

CPE101 Programming Languages I

Week 11

Multidimensional Arrays

Assoc. Prof. Dr. Caner ÖZCAN

Multidimensional Arrays

- ▶ An array can have more than one-dimensional.
- ▶ For example we will use 2-dimensional array for 3x4 matrix.
- ▶ In three-dimensional Euclidean space x, y, z , we prefer a 3-dimensional array to store our points.
- ▶ We write $M[i][j]$ to reach the elements in row i and column j .
 - For example, two-dimensional array (matrix) is defined as follows.
 - `int M[5][9]; /* has 5 rows and 9 columns */`
 - Conceptually, array M is similar to the following.

	0	1	2	3	4	5	6	7	8
0									
1									
2									
3									
4									

Reach 2-Dimensional Arrays

```
/* initialization */  
for (i=0; i<5; i++){  
    for (j=0; j<9; j++){  
        M[i][j] = 0;  
    }  
}
```

```
/* Sum */  
sum = 0;  
for (i=0; i<5; i++){  
    for (j=0; j<9; j++){  
        sum += M[i][j];  
    }  
}
```

```
/* finding min and max */  
min = M[0][0];  
max = M[0][0];  
for (i=0; i<5; i++){  
    for (j=0; j<9; j++){  
        if (M[i][j]<min)  
            min=M[i][j];  
        if (M[i][j]>max)  
            max=M[i][j];  
    }  
}  
printf("min: %d, max: %d\n", min, max);
```

Initialization of Multi-Dimensional Arrays

- ▶ Nested one-dimensional arrays can be used to assign initial values to the multi-dimensional arrays.

```
int M[5][9] = { {1, 1, 1, 1, 0, 1, 1, 1, 1},  
                {0, 1, 0, 1, 0, 1, 0, 1, 0},  
                {1, 0, 0, 1, 1, 1, 0, 0, 1},  
                {0, 0, 0, 0, 1, 1, 1, 1, 1},  
                {1, 1, 1, 1, 0, 0, 0, 1, 1}};
```

- ▶ If the first value is less than the rest, a multi-dimensional array element is filled with 0.

```
int M[5][9] = { {1, 1, 2, 1, 2, 0, 0, 1, 1},  
                {0, 0, 0, 1, 1, 1, 1, 2, 1}};  
/* 2., 3. and 4. rows are filled with zero*/
```

Initialization of Multi-Dimensional Arrays

- ▶ If it is less than the number of elements of a row in the list contains, the remainder of the line is filled with 0.

```
int M[5][9] = { {1, 1, 0, 0, 1, 1, 1, 1, 1},  
                {0, 1, 1, 2, 1, 1},  
                {1, 1, 2, 2, 3}};  
  
/* M[1][6], M[1][7], M[1][8] will be 0*/  
/* M[2][5], M[2][6], M[2][7], M[2][8] will be 0*/  
/* 3. and 4. rows will filled with 0 */
```

Higher Dimensional Arrays

- ▶ An array can be defined in any size.

```
int Cube[8][8][8];    /* a cube with size 8 */
int Prism[4][6][10]; /* a rectangular prism having
                    * dimensions 4x6x10
                    */
/* initialization can be done*/
float A[4][6][8] = {{{1, 2, 3}, {3, 4}}, {{3, 4}}};

Cube[2][3][4] = 2;
Prism[3][5][8] = 6;
A[0][0][4] = 3.34;
```

Multidimensional Arrays

- ▶ 8 tests are applied to a group of 5 students.
- ▶ Let's use 2-dimensional array to store their results.

```
#include<stdio.h>
int main( void ) {
    // Creating 5x8 matrix.
    int student_table[ 5 ][ 8 ];
    int i, j;
    for( i = 0; i < 5; i++ ) {
        for( j = 0; j < 8; j++ ) {
            printf( "%d no. student's ", ( i + 1 ) );
            printf( "%d no. exam> ", ( j + 1 ) );
            // We take value as the one-dimensional array.
            scanf( "%d", &student_table[ i ][ j ] );
        }
    }
    return 0;
}
```

Multidimensional Arrays

- ▶ Run this program, imagine we assign different grades values to the students. To put this into a visual form, it consists of a table as follows:

8 Exam

80	76	58	90	27	60	85	95
60	59	75	80	82	79	64	87
77	...						
				...	67	60	84

5 Student

6. Exam of 5. Student

- ▶ According to the table, 1. student seems to have taken 80, 76, 58, 90, 27, 60, 85 and 95 points. Or we understand that the 5. student takes the 67 from 6. exam. Similarly the necessary values assigned to other cells and related student's exam grades is being held in ₈ memory.

Example: Sum of Two Matrices

```
#include <stdio.h>
#define SAT 2
#define SUT 3

int main(){
    int a[SAT][SUT] = {5, 3, 7, 0, 1, 2};

    int b[SAT][SUT] = {1, 2, 3, 4, 5, 6};
    int c[SAT][SUT];
    int i, j;

    puts("A Matrix:");
    for(i=0; i<SAT; i++){
        for(j=0; j<SUT; j++)
            printf("%4d",a[i][j]);
        printf("\n");
    }
}
```

Example: Sum of Two Matrices

```
puts("B Matrix:");
for(i=0; i<SAT; i++){
    for(j=0; j<SUT; j++)
        printf("%4d",b[i][j]);
    printf("\n");
}
puts("\nC Matrix:");
for(i=0; i<SAT; i++){
    for(j=0; j<SUT; j++){
        c[i][j] = a[i][j] + b[i][j];
        printf("%4d",c[i][j]);
    }
    printf("\n");
}
return 0;
}
```

Example: Array to Matrix Conversion

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int dizi[100];
    int a[100][100];
    int i, j, n, x, sat, sut;
    printf("«How many elements does your array contain? > ");
    scanf("%d", &n);
    for(x=0; x<n; x++){
        printf("«Enter [%d]. element > ",x+1);
        scanf("%d",&dizi[x]);
    }
    printf("\n Enter matrix row count > ");
    scanf("%d", &sat);
    printf(" Enter matrix column count > ");
    scanf("%d",&sut);
```

Example: Array to Matrix Conversion

```
if(n%sat==0 && n%sut==0){
    x=0;
    for(i=0; i<sat; i++){
        for (j=0; j<sut; j++){
            a[i][j]=dizi[x];
            x++;
        }
    }
    printf("\n\n MATRIX > \n");
    for(i=0; i<sat; i++){
        for(j=0; j<sut; j++)
            printf("%3d", a[i][j]);
        printf("\n");
    }
} else
    printf("ERROR! Number of Array Elements can be full divided by row or column");
return 0;
}
```

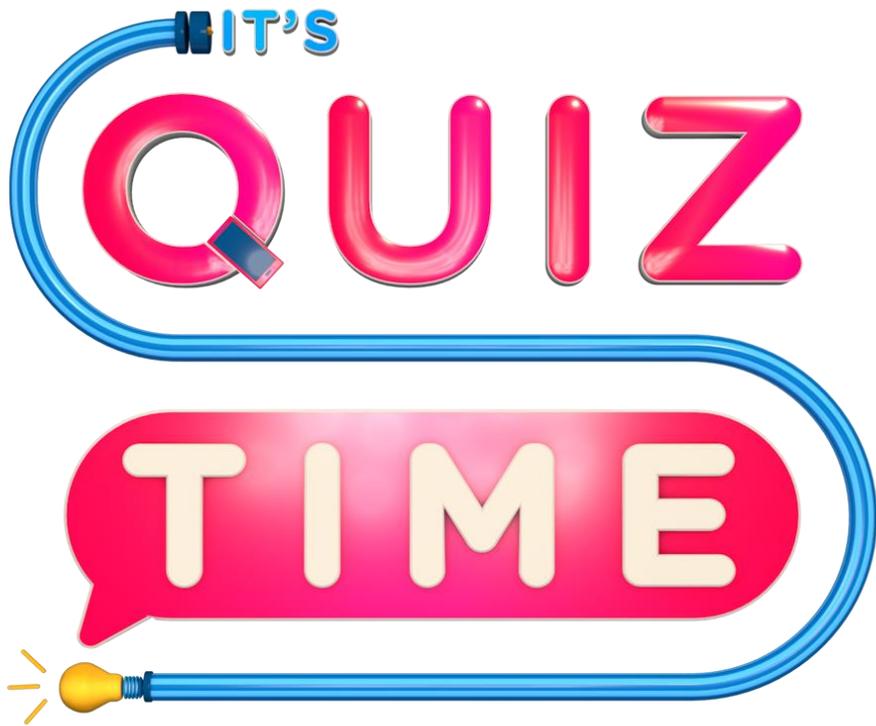
Example: Symmetric Matrix

```
#include <stdio.h>
#include <conio.h>
int main(){
    int a[100][100];
    int symmetry =1;
    int x, y, i, j;
    printf("Row size of matrix > ");
    scanf("%d", &x);
    printf("Column size of matrix > ");
    scanf("%d", &y);
    printf("Enter matrix values > ");
    for(i=0; i<x; i++) {
        for(j=0; j<y; j++) {
            printf("\n Value [%d] [%d] --> ", i+1, j+1);
            scanf("%d", &a[i][j]);
        }
    }
}
```

Example: Symmetric Matrix

```
//NOTE: if a[i][j]==a[j][i] this matrix is symmetric.
for(i=0; i<x; i++){
    for(j=0; j<y; j++){
        if(a[i][j]!=a[j][i])
            symmetry=0;
            break;
        }
    }
}
if(symmetry ==1)
    printf("\n Matrix is symmetric.\n");
else
    printf("\n Matrix is not symmetric\n");

return 0;
}
```



Write C program to find reverse of the given array.

References

- ▶ Doç. Dr. Fahri Vatansever, “Algoritma Geliştirme ve Programlamaya Giriş”, Seçkin Yayıncılık, 12. Baskı, 2015.
- ▶ J. G. Brookshear, “Computer Science: An Overview 10th Ed.”, Addison Wisley, 2009.
- ▶ Kaan Aslan, “A’dan Z’ye C Klavuzu 8. Basım”, Pusula Yayıncılık, 2002.
- ▶ Paul J. Deitel, “C How to Program”, Harvey Deitel.
- ▶ Bayram AKGÜL, C Programlama Ders notları