University of Mila

Faculty of Science and Technology

Department of Process Engineering

Practical Work 06: Solutions

Course: Introduction to Programming

Level: 1st Year ST - ENG & LMD

Semester 02

By:

Dr. Farouk KECITA

Academic Year: 2025/2026

Exercise 01: 1D Array Conversion Using Functions

Question: Write a C program that uses functions to:

- Fill an array of size N with user inputs
- Reverse the array
- Find the minimum and maximum elements
- Calculate the sum and average

Program:

```c
#include <stdio.h>
#define MAX 100

// Function prototypes
void fillArray(int arr[], int n);
void displayArray(int arr[], int n);
void reverseArray(int arr[], int n);
int findMin(int arr[], int n);
int findMax(int arr[], int n);
int sumArray(int arr[], int n);
float averageArray(int arr[], int n);

int main() {
    int arr[MAX], n;
    int min, max, sum;
    float avg;

    printf("Enter number of elements (n): ");
    scanf("%d", &n);

    fillArray(arr, n);

    printf("\nOriginal array: ");
    displayArray(arr, n);

    reverseArray(arr, n);
    printf("Reversed array: ");
    displayArray(arr, n);

    min = findMin(arr, n);
    max = findMax(arr, n);
    sum = sumArray(arr, n);
    avg = averageArray(arr, n);

    printf("\nMinimum element: %d\n", min);
    printf("Maximum element: %d\n", max);
    printf("Sum: %d\n", sum);
    printf("Average: %.2f\n", avg);
    return 0;
}
```

```c

void fillArray(int arr[], int n) {
    for(int i = 0; i < n; i++) {
        printf("arr[%d] = ", i);
        scanf("%d", &arr[i]);
    }
}

void displayArray(int arr[], int n) {
    for(int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

void reverseArray(int arr[], int n) {
    int temp[n];
    for(int i = 0; i < n; i++) {
        temp[i] = arr[n-1-i];
    }
    for(int i = 0; i < n; i++) {
        arr[i] = temp[i];
    }
}

int findMin(int arr[], int n) {
    int min = arr[0];
    for(int i = 1; i < n; i++) {
        if(arr[i] < min) min = arr[i];
    }
    return min;
}
int findMax(int arr[], int n) {
    int max = arr[0];
    for(int i = 1; i < n; i++) {
        if(arr[i] > max) max = arr[i];
    }
    return max;
}

int sumArray(int arr[], int n) {
    int sum = 0;
    for(int i = 0; i < n; i++) {
        sum += arr[i];
    }
    return sum;
}

float averageArray(int arr[], int n) {
    return (float)sumArray(arr, n) / n;
}
```

Sample Output

```
Enter number of elements (n): 5
arr[0] = 10
arr[1] = 20
arr[2] = 30
arr[3] = 40
arr[4] = 50

Original array: 10 20 30 40 50
Reversed array: 50 40 30 20 10

Minimum element: 10
Maximum element: 50
Sum: 150
Average: 30.00
```

Exercise 02: Matrix Multiplication Using Functions

Question: Write a C program that uses functions to multiply two matrices A (m×n) and B (n×p). The program should:

- Read matrix dimensions and elements
- Check if multiplication is possible
- Compute the product using a separate function
- Display the result

Program:

```c
#include <stdio.h>
#define MAX 10

// Function prototypes
void readMatrix(int mat[MAX][MAX], int rows, int cols, char name);
void displayMatrix(int mat[MAX][MAX], int rows, int cols, char name);
int multiplyMatrices(int A[MAX][MAX], int B[MAX][MAX], int
   C[MAX][MAX],
                     int m, int n,int l,int p);

int main() {
    int A[MAX][MAX], B[MAX][MAX], C[MAX][MAX];
    int m, n, p,l;

    printf("Matrix A dimensions:\n");
    printf("Enter rows (m): ");
    scanf("%d", &m);
    printf("Enter columns (n): ");
    scanf("%d", &n);

```

```
    printf("\nMatrix B dimensions:\n");
    printf("Enter rows (n): must equal %d\n", n);
    scanf("%d", &l);
    printf("Enter columns (p): ");
    scanf("%d", &p);

    // Read matrices
    readMatrix(A, m, n, 'A');
    readMatrix(B, l, p, 'B');

    // Display input matrices
    displayMatrix(A, m, n, 'A');
    displayMatrix(B, l, p, 'B');

    // Multiply matrices
    if(multiplyMatrices(A, B, C, m, n, l, p)) {
        displayMatrix(C, m, p, 'C');
    } else {
        printf("\nMatrix multiplication not possible!\n");
    }

    return 0;
}

void readMatrix(int mat[MAX][MAX], int rows, int cols, char name) {
    printf("\nEnter elements of matrix %c:\n", name);
    for(int i = 0; i < rows; i++) {
        for(int j = 0; j < cols; j++) {
            printf("%c[%d][%d] = ", name, i, j);
            scanf("%d", &mat[i][j]);
        }
    }
}

void displayMatrix(int mat[MAX][MAX], int rows, int cols, char name)
   {
    printf("\nMatrix %c (%dx%d):\n", name, rows, cols);
    for(int i = 0; i < rows; i++) {
        for(int j = 0; j < cols; j++) {
            printf("%d\t", mat[i][j]);
        }
        printf("\n");
    }
}

int multiplyMatrices(int A[MAX][MAX], int B[MAX][MAX], int
   C[MAX][MAX],
                     int m, int n, int l, int p) {
    if (n != l) {
        printf("Cannot multiply: columns of A (%d) != rows of B
           (%d)\n", n, l);
```

```
        return 0;
    }

    for(int i = 0; i < m; i++)
        for(int j = 0; j < p; j++) {
            C[i][j] = 0;
            for(int k = 0; k < n; k++)
                C[i][j] += A[i][k] * B[k][j];
        }
    return 1;
}
```

Sample Output

```
Matrix A dimensions:
Enter rows (m): 2
Enter columns (n): 3
Matrix B dimensions:
Enter rows (n): must equal 3
3
Enter columns (p): 2
Enter elements of matrix A:
A[0][0] = 1
A[0][1] = 2
A[0][2] = 3
A[1][0] = 4
A[1][1] = 5
A[1][2] = 6
Enter elements of matrix B:
B[0][0] = 7
B[0][1] = 8
B[1][0] = 9
B[1][1] = 10
B[2][0] = 11
B[2][1] = 12
Matrix A (2x3):
1       2       3
4       5       6
Matrix B (3x2):
7       8
9       10
11      12
Matrix C (2x2):
58      64
139     154
```

Exercise 03: Solving System of 2 Linear Equations

Question: Solve the following system of linear equations using the inverse matrix method:

$$\begin{cases} a_{11}x + a_{12}y = b_1 \\ a_{21}x + a_{22}y = b_2 \end{cases}$$

The solution is given by:

$$\begin{pmatrix} x \\ y \end{pmatrix} = A^{-1} \cdot B$$

where $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$, $B = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$

For a 2×2 matrix, $A^{-1} = \frac{1}{\det(A)} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}$, where $\det(A) = a_{11}a_{22} - a_{12}a_{21}$

Program:

```c
#include <stdio.h>
#include <math.h>

#define SIZE 2

int main() {
    float A[SIZE][SIZE], adjA[SIZE][SIZE], invA[SIZE][SIZE],
       B[SIZE], X[SIZE], det;

    printf("=== Solving System Using Inverse Matrix ===\n\n");

    // Input matrix A
    printf("Enter matrix A:\n");
    for(int i = 0; i < SIZE; i++)
        for(int j = 0; j < SIZE; j++) {
            printf("A[%d][%d] = ", i, j);
            scanf("%f", &A[i][j]);
        }

    // Input matrix B
    printf("\nEnter matrix B:\n");
    for(int i = 0; i < SIZE; i++) {
        printf("B[%d] = ", i);
        scanf("%f", &B[i]);
    }

    // Display the system
    printf("\nSystem:\n");
    printf("%.2f x + %.2f y = %.2f\n", A[0][0], A[0][1], B[0]);
    printf("%.2f x + %.2f y = %.2f\n", A[1][0], A[1][1], B[1]);

    // Calculate determinant
    det = A[0][0] * A[1][1] - A[0][1] * A[1][0];
    printf("\ndet = %.2f\n", det);

    if(det==0) {
```

```
        printf("\nError: Determinant = 0, inverse does not exist!\n");
    return 0;
    }

    // Calculate adjoint matrix
    adjA[0][0] = A[1][1];
    adjA[0][1] = -A[0][1];
    adjA[1][0] = -A[1][0];
    adjA[1][1] = A[0][0];

    // Calculate inverse matrix
    for(int i = 0; i < SIZE; i++)
        for(int j = 0; j < SIZE; j++)
            invA[i][j] = adjA[i][j] / det;

    // Calculate solution: X = invA * B (using loops)
    for(int i = 0; i < SIZE; i++) {
        X[i] = 0;
        for(int j = 0; j < SIZE; j++) {
            X[i] += invA[i][j] * B[j];
        }
    }

    // Display solution
    printf("\nSolution:\n");
    printf("x = %.2f\n", X[0]);
    printf("y = %.2f\n", X[1]);
    return 0;
}
```

Sample Output

```
=== Solving System Using Inverse Matrix ===
Enter matrix A:
A[0][0] = 2
A[0][1] = 3
A[1][0] = 1
A[1][1] = 4
Enter matrix B:
B[0] = 8
B[1] = 9
System:
2.00 x + 3.00 y = 8.00
1.00 x + 4.00 y = 9.00
det = 5.00
Solution:
x = 1.00
y = 2.00
```

Exercise 04: Solving Linear Equations Using Functions

Question: Rewrite Exercise 03 using functions to:

- Calculate the determinant of a 2×2 matrix
- Compute the inverse matrix
- Solve the system using a separate function

Program:

```c
#include <stdio.h>
#include <math.h>

#define SIZE 2

// Function prototypes
void readMatrixA(float A[SIZE][SIZE]);
void readMatrixB(float B[SIZE]);
float calculateDeterminant(float A[SIZE][SIZE]);
int inverseMatrix(float A[SIZE][SIZE], float invA[SIZE][SIZE]);
void solveSystem(float A[SIZE][SIZE], float B[SIZE], float X[SIZE]);
void displaySystem(float A[SIZE][SIZE], float B[SIZE]);
void verifySolution(float A[SIZE][SIZE], float X[SIZE], float
   B[SIZE]);

int main() {
    float A[SIZE][SIZE], B[SIZE], X[SIZE];

    printf("=== Solving 2x2 Linear System Using Functions ===\n\n");

    readMatrixA(A);
    readMatrixB(B);

    displaySystem(A, B);
    solveSystem(A, B, X);

    return 0;
}

// Function to read matrix A
void readMatrixA(float A[SIZE][SIZE]) {
    printf("Enter coefficients of matrix A:\n");
    printf("a11 = ");
    scanf("%f", &A[0][0]);
    printf("a12 = ");
    scanf("%f", &A[0][1]);
    printf("a21 = ");
    scanf("%f", &A[1][0]);
    printf("a22 = ");
    scanf("%f", &A[1][1]);
}

```

```c
// Function to read matrix B
void readMatrixB(float B[SIZE]) {
    printf("\nEnter constants of matrix B:\n");
    printf("b1 = ");
    scanf("%f", &B[0]);
    printf("b2 = ");
    scanf("%f", &B[1]);
}

float calculateDeterminant(float A[SIZE][SIZE]) {
    return A[0][0] * A[1][1] - A[0][1] * A[1][0];
}

int inverseMatrix(float A[SIZE][SIZE], float invA[SIZE][SIZE]) {
    float det = calculateDeterminant(A);

    if(det==0) return 0;

    invA[0][0] = A[1][1] / det;
    invA[0][1] = -A[0][1] / det;
    invA[1][0] = -A[1][0] / det;
    invA[1][1] = A[0][0] / det;

    return 1;
}

void solveSystem(float A[SIZE][SIZE], float B[SIZE], float X[SIZE]) {
    float invA[SIZE][SIZE];

    if(inverseMatrix(A, invA)) {
        X[0] = invA[0][0] * B[0] + invA[0][1] * B[1];
        X[1] = invA[1][0] * B[0] + invA[1][1] * B[1];

        printf("\nSolution:\n");
        printf("x = %.2f\n", X[0]);
        printf("y = %.2f\n", X[1]);

        verifySolution(A, X, B);
    } else {
        printf("\nError: Determinant = 0\n");
    }
}

void displaySystem(float A[SIZE][SIZE], float B[SIZE]) {
    printf("\nSystem:\n");
    printf("%.2f x + %.2f y = %.2f\n", A[0][0], A[0][1], B[0]);
    printf("%.2f x + %.2f y = %.2f\n", A[1][0], A[1][1], B[1]);
}

void verifySolution(float A[SIZE][SIZE], float X[SIZE], float
   B[SIZE]) {
```

```
    float check1, check2;

    printf("\nVerification:\n");
    check1 = A[0][0] * X[0] + A[0][1] * X[1];
    check2 = A[1][0] * X[0] + A[1][1] * X[1];

    printf("%.2f = %.2f\n", check1, B[0]);
    printf("%.2f = %.2f\n", check2, B[1]);
}
```

Sample Output

```
=== Solving 2x2 Linear System Using Functions ===

Enter coefficients:
Enter a11: 2
Enter a12: 3
Enter a21: 1
Enter a22: 4
Enter constants:
Enter b1: 8
Enter b2: 9

System of equations:
2.00 x + 3.00 y = 8.00
1.00 x + 4.00 y = 9.00

Determinant = 5.00

Solutions:
x = 1.00
y = 2.00

Verification:
Equation 1: 8.00 (expected 8.00)
Equation 2: 9.00 (expected 9.00)
```