# MATRIX CALCULATOR - AN EFFICIENT SOFTWARE PROGRAM FOR PERFORMING QUICK AND PRECISE MATHEMATICAL OPERATIONS ON MATRICES

Submitted in partial fulfillment of the requirements of

**University of Mumbai**

For the Degree of

## Bachelor of Engineering in CSE (AIML)

Submitted by

## MR. VIVEK KUMBHAR [ROLL NO.: 21]

## MR. NIMESH KUTHE [ROLL NO.: 23]

**MR. SUFIYAN CHOUGULE [ROLL NO.: 06]**

**MR. SATYAM SINGH [ROLL NO.:59]**

Under the guidance of

## PROF. BHAGYALAKSHMI



**DEPARTMENT OF CSE (AIML/IOT)**

**SMT. INDIRA GANDHI COLLEGE OF ENGINEERING**

Ghansoli, Navi Mumbai - 400701

## Academic Year: 2022-2023

**Project Report Approval for S.E.**

This project report entitled **“Matrix Calculator –** An efficient software program for performing quick and precise mathematical operations on matrices**.”**

By

**Mr. Vivek Kumbhar** (Roll No.: 21)

**Mr.Nimesh Kuthe** (Roll No.: 23)

**Mr. Sufiyan Chougule** (Roll No.: 06)

**Mr. Satyam Singh** (Roll No.: 59)

are approved for the degree of Bachelor of Engineering in Computer Engineering, Semester III, University of Mumbai.

Examiner 1 Examiner 2 Prof. Bhagyalakshmi Internal Guide

Prof. Sonali.Deshpande Dr. Sunil Chavan

Head of Department Principal

Date:

Place: Ghansoli, Navi Mumbai.

## Declaration

We declare that this written submission represents our own ideas in our own words and where others ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any act/data/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

****

--------------------------------------------------------

Mr. Vivek Kumbhar [Roll No.: 21]

****

--------------------------------------------------------

Mr. Nimesh Kuthe [Roll No.: 23]

****

Mr. Sufiyan Chougule [Roll No.: 06]

****

Mr. Satyam Singh [Roll No.: 59]

Date:

Place: Ghansoli, Navi Mumbai.

## Abstract

Title: Matrix Calculator – Anefficient **software program** for performing quick and precise mathematical operations on matrices.

When large numbers are involved in the already complex calculation methods for matrices, an average human mind is very likely to run into errors or complications such as low calculation speed, imprecise methodology and inaccurate results.

By means of a menu-driven matrix calculator, the user can get accurate results for the chosen matrix operation in matter of seconds. This not only eliminates possibility of error and duration of calculation but provides the user a structured interface to work with to exactly describe the required matrices. Unlike a simple calculator, the user doesn’t have to perform individual operations and will just get the final answer after providing the correct input in the calculator interface.

The Matrix Calculator can be used by Students to cross-check their self-calculated answers while understanding and performing matrix calculations.

It can also be of use to the examiners to verify a student’s answers accurately and in a short time span.

## List of Abbreviations

* B.E.: Bachelor of Engineering
* DFD: Data Flow Diagram
* VS Code: Visual Studio Code
* IDE: Integrated Development Environment
* GCC: GNU Compiler Collection
* UNIX: UNiplexed Information Computing System
* GUI: Graphical User Interface

## List of Figures

* Fig. 2.1 VS Code
* Fig. 2.2 Basic C Program
* Fig. 2.3 GitHub Homepage
* Fig. 4.5 Data Flow Diagram
* Fig. 5.1 Flow Chart
* Fig. 6.2.1 Main Code
* Fig. 6.2.2 Display Output

## List of Tables

* Table 3.1 Timeline Chart

**Index**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Chapter No.** | **Content** | | | **Page No.** |
| **I** | **Abstract** | | | **i** |
| **II** | **List of Abbreviations** | | | **ii** |
| **III** | **List of Figures & Tables** | | | **iii** |
| **1** | **Introduction** | | | **1** |
|  | 1.1 | | Problem Statement | 2 |
|  | 1.2 | | Objectives | 2 |
|  | 1.3 | | Scope | 2 |
|  | 1.4 | | Report Organization | 2 |
| **2** | **Methodology** | | | **3** |
|  | 2.1 | Visual Studio Code (VS Code) | | 4 |
|  | 2.2 | C Programming Language | | 5 |
|  | 2.3 | GitHub | | 6 |
| **3** | **Planning and Formulation** | | | **7** |
|  | 3.1 | | Timeline Chart | 8 |
|  | 3.2 | | Feasibility Analysis | 9 |
| 4 | **Requirement Analysis** | | | 10 |
|  | 4.1 | | Hardware Requirements | 11 |
|  | 4.2 | | Software Requirements | 11 |
|  | 4.3 | | Functional Requirements | 11 |
|  | 4.4 | | Non-functional Requirements | 11 |
|  | 4.5 | | Data Flow Diagram | 12 |
| **5** | **System Design** | | | **13** |
|  | 5.1 | | Flow Chart | 14 |
| **6** | **Development** | | | **15** |
|  | 6.1 | | Source Code | 16 |
|  | 6.2 | | Main Code & Display Output | 25 |
| **7** | **Conclusion & Future Scope** | | | **26** |
|  | 7..1 | | Conclusion | 27 |
|  | 7..2 | | Future Scope | 27 |
|  | **Acknowledgement** | | | **28** |
|  | **References** | | | **29** |

# INTRODUCTION

## Introduction

Whenever a certain mathematical concept is to be learnt, it is important for the learner to solve problems based on the concept as much as it is to learn the concept itself. However, one cannot be sure of the resulting answers every time since there is always a possibility of human error in any calculation performed by humans. Also, the amount of time required for accurately performing complex calculations keeps on increasing with the length of the problem. Thus, it becomes clear that a medium or tool is necessary to guarantee the correct answers within a short span of time. In this case, we are taking the calculation of matrices and different operations to be performed on them as a context to the previously implied idea.

### Problem Statement

Calculators are used almost everywhere in the present day. Taking into consideration the idea of a calculator which provides accurate answers quickly, we are to develop a calculator that consists of operations which are to be performed on matrices. Thus, we will have to design a program that can take matrices as input data and provide the required output based on the operations selected by the user.

### Objectives

1. Perform Addition of 2 Matrices as per user input.

2. Perform Subtraction of 2 Matrices as per user input.

3. Perform Multiplication of 2 Matrices as per user input.

4. Find and Display Transpose of a Matrix.

5. Calculate Determinant of a Square Matrix.

6. Find the Inverse of a Non-Singular Matrix.

### Scope

* + 1. The Matrix Calculator can be used by Students to cross-check their self-calculated answers while understanding and performing matrix calculations.
    2. It can also be of use to the examiners to verify a student’s answers accurately and in a short time span.

### Report Organization

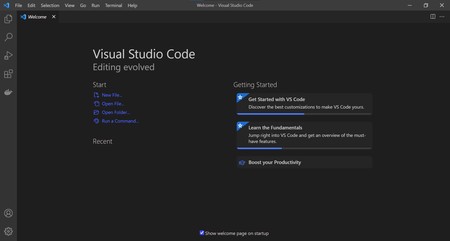
* **Methodology.**
* **Planning and Formulation**.
* **Requirement Analysis**
* **System Design**
* **Development**
* **Conclusion**

# METHODOLOGY

## Methodology

* 1. **Visual Studio Code (VS Code)**

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE. At its heart, Visual Studio Code features a lightning fast source code editor, perfect for day-to-day use. With support for hundreds of languages, VS Code helps you be instantly productive with syntax highlighting, bracket-matching, auto-indentation, box-selection, snippets, and more [1].



### Fig 2.1 Visual Studio

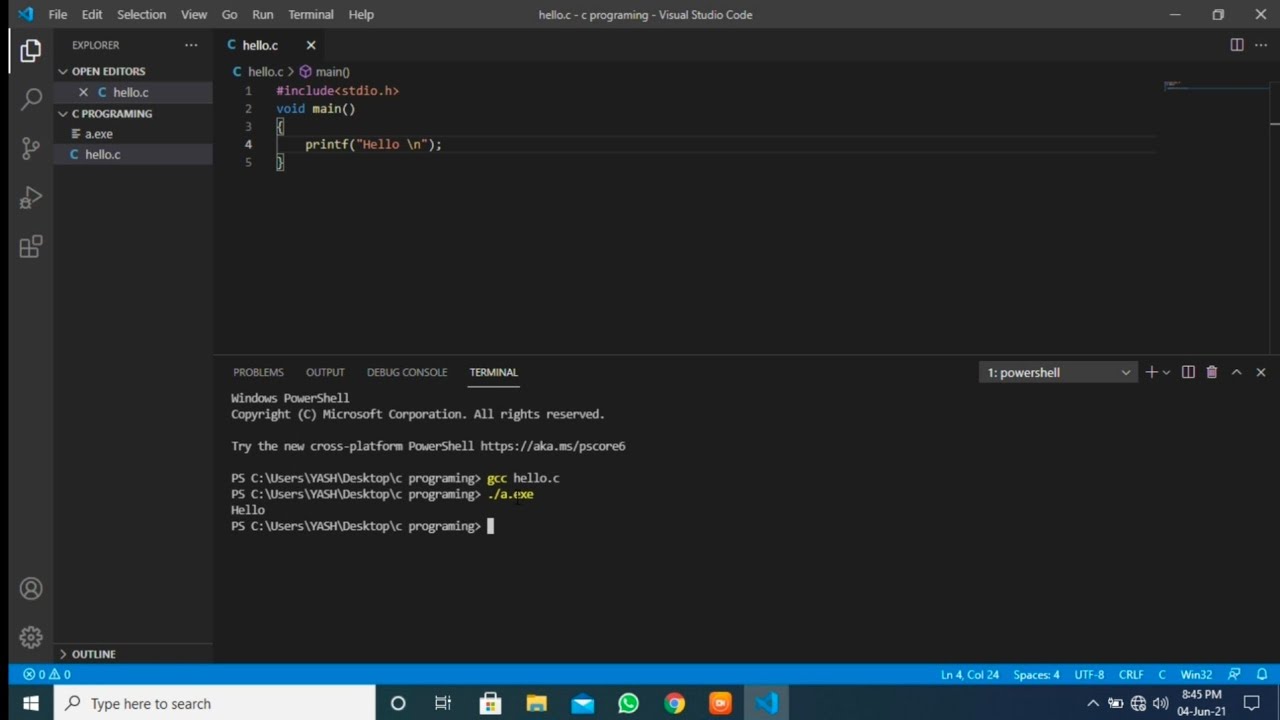
* 1. **C Programming Language**

C is a general-purpose, high-level language that was originally developed by Dennis M. Ritchie to develop the UNIX operating system at Bell Labs. C was originally first implemented on the DEC PDP-11 computer in 1972.

In 1978, Brian Kernighan and Dennis Ritchie produced the first publicly available description of C, now known as the K&R standard.

The UNIX operating system, the C compiler, and essentially all UNIX application programs have been written in C. C has now become a widely used professional language for various reasons –

* Easy to learn
* Structured language
* It produces efficient programs
* It can handle low-level activities
* It can be compiled on a variety of computer platforms [2].



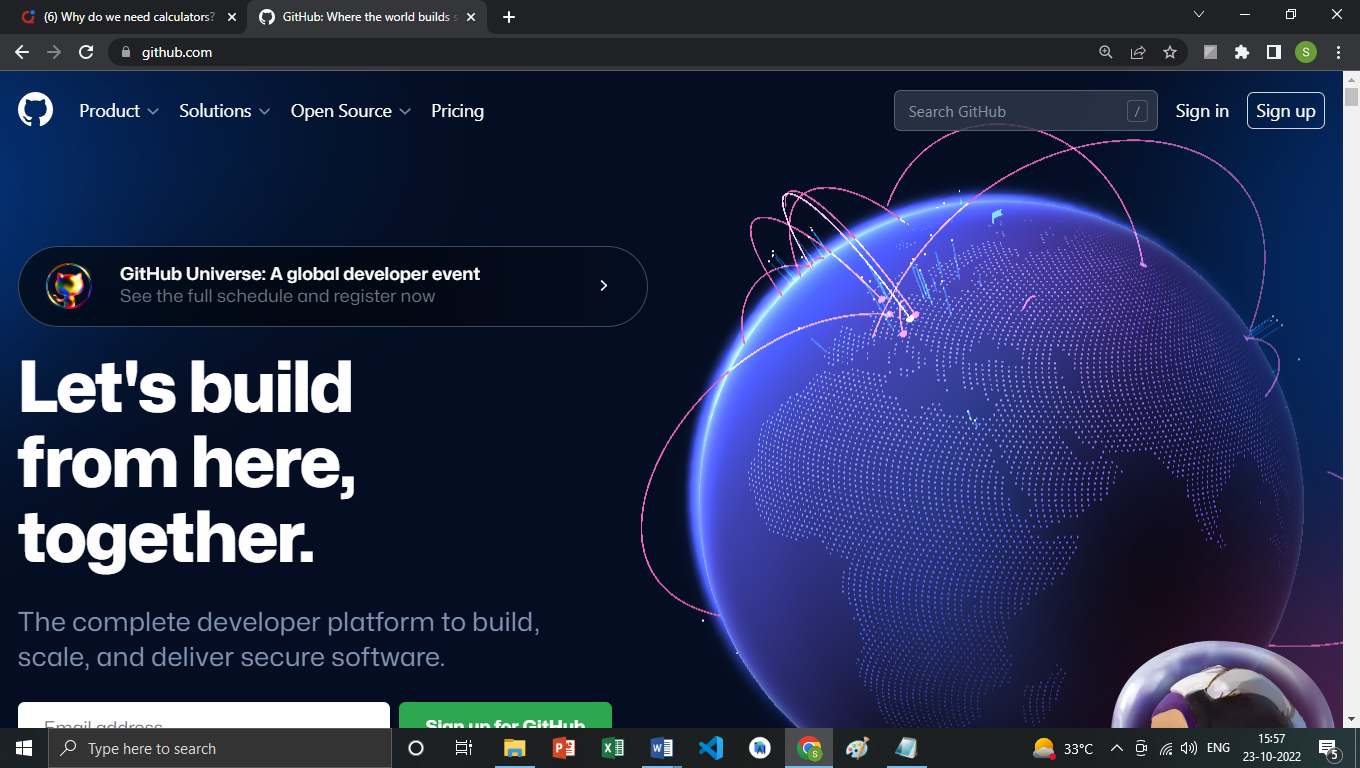
**Fig. 2.2 Basic C Program**

**2.3 GitHub**

## What is GitHub?

[GitHub](https://github.com/) is a web-based interface that uses [Git](https://git-scm.com/), the open source version control software that lets multiple people make separate changes to web pages at the same time. As Carpenter notes, because it allows for real-time collaboration, GitHub encourages teams to work together to build and edit their site content.

GitHub allows multiple developers to work on a single project at the same time, reduces the risk of duplicative or conflicting work, and can help decrease production time. With GitHub, developers can build code, track changes, and innovate solutions to problems that might arise during the site development process simultaneously [3].



**Fig. 2.3 GitHub Homepage**

# PLANNING AND FORMULATION

### Timeline Chart

**Table 3.1 Timeline Chart**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Timeline Chart for the project Matrix Calculator – An efficient software program for performing quick and precise mathematical operations on matrices.** | | | | | | | | | | | | | | | | |
| **Months** | **August** | | | | **September** | | | | **October** | | | | **November** | | | |
| **Phases** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** |
| Requirements Analysis |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Feasibility Study |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Design and Architecture |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Product Development |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Implementation on |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Testing |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Report Organization |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

### Feasibility Analysis

Next step in analysis is feasibility study. By performing a feasibility study the scope of the system will be defined completely. Most computer systems are developed to satisfy a known user requirement. This means that the first event in the life cycle of a system is usually the task of studying whether it is feasible to computerize a system under consideration or not. Once the decision is made, a report is forwarded and it is known as Feasibility Report. The feasibility is studied under the following contexts:

### Technical Feasibility

It involves determining whether or not a system can actually be constructed to solve the problem at hand. The technical issues raised during the feasibility stage of investigation are related to achievability of project’s goal and possibility of completion of project.

### Economical Feasibility:

This feasibility deals with the cost/benefit analysis. A number of intangible benefits like user friendliness, robustness and security were pointed out. The cost that will be incurred upon the implementation of this project would be quite nominal.

### Operational Feasibility:

The developed system will be very reliable and user friendly. All the features and operations that we will implement in our project are possible to implement and thus feasible. This will facilitate easy use and adoptability of the system. With the use of menus, and proper validation required it becomes fully understandable to the common user and operational with the user [4].

# REQUIREMENTS ANALYSIS

## Requirements Analysis

### Hardware Requirement

* + 1. Processor: 2.6 GHz or faster
    2. Hard Disk: 2 GB or more
    3. RAM: 4 GB or above
    4. Computer / Laptop

### Software Requirement

* + 1. Operating System : Windows 7 or above
    2. Visual Studio
    3. GCC Compiler (MinGW)

### Functional Requirement

* + 1. In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish.
    2. Since, our software is developed in plain C language and does not contain a GUI, it isn’t necessary to classify it into a front and back end as it operates in the command line window.
    3. The user-input will be accepted in the command line window via keyboard based on which other results and output screens will be displayed.

### Non-Functional Requirements

* + 1. In system engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors.
    2. This should be contrasted with functional requirements so that our design can be more effective and more beneficial for the user to use our application.
    3. To make our software more user-friendly, we thought of designing a GUI. However, due to some discrepancies and our lack of experience in building a GUI, we were unable to do so.

### Data Flow Diagram

Output

Keyboard Input

Display Text

User

Database

Fig. 4.5 Data Flow Diagram

## SYSTEM DESIGN

* 1. **Flow Chart**

## System Design

A flowchart is a type of [diagram](https://en.wikipedia.org/wiki/Diagram) that represents a [workflow](https://en.wikipedia.org/wiki/Workflow) or [process](https://en.wikipedia.org/wiki/Process). A flowchart can also be defined as a diagrammatic representation of an [algorithm](https://en.wikipedia.org/wiki/Algorithm), a step-by-step approach to solving a task.

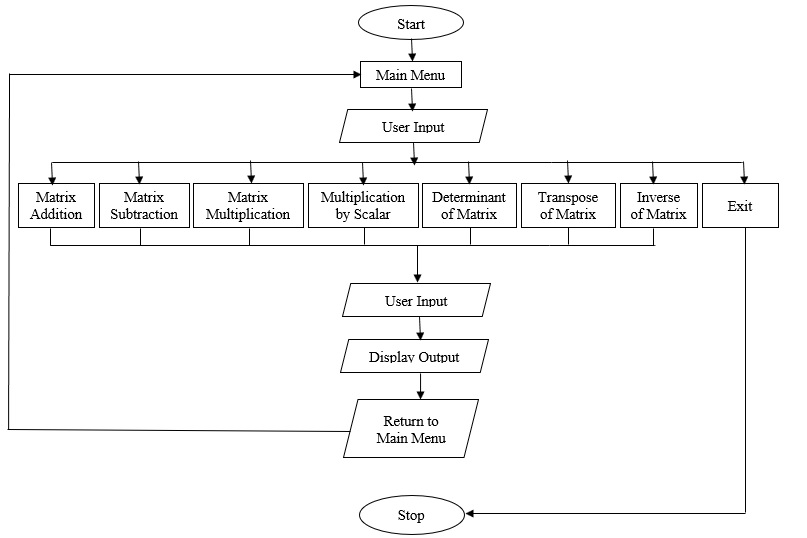


Fig. 5.1 Flowchart

*This flow chart represents flow of data or say the users flow while accessing the software.*

**DEVELOPMENT**

## Development

### 6.1 Source Code

Separate source codes are to be written for the various operations available in the calculator that are to be performed on the matrices. Considering our objectives, we need to develop source codes for in the C Programming Language:

* Addition of two Matrices
* Subtraction of two Matrices
* Multiplication of two Matrices
* Multiplication of Matrix by Scalar
* Determinant of a Matrix
* Transpose of a Matrix
* Inverse of a Matrix

We have constructed a separate library which contains all the source code for operational functions and the library file has been inserted as a header in the main program file.

**6.1.1 Source Code for Addition of two Matrices:**

void addMat()

{

int row1, col1, row2, col2;

printf("Rows of MatA = ");

scanf("%d", &row1);

printf("Columns of MatA = ");

scanf("%d", &col1);

printf("\nRows of MatB = ");

scanf("%d", &row2);

printf("Columns of MatB = ");

scanf("%d", &col2);

float a[row1][col1], b[row2][col2];

if (row1 == row2 && col1 == col2)

{

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col1; j++)

{

printf("a[%d][%d] = ", i + 1, j + 1);

scanf("%f", &a[i][j]);

}

}

printf("\n");

for (int i = 0; i < row2; i++)

{

for (int j = 0; j < col2; j++)

{

printf("b[%d][%d] = ", i + 1, j + 1);

scanf("%f", &b[i][j]);

}

}

printf("\nMatrix A:\n");

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col1; j++)

{

printf(" %.2f", a[i][j]);

}

printf("\n");

}

printf("\nMatrix B:\n");

for (int i = 0; i < row2; i++)

{

for (int j = 0; j < col2; j++)

{

printf(" %.2f", b[i][j]);

}

printf("\n");

}

printf("\nMatrix A + Matrix B: \n");

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col1; j++)

{

printf(" %.2f", a[i][j] + b[i][j]);

}

printf("\n");

}

printf("\n");

}

else

printf("\nOperation not possible.");

}

**6.1.2 Source Code for Subtraction of two Matrices:**

void subtMat()

{

int row1, col1, row2, col2;

printf("Rows of MatA = ");

scanf("%d", &row1);

printf("Columns of MatA = ");

scanf("%d", &col1);

printf("\nRows of MatB = ");

scanf("%d", &row2);

printf("Columns of MatB = ");

scanf("%d", &col2);

float a[row1][col1], b[row2][col2];

if (row1 == row2 && col1 == col2)

{

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col1; j++)

{

printf("a[%d][%d] = ", i + 1, j + 1);

scanf("%f", &a[i][j]);

}

}

printf("\n");

for (int i = 0; i < row2; i++)

{

for (int j = 0; j < col2; j++)

{

printf("b[%d][%d] = ", i + 1, j + 1);

scanf("%f", &b[i][j]);

}

}

printf("\nMatrix A:\n");

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col1; j++)

{

printf(" %.2f", a[i][j]);

}

printf("\n");

}

printf("\nMatrix B:\n");

for (int i = 0; i < row2; i++)

{

for (int j = 0; j < col2; j++)

{

printf(" %.2f", b[i][j]);

}

printf("\n");

}

printf("\nMatrix A - Matrix B: \n");

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col1; j++)

{

printf(" %.2f", a[i][j] - b[i][j]);

}

printf("\n");

}

printf("\n");

}

else

printf("\nOperation not possible.");

}

**6.1.3 Source Code for Multiplication of Matrices:**

void multMat()

{

int i, j, row1, col1, row2, col2, k;

printf("Rows of MatA = ");

scanf("%d", &row1);

printf("Columns of MatA = ");

scanf("%d", &col1);

printf("\nRows of MatB = ");

scanf("%d", &row2);

printf("Columns of MatB = ");

scanf("%d", &col2);

float a[row1][col1], b[row2][col2], p[row1][col2];

if (col1 == row2)

{

printf("\nEnter elements of Matrix A:\n");

for (i = 0; i < row1; i++)

{

for (j = 0; j < col1; j++)

{

printf("a[%d][%d] = ", i + 1, j + 1);

scanf("%f", &a[i][j]);

}

}

printf("\nEnter elements of Matrix B:\n");

for (i = 0; i < row2; i++)

{

for (j = 0; j < col2; j++)

{

printf("b[%d][%d] = ", i + 1, j + 1);

scanf("%f", &b[i][j]);

}

}

printf("\nMatrix A:\n");

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col1; j++)

{

printf(" %.2f", a[i][j]);

}

printf("\n");

}

printf("\nMatrix B:\n");

for (int i = 0; i < row2; i++)

{

for (int j = 0; j < col2; j++)

{

printf(" %.2f", b[i][j]);

}

printf("\n");

}

printf("\nMatrix A \* Matrix B: \n");

for (i = 0; i < row1; i++)

{

for (j = 0; j < col2; j++)

{

p[i][j] = 0;

for (k = 0; k < col1; k++)

{

p[i][j] += a[i][k] \* b[k][j];

}

printf(" %.2f", p[i][j]);

}

printf("\n");

}

}

else

printf("\nOperation not possible.");

}

**6.1.4 Source Code for Multiplication of Matrix by Scalar:**

void scalMultMat()

{

int i, j, row, col;

float k;

printf("Rows of Matrix = ");

scanf("%d", &row);

printf("Columns of Matrix = ");

scanf("%d", &col);

float a[row][col];

printf("\nEnter the scalar for multiplication:");

scanf("%f", &k);

printf("Enter elements of matrix:\n");

for (int i = 0; i < row; i++)

{

for (int j = 0; j < col; j++)

{

printf("a[%d][%d] = ", i + 1, j + 1);

scanf("%f", &a[i][j]);

}

}

printf("\nThe Entered Matrix is:\n");

for (int i = 0; i < row; i++)

{

for (int j = 0; j < col; j++)

{

printf(" %.2f", a[i][j]);

}

printf("\n");

}

printf("\nThe result of multiplication of Matrix by Scalar is:\n");

for (int i = 0; i < row; i++)

{

for (int j = 0; j < col; j++)

{

printf(" %.2f", k \* a[i][j]);

}

printf("\n");

}

}

**6.1.5 Source Code for Determinant of Matrix:**

void detMat()

{

int i, j, choice, row, col;

float Dmt;

printf("Select Dimension of Matrix:(Enter 1 or 2)\n");

printf("1. 2 X 2\n2. 3 X 3\n");

scanf("%d", &choice);

switch (choice)

{

case 1:

row = 2;

col = 2;

break;

case 2:

row = 3;

col = 3;

break;

default:

printf("\nInvalid choice.\n");

break;

}

float a[row][col];

printf("\nEnter the elements of the matrix:\n");

for (i = 0; i < row; i++)

{

for (j = 0; j < col; j++)

{

printf("a[%d][%d] = ", i + 1, j + 1);

scanf("%f", &a[i][j]);

}

}

printf("\nThe Entered Matrix is:\n");

for (int i = 0; i < row; i++)

{

for (int j = 0; j < col; j++)

{

printf(" %.2f", a[i][j]);

}

printf("\n");

}

if (row == 2 && col == 2)

{

Dmt = (a[0][0] \* a[1][1]) - (a[0][1] \* a[1][0]);

}

else if (row == 3 && col == 3)

{

Dmt = a[0][0] \* ((a[1][1] \* a[2][2]) - (a[1][2] \* a[2][1])) - a[0][1] \* ((a[1][0] \* a[2][2]) - (a[1][2] \* a[2][0])) + a[0][2] \* ((a[1][0] \* a[2][1]) - (a[1][1] \* a[2][0]));

}

printf("\nThe determinant for the given matrix is:\n%.2f", Dmt);

}

**6.1.6 Source Code for Transpose of Matrix:**

void transMat()

{

int i, j, row, col;

printf("Rows of Matrix = ");

scanf("%d", &row);

printf("Columns of Matrix = ");

scanf("%d", &col);

float a[row][col], transpose[row][col];

printf("Enter elements of matrix:\n");

for (int i = 0; i < row; i++)

{

for (int j = 0; j < col; j++)

{

printf("a[%d][%d] = ", i + 1, j + 1);

scanf("%f", &a[i][j]);

}

}

printf("\nThe Entered Matrix is:\n");

for (int i = 0; i < row; i++)

{

for (int j = 0; j < col; j++)

{

printf(" %.2f", a[i][j]);

}

printf("\n");

}

printf("\nThe transpose of the matrix is:\n");

for (int i = 0; i < col; i++)

{

for (int j = 0; j < row; j++)

{

transpose[i][j] = a[j][i];

printf(" %.2f", transpose[i][j]);

}

printf("\n");

}

}

**6.1.7 Source Code for Inverse of Matrix:**

void invMat()

{

int i, j, choice, row, col;

float Dmt;

printf("Select Dimension of Matrix:(Enter 1 or 2)\n");

printf("1. 2 X 2\n2. 3 X 3\n");

scanf("%d", &choice);

switch (choice)

{

case 1:

row = 2;

col = 2;

break;

case 2:

row = 3;

col = 3;

break;

default:

printf("\nInvalid choice.\n");

break;

}

float a[row][col];

printf("\nEnter the elements of the matrix:\n");

for (i = 0; i < row; i++)

{

for (j = 0; j < col; j++)

{

printf("a[%d][%d] = ", i + 1, j + 1);

scanf("%f", &a[i][j]);

}

}

printf("\nThe Entered Matrix is:\n");

for (int i = 0; i < row; i++)

{

for (int j = 0; j < col; j++)

{

printf(" %.2f", a[i][j]);

}

printf("\n");

}

if (row == 2 && col == 2)

{

Dmt = (a[0][0] \* a[1][1]) - (a[0][1] \* a[1][0]);

}

else if (row == 3 && col == 3)

{

Dmt = a[0][0] \* ((a[1][1] \* a[2][2]) - (a[1][2] \* a[2][1])) - a[0][1] \* ((a[1][0] \* a[2][2]) - (a[1][2] \* a[2][0])) + a[0][2] \* ((a[1][0] \* a[2][1]) - (a[1][1] \* a[2][0]));

}

if (Dmt == 0)

{

printf("\nThe matrix is singular, and does not have an inverse.\n");

}

else if (row == 2 && col == 2)

{

float inv[row][col], adj[row][col];

adj[0][0] = a[1][1];

adj[0][1] = -a[0][1];

adj[1][0] = -a[1][0];

adj[1][1] = a[0][0];

for (i = 0; i < row; i++)

{

for (j = 0; j < col; j++)

{

inv[i][j] = (adj[i][j]) / Dmt;

}

}

printf("\n The Inverse of given matrix is:\n");

for (i = 0; i < row; i++)

{

for (j = 0; j < col; j++)

{

printf(" %.2f", inv[i][j]);

}

printf("\n");

}

}

else if (row == 3 && col == 3)

{

float inv[row][col], adj[row][col];

adj[0][0] = (a[1][1] \* a[2][2] - a[1][2] \* a[2][1]);

adj[0][1] = -((a[0][1] \* a[2][2] - a[0][2] \* a[2][1]));

adj[0][2] = (a[0][1] \* a[1][2] - a[0][2] \* a[1][1]);

adj[1][0] = -((a[1][0] \* a[2][2] - a[1][2] \* a[2][0]));

adj[1][1] = (a[0][0] \* a[2][2] - a[0][2] \* a[2][0]);

adj[1][2] = -((a[0][0] \* a[1][2] - a[0][2] \* a[1][0]));

adj[2][0] = (a[1][0] \* a[2][1] - a[1][1] \* a[2][0]);

adj[2][1] = -((a[0][0] \* a[2][1] - a[0][1] \* a[2][0]));

adj[2][2] = (a[0][0] \* a[1][1] - a[0][1] \* a[1][0]);

for (i = 0; i < row; i++)

{

for (j = 0; j < col; j++)

{

inv[i][j] = (adj[i][j]) / Dmt;

}

}

printf("\n The Inverse of given matrix is:\n");

for (i = 0; i < row; i++)

{

for (j = 0; j < col; j++)

{

printf(" %.2f", inv[i][j]);

}

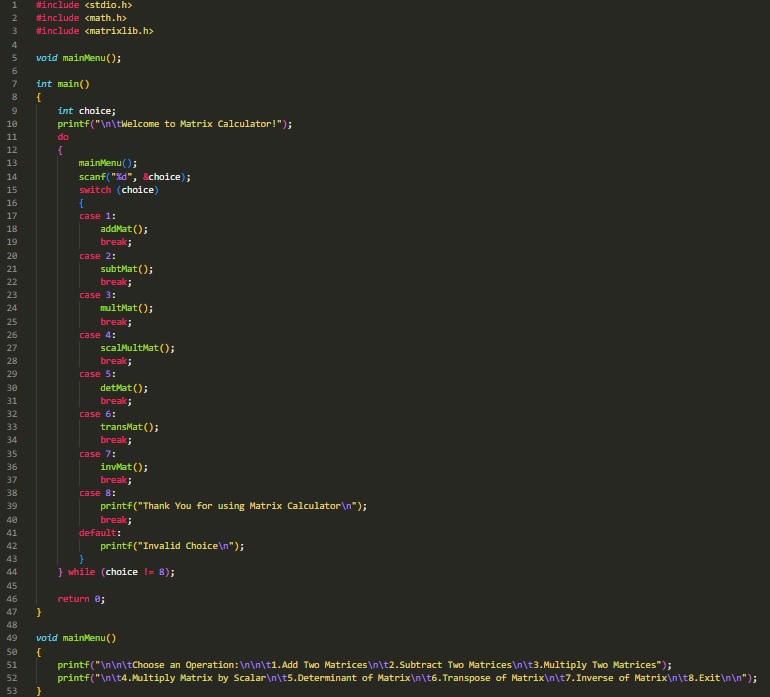
printf("\n");

}

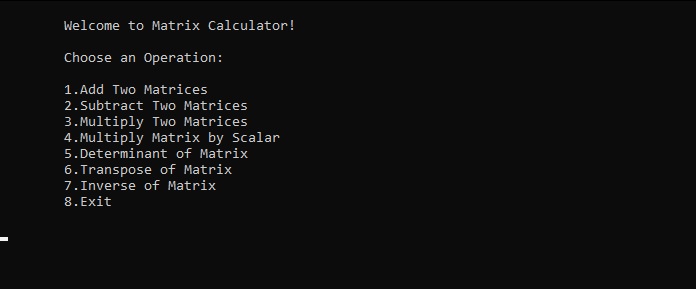
}

}

**6.2 Main Code & Display Output**

****

**Fig. 6.2.1 Main Code**



**Fig. 6.2.2 Display Output**

**CONCLUSION AND FUTURE SCOPE**

* 1. **Conclusion**

## Conclusion and Future Scope

In this study, we have developed an effective matrix calculator that can be used for performing a variety of complex calculations and operations on matrices. After starting the program, the user can choose from within the given options. On selecting one of the options the user will then be requested to enter the matrices in question. Once entered, we get the desired result in a matter of seconds in the output window. Since the program is menu-driven, the user will not be forced out of it after a single iteration and they will have control over whether or not to exit the program.

### Future Scope

* + 1. It can be further improved by making use of GUI on any platform, making it more user-friendly.
    2. It can become a part of and work in unison with another system where matrix calculation is required.

## Acknowledgement

As every project is ever complete with the guidance of experts. So, we would like to take this opportunity to thank all those individuals who have contributed in visualizing this project.

We express our deepest gratitude to our project guide Prof. ------------(CSE(AIML/IOT) Department, Smt. Indira Gandhi College of Engineering, University of Mumbai) for her valuable guidance, moral support and devotion bestowed on us throughout our work.

We would also take this opportunity to thank our project coordinator Prof. Bhagyalakshmi for her guidance in selecting this project and also for providing us all the details on proper presentation of this project.

We extend our sincere appreciation to our entire professors from Smt. Indira Gandhi College of Engineering for their valuable inside and tip during the designing the project. Their contributions have been valuable in many ways that we find it difficult to acknowledge them individually.

We are also grateful to our HOD Prof. Sonali Deshpande for extending his help directly and indirectly through various channels in our project.

If I can say in words I must at the outset my intimacy for receipt of affectionate care to Smt. Indira Gandhi College of Engineering for providing such a simulating atmosphere and wonderful work environment.

## References

1. <https://code.visualstudio.com/Docs/editor/whyvscode>
2. <https://www.tutorialspoint.com/cprogramming/c_overview.htm>
3. <https://digital.gov/resources/an-introduction-github/>
4. <https://123seminarsonly.com/>