HIMALAYA COLLEGE OF ENGINEERING

Advanced C++ Programming Lab Report

Lab 1: Quadratic Equations, Triangle Classification and Password Strength Checking

Prepared By: Inisha Mali

Subject: Object Oriented Programming (OOP)

Program: Bachelor's of Electronics and Computer Engineering

Institution: Himalaya College of Engineering

Date: May 25, 2025

Objectives:

- To utilize control structures, functions, and built-in libraries in C++ programming.
- To perform calculations involving quadratic equations using the standard formula.
- To implement logic that checks if a triangle is valid and determines its type.
- To use string manipulation and character analysis for evaluating password strength.

Tools and Libraries Used:

- Programming Language: C++
- IDE: G++
- Libraries: **`#include <iostream>, include <string>, #include <math>`**

Theory:

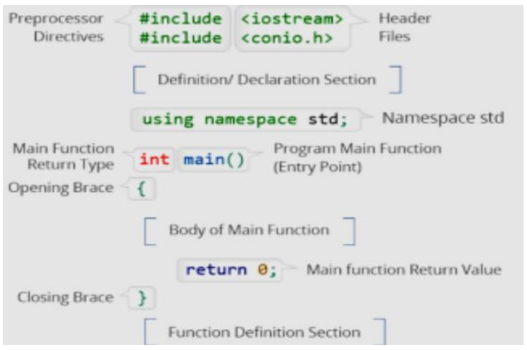
C++ Programming

C++ is a versatile language used to build efficient programs. Beginners start by learning variables, conditional statements, and loops to solve simple problems.

C++ Program Structure

A basic structure of C++ program consists of many key components arranged in a logical sequence. These parts are essential for the compilation and execution of C++ program. It includes header files such as '**`<iostream>`**' and it starts with **`main()`**, which is the entry point. '**`using namespace std`**' is included to access standard features easily.

Example:



```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{
return 0;
}
```

Preprocessor Directives
Header Files
Definition/ Declaration Section
Namespace std
Main Function Return Type
Program Main Function (Entry Point)
Opening Brace
Body of Main Function
Main function Return Value
Closing Brace
Function Definition Section

Declaration of Variables and data types

Variables are declared and assigned in the following ways:

```
int age; //declaration
age = 20;//initialization
int score = 100;  // Declaration and initialization
```

Some Common Data Types in C++:

- int – used for whole numbers (e.g: int x = 10;)
- float – used for decimal numbers (e.g: float f = 2.2;)
- double – used for more precise decimals (e.g: double d = 10.3527;)
- char – used for single characters (e.g: char c = 'B';)

Rules for variable naming

- Starting should be a letter or an underscore
- No digits at the beginning
- Space or special characters should not be used.
- Case-sensitive (num ≠ NUM)

Conditional Statements

Controls the flow of execution based on conditions.

if statement:

It is used to continue the flow through checking the condition.

Syntax:

```
 if (condition) {

    // Code executes if condition is true
}
```

if...else statement:

It is used for two different contitions. The code is executed as per the condition.

Syntax:

```
 if (condition) {

    // Code if true
} else {

    // Code if false
}
```

else…if ladder:

It is used for multiple conditions to be checked one after another simultaneously.

Syntax:

```
if (condition1) {

    // Code for condition1
} else if (condition2) {

    // Code for condition2
} else {

    // Default case
}
```

Switch Statement:

It is used when a variable is compared with multiple constant values.

Syntax:

```
switch (expression) {

  case value1:
    // Code for case 1
    break;

  case value2:
    // Code for case 2
    break;

  default:
    // Code if no case matches
}
```

Loops in C++

for Loop

It is used when the number of iterations are fixed.

Syntax:

```
for(initialization, condition, modification)
{ statements;
 }
```

while Loop

Used when the condition is checked before the loop body and the number of repetitions is not fixed.

Syntax:

```
while (condition) {
// code to repeat
}
```

do...while Loop

Runs the loop body at least once before checking the condition.

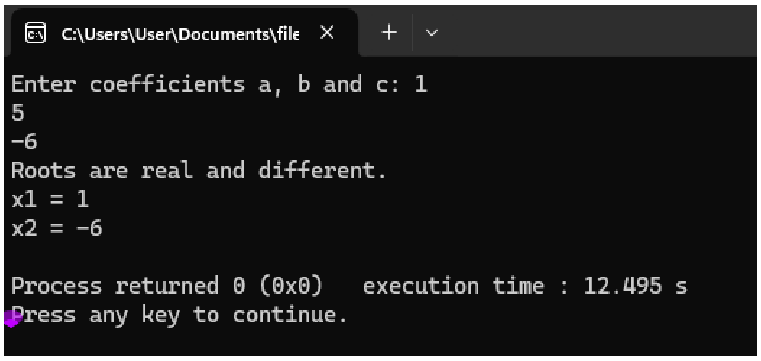
Syntax:

```
do {
// code to repeat
} while (condition);
```

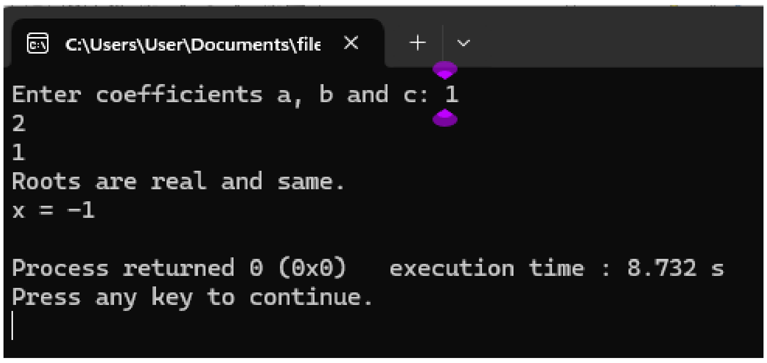
1. WAP to calculate the value of a variable from the given quadratic equation.

```
#include <iostream>

#include <cmath>

using namespace std;

int main() {

    double a, b, c;

    double discriminant, x1, x2;

    cout << "Enter coefficients a, b and c: ";

    cin >> a >> b >> c;

    if (a == 0) {

        cout << "This is not a quadratic equation (a cannot be 0)." << endl;

    }

    discriminant = b * b - 4 * a * c;

    if (discriminant > 0) {

        // Two real and distinct roots

        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) {

        // One real root

        x1 = -b / (2 * a);

        cout << "Roots are real and same." << endl;

        cout << "x = " << x1 << endl;

    }

    else {

        // Complex roots

        double realPart = -b / (2 * a);

        double imaginaryPart = sqrt(-discriminant) / (2 * a);

        cout << "Roots are complex and imaginary." << endl;

        cout << "x1 = " << realPart << " + " << imaginaryPart << "i" << endl;

        cout << "x2 = " << realPart << " - " << imaginaryPart << "i" << endl;

    }

    return 0;

}
```

Outputs:



```
Enter coefficients a, b and c: 1
5
-6
Roots are real and different.
x1 = 1
x2 = -6

Process returned 0 (0x0)   execution time : 12.495 s
Press any key to continue.
```

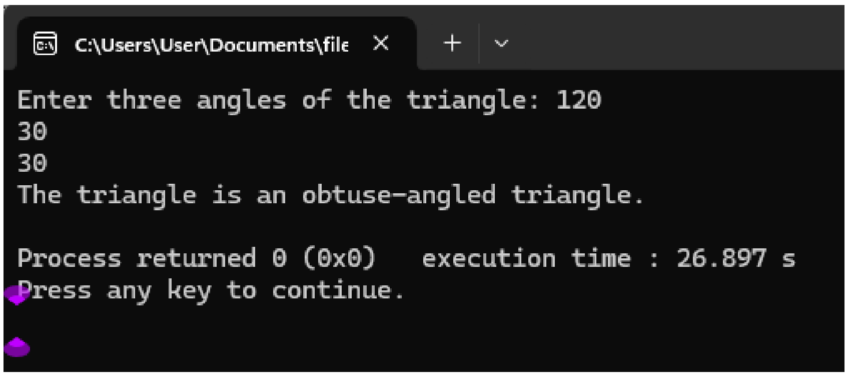


```
Enter coefficients a, b and c: 1
2
1
Roots are real and same.
x = -1

Process returned 0 (0x0)   execution time : 8.732 s
Press any key to continue.
```

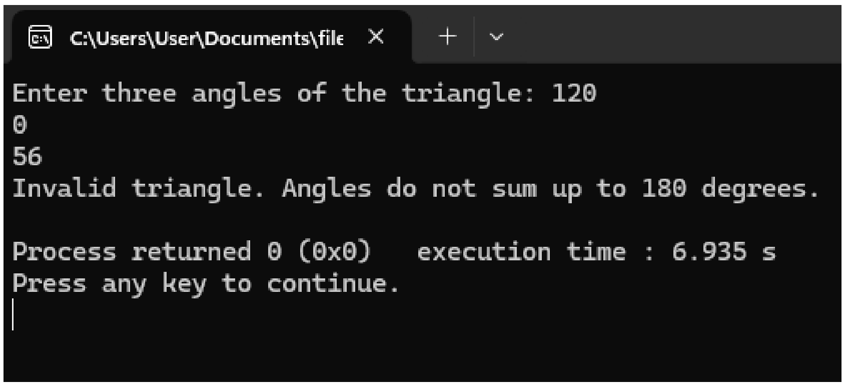
2. WAP to check whether the given angles form acute, obtuse or right angled triangle.

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

int main() {
   int angle1, angle2, angle3;
   cout << "Enter three angles of the triangle: ";
   cin >> angle1 >> angle2 >> angle3;

   int sum = angle1 + angle2 + angle3;
   // Check if the angles form a valid triangle
   if (sum != 180 || angle1 <= 0 || angle2 <= 0 || angle3 <= 0) {
      cout << "Invalid triangle. Angles do not sum up to 180 degrees." << endl;
   }
   else if (angle1 == 90 || angle2 == 90 || angle3 == 90) {
      cout << "The triangle is a right-angled triangle." << endl;
   }
   else if (angle1 > 90 || angle2 > 90 || angle3 > 90) {
      cout << "The triangle is an obtuse-angled triangle." << endl;
   }
   else {
      cout << "The triangle is an acute-angled triangle." << endl;
   }

   return 0;
}
```

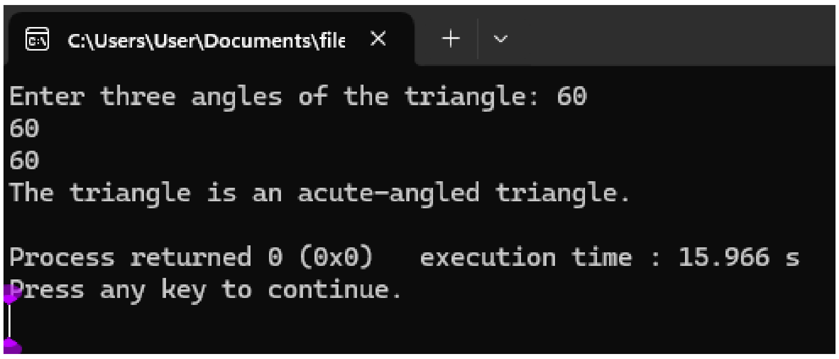
Output



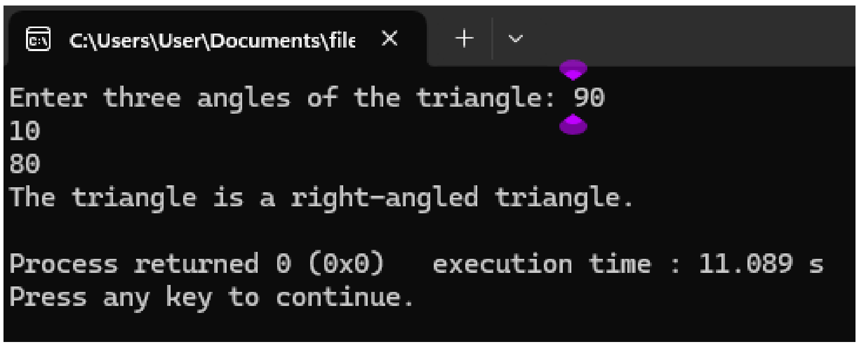
```
Enter three angles of the triangle: 120
30
30
The triangle is an obtuse-angled triangle.

Process returned 0 (0x0)   execution time : 26.897 s
Press any key to continue.
```



```
Enter three angles of the triangle: 120
0
56
Invalid triangle. Angles do not sum up to 180 degrees.

Process returned 0 (0x0)   execution time : 6.935 s
Press any key to continue.
```



```
Enter three angles of the triangle: 60
60
60
The triangle is an acute-angled triangle.

Process returned 0 (0x0)   execution time : 15.966 s
Press any key to continue.
```

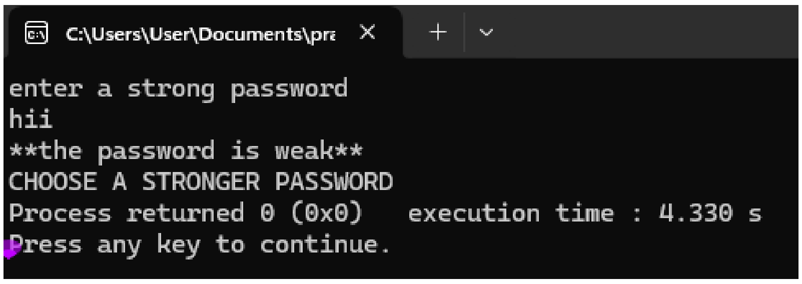


```
Enter three angles of the triangle: 90
10
80
The triangle is a right-angled triangle.

Process returned 0 (0x0)   execution time : 11.089 s
Press any key to continue.
```

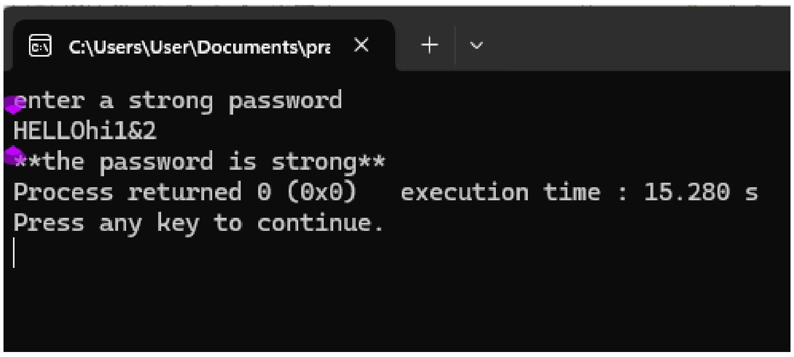
3. WAP to check whether the entered password is a strong password or not. For a password to be strong it must have numbers, special keys, alpha keys(both caps and small).

```
#include<iostream>
#include<string>
using namespace std;
//special characters
int scharacter(string password,int len)
{
   int temp1=0;
   while(len>=0)
   {

if(password[len]>=34&&password[len]<=47||password[len]>=58&&password[len]<=6
4||password[len]>=123&&password[len]<=126)
   {
      temp1++;
   }
   len--;
   }
   return temp1;
}
//caps letter
int caps(string password, int len)
{

   int temp2=0;
   while(len>=0)
   {

   if(password[len]>=65&&password[len]<=90)
   {
      temp2++;
   }
   len--;
   }
   return temp2;
}
//small letters
int small(string password, int len)
{

   int temp3=0;
   while(len>=0)
   {
   if(password[len]>=97&&password[len]<=122)
   {
      temp3++;
   }
   len--;
   }
   return temp3;
}
//numbers
int num(string password, int len)
{
   int temp4=0;
   while(len>=0)
   {
   if(password[len]>=48&&password[len]<=57)
   {
      temp4++;
   }
   len--;
   }
   return temp4;
}

int main()
{
   int len,a,b,c,d;
   string password;
   cout<<"enter a strong password"<<endl;
   cin>>password;
   len=password.length();
   a=scharacter(password,len);
   b=caps(password, len);
   c=small(password, len);
   d=num(password,len);
   if(a>=1&&b>=1&&c>=1&&d>=1)
   {
      cout<<"**the password is strong**";
   }
   else{
      cout<<"**the password is weak**"<<endl<<"CHOOSE A STRONGER
PASSWORD";
   }
   return 0;
}
```

Output:



```
C:\Users\User\Documents\pra

enter a strong password
hii
**the password is weak**
CHOOSE A STRONGER PASSWORD
Process returned 0 (0x0)   execution time : 4.330 s
Press any key to continue.
```



```
C:\Users\User\Documents\pra

enter a strong password
HELLOhi1&2
**the password is strong**
Process returned 0 (0x0)   execution time : 15.280 s
Press any key to continue.
```

Discussion:
Through this lab we were able to get familiar with the various types of loops and control statements such as if else, while, do-while, etc. Various problems were solved which required logical thinking along with the proper use of the control statements. The errors that occurred were solved by proper analysis through the code with the help of teachers.

Conclusion:
In this lab, we explored the fundamental concepts and practical applications of control statements in programming, including conditional statements (if, if-else, switch) and loops (for, while, do-while). These constructs are essential for directing the flow of a program based on specific conditions and for performing repetitive tasks efficiently. Through hands-on practice, we learned how to implement decision-making logic and control the iteration of code blocks.