Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering
Semester: (Spring, Year:2022), B.Sc. in CSE (Day)

Course Title: Structured Programming Lab

Course Code: CSE 104 **Section: DA**

Lab Project Name: Matrix Calculator in C

Student Details

Name	ID
Md. Kawsar Ahamed	213902013

Submission Date **: 26 April 2022**

Course Teacher's Name **: Md. Solaiman Mia**

[For Teachers use only: Don't Write Anything inside this box]

Lab Project Status

Marks: …… **Signature: .....**

Comments: ..... **Date: .....**

Table of Contents

Chapter 1

Introduction

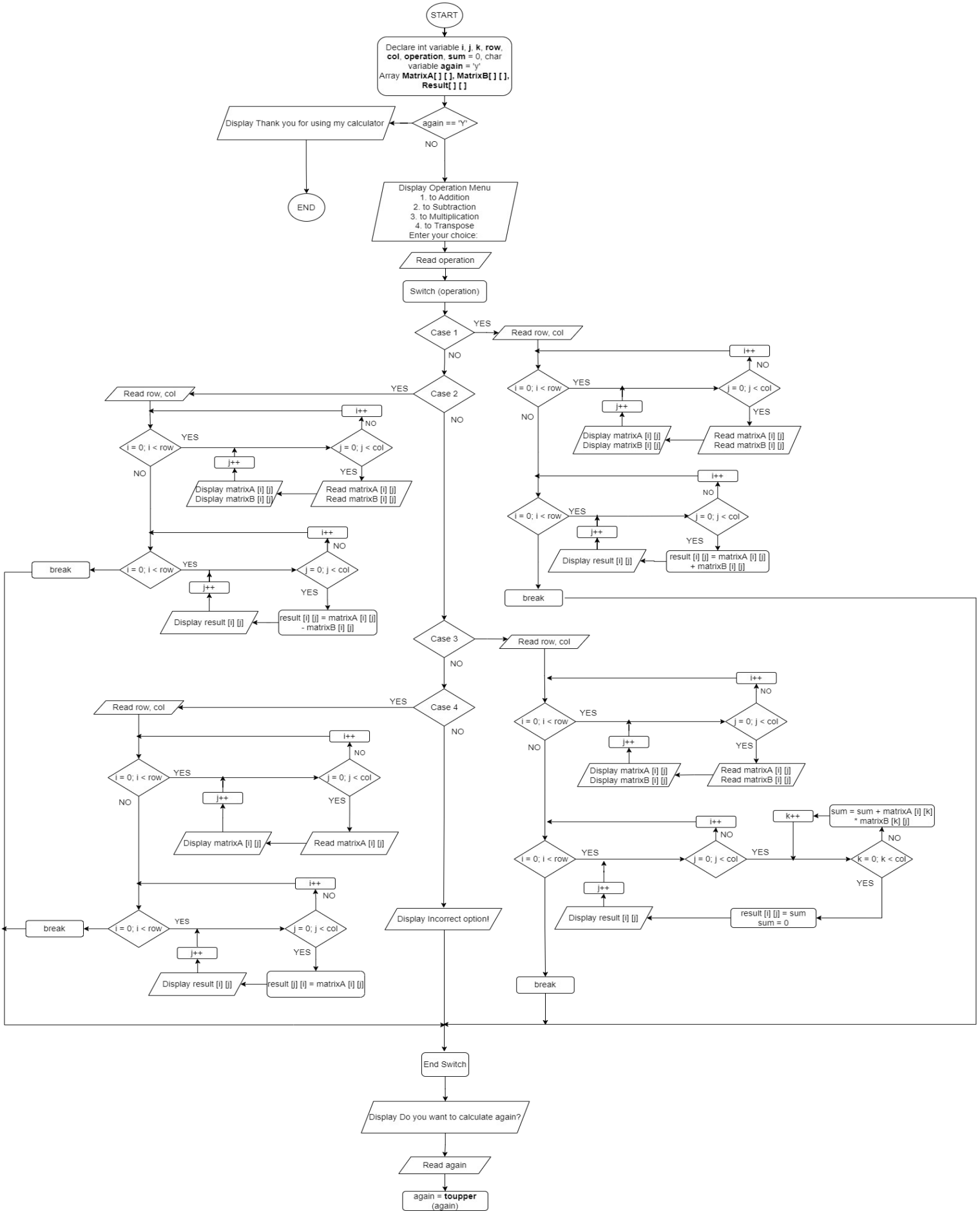
1.1 Introduction

Matrix, a set of numbers arranged in rows and columns so as to form a rectangular array. The numbers are called the elements, or entries, of the matrix. Matrices have wide applications in engineering, physics, economics, and statistics as well as in various branches of mathematics. Matrix operations such as addition, multiplication, subtraction, etc., are similar to what most people are likely accustomed to seeing in basic arithmetic and algebra, but do differ in some ways, and are subject to certain constraints. Below are descriptions of the matrix operations that this calculator can perform.

1.2 Objective

Matrix calculators and *formulas* for 2×2, 3×3, 4×4, nxn matrix or matrices addition, subtraction, multiplication, determinant, inverse or transpose matrix. The main objective of these matrix tools is to assist students, professionals and researchers to quickly perform matrix related calculations or verify the results of such calculations to analyze, determine or solve the linear functions and equations. These matrix formulas & calculators may provide the answers in many complex algorithms of digital information, image and video processing applications.

1.3 Flowchart-



Chapter 2

Implementation of the Project

1 Pseudo Code -

Step 1- Start.

Step 2- Declare integer variable i, j, k, row, col, operation, sum = 0, Array matrixA, matrixB, result, char variable again.

Step 3- While again = y then display message Welcome and Operation Menu.

Step 4- Read operation and switch operation.

Step 5- Do what user give number in operation for case.

Case 1 for addition- Display message and read row and col. Using nested for loop, initial i = 0 and i < row

j = 0, j < col and read matrixA [i] [j] .

Same as matrixB. After reading matrixA and matrixB display those number using same loop.

Using for loop, result [i] [j] = matrixA [i] [j] + matrixB [i] [j] and display result. After display result break Case 1

Case 2 for subtraction- If Case 1 false then display message and read row and col. Using nested for loop, initial i = 0 and

i < row j = 0, j < col and read matrixA [i] [j] .

Same as matrixB. After reading matrixA and matrixB display those number using same loop.

Using for loop, result [i] [j] = matrixA [i] [j] - matrixB [i] [j] and display result. After display result break Case 2

Case 3 for multiplication- If Case 2 false then display message and read row and col. If row = col then using nested for loop, initial i = 0 and i < row j = 0, j < col and read matrixA [i] [j]. Same as matrixB.

Using for loop for 3 times, sum = sum + matrixA [i] [j] × matrixB [k] [j] then result [i] [j] = sum and display matrixA, matrixB and result. After display result break Case 3

Case 4 for transpose- If Case 3 false then display message and read row and col. Using nested for loop, initial i = 0 and

i < row j = 0, j < col and read matrixA [i] [j] .

Using for loop, result [j] [i] = matrixA [i] [j] and display result. After display result break Case 4

Default – If all case are false then display Incorrect option and break and end switch.

Step 6- After end switch read again. If again = y then using toupper function and then again switch operation start and repeat all. If again not equal y then display message and close the operations.

Step 7- End.

2 Implementation -

```
#include <stdio.h>

int main()
{
   int i, j, k;

   int matrixA[10][10]; // initialized at 10 just to have it initialized
   int matrixB[10][10];
   int result[10][10];
   int row, col;

   int operation;//for switch statements
   char again = 'Y';
   int sum = 0;

   while (again == 'Y')
   {
      printf("\t\tWelcome to Matrix calculator\n");
      printf("*******************************************************\n");

      printf("\nOperation Menu:\n");
      printf("\t1. to Addition\n");
      printf("\t2. to Subtraction\n");
      printf("\t3. to Multiplication\n");
      printf("\t4. to Transpose\n");
      printf("\nEnter your choice: ");
      scanf(" %d", &operation);
```

```
switch (operation)
{

//addition
case 1:
   printf("\n\t*****You choice Matrix addition******");
   printf("\n************************************************************");
   printf("\nEnter Number of Row: ");
   scanf("%d", &row);
   printf("Enter Number of Column: ");
   scanf("%d", &col);
   printf("\n\nEnter elements for Matrix A\n");
   for (i=0; i<row; i++)
   {
      for (j=0; j<col; j++)
      {
         printf("\A[%d][%d] = ", i, j);
         scanf("%d", &matrixA[i][j]);
      }
      printf("\n");
   }
   printf("\nEnter elements for Matrix B\n");
   for (i=0; i<row; i++)
   {
      for (j=0; j<col; j++)
      {
         printf("B[%d][%d] = ", i, j);
         scanf("%d", &matrixB[i][j]);
      }
      printf("\n");
   }
```

```
printf("\n[A]= \n");
for (i=0; i<row; i++)
{
   for (j=0; j<col; j++)
   {
      printf("\t%d ", matrixA[i][j]);
   }
   printf("\n");
}
printf("\n[B]= \n");
for (i=0; i<row; i++)
{
   for (j=0; j<col; j++)
   {
      printf("\t%d ", matrixB[i][j]);
   }
   printf("\n");
}

printf("\n");
for (i=0; i<row; i++)
{
   for (j=0; j<col; j++)
   {
      result[i][j] = matrixA[i][j] + matrixB[i][j];
   }
}
printf("[A+B]= \n");
for (i=0; i<row; i++)
{
```

```
        for (j=0; j<col; j++)
        {
            printf("\t%d ", result[i][j]);
        }
        printf("\n");
    }
    break;

//subtraction
case 2:
    printf("\n\t*****You choice Matrix subtraction******");
    printf("\n************************************************************");

    printf("\nEnter Number of Row: ");
    scanf("%d", &row);
    printf("Enter Number of Column: ");
    scanf("%d", &col);
    printf("\n\nEnter elements for Matrix A\n");

    for (i=0; i<row; i++)
    {
        for (j=0; j<col; j++)
        {
            printf("\A[%d][%d] = ", i, j);
            scanf("%d", &matrixA[i][j]);
        }
        printf("\n");
    }
    printf("\nEnter elements for Matrix B\n");
    for (i=0; i<row; i++)
    {
        for (j=0; j<col; j++)
```

```
    {
        printf("B[%d][%d] = ", i, j);
        scanf("%d", &matrixB[i][j]);
    }
    printf("\n");
}
printf("\n[A]= \n");
for (i=0; i<row; i++)
{
    for (j=0; j<col; j++)
    {
        printf("\t%d ", matrixA[i][j]);
    }
    printf("\n");
}

printf("\n[B]= \n");

for (i=0; i<row; i++)
{
    for (j=0; j<col; j++)
    {
        printf("\t%d ", matrixB[i][j]);
    }
    printf("\n");
}
printf("\n[A-B]= \n");
for (i = 0; i < row; i++)
{
    for (j = 0; j < col; j++)
```

```
        {
            result[i][j] = matrixA[i][j] - matrixB[i][j];
            printf("\t%d ", result[i][j]);
        }
        printf("\n");
    }
    break;
//multiplication
case 3:
    printf("\n\t*****You choice Matrix multiplication******");
    printf("\n************************************************************");
    printf("\nEnter Number of Row: ");
    scanf("%d", &row);
    printf("Enter Number of Column: ");
    scanf("%d", &col);
    if (row == col)
    {
        printf("\n\nEnter elements for Matrix A\n");
        for(i=0; i<row; i++)
        {
            for(j=0; j<col; j++)
            {
                printf("A[%d][%d]= ",i,j);
                scanf("%d", &matrixA[i][j]);
            }
            printf("\n");
        }
        printf("\nEnter elements for Matrix B\n");
        for(i=0; i<row; i++)
        {
            for(j=0; j<col; j++)
```

```
    {
        printf("B[%d][%d]= ",i,j);
        scanf("%d", &matrixB[i][j]);
    }
    printf("\n");
}
for(i=0; i<row; i++)
{
    for(j=0; j<col; j++)
    {
        for(k=0; k<col; k++)
        {
            sum = sum + matrixA[i][k]*matrixB[k][j];
        }
        result[i][j] = sum;
        sum = 0;
    }
}
printf("\n[A]= \n");
for(i=0; i<row; i++)
{
    for(j=0; j<col; j++)
    {
        printf("\t");
        printf("%d ", matrixA[i][j]);
    }
    printf("\n");
}
printf("\n[B]= \n");
for(i=0; i<row; i++)
```

```
        {
            for(j=0; j<col; j++)
            {
                printf("\t");
                printf("%d ", matrixB[i][j]);
            }
            printf("\n");
        }
        printf("\n[A*B]=\n");
        for(i=0; i<row; i++)
        {
            for(j=0; j<col; j++)
            {
                printf("\t");
                printf("%d ", result[i][j]);
            }
            printf("\n");
        }
        break;
    }
    else
    {
        printf("\nColumn number and Row number are not same\n");
        break;
    }

//transpose
case 4:
    printf("\n\t*****You choice transpose of Matrix******");
    printf("\n*********************************************************");
    printf("\nEnter Number of Row: ");
    scanf("%d", &row);
```

```
printf("Enter Number of Column: ");
scanf("%d", &col);
printf("\n\nEnter elements for Matrix A\n");
for(i=0; i<row; i++)
{
   for(j=0; j<col; j++)

   {
      printf("\A[%d][%d]=",i,j);
      scanf("%d",&matrixA[i][j]);
   }
   printf("\n");
}
printf("\n[A]=\n");
for(i=0; i<row; i++)
{
   for(j=0; j<col; j++)
   {
      result[j][i]=matrixA[i][j];
   }
}
for(i=0; i<row; i++)
{
   printf("\t");
   for(j=0; j<col; j++)
   {
      printf("\t%d ",matrixA[i][j]);
   }
   printf("\n");
}
printf("\nTranspose of [A]= \n");
for(i=0; i<col; i++)
```

```
        {
            printf("\t");
            for(j=0; j<row; j++)
            {
                printf("\t%d ",result[i][j]);
            }
            printf("\n");
        }
        break;
    default:
        printf("\nIncorrect option! Please choose the correct option.");
        break;
    }
    printf("\n\nDo you want to calculate again? Press Y for calculate again.\nTo EXIT, press any letter.\n");
    scanf(" %c", &again);
    again = toupper(again);
  }

  printf("\n\nThank you for using my calculator!\n\n");
  getch ();
  return 0;
}
```

Chapter 3

Performance Evaluation

3.1 Results and Discussions

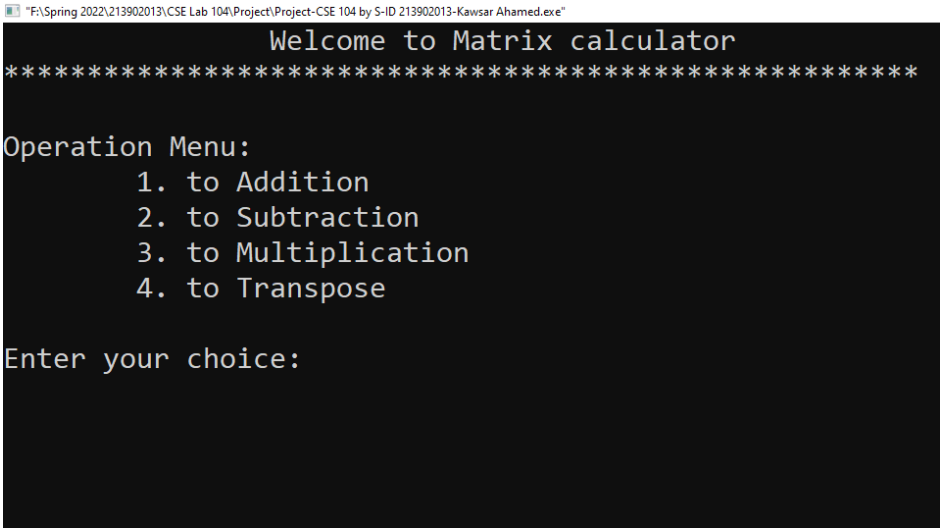


figure 3.1 Contents

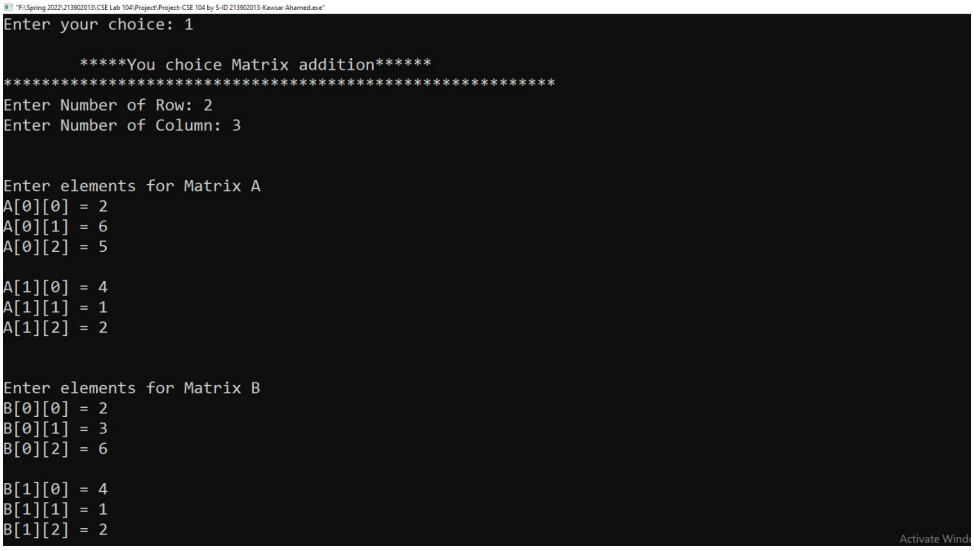


figure 3.2 Matrix addition

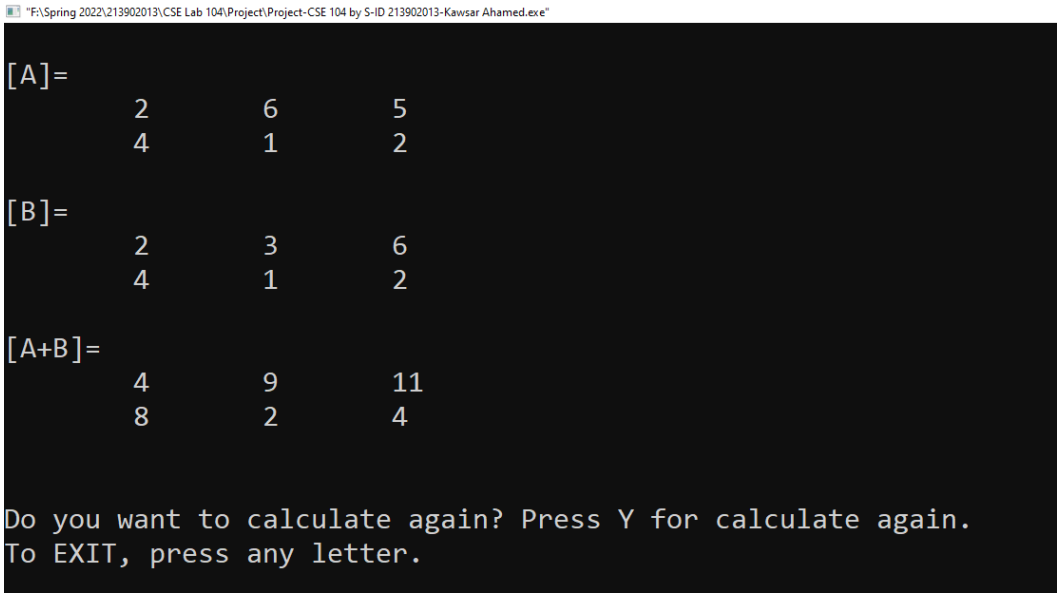


figure 3.3 Addition result

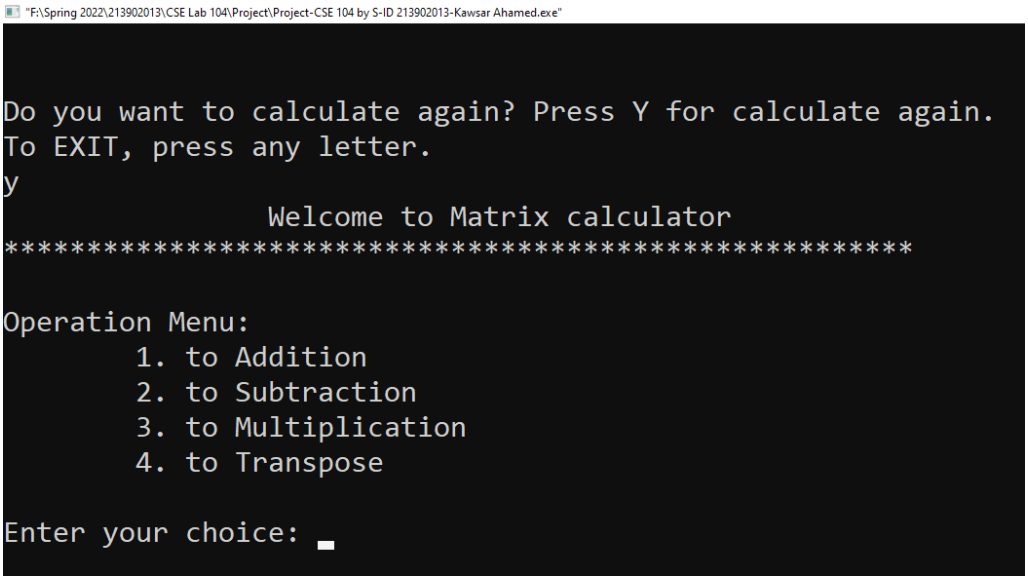


figure 3.4 After result

3.2 Analysis and Outcome

- Here is a program using C language.
- There are some operations on the matrix.
- Array has been used to get the value of each row and column correctly.
- Switch function have been used for separately use.
- He we use toupper keywords for reuse.
- Using function would have been more efficient.
- Due to the large amount of code, the program was time consuming.
- There were no major issues with the program.
- Finally we were able to run the program properly.
- It’s working.

Chapter 4

Conclusion

4.1 Introduction

The matrix we are is useful and powerful in mathematical analysis and collecting data. Besides the simultaneous equations, the characteristic of the matrices is useful in programming where we put in an array that is a matrix also to store the data.

.1 Practical Implications

Matrices are used in the science of optics to account for reflection and for refraction. Matrices are also useful in electrical circuits and quantum mechanics and resistor conversion of electrical energy. Matrices are used to solve AC network equations in electric circuits.

.2 Scope of Future Work

- Here users can easily calculate the matrix sum, subtraction, and multiplication between two matrices.
- Users can also transpose here.
- In the future, we may perform some more operations on the matrix like find the determinant.
- We also can add inverse operations.
- We can find the rank of a matrix.
- We can make operations of triangular matrix. etc.