**UNIVERSITY INSTITUTE OF COMPUTING**

**PROJECT REPORT ON**

**SCIENTIFIC CALCULATOR**

Program Name: BCA

Subject Name/Code: Data Structures(23CAT-201)

Submitted by: Submitted to:

Name: Afjal khan Name: Mrs. Koniaka Rani

UID: 23BCA10246 DATE: 06/11/24

Section: BCA – 1 “A”

ABSTRACT

# Introduction:

A scientific calculator is a device or software tool used to perform a wide range of mathematical operations, from basic arithmetic to advanced scientific calculations like trigonometry, logarithms, exponentiation, and more. The goal of this project is to build a **Scientific Calculator** using **Data Structures and Algorithms (DSA)** in **C++**. The calculator will be able to handle both basic operations like addition, subtraction, multiplication, and division, as well as scientific functions such as sine, cosine, tangent, logarithmic operations, and power.

The main aim of this project is to demonstrate how fundamental data structures like stacks, queues, and linked lists can be used to implement operations and improve the overall efficiency and scalability of the program. Additionally, this project helps in understanding the application of algorithms for evaluating expressions and processing user input.

# System Requirements

For building and running the Scientific Calculator, the following system requirements should be met:

* **Operating System**: Windows, Linux, or macOS
* **Compiler**: A C++ compiler such as GCC or Visual C++
* **IDE/Editor**: Code::Blocks, Visual Studio Code, or any C++ compatible IDE
* **Memory**: Minimum 1 GB of RAM
* **Disk Space**: At least 50 MB of free space for storing the project files
* **C++ Version**: C++11 or above

# CODE

# #include <iostream>

# #include <cmath>

# #include <stack>

# #include <cctype>

# #include <string>

# #include <sstream>

# using namespace std;

# double performOperation(double a, double b, char op) {

# switch (op) {

# case '+': return a + b;

# case '-': return a - b;

# case '\*': return a \* b;

# case '/': return a / b;

# case '^': return pow(a, b);

# default: return 0;

# }

# }

# double evaluateExpression(string expression) {

# stack<double> values;

# stack<char> ops;

# for (int i = 0; i < expression.length(); i++) {

# 

# if (isdigit(expression[i])) {

# double value = 0;

# while (i < expression.length() && isdigit(expression[i])) {

# value = value \* 10 + (expression[i] - '0');

# i++;

# }

# values.push(value);

# i--;

# }

# else if (expression[i] == '+' || expression[i] == '-' ||

# expression[i] == '\*' || expression[i] == '/' ||

# expression[i] == '^') {

# while (!ops.empty() && (ops.top() == '\*' || ops.top() == '/' ||

# ops.top() == '^')) {

# double val2 = values.top(); values.pop();

# double val1 = values.top(); values.pop();

# char op = ops.top(); ops.pop();

# values.push(performOperation(val1, val2, op));

# }

# ops.push(expression[i]);

# }

# else if (expression[i] == '(') {

# ops.push(expression[i]);

# }

# else if (expression[i] == ')') {

# while (ops.top() != '(') {

# double val2 = values.top(); values.pop();

# double val1 = values.top(); values.pop();

# char op = ops.top(); ops.pop();

# values.push(performOperation(val1, val2, op));

# }

# ops.pop();

# }

# }

# while (!ops.empty()) {

# double val2 = values.top(); values.pop();

# double val1 = values.top(); values.pop();

# char op = ops.top(); ops.pop();

# values.push(performOperation(val1, val2, op));

# }

# return values.top();

# }

# int main() {

# string input;

# cout << "Scientific Calculator\n";

# cout << "Enter an expression: ";

# getline(cin, input);

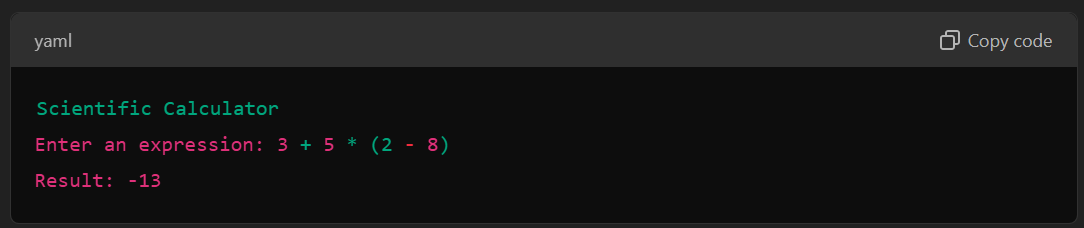
# cout << "Result: " << evaluateExpression(input) << endl;

# return 0;

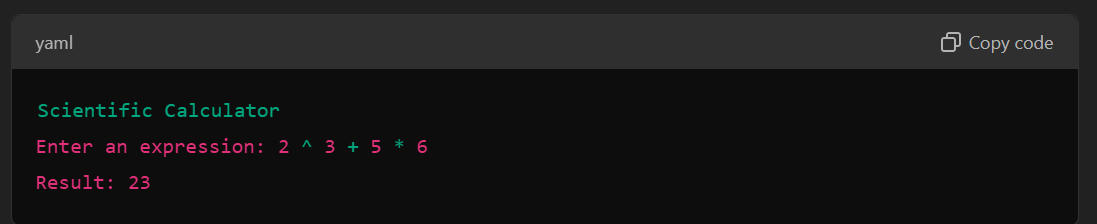
# }

# Output:

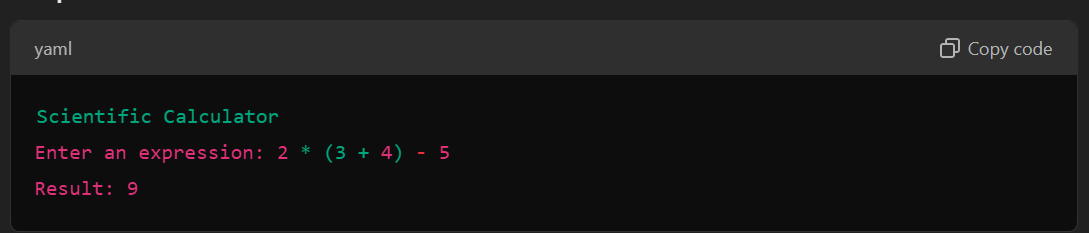
Sample 1:



Sample 2:



Sample 3:



New Account Creation:

