TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

HIMALAYA COLLEGE OF ENGINEERING

CHYASAL, LALITPUR

Lab Report No: - 01

Title: - Quadratic Equations, Triangle Classification and Password Strength Checking

Submitted by: -

Name: - Diwas Pokhrel

Roll NO: -HCE081BEI014

Submitted To: -

Department of Electronics and Computer Engineering

Checked by: -

Date of submission: - 2025/05/25

Objectives:

- To understand and apply C++ control structures, functions, and standard libraries.
- To implement mathematical computations using the quadratic formula.
- To develop logic for triangle validation and classification.
- T apply string handling and character checking for password strength.

Tools and Libraries Used

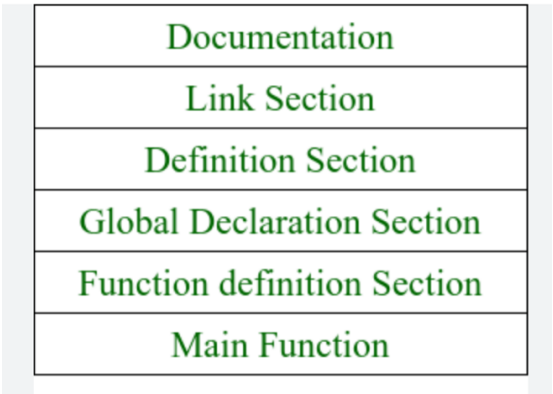
- Programming Language: C++
- IDE: Clyang
- Libraries: #include <iostream>, #include <cmath>, #include <string>

THEORY

INTRODUCTION

C++ is a powerful, high-performance programming language widely used for system/software development, game programming, and competitive coding. It supports **object-oriented programming (OOP), procedural programming,** and **generic programming**. C++ provides low-level memory manipulation while maintaining high-level abstractions, making it efficient for complex computations.

Structure of C++ Program:



Key features of C++:

- **Speed & Efficiency:** Compiles directly to machine code.
- **Rich Library Support (STL):** Includes data structures and algorithms.
- **Control Structures:** Like `if-else`, loops, and switch-case for decision-making.
- **Modularity:** Supports functions and classes for code reusability.

Conditional Statements in C++

In C++, **conditional statements** allow programs to make decisions based on certain conditions. The primary conditional structures are:

- **`if` statement** (single condition check).
- **`if-else` statement** (two possible paths).
- **`else-if` ladder** (multiple conditions).

These structures help control the flow of execution based on logical comparisons.

- **IF STATEMENT :**

SYNTAX:

```
if (condition) {
// Code executes if condition is true
}
```

- **IF-ELSE STATEMENT:**

SYNTAX:

```
if (condition) {
// Executes if condition is true  }
else {
/ Executes if condition is false
}
```

- **IF-ELSE LADDER**

SYNTAX:

```
if (condition1) {
/ Runs if condition1 is true  }
else if (condition2) {
```

```
// Runs if condition2 is true
}
else if (condition3) {
// Runs if condition3 is true
}
else {
// Default case (if all conditions fail)
}
```

- **NESTED IF-ELSE STATEMENT**

SYNTAX:

```
if (condition1) {
if (condition2) {
// Runs if both conditions are true
}
else {
// Runs if condition1 is true but condition2 is false
}
}
```

LOOPS IN C++

- **FOR LOOP**

```
for (initialization; condition; increment/decrement)
{
// code to be executed
}
```

- **while loop**

  Syntax:

```
while (condition) {

// code to be executed

}
```

- **do_while loop**

  Syntax:

```
do {

// code to be executed

} while (condition);
```

Certain Programs of C++ are:

Solve $ax^2+bx+c=0$ and handle all the discriminant cases

```
#include<iostream>
#include<cmath>
 using namespace std ;
 int main () {
double a , b , c , discriminant , root1 , root2 ;
cout << " Enter coefficients a, b, and c: ";
cin >> a >> b >> c ;

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

if ( discriminant > 0) {
 root1 = ( - b + sqrt ( discriminant ) ) / (2* a ) ;
 root2 = ( - b - sqrt ( discriminant ) ) / (2* a ) ;
 cout << " Real and distinct roots : " << root1 << " and "
<< root2 ;
 } else if ( discriminant == 0) {
 root1 = -b / (2* a ) ;
cout << " Real and equal roots : " << root1 ;
 } else {
double realPart = -b / (2* a ) ;
double imagPart = sqrt ( - discriminant ) / (2* a ) ;
cout << " Complex roots : " << realPart << " " <<
imagPart << "i";
}
```

The output of the program:

Output 1:

```
Output

Enter coefficients a, b, and c: 1 -5 6
Real and distinct roots : 3 and 2

=== Code Execution Successful ===
```

Output 2:

```
Output

Enter coefficients a, b, and c: 2 3 4
Complex roots : -0.75 1.19896i

=== Code Execution Successful ===
```

Output 3:

```
Output

Enter coefficients a, b, and c: 1 -4 4
Real and equal roots : 2

=== Code Execution Successful ===
```

2)Check if three angles form a triangle and classify it

```
#include<iostream>
using namespace std;
int main()
{
    int a,b,c;
    cout<<"Enter 1st angle: ";
    cin>>a;
    cout<<"Enter 2nd angle: ";
    cin>>b;
    cout<<"Enter 3rd angle: ";
    cin>>c;
    if(a+b+c==180)
    {
        if(a==90 || b==90 || c==90)
        {
            cout<<"Figure is right angled triangle. ";
        }
        else if(a>90 || b>90 || c>90)
        {
            cout<<"Figure is obtuse angled triangle. ";
        }
        else
        {
            cout<<"Figure is acute angled triangle. ";
        }
    }
    else
    {
        cout<<"Given angles don't form a triangle.";
    }
}
```

Output of the program

```
Enter 1st angle: 90
Enter 2nd angle: 90
Enter 3rd angle: 0
Figure is right angled triangle.

=== Code Execution Successful ===
```

```
Enter 1st angle: 95
Enter 2nd angle: 65
Enter 3rd angle: 20
Figure is obtuse angled triangle.

=== Code Execution Successful ===
```

```
Enter 1st angle: 50
Enter 2nd angle: 50
Enter 3rd angle: 80
Figure is acute angled triangle.

=== Code Execution Successful ===
```

Output

```
Enter 1st angle: 33
Enter 2nd angle: 22
Enter 3rd angle: 11
Given angles don't form a triangle.

=== Code Execution Successful ===
```

3) Password Strength Checker

```cpp
#include <iostream>
#include <string>
using namespace std;
class Password {
    int hasUpper = 0, hasSymbol = 0, hasLower = 0, hasNumber = 0;
public:
    int isUpper(char ch) {
        if (ch >= 65 && ch <= 90) return 1;
        return 0;
    }
    int isLower(char ch) {
        if (ch >= 97 && ch <= 122) return 1;
        return 0;}
    int isNumber(char ch) {
        if (ch >= 48 && ch <= 57) return 1;
        return 0; }
    int isSymbol(char ch) {
        if ((ch >= 33 && ch <= 47) ||(ch >= 58 && ch <= 64) || (ch >= 91 && ch <= 96) || (ch >= 123 && ch <= 126)) {
            return 1;
        }return 0;
    }
    int checkPass(string password) {
        if (password.length() <= 8) {
            return 0;
        }
    for (int i = 0; i < password.length(); i++) {
            char ch = password[i];
            if (isUpper(ch)==1) hasUpper = 1;
            else if (isLower(ch)==1) hasLower = 1;
            else if (isNumber(ch)==1) hasNumber = 1;
            else if (isSymbol(ch)==1) hasSymbol = 1;
        }
    if (hasUpper == 1 && hasLower == 1 && hasNumber == 1 && hasSymbol == 1)
            return 1;
        else
            return 0; }
};
int main() {
    Password pa;
    string pass;
    cout << "Enter password: ";
    cin >> pass;
     if (pa.checkPass(pass) == 1) {
        cout << "Password is strong." << endl;
    } else {
        cout << "Password is not strong." << endl;
    }
 return 0;
}
```

Ouput :

```
Output

Enter password: Easypass@1
Password is strong.


=== Code Execution Successful ===
```

```
Output

Enter password: notstrong
Password is not strong.


=== Code Execution Successful ===
```

Discussion:

This lab successfully implemented key C++ programming concepts through four practical applications. The quadratic equation solver demonstrated effective use of mathematical functions and conditional logic to handle different discriminant cases. The triangle classifier showed robust validation of geometric properties using logical operators, while the password checker illustrated comprehensive string manipulation and character analysis. Each program validated the theoretical concepts while revealing practical considerations - particularly in handling edge cases, optimizing conditional checks, and managing different data types. The exercises collectively strengthened understanding of core programming constructs and their real-world applications.

Conclusion:

Through these well-designed exercises, the lab achieved its objectives of reinforcing fundamental C++ programming skills. The implementation of mathematical computations, geometric validation, string analysis, and algorithmic generation provided comprehensive practice with control structures, functions, and standard libraries. These practical applications not only solidified theoretical knowledge but also developed problem-solving approaches essential for advanced programming. The lab successfully bridged classroom concepts with hands-on implementation, creating a strong foundation for future object-oriented programming studies and more complex software development challenges.