C-style Strings

```

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

int main()
{
    char s1[50], s2[50];

    cout << "Enter string s1: ";
    cin.getline(s1, 50);

    cout << "Enter string s2: ";
    cin.getline(s2, 50);

    strcat(s1, s2);

    cout << "s1 = " << s1 << endl;
    cout << "s2 = " << s2;

    return 0;
}
*****

```

Example 1: Copy String Object

```

#include <iostream>
using namespace std;

int main()
{
    string s1, s2;

    cout << "Enter string s1: ";
    getline (cin, s1);

    s2 = s1;

    cout << "s1 = "<< s1 << endl;
    cout << "s2 = "<< s2;

    return 0;
}
*****

```


Example 1: Copy C-Strings

```
#include <iostream>
```

```
#include <cstring>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    char s1[100], s2[100];
```

```
    cout << "Enter string s1: ";
```

```
    cin.getline(s1, 100);
```

```
    strcpy(s2, s1);
```

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

```
    cout << "s2 = " << s2;
```

```
    return 0;
```

```
}
```

```
*****
```

Example: Sort Words in Dictionary Order

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    string str[10], temp;
```

```
    cout << "Enter 10 words: " << endl;
```

```
    for(int i = 0; i < 10; ++i)
```

```
    {
```

```
        getline(cin, str[i]);
```

```
    }
```

```
// Use Bubble Sort to arrange words
```

```
for(int i = 0; i < 9; ++i) {
```

```
    for(int j = 0; j < 9 - i; ++j) {
```

```
        if (str[j] > str[j + 1]) {
```

```
            temp = str[j];
```

```
            str[j] = str[j + 1];
```

```
            str[j + 1] = temp;
```

```
        }
```

```
    }
```

```
}
```

```
cout << "In lexicographical order: " << endl;
```

```
for(int i = 0; i < 10; ++i)
```

```
{
```

```
    cout << str[i] << endl;
```

```
}
```

```

    return 0;
}
*****

```

Example: Store and Display Information Using Structure

```

#include <iostream>
using namespace std;

```

```

struct student
{
    char name[50];
    int roll;
    float marks;
};

```

```

int main()
{
    student s;
    cout << "Enter information," << endl;
    cout << "Enter name: ";
    cin >> s.name;
    cout << "Enter roll number: ";
    cin >> s.roll;
    cout << "Enter marks: ";
    cin >> s.marks;

    cout << "\nDisplaying Information," << endl;
    cout << "Name: " << s.name << endl;
    cout << "Roll: " << s.roll << endl;
    cout << "Marks: " << s.marks << endl;
    return 0;
}

```

Example: Add Distances Using Structures

```

#include <iostream>
using namespace std;

```

```

struct Distance {
    int feet;
    float inch;
}d1 , d2, sum;

```

```

int main() {
    cout << "Enter 1st distance," << endl;
    cout << "Enter feet: ";
    cin >> d1.feet;
    cout << "Enter inch: ";
    cin >> d1.inch;

    cout << "\nEnter information for 2nd distance" << endl;
    cout << "Enter feet: ";
    cin >> d2.feet;
    cout << "Enter inch: ";

```

```

cin >> d2.inch;

sum.feet = d1.feet+d2.feet;
sum.inch = d1.inch+d2.inch;

// changing to feet if inch is greater than 12
if(sum.inch > 12) {
    // extra feet
    int extra = sum.inch / 12;

    sum.feet += extra;
    sum.inch -= (extra * 12);
}

cout << endl << "Sum of distances = " << sum.feet << " feet " << sum.inch << " inches";
return 0;
}

```

Example: Source Code to Add Two Complex Numbers

// Complex numbers are entered by the user

```

#include <iostream>
using namespace std;

typedef struct complex {
    float real;
    float imag;
} complexNumber;

complexNumber addComplexNumbers(complex, complex);

int main() {
    complexNumber num1, num2, complexSum;
    char signOfImag;

    cout << "For 1st complex number," << endl;
    cout << "Enter real and imaginary parts respectively:" << endl;
    cin >> num1.real >> num1.imag;

    cout << endl
        << "For 2nd complex number," << endl;
    cout << "Enter real and imaginary parts respectively:" << endl;
    cin >> num2.real >> num2.imag;

    // Call add function and store result in complexSum
    complexSum = addComplexNumbers(num1, num2);

    // Use Ternary Operator to check the sign of the imaginary number
    signOfImag = (complexSum.imag > 0) ? '+' : '-';

    // Use Ternary Operator to adjust the sign of the imaginary number

```

contact@talentbattle.in |

www.talentbattle.in

```

    complexSum.imag = (complexSum.imag > 0) ? complexSum.imag : -complexSum.imag;

    cout << "Sum = " << complexSum.real << signOfImag << complexSum.imag << "i";

    return 0;
}

complexNumber addComplexNumbers(complex num1, complex num2) {
    complex temp;
    temp.real = num1.real + num2.real;
    temp.imag = num1.imag + num2.imag;
    return (temp);
}
*****

Example: Program to Time Difference
// Computes time difference of two time period
// Time periods are entered by the user

#include <iostream>
using namespace std;

struct TIME
{
    int seconds;
    int minutes;
    int hours;
};

void computeTimeDifference(struct TIME, struct TIME, struct TIME *);

int main()
{
    struct TIME t1, t2, difference;

    cout << "Enter start time." << endl;
    cout << "Enter hours, minutes and seconds respectively: ";
    cin >> t1.hours >> t1.minutes >> t1.seconds;

    cout << "Enter stop time." << endl;
    cout << "Enter hours, minutes and seconds respectively: ";
    cin >> t2.hours >> t2.minutes >> t2.seconds;

    computeTimeDifference(t1, t2, &difference);

    cout << endl << "TIME DIFFERENCE: " << t1.hours << ":" << t1.minutes << ":" << t1.seconds;
    cout << " - " << t2.hours << ":" << t2.minutes << ":" << t2.seconds;
    cout << " = " << difference.hours << ":" << difference.minutes << ":" << difference.seconds;
    return 0;
}

void computeTimeDifference(struct TIME t1, struct TIME t2, struct TIME *difference){

    if(t2.seconds > t1.seconds)

```

```

{
    --t1.minutes;
    t1.seconds += 60;
}

difference->seconds = t1.seconds - t2.seconds;
if(t2.minutes > t1.minutes)
{
    --t1.hours;
    t1.minutes += 60;
}
difference->minutes = t1.minutes - t2.minutes;
difference->hours = t1.hours - t2.hours;
}
*****

```

Example: Store Information in Structure and Display it.

```

#include <iostream>
using namespace std;

struct student
{
    char name[50];
    int roll;
    float marks;
} s[10];

int main()
{
    cout << "Enter information of students: " << endl;

    // storing information
    for(int i = 0; i < 10; ++i)
    {
        s[i].roll = i+1;
        cout << "For roll number" << s[i].roll << ", " << endl;

        cout << "Enter name: ";
        cin >> s[i].name;

        cout << "Enter marks: ";
        cin >> s[i].marks;

        cout << endl;
    }

    cout << "Displaying Information: " << endl;

    // Displaying information
    for(int i = 0; i < 10; ++i)
    {
        cout << "\nRoll number: " << i+1 << endl;
        cout << "Name: " << s[i].name << endl;
    }
}

```

```
        cout << "Marks: " << s[i].marks << endl;
    }

    return 0;
}
```
