

1. Write a C++ program that reads three coefficients a , b and c for quadratic equation and finds whether the solutions are real or imaginary. (for $ax^2 + bx + c = 0$, if $b^2 - 4ac = 0$ then the solutions are real)

Source Code:

```
#include <iostream>
```

```
#include <cmath>
```

```
#include <iomanip>
```

```
int main() {
```

```
    float a, b, c;
```

```
    std::cout << "Enter a, b and c for a quadratic equation: ";  
    std::cin >> a >> b >> c;
```

```
    float x1 = (-b + sqrt(pow(b,2) - 4*a*c)) / (2*a);
```

```
    float x2 = (-b - sqrt(pow(b,2) - 4*a*c)) / (2*a);
```

```
    float j = pow(b,2) - 4*a*c;
```

```
    if (j < 0) {
```

```
        std::cout << "The solutions are imaginary." <<  
        std::endl;
```

```
    }
```

```
    else {
```

```
        std::cout << "The solutions are real: x1 = " <<
```

```
        x1 << " & x2 = " << std::fixed <<
```

```
        std::setprecision(2) << x2 << "." << std::endl;
```

```
    }
```

```
    return 0;
```

```
}
```

Description:

The program takes in input for variables a , b and c .

The values of x_1 and x_2 are calculated.

Variable j represents the condition for the solutions of the quadratic equation being real, i.e., the solutions x_1 and x_2 are only real if j ^{is not} ~~is~~ less than zero.

The program states that the solutions are imaginary if j is less than zero. Otherwise, it says that the solutions are real while also displaying their values.

Output:

Enter a , b and c for a quadratic equation: 3 -4 1

The solutions are real: $x_1 = 1$ & $x_2 = 0.33$.

2. Write a C++ program that reads ten positive numbers from user and finally prints the largest of all. (use for loop, if condition and function)

Source Code:

```
#include <iostream>
```

```
int greatest_num (int a[10]) {  
    int g = a[0];  
    for (int i = 0; i < 10; i++) {  
        if (g < a[i]) {  
            g = a[i];  
        }  
    }  
    return g;  
}
```

```
}
```

```
int main () {
```

```
    int a[10];
```

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

```
        jump-here:
```

```
        std::cout << "Enter number " << (i+1) << ". : ";
```

```
        std::cin >> a[i];
```

```
        if (a[i] <= 0) {
```

```
            std::cout << "Enter only positive integers" <<  
            std::endl;
```

```
            goto jump-here;
```

```
        }
```

```
    }
```

```
    std::cout << "The greatest number is " << greatest_num(a)  
    << "." << std::endl;
```

```
    return 0;
```

```
}
```

Description:

The program starts by taking input for elements in an array `a` of size 10.

It uses `if` `goto` statements and `if` statements to ensure only natural numbers are entered by the user.

Then the greatest number out of those in the array is displayed using a function called `greatest-num`.

The array `a` is passed by pointer to the function `greatest-num`.

Inside this function, a variable `g` takes on the value of the first element of array `a`. Through a `for` loop, its value is set to a new element anytime the current value of `g` is less than the element of array in that instance of the loop.

When the loop ends, the function returns `g`, which is now equal to the largest number inside the array.

Output:

Enter number 1. : 1

Enter number 2. : 6

Enter number 3. : 5

Enter number 4. : 3

Enter number 5. : 98

Enter number 6. : 54

Enter number 7. : -1

Enter only positive integers.

Enter number 7. : 32

Enter number 8. : 2

Enter number 9. : 32

Enter number 10. : 45

The greatest number is 98.

3. Write a C++ program using function (pass by reference) that calculates the values of x and y from the two linear equations.

$$ax + by = m$$

$$cx + dy = n$$

The solutions are given as

$$x = (md - bn) / (ad - cb)$$

$$y = (na - mc) / (ad - cb)$$

The function should take eight arguments and return nothing.

Source Code:

```
#include <iostream>
```

```
void solve(int &a, int &b, int &c, int &d, int &m, int &n,  
float &x, float &y) {
```

```
    x = ((m*d) - (float)(b*n)) / ((a*d) - (c*b));
```

```
    y = ((n*a) - (float)(m*c)) / ((a*d) - (c*b));
```

```
}
```

```
int main() {
```

```
    int a, b, c, d, m, n;
```

```
    std::cout << "Enter the values of a, b, c, d, m and n: ";
```

```
    std::cin >> a >> b >> c >> d >> m >> n;
```

```
    float x, y;
```

```
    solve(a, b, c, d, m, n, x, y);
```

```
    std::cout << "x = " << x << " and y = " << y << ". " <<
```

```
    std::endl;
```

```
    return 0;
```

```
}
```

Description :

The program starts by taking input for the variables a, b, c, d, m and n .

After that, it passes those variables and the variables x and y to a function called `solve`, which has corresponding reference variables as its parameters.

Then, the value of x and y are simply calculated according to the given equations. The function needs to return nothing as we are using reference variables. Also, we must typecast integers to floats for an accurate ~~set~~ value of x and y .

Then, the values of x and y are displayed by the main function.

Output :

Enter the values of a, b, c, d, m and n : 2 1 2 2 3 1

$x = 2.5$ and $y = -2$.

in cpp can u have a reference array as parameter for a function •

In C++, you cannot have a true reference to an entire array unless you specify its size, because references to arrays are quite strict about the type, including the size of the array.

Here's how it works:

✓ Valid: Passing array by reference with a specified size

cpp

Copy

```
void process(int (&arr)[5]) {  
    for (int i = 0; i < 5; ++i)  
        std::cout << arr[i] << " ";  
}
```

This function only accepts an array of **exactly 5 ints**. You're passing the array **by reference**, so no copy is made.

Usage:

cpp

Copy

```
int myArray[5] = {1, 2, 3, 4, 5};  
process(myArray); // OK
```

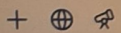
✗ Invalid: Reference to an array without size



cpp

Copy

Ask anything



ChatGPT >

✓ **valid: Passing array by reference with a specified size**

cpp

```
void process(int (&arr)[5]) {  
    for (int i = 0; i < 5; ++i)  
        std::cout << arr[i] << " ";  
}
```

This function only accepts an array of **exactly 5 ints**. You're passing the array **by reference**, so no copy is made.

Usage:

cpp

```
int myArray[5] = {1, 2, 3, 4, 5};  
process(myArray); // OK
```

✗ **Invalid: Reference to an array without size**

cpp

```
void process(int (&arr)[]) { // ✗ Compile error: array size must be known  
    ...  
}
```

✓ **Alternative: Pass pointer and size**



If the size isn't fixed, you usually do this:

Ask anything

