Simple GUI Calculator with History

CCP Mini Project

PROJECT TEAM

1. Ishan Shastri
2. Kunj Sheth
3. Yug Shah

INTRODUCTION

Objective:

To develop a basic GUI-based calculator using C that performs arithmetic operations and maintains a history of calculations.

Technologies and Tools Used:

- Programming Language: C
- Library: Win32 API (for GUI development)
- Development Environment (IDE): Code::Blocks

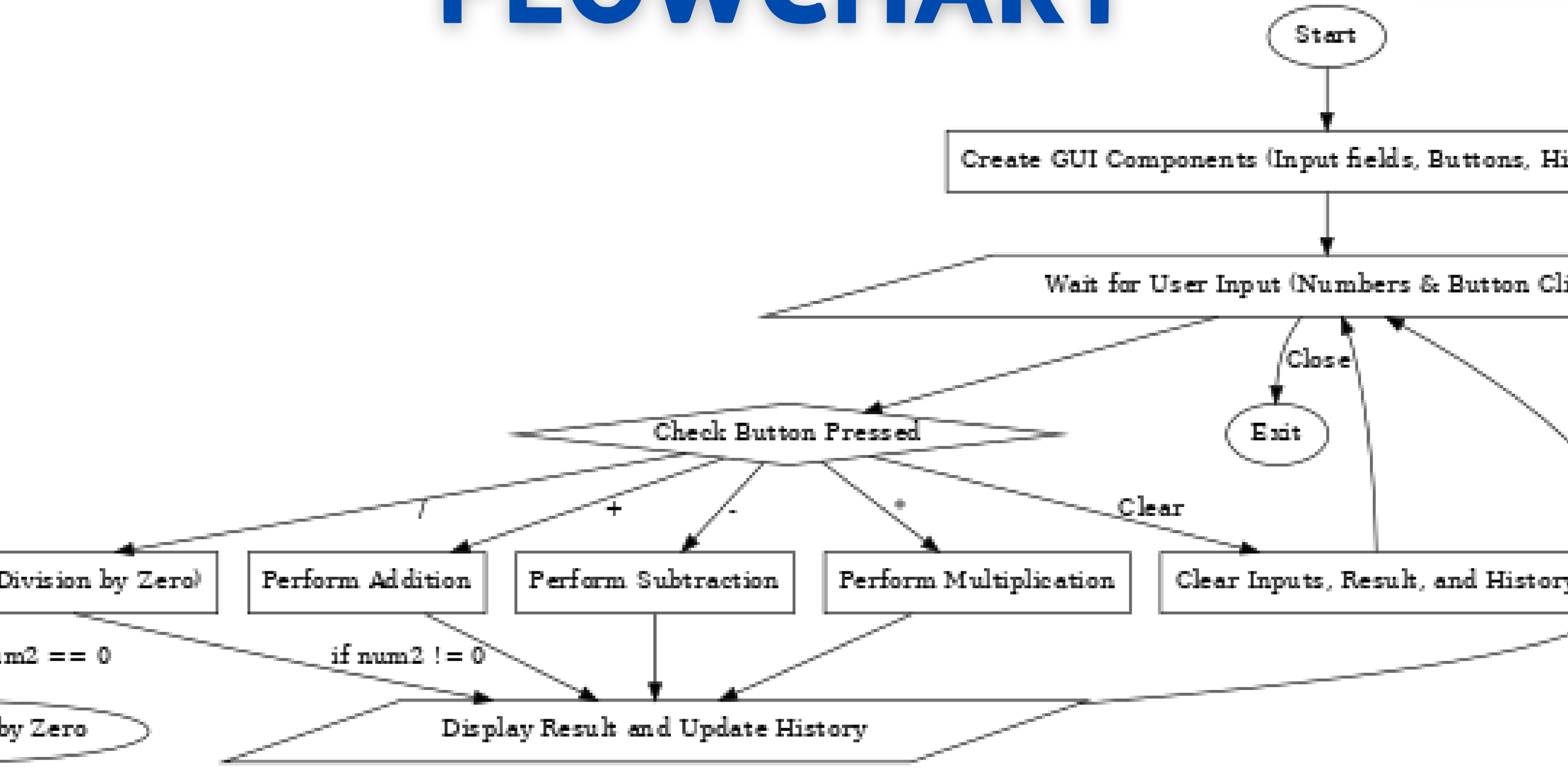
FEATURES

- Perform basic arithmetic operations (Add, Subtract, Multiply, Divide)
- Real-time input validation (numbers are only allowed)
- Division error handling (e.g., Cannot divide by zero)
- Clear/reset functionality - Which can clear all history and input and arithmetic operator selected.
- History of all the calculations performed can be viewed before until the clear option is selected.

KEY CONCEPTS USED

- GUI Components (Buttons, Text Boxes, Static Text, List Box)
- Message Handling (Win32 API)

FLOWCHART


Start
Create GUI Components (Input fields, Buttons, Hi
Wait for User Input (Numbers & Button Cli
Close
Exit
Check Button Pressed
/
+
-
*
Clear
Division by Zero)
Perform Addition
Perform Subtraction
Perform Multiplication
Clear Inputs, Result, and History
m2 == 0
if num2 != 0
by Zero
Display Result and Update History

CODE

1

```
 <windows.h>
 <stdio.h>

BUTTON_ADD      1
BUTTON_SUBTRACT 2
BUTTON_MULTIPLY 3
BUTTON_DIVIDE   4
BUTTON_EQUALS   5
BUTTON_CLEAR    6
BUTTON_HISTORY  7

CALLBACK WindowProc(HWND, UINT, WPARAM, LPARAM);

put1, hInput2, hResult, hListHistory;
tory[1024] = "";

PI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow
t char CLASS_NAME[] = "SimpleCalculator";

LASS wc = { };
pfnWndProc = WindowProc;
Instance = hInstance;
pszClassName = CLASS_NAME;

sterClass(&wc);

 hwnd = CreateWindowEx(0, CLASS_NAME, "Simple Calculator with History",
                       WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT,
                       300, 400, NULL, NULL, hInstance, NULL);

hwnd == NULL) return 0;

Window(hwnd, nCmdShow);

msg = { };
e (GetMessage(&msg, NULL, 0, 0)) {
TranslateMessage(&msg);
DispatchMessage(&msg);


rn 0;
```

2

```
ddToHistory(const char *entry) {
:cat(history, entry);
:cat(history, "\r\n");
ndMessage(hListHistory, LB_ADDSTRING, 0, (LPARAM)entry);


alculateAndDisplay(HWND hwnd, int operation) {
ar buffer1[256], buffer2[256], resultText[256];
uble num1, num2, result;

:WindowText(hInput1, buffer1, sizeof(buffer1));
:WindowText(hInput2, buffer2, sizeof(buffer2));

n1 = atof(buffer1);
n2 = atof(buffer2);

itch (operation) {
  case BUTTON_ADD:      result = num1 + num2; sprintf(resultText, "%.2f + %.2f = %.2f", num1, num2, result); b
  case BUTTON_SUBTRACT: result = num1 - num2; sprintf(resultText, "%.2f - %.2f = %.2f", num1, num2, result); b
  case BUTTON_MULTIPLY: result = num1 * num2; sprintf(resultText, "%.2f * %.2f = %.2f", num1, num2, result); b
  case BUTTON_DIVIDE:
      if (num2 == 0) {
          MessageBox(hwnd, "Cannot divide by zero", "Error", MB_OK | MB_ICONERROR);
          return;
      }
      result = num1 / num2;
      sprintf(resultText, "%.2f / %.2f = %.2f", num1, num2, result);
      break;
  default: return;


:WindowText(hResult, resultText);
dToHistory(resultText);


 CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam) {
itch (uMsg) {
  case WM_CREATE:
      hInput1 = CreateWindow("EDIT", "", WS_CHILD | WS_VISIBLE | WS_BORDER | ES_NUMBER,
                             20, 20, 100, 25, hwnd, NULL, NULL, NULL);
      hInput2 = CreateWindow("EDIT", "", WS_CHILD | WS_VISIBLE | WS_BORDER | ES_NUMBER,
                             150, 20, 100, 25, hwnd, NULL, NULL, NULL);
```

CODE

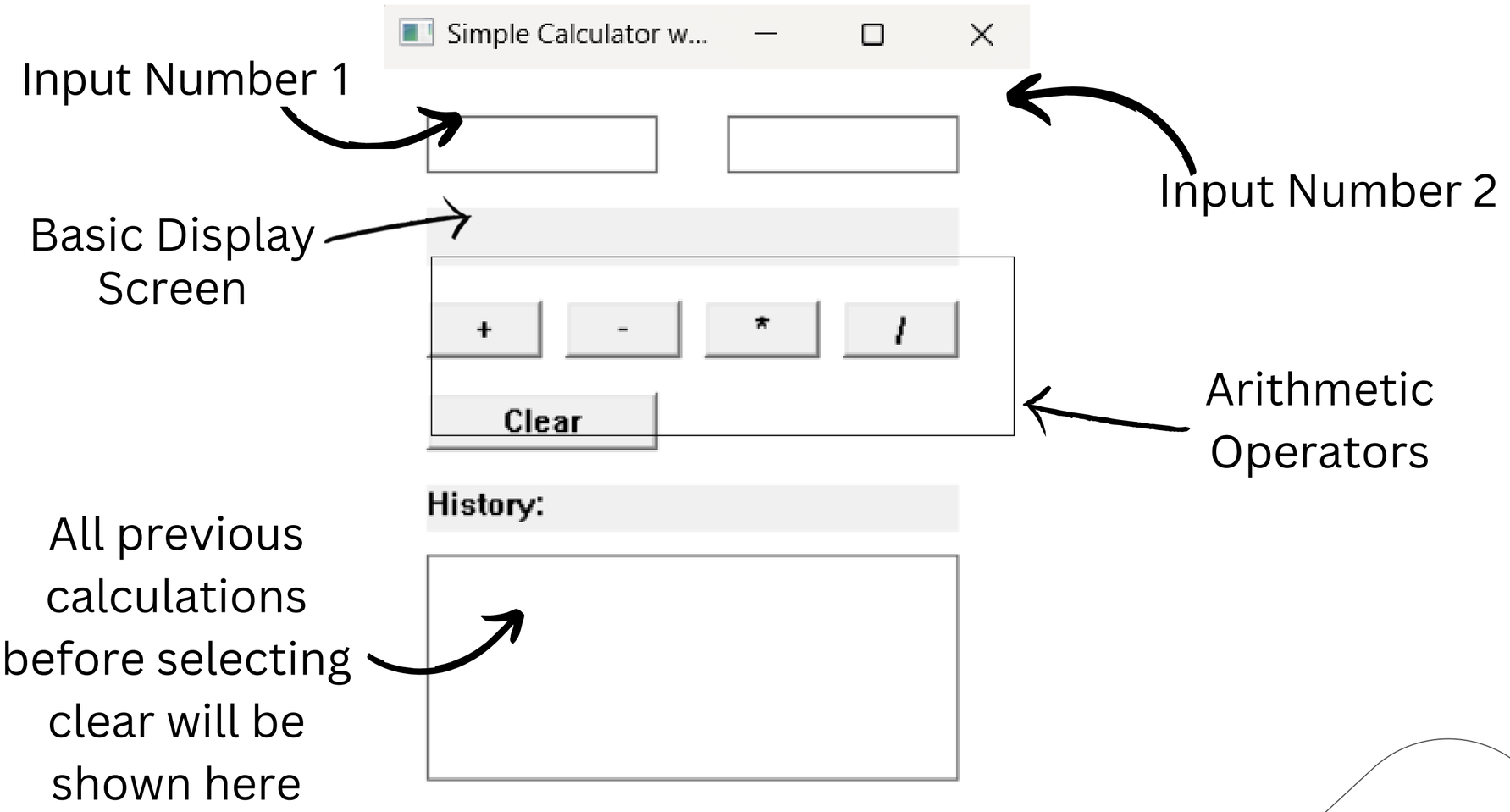
3

```
            hResult = CreateWindow("STATIC", "", WS_CHILD | WS_VISIBLE,
                                   20, 60, 230, 25, hwnd, NULL, NULL, NULL);

            CreateWindow("BUTTON", "+", WS_CHILD | WS_VISIBLE, 20, 100, 50, 25, hwnd, (HMENU) BUTTON_ADD, NULL, NULL);
            CreateWindow("BUTTON", "-", WS_CHILD | WS_VISIBLE, 80, 100, 50, 25, hwnd, (HMENU) BUTTON_SUBTRACT, NULL, NULL);
            CreateWindow("BUTTON", "*", WS_CHILD | WS_VISIBLE, 140, 100, 50, 25, hwnd, (HMENU) BUTTON_MULTIPLY, NULL, NULL);
            CreateWindow("BUTTON", "/", WS_CHILD | WS_VISIBLE, 200, 100, 50, 25, hwnd, (HMENU) BUTTON_DIVIDE, NULL, NULL);
            CreateWindow("BUTTON", "Clear", WS_CHILD | WS_VISIBLE, 20, 140, 100, 25, hwnd, (HMENU) BUTTON_CLEAR, NULL, NULL);

            CreateWindow("STATIC", "History:", WS_CHILD | WS_VISIBLE, 20, 180, 230, 20, hwnd, NULL, NULL, NULL);
            hListHistory = CreateWindow("LISTBOX", NULL, WS_CHILD | WS_VISIBLE | WS_BORDER | WS_VSCROLL,
                                        20, 210, 230, 100, hwnd, (HMENU) BUTTON_HISTORY, NULL, NULL);
            break;

        case WM_COMMAND:
            switch (LOWORD(wParam)) {
                case BUTTON_ADD: case BUTTON_SUBTRACT:
                case BUTTON_MULTIPLY: case BUTTON_DIVIDE:
                    CalculateAndDisplay(hwnd, LOWORD(wParam));
                    break;
                case BUTTON_CLEAR:
                    SetWindowText(hInput1, "");
                    SetWindowText(hInput2, "");
                    SetWindowText(hResult, "");
                    SendMessage(hListHistory, LB_RESETCONTENT, 0, 0);
                    strcpy(history, "");
                    break;
            }
            break;

        case WM_DESTROY:
            PostQuitMessage(0);
            break;

        default:
            return DefWindowProc(hwnd, uMsg, wParam, lParam);
    }
    return 0;
}
```

CODE HIGHLIGHTS

- ‘CalculateAndDisplay()’ for handling operations and displaying results
- ‘AddToHistory()’ for dynamically updating the history
- ‘WM_CREATE’ and ‘WM_COMMAND’ for GUI component creation and event handling

CALCULATOR DESCRIPTION



LIVE DEMONSTRATION

CONCLUSION

- Successfully created a simple calculator using GUI with history.
- From this project, we learnt about GUI programming and Win32 API.

THANK YOU