Omar Mohamed Atia Mohamed Shehab

Professor Mohamed Eid

Computer Programming for Engineers

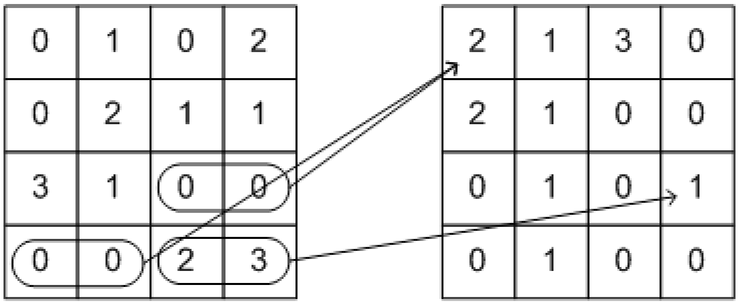
November 13, 2022

Step 1: Problem Identification and Statement:

- **Gray Level Co-occurrence Matrix (GLCM)** is used for **texture analysis** of an image. We loop through two pixels at a time to determine the **frequency** of each index in a specific matrix. Those two pixels are called the **reference and the neighbor pixel.** To calculate the GLCM, a particular spatial relationship between the reference and neighbor pixel should be defined. We can define the current pixel to be the 1, and the neighbor to be the right pixel next to the current one or the one below it.
- In order for the GLCM to be created, we should find the maximum number in the grayscale matrix and it will be the dimensions of the new GLCM Matrix (if the max size is 8 then the dimensions will be 8*8). Then we should loop through the grayscale matrix and each successive index in it **(ie: 1,1)** will be taken as the index of the new GLCM Matrix and will be incremented by one each time it appears.
- The next step is to calculate **the normalized matrix**, which is calculated by dividing each number in the matrix by the summation of the GLCM Matrix.
- Those data will help in calculating some statistical information about the image such as the **contrast, energy, homogeneity, and variance**.

Step 2: Gathering Information:

- **The GLCM Matrix process:**
    - We calculate the GLCM matrix by looping through the grayscale matrix and getting each two consecutive indexes to be the index of the GLCM Matrix then this index should be incremented as shown in the **figure 1:**



4 x 4 image

0	1	0	2
0	2	1	1
3	1	0	0
0	0	2	3

GLCM

2	1	3	0
2	1	0	0
0	1	0	1
0	1	0	0

Figure 1

- **The Normalized Matrix process:**
    - This matrix needs the summation of numbers in the GLCM Matrix **M(i,j)**, then we divide the summation by each number in the index, and the following formula is used to calculate the Normalized Matrix **p(i,j)**:

$$p(i,j) = \frac{M(i,j)}{\sum_{i,j} M(i,j)}$$

    - The Normalized Matrix is going to be as shown in **figure 2** based on the example in **figure1:**

0.16	0.08	0.25	0
0.16	0.08	0	0
0	0.08	0	0.08
0	0.08	0	0

Figure 2

Nomalized GLCM

- The texture statistical information about the normalized matrix:
    - **Contrast:** it’s calculated through calculating the summation of the **square of the difference** between the **ith (rows)** and **jth(columns)** index of a number in the matrix and **multiplying** it with the number itself in the normalized matrix. The formula below shows how can we calculate the contrast:-

$$: \sum_{i,j} |i-j|^2 \times p(i,j)$$

    - **Energy:** it’s calculated by summing the square of all the numbers in the normalized matrix. The formula below shows how can we calculate the energy :-

$$\sum_{i,j} p(i,j)^2$$

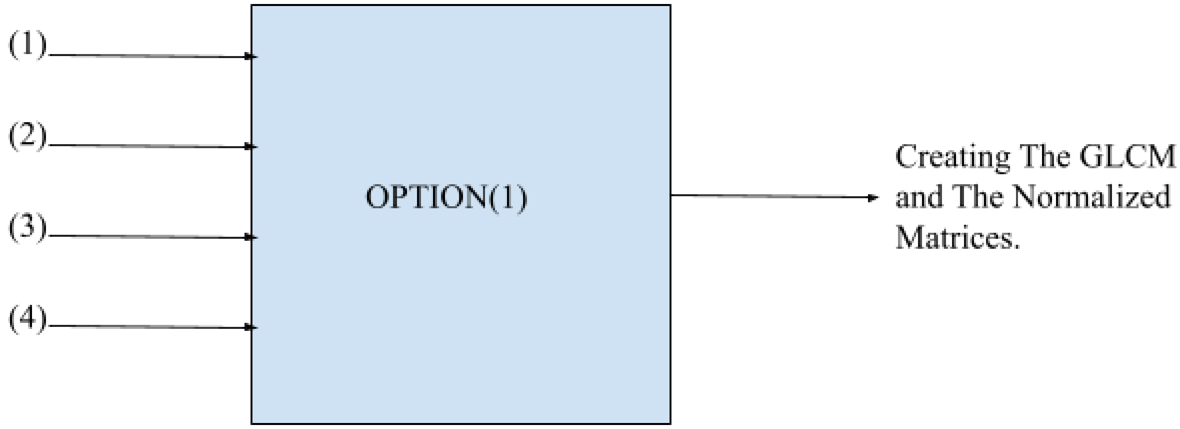
    - **Homogeneity:** it’s calculated through calculating the summation of the **division** of each number in the normalized matrix by the **square of the difference** between the **ith (rows)** and **jth(columns)** index of a number in the matrix then adding 1 to it. The formula below shows how can we calculate the homogeneity :-

$$\sum_{i,j} \frac{p(i,j)}{1+|i-j|^2}$$

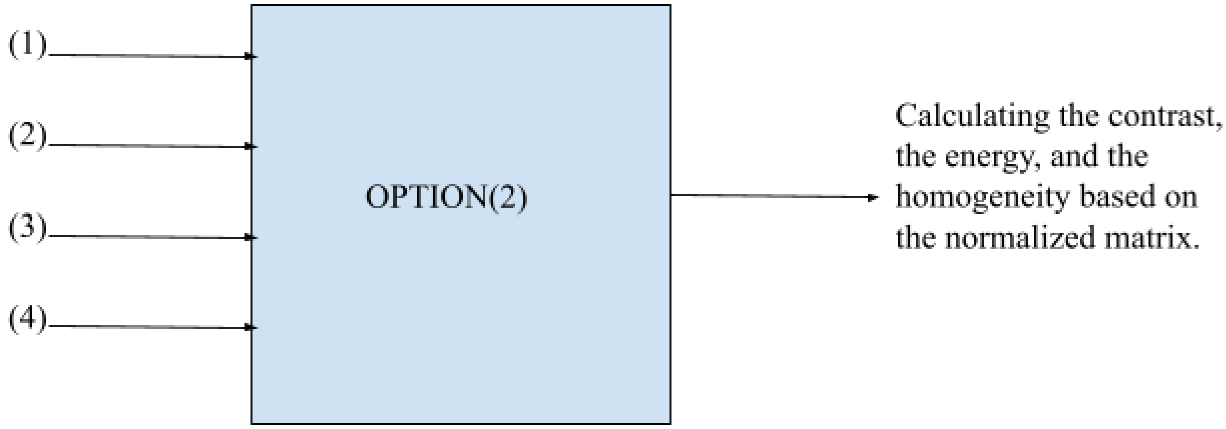
- **I/O Menu:**

First, the user will be asked to enter the names of two files, o**ne for the coordinates of the Grayscale Matrix, and the other for displaying the GLCM Matrix.** Then, a menu will display to the user and he has to choose a valid number.

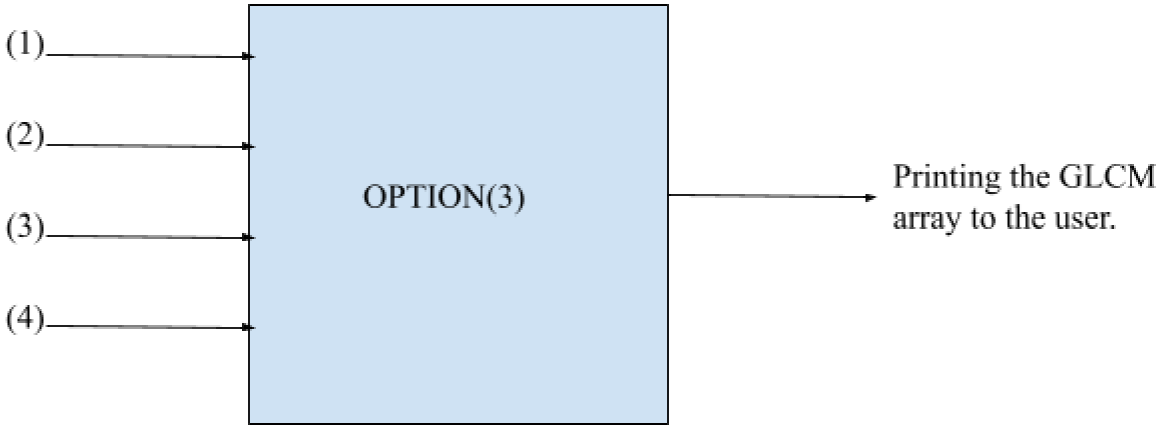
- **The menu is:**
  - 1- Create and normalize the GLCM matrix.
  - 2- Compute the statistical parameters of the texture.
  - 3- Print the GLCM matrix.
  - 4- Exit.
- **Input/output for option (1):**



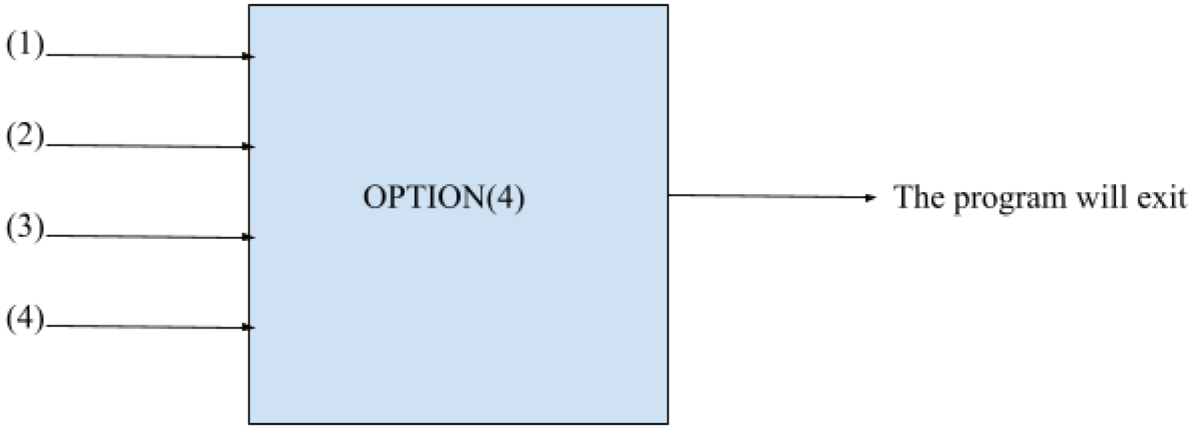
- **Input/output for option (2):**


(1)
(2)
(3)
(4)
OPTION(2)
Calculating the contrast, the energy, and the homogeneity based on the normalized matrix.

- **Input/output for option (3):**



- **Input/output for option (4):**



Step 3: Test cases and Algorithms:

A. Test cases:

Grayscale matrix	**GLCM Matrix**	**Normalized Matrix**	**GLCM Summation**	**Contrast**	**Energy**	**Homogeneity**
1 1 5 6 8 2 3 5 7 1 4 5 7 1 2 8 5 1 2 5	**0 0 0 0 0 0 0 0 0 0 1 2 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 2 0 0 0 0 0 0 0 0 0 1 0 2 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0**	**0 0 0 0 0 0 0 0 0 0 0.0625 0.125 0 0 0.0625 0 0 0 0 0 0 0.0625 0 0.0625 0 0 0 0 0 0 0 0 0.0625 0 0 0 0 0 0 0 0 0.0625 0 0 0 0 0.0625 0 0 0 0 0.0625 0.125 0 0 0 0 0 0 0 0 0 0.0625 0 0.125 0 0 0 0 0 0 0 0 0 0 0 0 0.0625 0 0 0**	**16**	**8.9375**	**0.0859375**	**0.291981**
1 1 1 1 1 1	**0 0 0 3**	**0 0 0 1**	**3**	**0**	**1**	**1**
1 2 3 1 2 3 1 2 3	**0 0 0 0 0 0 3 0 0 0 0 3 0 0 0 0**	**0 0 0 0 0 0 0.5 0 0 0 0 0.5 0 0 0 0**	**6**	**1**	**0.5**	**0.5**
1 4	**0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0**	**0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0**	**1**	**9**	**1**	**0.1**

For the first test case, we can notice that the biggest number in the Grayscale matrix. This number will be the dimensions of the GLCM Matrix and the Normalized Matrix. For the **GLCM Matrix**, we will loop through the grayscale matrix consecutively and each two numbers will be the new position of the index in the GLCM Matrix and it will be incremented based on the time of the occurrence of the indexes. The Normalized Matrix will be created by dividing the summation of the GLCM Matrix. Then the statistical information will be calculated based on the normalized matrix and using the given the formulas in the “gathering information” section

B. Algorithms:

```
Declare max_ArrNumber(int **arr, int rows, int cols) as integer function
    Declare max_num as integer and assign its value to arr[0][0]
    For int i = 0 to i < number of rows repeat
        For int j = 0 to i < number of columns repeat
            If (arr[i][j] greater than max_num)
                Max_num = arr[i][j]
    Return max_num
Declare GLCM_matrix(double **glcm, int **arr, int rows, int cols, int MAX_NUM) as void function
    For int i = 0 to i < number of rows repeat
        For int j = 0 to i < number of columns repeat
           Declare x as integer and assign to it arr[i][j]
           Declare y as integer and assign to it arr[i][j+1]
           Increment glcm[x][y] by 1
Declare normalize_GLCM(double **normalizedArr, double **glcm, int **arr, int rows, int cols, int MAX_NUM) as void function
    Declare sum as double and assign 0 to it
    For int i = 0 to i < MAX_NUM repeat
        For int j = 0 to i < MAX_NUM repeat
            Increment sum by glcm[i][j]
    For int i = 0 to i < MAX_NUM repeat
        For int j = 0 to i < MAX_NUM repeat
```

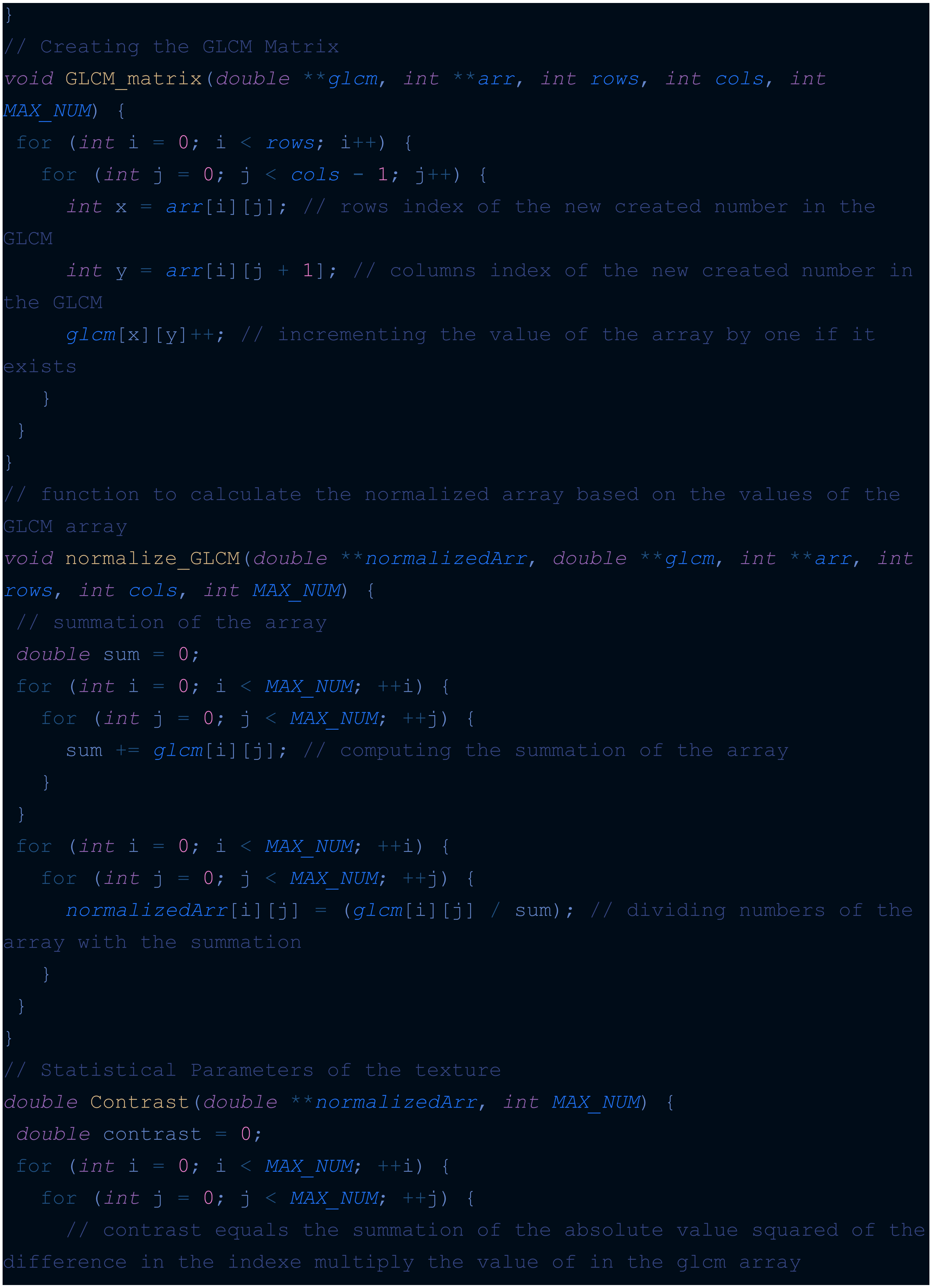
```
            Assign normalizedArr[i][j] to glcm[i][j] divided by sum
Declare Contrast(double **normalizedArr, int MAX_NUM) as double function
    Declare contrast as double and assign it to 0
    For int i = 0 to i < MAX_NUM repeat
        For int j = 0 to i < MAX_NUM repeat
            Increment contrast by (power two of the absolute value of (i - j) )* normalizedArr[i][j]
    Return contrast
Declare Energy(double **normalizedArr, int MAX_NUM) as double function
    Declare energy as double and assign it to 0
    For int i = 0 to i < MAX_NUM repeat
        For int j = 0 to i < MAX_NUM repeat
            Increment energy by power two of normalizedArr[i][j]
    Return energy
Declare Homogeneity(double **normalizedArr, int MAX_NUM) as double function
    Declare homo as double and assign it to 0
    For int i = 0 to i < MAX_NUM repeat
        For int j = 0 to i < MAX_NUM repeat
            Increment homo by (normalizedArr[i][j] divided by ( 1 + power two of the absolute value of (i-j))
    Return homo
MAIN:
    Declare choice, rows, cols, arr ptr ptr, MAX_NUM ,check(0) as integer
    Declare filenameIn, filenameOut as string
    Declare glcm ptr ptr, normalizedArr ptr ptr as double
    Declare filenameIn as input file
    Menu:
    Print”1- Create and normalize the GLCM matrix.‟ newline
    Print”2- Compute the statistical parameters of the texture.‟ newline
    Print”3- Print the GLCM matrix. ‟ newline
    Print”4- Exit. ”newline
    Print”Enter a Number: ”newline
    Read choice from the user
    if(choice equals 1)
        Increment check by 1
        files:
        Declare filenameIn as input file
        If(infile.fail())
            Print”file failed to operate.. Please enter again” newline
            goto files
        Read cols and rows from infile
        Create 2d dynamic array (arr) with dimensions rows and cols
    For int i = 0 to i < number of rows repeat
        For int j = 0 to i < number of columns repeat
            Read arr[i][j] from infile
    Assign MAX_NUM to max_ArrNumber(arr,rows,cols)
    Increment MAX_NUM
    Create 2d dynamic array (glcm) with dimensions MAX_NUM and MAX_NUM
```

```
    Call function GLCM_matrix(glcm,arr,rows,cols, MAX_NUM)
    Create 2d dynamic array (normalizedArr) with dimensions MAX_NUM and MAX_NUM
    Call function normalized_GLCM(normalizedArr,glcm, arr,rows,cols,MAX_NUM)
    Print”The Normalized Matrix is created” newline
    goto Menu
    Else if(choice equals 2 and check not equal 0)
        Print “The Contrast: ” call Contrast(normalizedArr, MAX_NUM) newline
        Print “The Energy: ” call Energy(normalizedArr, MAX_NUM) newline
        Print “The Homogeneity: ” call Homogeneity(normalizedArr, MAX_NUM) newline
        goto Menu
    Else if(choice equals 3 and check not equal 0)
        For int i = 0 to i < number of rows repeat
            For int j = 0 to i < number of columns repeat
                    Print glcm[i][j] “ “
            Print newline
        goto Menu
    Else if (choice equals 4)
        Print “Thanks for using the program….exit” newline
    Else
        if(check equals 0)
            Print “enter 1 please: ” newline
        Else if(choice > 4)
            Print “Invalid Input, enter again: “ newline
    Goto Menu
```

```
For int i = 0 to i < rows repeat
     Delete[] arr[i]
Delete[] arr
For int i = 0 to i < MAX_NUM repeat
     Delete[] glcm[i]
Delete[] glcm
For int = 0 to i < MAX_NUM repeat
     Delete[] normalizedArr[i]
Delete[] normalizedArr
Close infile
Return 0
```

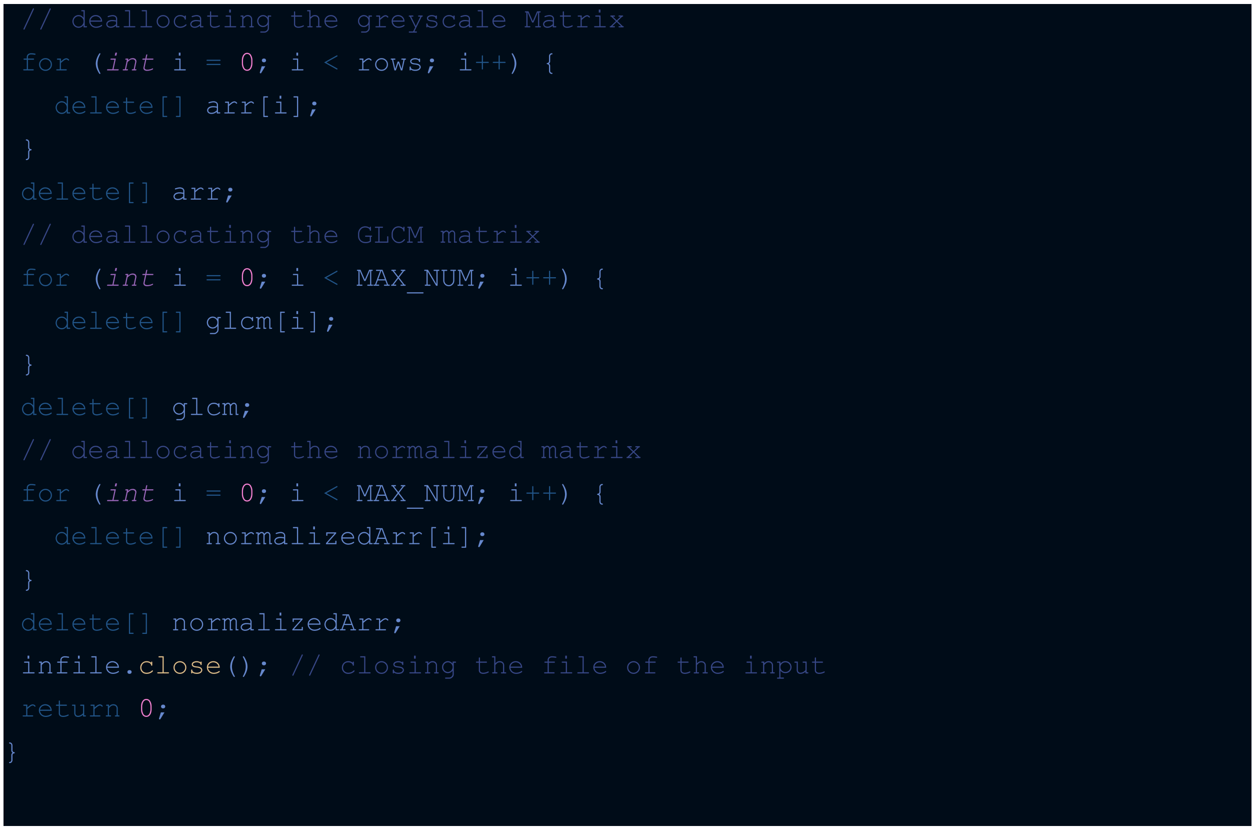
Step 4: Code

```
/*--------------------------*/
/* Omar Mohamed Atia Shehab */
/* Date:   November 13, 2022   */
/*       Assignment2        */
/* Image Features Extraction using GLCM  */
#include <cmath>
#include <fstream>
#include <iostream>
#include <string>
using namespace std;
// Function to find the Maximum number in the Grey Scale Matrix
int max_ArrNumber(int **arr, int rows, int cols) {
 int max_num = arr[0][0];
 for (int i = 0; i < rows; i++) {
   for (int j = 0; j < cols; j++) {
     if (arr[i][j] > max_num) {
       max_num = arr[i][j]; // assigning the max number to the variable
     }
   }
 }
 return max_num;
```



```
}
// Creating the GLCM Matrix
void GLCM_matrix(double **glcm, int **arr, int rows, int cols, int
MAX_NUM) {
 for (int i = 0; i < rows; i++) {
   for (int j = 0; j < cols - 1; j++) {
     int x = arr[i][j]; // rows index of the new created number in the
GLCM
     int y = arr[i][j + 1]; // columns index of the new created number in
the GLCM
     glcm[x][y]++; // incrementing the value of the array by one if it
exists
   }
 }
}
// function to calculate the normalized array based on the values of the
GLCM array
void normalize_GLCM(double **normalizedArr, double **glcm, int **arr, int
rows, int cols, int MAX_NUM) {
 // summation of the array
 double sum = 0;
 for (int i = 0; i < MAX_NUM; ++i) {
   for (int j = 0; j < MAX_NUM; ++j) {
     sum += glcm[i][j]; // computing the summation of the array
   }
 }
 for (int i = 0; i < MAX_NUM; ++i) {
   for (int j = 0; j < MAX_NUM; ++j) {
     normalizedArr[i][j] = (glcm[i][j] / sum); // dividing numbers of the
array with the summation
   }
 }
}
// Statistical Parameters of the texture
double Contrast(double **normalizedArr, int MAX_NUM) {
 double contrast = 0;
 for (int i = 0; i < MAX_NUM; ++i) {
   for (int j = 0; j < MAX_NUM; ++j) {
     // contrast equals the summation of the absolute value squared of the
difference in the indexe multiply the value of in the glcm array
```

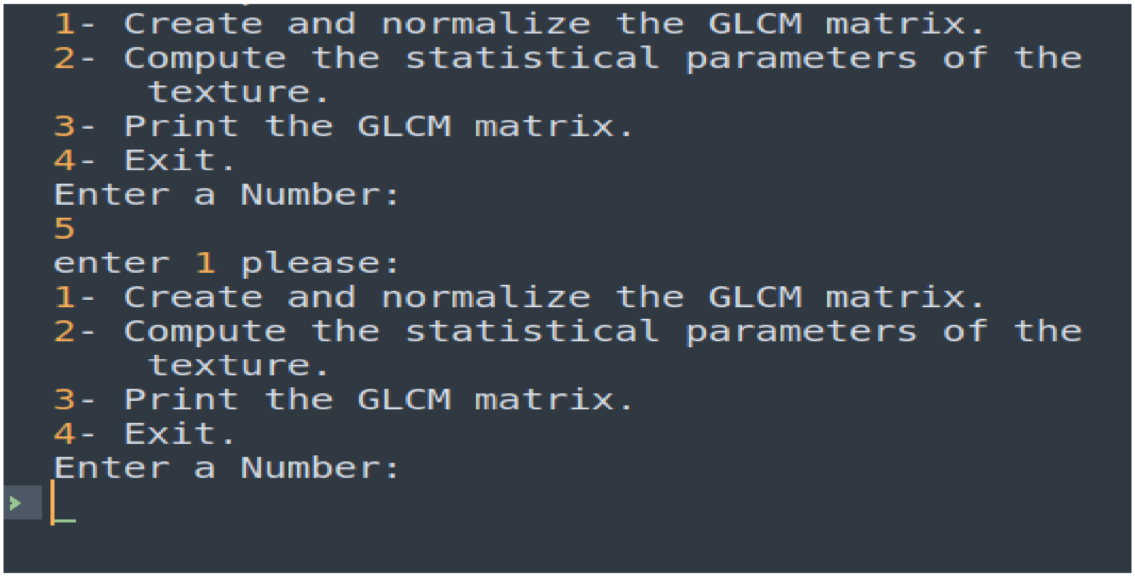
```
     contrast += (pow(abs(i - j), 2) * normalizedArr[i][j]);
   }
 }
 return contrast;
}
double Energy(double **normalizedArr, int MAX_NUM) {
 double energy = 0;
 for (int i = 0; i < MAX_NUM; i++) {
   for (int j = 0; j < MAX_NUM; j++) {
     // energy equals the summation of the values of the glcm matrix
squared
     energy += pow(normalizedArr[i][j], 2);
   }
 }
 return energy;
}
double Homogeneity(double **normalizedArr, int MAX_NUM) {
 double homo = 0;
 for (int i = 0; i < MAX_NUM; i++) {
   for (int j = 0; j < MAX_NUM; j++) {
     homo += (normalizedArr[i][j] / (1 + pow(abs(i - j), 2)));
   }
 }
 return homo;
}
int main() {
 // The choice the user has to choose, the rows and columns of the grey
scale array
 // intializing the grayscale array and the maximum number of the array
variable
 int choice, rows, cols, **arr, MAX_NUM ,check = 0;
 string filenameIn, filenameOut; // files to store the data of the grey
scale and print the results
 double **glcm, **normalizedArr; // intrializing the glcm matrix and the
normalized matrix
 ifstream infile(filenameIn);//intializing the input file
 // Displaying the menu function serveral times for the user to print the
results of what he chooses
Menu:
 // calling the menu function
 cout << "1- Create and normalize the GLCM matrix. \n";
 cout << "2- Compute the statistical parameters of the texture. \n";
 cout << "3- Print the GLCM matrix. \n";
 cout << "4- Exit. \n";
 cout << "Enter a Number: \n";
 cin >> choice;
 if (choice == 1) {
   check++; // incrementing check varible to let the other processes to be
created (GLCM, Normalized Matrixes)
   files:
   cout << "Please enter the name of the file: \n";
   cin >> filenameIn;
   ifstream infile(filenameIn);
   // checking if the file exists or not
   if (infile.fail()) {
     cout << "file failed to operate.. please enter again\n";
     goto files;
   }
   // geting the data of the matrix from the file
   infile >> cols >> rows;
   // Creatring the Grey Scale Matrix
   arr = new int *[rows];
   for (int i = 0; i < rows; i++) {
     arr[i] = new int[cols];
   }
   for (int i = 0; i < rows; i++) {
     for (int j = 0; j < cols; j++) {
       infile >>arr[i][j]; // entering the values of the grey scale array
from the file
     }
   }
   // assigning the value that the max function will return to a variable
   MAX_NUM = max_ArrNumber(arr, rows, cols);
   MAX_NUM++; // incrememting the value of the matrix to print till 8
columns and
              // rows
   // GLCM Array
   glcm = new double *[MAX_NUM];
   for (int i = 0; i < MAX_NUM; ++i) {
     glcm[i] = new double[MAX_NUM];
   }
   GLCM_matrix(glcm, arr, rows, cols, MAX_NUM); // calling the function
here for the sake of creating the normalized Matrix
   normalizedArr = new double *[MAX_NUM];
   for (int i = 0; i < MAX_NUM; ++i) {
     normalizedArr[i] = new double[MAX_NUM];
   }
   // calling the function to create the normalized Matrix
   normalize_GLCM(normalizedArr, glcm, arr, rows, cols, MAX_NUM);
   cout << "The Normalized Matrix is created\n";
   goto Menu;
 } else if (choice == 2 && check != 0) {
   // computing the statistical parameters
   cout << "The Contrast: " << Contrast(normalizedArr, MAX_NUM) << "\n";
   cout << "The Energy: " << Energy(normalizedArr, MAX_NUM) << "\n";
   cout << "The Homogeneity: " << Homogeneity(normalizedArr, MAX_NUM) <<
"\n";
   goto Menu;
 } // when user chooses option 3
 else if (choice == 3 && check != 0) {
   // printing the matrix into file to display this matrix later
   for (int i = 0; i < MAX_NUM; ++i) {
     for (int j = 0; j < MAX_NUM; ++j) {
       cout << glcm[i][j] << " ";
     }
     cout << endl;
   }
   goto Menu;
 } else if (choice == 4) {
   // exiting the porgram
   cout << "Thanks for using the program....exit\n";
 } else {
   // if the user enetered an invalid input
   if(check == 0){
     cout<<"enter 1 please: \n";
   }else if(choice > 4){
     cout << "Invalid Input, enter again: \n";
   }
   goto Menu;
 }
```



```
  // deallocating the greyscale Matrix
  for (int i = 0; i < rows; i++) {
    delete[] arr[i];
  }
  delete[] arr;
  // deallocating the GLCM matrix
  for (int i = 0; i < MAX_NUM; i++) {
    delete[] glcm[i];
  }
  delete[] glcm;
  // deallocating the normalized matrix
  for (int i = 0; i < MAX_NUM; i++) {
    delete[] normalizedArr[i];
  }
  delete[] normalizedArr;
  infile.close(); // closing the file of the input
  return 0;
}
```

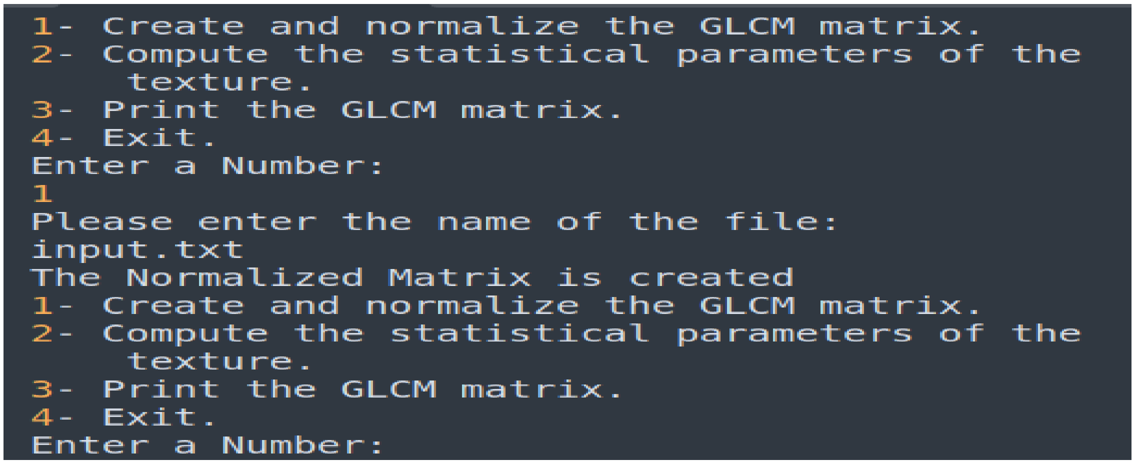
Step 5: Test and verification:

1. **Test 1: enter invalid input rather than the options in the menu**



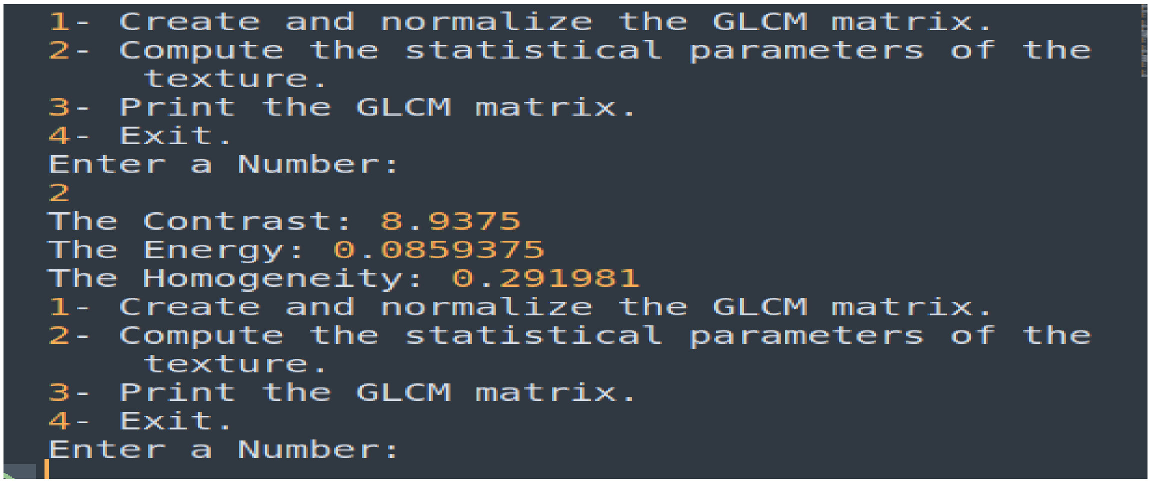
```
1- Create and normalize the GLCM matrix.
2- Compute the statistical parameters of the
    texture.
3- Print the GLCM matrix.
4- Exit.
Enter a Number:
5
enter 1 please:
1- Create and normalize the GLCM matrix.
2- Compute the statistical parameters of the
    texture.
3- Print the GLCM matrix.
4- Exit.
Enter a Number:
```

2. **Test 2: option 1**



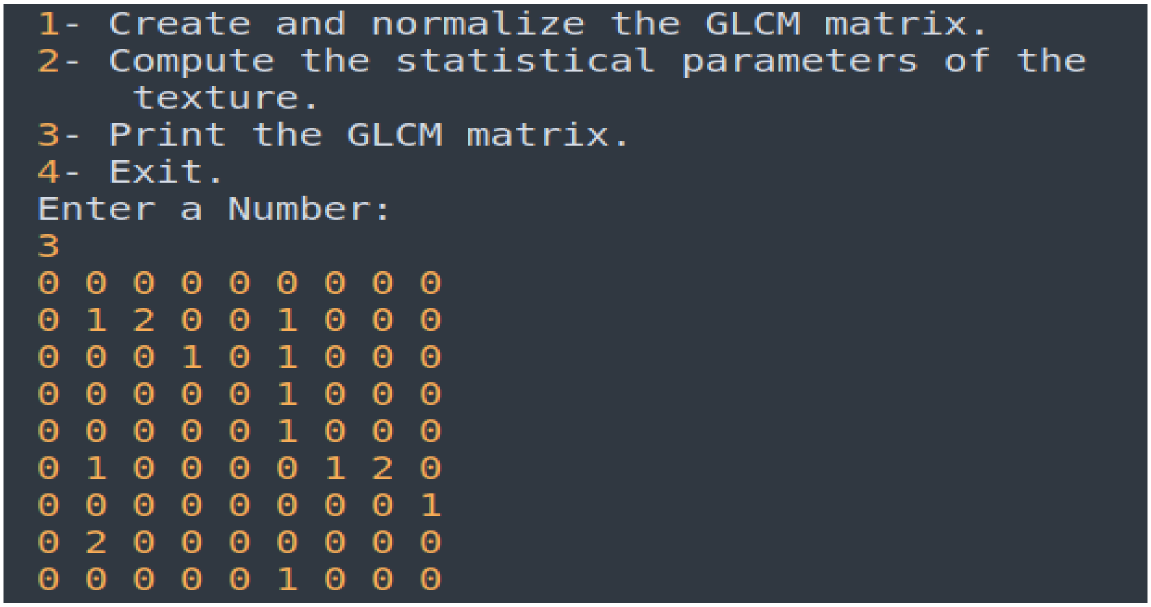
```
1- Create and normalize the GLCM matrix.
2- Compute the statistical parameters of the
   texture.
3- Print the GLCM matrix.
4- Exit.
Enter a Number:
1
Please enter the name of the file:
input.txt
The Normalized Matrix is created
1- Create and normalize the GLCM matrix.
2- Compute the statistical parameters of the
   texture.
3- Print the GLCM matrix.
4- Exit.
Enter a Number:
```

3. **Test 3: option 2 calculations**



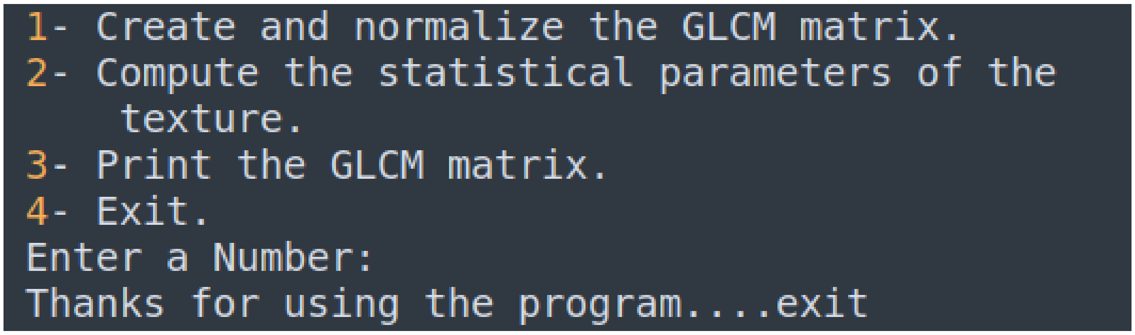
```
1- Create and normalize the GLCM matrix.
2- Compute the statistical parameters of the
   texture.
3- Print the GLCM matrix.
4- Exit.
Enter a Number:
2
The Contrast: 8.9375
The Energy: 0.0859375
The Homogeneity: 0.291981
1- Create and normalize the GLCM matrix.
2- Compute the statistical parameters of the
   texture.
3- Print the GLCM matrix.
4- Exit.
Enter a Number:
```

4. **Test 4: option 3 calculations**



```
1- Create and normalize the GLCM matrix.
2- Compute the statistical parameters of the
   texture.
3- Print the GLCM matrix.
4- Exit.
Enter a Number:
3
0 0 0 0 0 0 0 0 0
0 1 2 0 0 1 0 0 0
0 0 0 1 0 1 0 0 0
0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0 0
0 1 0 0 0 0 1 2 0
0 0 0 0 0 0 0 0 1
0 2 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0
```

5. **Test 5: option 4 calculations**



```
1- Create and normalize the GLCM matrix.
2- Compute the statistical parameters of the
    texture.
3- Print the GLCM matrix.
4- Exit.
Enter a Number:
Thanks for using the program....exit
```

Verification:

- **The menu accepts only four options: 1, 2, 3, and 4.**
- **The Grayscale matrix can’t contain a negative number. All the test cases should be positive numbers.**
- **The input file should be created before running the program and it’s preferred to name it input.txt.**
- **The columns should be provided before the rows in the input file.**
- **The columns should be greater than or equal to 1.**