TECNIA INSTITUTE OF ADVANCED STUDIES
GRADE "A" INSTITUTE
Approved by AICTC, Ministry of HRD, Govt. of India, Affiliated to GGSIP University
Recognized Under Sec. 2(f) of UGC Act 1956
INSTITUTIONAL AREA MADHUBAN CHOWK,ROHINI,DELHI 110085
Tel:91-11-27555121-24, E-Mail : directortias@tecnia.in, Website: www.tiaspg.tecnia.in

EGAC Accredited 21001:2018 51001:2018

DEPARTMENT OF INFORMATION COMMUNICATION & TECHNOLOGY

MINI PROJECT
OF

BCA-108

Software Engineering

Academic Session: 2024-25

Batch: 2024-28

Submittedto:
Dr Anshu Arora
(Subject-In-Charge)

Submitted by:
Name: Preeti Raj
Enrolment No.:06617002024
Program: BCA
Semester: 2nd
Shift: Morning
Division: B

TECNIA INSTITUTE OF ADVANCED STUDIES
GRADE "A" INSTITUTE
Approved by AICTC, Ministry of HRD, Govt. of India, Affiliated to GGSIP University
Recognized Under Sec. 2(f) of UGC Act 1956
INSTITUTIONAL AREA MADHUBAN CHOWK,ROHINI,DELHI 110085
Tel:91-11-27555121-24, E-Mail : directortias@tecnia.in, Website: www.tiaspg.tecnia.in

EGAC Accredited 21001:2018 51001:2018

DEPARTMENT OF INFORMATION COMMUNICATION & TECHNOLOGY

Project Title:	Calculator
Subject:	Software Engineering
Session:	2024-25
Submitted to:	Dr Anshu Arora
Program & Sem:	BCA II-SEMESTER
Submitted by:	Preeti Raj, Vedika Aggarwal, Tanvi Taygi
Software Used:	Visual Studio

ABSTRACT

This project presents a simple calculator program written in the C programming language that performs basic arithmetic operations such as addition, subtraction, multiplication, division, and modulus. A key feature of this project is the use of a circular queue to store the last three results, allowing the user to refer back to previous calculations. The goal is to demonstrate both arithmetic logic and the implementation of an efficient data structure for limited memory storage. The project offers a practical example of applying fundamental C programming concepts in a structured, user-friendly way.

INTRODUCTION

In the digital age, calculators are essential tools for performing mathematical operations quickly and accurately. While most modern systems provide built-in calculators, understanding how to build one from scratch strengthens programming logic and problem-solving skills. This project aims to create a console-based calculator using the C language that performs basic operations and stores recent results using a circular queue. The calculator not only takes user input for calculations but also allows the user to continue operations with previous results. By incorporating a circular queue, the project also introduces an efficient way to manage a limited history of results, thereby simulating how memory and data can be handled effectively in real-world applications.

PROJECT OVERVIEW

This project focuses on the creation of a simple console-based calculator using the C programming language. The calculator not only performs basic arithmetic operations like addition, subtraction, multiplication, division, and modulus, but it also features a mechanism to store the most recent results using a circular queue of fixed size. This allows users to continue performing calculations by reusing the last result and review up to the last three calculations before the program exits.

The program is designed to accept two numbers and an arithmetic operator from the user, then process the expression and store the result in a circular queue. A circular queue is used instead of a normal array to efficiently manage memory and automatically overwrite the oldest results once the storage limit is reached. This type of data structure is particularly useful when working with limited storage space, as it avoids the need to shift data manually. In this project, the queue is of size 3, so it holds only the last three computed values.

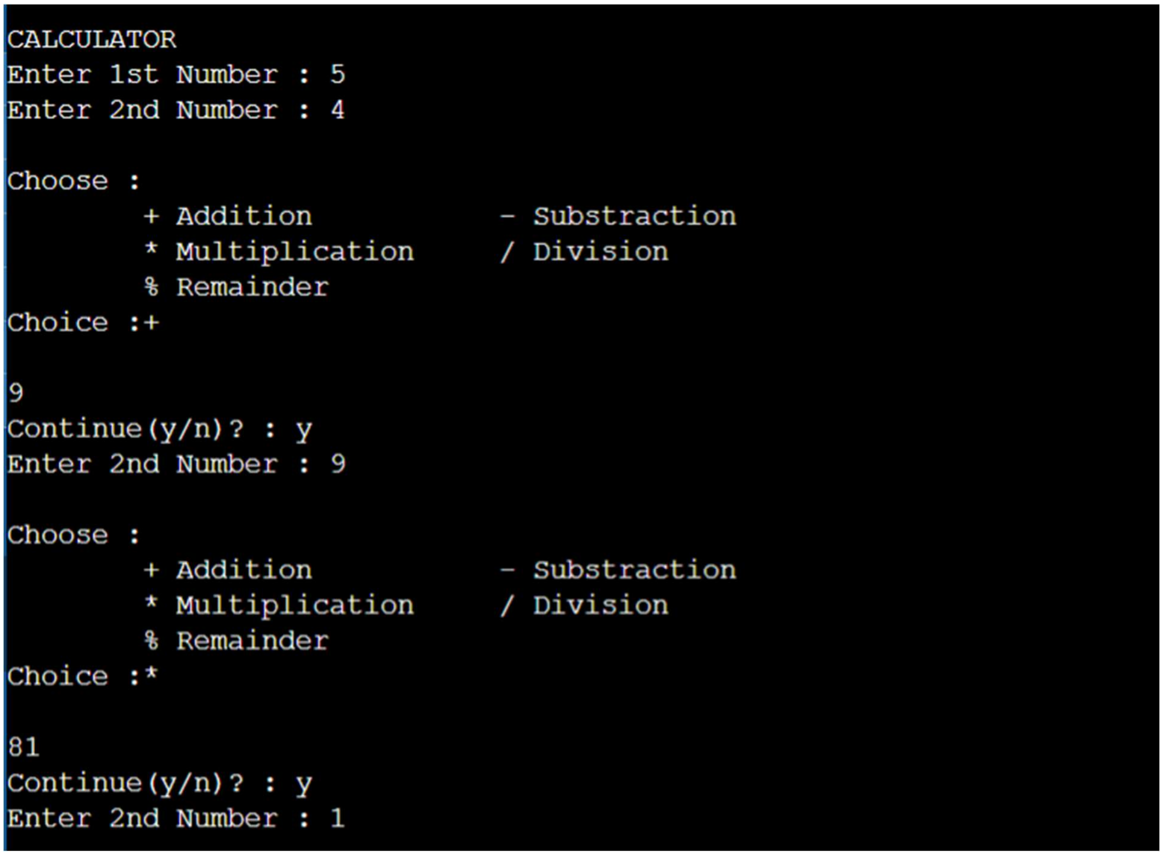
The logic of the calculator is handled through a switch-case statement inside a math() function that performs the desired operation. After each calculation, the result is stored using the insert() function, which updates the circular queue correctly, handling cases when the rear reaches the end and needs to wrap back to the beginning. The display() function is responsible for printing only the stored results in correct order, avoiding any unused or garbage values.

This project helped deepen the understanding of not just arithmetic logic and user input in C, but also introduced the practical usage of data structures like circular queues. It reinforces how circular queues work, how to avoid overflow, and how to control access to only valid data. The overall result is a functional calculator application that is lightweight, efficient, and educational.

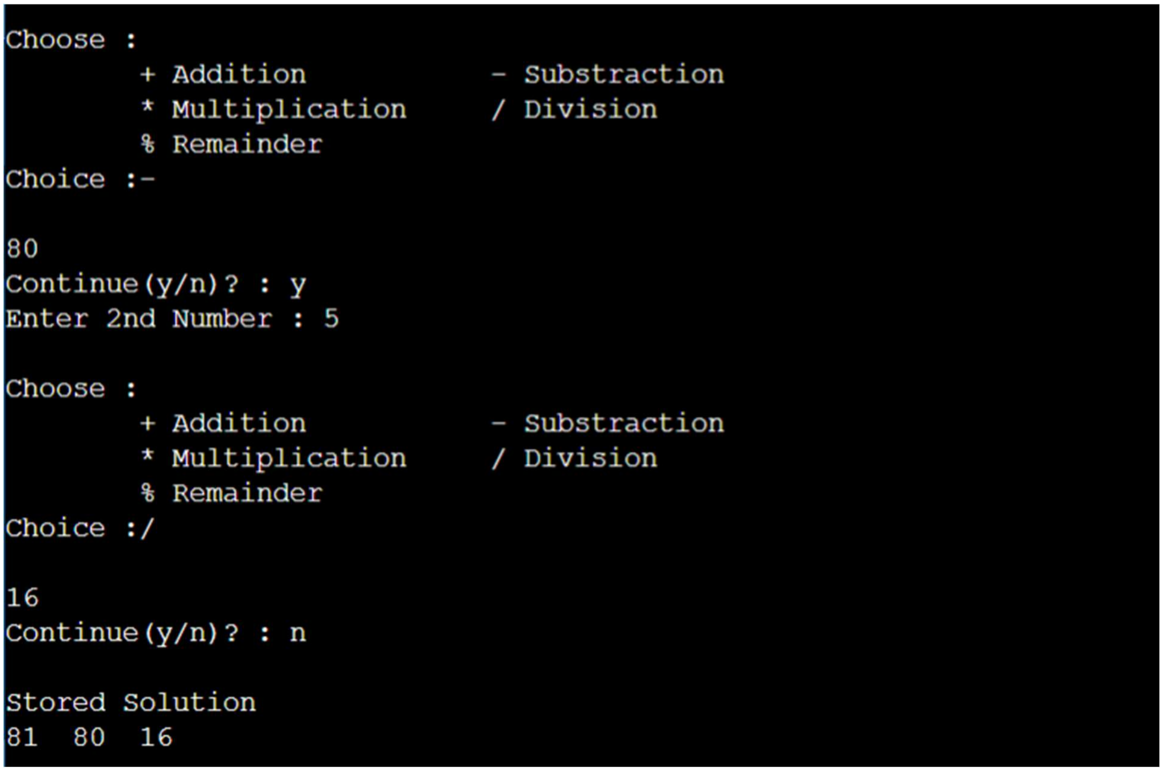
CODE

```
#include<stdio.h>
#include<stdlib.h>

int queue[3];
int front=-1;
int rear=-1;

void insert(int data){
    if(rear == -1 && front == -1){
        queue[++rear]=data;
        ++front;
    }
    else if(rear==2){
        rear=-1;
        queue[++rear]=data;
        ++front;
    }
    else{
        queue[++rear]=data;
    }
}

void display(){
    int step=front;
    while(step != 3){
    printf("%d  ",queue[step]);
    ++step;
    }
    if(front != 0){
        step=0;
        while(step <= rear){
            printf("%d  ",queue[step]);
            ++step;
        }
    }
}


int math(int first, int second, char sign){
    switch(sign){
        case '+':
            return first+second;
        case '-':
            return first-second;
        case '*':
            return first*second;
        case '/':
            return first/second;
        case '%':
            return first%second;
        default:
            printf("\nSomethings Wrong");
    }
}

int main(){
    int val1,val2;
    char sign,option;
    printf("\nCALCULATOR");
    printf("\nEnter 1st Number : ");
    scanf("%d",&val1);
    here:
    printf("Enter 2nd Number : ");
    scanf("%d",&val2);
    printf("\nChoose : \n\t+ Addition           - Substraction\n\t* Multiplication     /
Division\n\t%% Remainder\nChoice :");
    scanf(" %c",&sign);
    int result=math(val1,val2,sign);
    printf("\n%d",result);
    insert(result);
    printf("\nContinue(y/n)? : ");
    scanf(" %c",&option);
    if(option == 'y'){
        val1= result;
        goto here;
    }
    else{
        printf("\nStored Solution\n");
        display();
    }
}
```

OUTPUT



```
CALCULATOR
Enter 1st Number : 5
Enter 2nd Number : 4

Choose :
        + Addition           - Substraction
        * Multiplication     / Division
        % Remainder
Choice :+

9
Continue(y/n)? : y
Enter 2nd Number : 9

Choose :
        + Addition           - Substraction
        * Multiplication     / Division
        % Remainder
Choice :*

81
Continue(y/n)? : y
Enter 2nd Number : 1
```



```
Choose :
        + Addition           - Substraction
        * Multiplication     / Division
        % Remainder
Choice :-

80
Continue(y/n)? : y
Enter 2nd Number : 5

Choose :
        + Addition           - Substraction
        * Multiplication     / Division
        % Remainder
Choice :/

16
Continue(y/n)? : n

Stored Solution
81  80  16
```

LEARNING OUTCOMES

Through the development of this calculator project in C, several important programming concepts and skills were reinforced:
Understanding of Basic C Syntax: The project strengthened the foundation of C programming, including the use of variables, functions, loops, conditional statements, and switch-case logic.
User Input and Output Handling: You learned how to interact with users through console input/output using scanf and printf.
Implementation of Data Structures: A key learning point was the implementation of a circular queue, which taught how to manage memory efficiently and store a fixed number of recent results.
Problem-Solving and Debugging: Throughout the project, logical issues and edge cases were identified and corrected, improving analytical and debugging skills.
Use of Modular Programming: Breaking down the calculator logic into functions (like math(), insert(), and display()) helped in understanding code organization and modular design.

CONCLUSION

This project successfully demonstrates how a simple calculator can be developed using the C programming language, enhanced with the capability to store past results through a circular queue. The inclusion of such a data structure showcases efficient memory usage and introduces students to concepts commonly used in real-world applications such as buffer management and history tracking. Overall, the project helped in combining theoretical knowledge with hands-on coding, making it a valuable learning experience in both C programming and data structure implementation.

REFERENCES

- OpenAI. ChatGPT
- GeeksforGeeks.