

**VARDHAMAN COLLEGE OF ENGINEERING**

(AUTONOMOUS)

Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with A++ Grade, ISO 9001:2015 Certified

Kacharam, Shamshabad, Hyderabad – 501218, Telangana, India

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

A Course End Project Report towards Data Structures Laboratory titled

# Matrix Calculator

Submitted in the partial fulfilment of the requirements for the course end project of

**BACHELOR TECHNOLOGY OF TECHNOLOGY**

**IN**

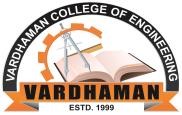
**COMPUTER SCIENCE AND ENGINEERING**

Submitted

By

Roll Number Name of the Student

22881A05A1 N.Siddartha Gowtham



**VARDHAMAN COLLEGE OF ENGINEERING**

(AUTONOMOUS)

Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with A++ Grade, ISO 9001:2015 Certified

Kacharam, Shamshabad, Hyderabad – 501218, Telangana, India

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**CERTIFICATE**

This is to certify that the project titled “Simulation of Data structures” is submitted by

Roll Number Name of the Student

22881A05A1 N.SiddarthGowtham

in partial fulfilment of the requirements for the course end project for the course Data Structures Laboratory for the academic year 2022-23

Signature of the Instructor Signature of the HOD

**Abstract:**

The Matrix Calculator using Files is a software application that provides a userfriendly interface for performing various matrix operations with the support of file input and output. This calculator offers a versatile tool for matrix manipulations, enabling users to efficiently perform operations such as addition, subtraction, multiplication, determinant computation, inverse calculation, and more.

The application is designed to accommodate large matrices by leveraging file handling techniques. Users can input matrices from external files, allowing for the manipulation of matrices beyond the constraints of available memory. Additionally, the results of matrix operations can be saved to output files for future reference and analysis.

**Table of Contents**

CHAPTER NO. TITLE PAGE NO

ABSTRACT

LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| 1 | INTRODUCTION  1.1 PROBLEM DEFENITION | 9 |
|  | 1.2 OBJECTIVES OF PROJECT | 9 |
| 2 | System REQUIREMENTS | 14 |
|  | 2.1 Algorithms | 14 |
|  | 2.2 Data Structures | 15 |
| 3 | Design & Implementation | 20 |
|  | 3.1 Design/Solution | 21 |
|  | 3.2 Flowchart | 22 |
|  | 3.3 Programs | 23 |
| 4 | System Testing | 25 |
|  | 4.1 Input |  |
|  | 4.2 Output |  |
| 5 | Conclusion | 26 |

**INTRODUCTION:**

The Matrix Calculator is a powerful tool designed to perform various matrix operations conveniently by reading matrix data from files. Matrices are fundamental data structures used in a wide range of fields, such as engineering, physics, computer graphics, and data analysis. This calculator aims to simplify matrix calculations and automate repetitive tasks by allowing users to store matrix data in files and retrieve results efficiently.

* 1. **PROBLEM DEFENITION:**

The problem addressed by the Matrix Calculator using Files is to develop a software application that facilitates efficient manipulation of matrices while accommodating large matrices beyond the constraints of available memory. The primary objective is to provide users with a userfriendly interface to perform a variety of matrix operations with the support of file-based input and output.

* 1. **OBJECTIVES OF PROJECT:**

The main objective of the Matrix Calculator using Files is to develop a software application that enables efficient manipulation of matrices, providing users with a user-friendly interface for performing various matrix operations while accommodating large matrices beyond the constraints of available memory. The specific objectives of the Matrix Calculator are as follows:

**SYSTEM REQUIREMENTS:**

**2.1 Algorithms**:

To implement a matrix calculator using files, you need to design an algorithm that can read matrix data from files, perform various matrix operations, and write the results back to files.

Below is a high-level algorithm for a matrix calculator using files:

**1. Read Input Matrices from Files:**

* Prompt the user to input the names of the files containing the matrices they want to perform operations on.
* Open the files and read the matrix data from them into appropriate data structures (e.g., 2D array, list of lists, or sparse matrix format) as discussed in the previous response.

**2. Display Menu and Get Operation Choice:**

* Present the user with a menu of matrix operations they can perform (e.g., addition, subtraction, multiplication, etc.).
* Prompt the user to select an operation.

**3. Perform the Chosen Operation:**

- Based on the user's choice, perform the corresponding matrix operation using appropriate algorithms (e.g., matrix addition, multiplication, etc.).

**4. Write Result to File:**

* After performing the matrix operation, open a new file or overwrite one of the input files to store the result.
* Write the result matrix data to the file in the same format as the input files.

**5. Display or Continue:**

* Optionally, you can display the result matrix to the user on the screen.
* Ask the user if they want to continue using the calculator or exit the program.

**6. Error Handling:**

* Implement appropriate error handling to deal with cases like file not found, invalid matrix data in the file, incompatible matrix dimensions for certain operations, etc.
* Provide informative error messages to the user to guide them on how to fix the issues.

**7. Repeat:**

- If the user wants to continue, repeat steps 1 to 6 for the new set of input files and

operations.

**2.2 Data Structures**

When implementing a matrix calculator that reads data from files, you can use various data structures to store and manipulate the matrix data efficiently. Here are some common data structures you can consider:

1. **2D Array:**A simple and commonly used data structure to represent a matrix is a 2D array. You can use a 2D array to store the elements of the matrix. Each row of the array represents a row in the matrix, and each element of the array represents an individual cell in the matrix.
2. **List of Lists (List of Lists of Integers):**If your matrix contains a variable number of rows and columns, you can use a list of lists to represent it. Each element of the outer list would be a list representing a row of the matrix.
3. **List of Lists (List of Lists of Floats):** Similar to the previous option, but if your matrix contains floating-point numbers, you can use this data structure to store the matrix elements.
4. **Dictionary of Dictionaries:**You can use a nested dictionary where the outer dictionary represents rows, and each inner dictionary represents columns. The keys of the inner dictionaries would be column indices, and the corresponding values would be the matrix elements.

**Design & Implementation**:

**3.1 Design/Solution:**

To create a Matrix Calculator using files, we need to design a solution that encompasses reading input matrices from files, performing various matrix operations, and writing the results back to files. Here's a step-by-step design for such a calculator**:**

**1. Input:**

* Prompt the user to provide the file names for the input matrices (e.g., matrix1.txt, matrix2.txt).
* Each file should contain the matrix data in a specific format (e.g., CSV, TSV, custom format).

**2. Read Matrices from Files:**

* Open the input files and read the matrix data from them.
* Parse the data and store it in appropriate data structures, such as a 2D array or list of lists.

**3. Display Menu:**

* Show a menu of available matrix operations to the user (e.g., addition, subtraction, multiplication, etc.).
* Let the user choose the desired operation by entering the corresponding operation code or name.

**4. Perform the Chosen Operation:**

* Based on the user's choice, perform the corresponding matrix operation using efficient algorithms.
* Ensure that the dimensions of the matrices are compatible for the selected operation.

**5. Write Result to File:**

* After performing the matrix operation, open a new file or overwrite one of the input files to store the result.
* Write the result matrix data to the file in the same format as the input files.

**6. Error Handling:**

* Implement error handling to catch and handle potential issues, such as file not found, invalid matrix data, or incompatible dimensions for operations.
* Provide informative error messages to guide users on resolving issues.

**7. Display Result or Continue:**

* Display the result matrix to the user on the screen.
* Ask the user if they want to continue using the calculator or exit the program.

**8. Loop for Multiple Operations:**

* Offer the option to perform multiple matrix operations in a single run by utilizing a loop.
* Allow users to keep performing operations until they choose to exit the calculator.

**9. Finalization:**

- After the user exits the calculator, display a closing message or prompt for further actions.

**3.2 Flowchart**

**3.2 Flowchart**

Matrix Calculator

Prompt for file names

Read the Matrices

Validate Matrix Data

Display Menu

Choose Operation

Perform the choosen Opertion

Validate Result Matrix

Write Result to File

Display Result

Ask to Continue

EXIT

**3.3 PROGRAMS:**

**#include <stdio.h>**

**#include <stdlib.h>**

**// Function to read a matrix from a file void readMatrixFromFile(char \*filename, int rows, int cols, int matrix[rows][cols]) { FILE \*file = fopen(filename, "r"); if (file == NULL) {**

**printf("Error opening file: %s\n", filename); exit(1);**

**}**

**for (int i = 0; i< rows; i++) { for (int j = 0; j < cols; j++) {**

**fscanf(file, "%d", &matrix[i][j]);**

**}**

**}**

**fclose(file);**

**}**

**// Function to write a matrix to a file void writeMatrixToFile(char \*filename, int rows, int cols, int matrix[rows][cols]) { FILE \*file = fopen(filename, "w");**

**if (file == NULL) {**

**printf("Error opening file: %s\n", filename); exit(1);**

**}**

**for (int i = 0; i< rows; i++) { for (int j = 0; j < cols; j++) { fprintf(file, "%d ", matrix[i][j]);**

**}**

**fprintf(file, "\n");**

**}**

**fclose(file);**

**}**

**// Function to add two matrices**

**void addMatrices(int rows, int cols, int matrix1[rows][cols], int matrix2[rows][cols], int result[rows][cols]) { for (int i = 0; i< rows; i++) { for (int j = 0; j < cols; j++) { result[i][j] = matrix1[i][j] + matrix2[i][j];**

**}**

**}**

**}**

**// Function to subtract two matrices**

**void subtractMatrices(int rows, int cols, int matrix1[rows][cols], int matrix2[rows][cols], int result[rows][cols]) { for (int i = 0; i< rows; i++) { for (int j = 0; j < cols; j++) { result[i][j] = matrix1[i][j] - matrix2[i][j];**

**}**

**}**

**}**

**// Function to multiply two matrices**

**void multiplyMatrices(int rows1, int cols1, int matrix1[rows1][cols1], int rows2, int cols2, int matrix2[rows2][cols2], int result[rows1][cols2]) {**

**if (cols1 != rows2) {**

**printf("Error: The number of columns in the first matrix must be equal to the number of rows in the second matrix for multiplication.\n");**

**exit(1);**

**}**

**for (int i = 0; i< rows1; i++) { for (int j = 0; j < cols2; j++) { result[i][j] = 0; for (int k = 0; k < cols1; k++) { result[i][j] += matrix1[i][k] \* matrix2[k][j];**

**}**

**}**

**}**

**}**

**// Function to divide two matrices (element-wise division)**

**void divideMatrices(int rows, int cols, int matrix1[rows][cols], int matrix2[rows][cols], int result[rows][cols]) {**

**for (int i = 0; i< rows; i++) { for (int j = 0; j < cols; j++) { if (matrix2[i][j] == 0) { printf("Error: Division by zero is not allowed.\n"); exit(1);**

**}**

**result[i][j] = matrix1[i][j] / matrix2[i][j];**

**}**

**}**

**}**

**int main() { int rows, cols;**

**// Read matrix dimensions from the user printf("Enter the number of rows: "); scanf("%d", &rows); printf("Enter the number of columns: "); scanf("%d", &cols);**

**// Allocate memory for matrices int matrix1[rows][cols]; int matrix2[rows][cols]; int result[rows][cols];**

**// Read matrices from input files readMatrixFromFile("matrix1.txt", rows, cols, matrix1); readMatrixFromFile("matrix2.txt", rows, cols, matrix2);**

**// Perform addition addMatrices(rows, cols, matrix1, matrix2, result); writeMatrixToFile("add\_result.txt", rows, cols, result);**

**// Perform subtraction subtractMatrices(rows, cols, matrix1, matrix2, result); writeMatrixToFile("subtract\_result.txt", rows, cols, result);**

**// Perform multiplication int rows2, cols2;**

**printf("Enter the number of rows for the second matrix: "); scanf("%d", &rows2); printf("Enter the number of columns for the second matrix: "); scanf("%d", &cols2);**

**if (cols2 != rows) {**

**printf("Error: The number of columns in the first matrix must be equal to the number of rows in the second matrix for multiplication.\n");**

**return 1;**

**}**

**int result\_multiply[rows][cols2];**

**readMatrixFromFile("matrix2.txt", rows2, cols2, matrix2); multiplyMatrices(rows, cols, matrix1, rows2, cols2, matrix2, result\_multiply); writeMatrixToFile("multiply\_result.txt", rows, cols2, result\_multiply);**

**// Perform division divideMatrices(rows, cols, matrix1, matrix2, result); writeMatrixToFile("divide\_result.txt", rows, cols, result);**

**printf("Matrices added, subtracted, multiplied, and divided successfully, and results saved to files.\n");**

**System Testing:**

**4.1 Files Input: matrix1.txt**

1. 1
2. 2

**matrix2.txt**

1. 1
2. 2

**4.2 Output: add\_result.txt**

1. 1
2. 2

**subract\_result.txt**

0 0

0 0

**multiply\_result.txt**

0 0

1. 0

**divide\_result.txt**

1. 1

1 1

**CONCLUSION:**

In conclusion, a matrix calculator using files is a useful tool that allows users to perform various matrix operations by reading input matrices from files and writing the results back to files. Overall, a matrix calculator using files provides a convenient way for users to work with matrices without having to input data manually and allows for the automation of repetitive tasks. It can be further enhanced by incorporating additional features, optimizing algorithms for efficiency, and ensuring a user-friendly interface. With proper implementation, such a calculator can become a valuable tool for various applications in engineering, data analysis, and scientific computing.