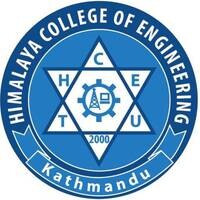


**TRIBHUVAN UNIVERSITY**

**INSTITUTE OF ENGINEERING**



**HIMALAYA COLLEGE OF ENGINEERING**

**CHYASAL, LALITPUR**

**Lab Report No: -**

**Title: -**

**Submitted by: - Submitted To: -**

**Name: - Atul Bhattarai Department Of Computer and Electronics**

**Roll NO: - HCOE 081 BEI 011 Checked by: -**

**Date of submission: -**

**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:**

# Basics of 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.

# Structure of a C++ Program

A basic C++ program includes header files (like **<**iostream**>**) and starts with main**()**, which is the entry point. The program uses using namespace std**;** to access standard features easily. The main function contains the code and ends with return 0**;** to indicate success.

Example:

#include<iostream>

using namespace std;

int main()

{

cout << "Hello world!";

return 0;

}

# Variables and Data Types

**Common Data Types:**

* int –intezer numbers (e.g., int x = 5;)
* float – decimal numbers (e.g., float pi = 3.14;)
* double – more precise decimals (e.g., double d = 2.718;)
* char – single characters (e.g., char c = 'A';)

**Variable Naming Rules**

* Start with a letter or underscore
* No digits at the beginning
* No space or special characters (except \_)
* Case-sensitive (Age ≠ age)

# Conditional Statements

Conditional statements control program flow based on conditions.

**if statement:**

Used when we must check the condition.

Syntax: if (condition)

{

Statement // Code runs if condition is true

}

**if...else statement:**

Used when we must check the condition and execute true and false condition separately.

Syntax: if (condition) {

Statement // Runs if true

}

else

{

Statement // Runs if false

}

**else….if ladder:**

Used when multiple conditions are to be checked one after another.

Syntax: if (condition1) {

// code if condition1 is true

} else if (condition2) {

// code if condition2 is true

} else if (condition3) {

// code if condition3 is true

} else {

// code if none are true

}

**switch Statement:**

Used to select one block of code from many options based on a variable's value.

Syntax: switch (expression) {

case value1:

// code for case 1

break;

case value2:

// code for case 2

break; ...

default:

// code if no cases match

}

# Loops in C++

**for Loop**

Used when the number of iterations is known.

for (initialization; condition; update) {

// code to repeat

}

Syntax:

1. Password strength checker

Source code:

#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;

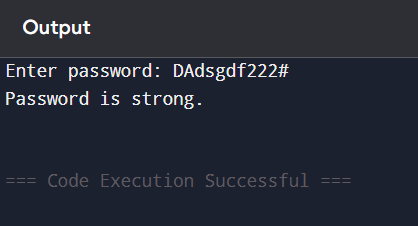
} else {

cout << "Password is not strong." << endl;

}

return 0;

}



1. Triangle classification

Source code:

#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. ";

}

}

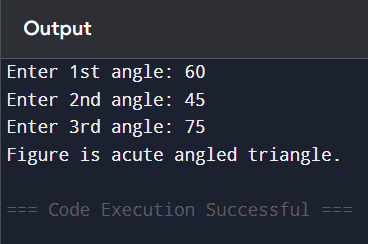
else

{

cout<<"Given angles don't form a triangle.";

}

}



1. Quadratic equation solver

Source code:

#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;

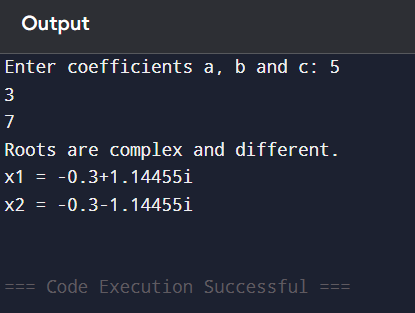
cout << "x1 = " << realPart << "+" << imaginaryPart << "i" << endl;

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

}

return 0;

}



**Discussion**

In this lab, I implemented C++ programs to solve quadratic equations, classify triangles, and check password strength. I practiced using control structures, loops, and string functions. The quadratic equation task helped me understand conditional logic and mathematical functions. Triangle classification taught me about validation and nested conditions. The password checker improved my skills in character analysis and string handling. Overall, this lab strengthened my understanding of basic C++ programming concepts.

**Conclusion**

The lab effectively enhanced my understanding of fundamental C++ programming through practical problem-solving. By implementing tasks like solving quadratic equations, validating and classifying triangles, and evaluating password strength, I developed a clearer grasp of control structures, logical conditions, and string operations. This hands-on experience not only strengthened my coding skills but also laid a solid groundwork for more advanced topics in object-oriented programming.