CPE101 Programming Languages I

Week 10
Arrays

Assist. Prof. Dr. Caner ÖZCAN

Arrays

- In the operations performed with the help of computers, it may be necessary to enter a large number of data and process the entered data according to a certain systematic.

- Processing of data in a specific order is both easier and more practical.

- Therefore, computer programs use sequential data fields called "arrays" to process multiple data.

- These data fields, named by a single name, are placed in memory consecutively.

Arrays

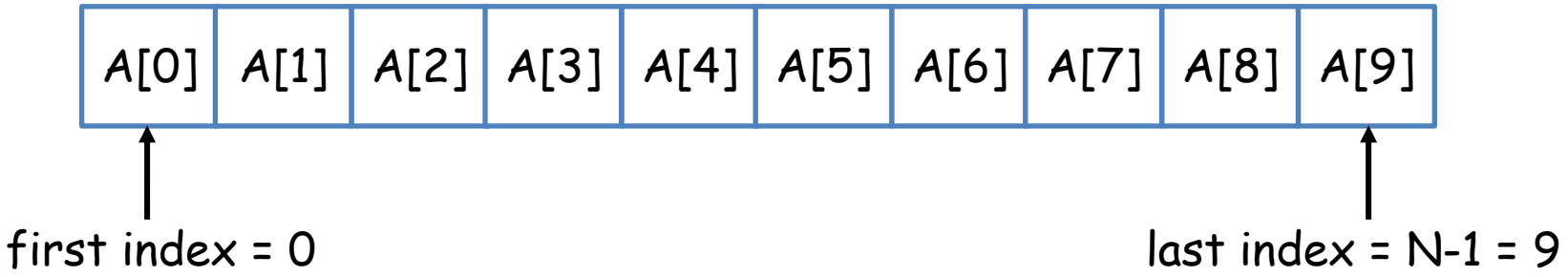
- We would need more than one variable of the same type for the same purpose.
- For example, a class of 100 students take "Programming Languages" course and have grades. Rather than to use individual variables (100-variable name required), these grades can be stored in an array called "Notlar".
- This way, many variable names and fields are not required.
- Information is kept under a specific structure with a single name and processed quickly.

Arrays

- Data structure that holds multiple variables of the same type together.
- The simplest type is a one dimensional.
  - The elements of a dimensional array are assumed to be arranged one after the other in a row.

Code:

```
#define N 10
…
int  A[N];
```



Arrays

- n. element of array is indicated by c[n-1].
  - c[0]+ c[1]+ c[2]+…….c[n-1]
- Array elements are like normal variables.
  - c[0] = 3;
  - printf(“%d”, c[0]);
- Operations can be performed on the index number. If a= 2, b=3
  - c[a+b] += 8; // c[5] adds 8 to element value
- To print out sum of the values of the first three elements of the array:
  - printf(“%d”, c[0]+c[1]+c[2]);

Array Initialization

- Initial value can be assigned during the definition of arrays.

```
int A[10]={8, 4, 10, 2, 5, 6, 7, 8, 9, 4};
```

- If the first values are less than the number of elements in the array, value of the remaining elements will be 0.

```
int A[10]={1, 2, 3, 4};
/* A[10] first values of array {1, 2, 3, 4, 0, 0, 0, 0, 0, 0}*/
```

- If you define an array with initial values, you can leave the size of the array empty.

```
int A[]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
/* Array A has 10 elements A[0]..A[9] */
```

Array Initialization

- The initial values of the array elements are not automatically zero. For this, at least the first element value must be set to zero.

```
int n[5] = {0}; // values of all elements will be 0
```

- If there is too much initial value, error will occur.

```
int n[5] = {1, 2, 3, 4, 5, 6}; //six initial value
```

Array Usage

- To reach each element of the array, we need to use the index of each element.
- An index indicates the position of the element in the array.
- The elements of the arrays are listed one after the other. (no gaps)
- Each element of the array is defined in sequence, and this sequence starts at 0.

Array Usage

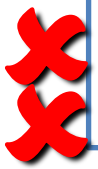
- Example

```
#define MAX_STD_NUMBER 5
…
int grades[MAX_STD_NUMBER];

…
grades[0] = 98;
grades[1] = 87;
grades[2] = 92;
grades[3] = 79;
grades[4] = 85;
```

Array Usage

- Warning!
  - C does not check indexes about proper range.

```
#define MAX_STD_NUMBER 5
…
int grades [MAX_STD_NUMBER];

…

grades[53] = 98;
grades[5] = 98;
```



Array Usage

- Loops are usually used when accessing elements of an array, and an element of the array is used for each iteration of the loop.
- The most commonly used loop is the for loop. Because both the initial value assignments and the index variable can be explicitly used in the loop expression:

```
grades[0] = 0;
grades[1] = 0;
grades[2] = 0;
grades[3] = 0;
grades[4] = 0;
```

```
int i;
for(i = 0; i < MAX_STD_NUMBER; i++)
        grades[i] = 0;
```

Example: Read

```
#include <stdio.h>
#define SIZE 5

int main(void)
{
   int i;
   double a[SIZE];
   printf("Enter %d array elements: ", SIZE);
   /* read array elements*/
   for(i = 0; i < SIZE; i++)
        scanf("%lf", &a[i]);
   return 0;
}
```

```
Enter 5 array element: 1.2 3.4 5.6 7.8 9.0
```

Example: Write

```
#include <stdio.h>
#define SIZE 5

int main(void)
{
   int i;
   double a[SIZE] = { 1.2, 3.4, 5.6, 7.8,
                      9.0 };
   /* Print array elements*/
   for(i = 0; i < SIZE; i++)
       printf("a[%d] = %.2lf\n", i, a[i]);

   return 0;
}
```

```
a[0] = 1.20
a[1] = 3.40
a[2] = 5.60
a[3] = 7.80
a[4] = 9.00
```

Example: Maximum Element

```
#include<stdio.h>

#define SIZE 5

int main(void)
{
    int i;
    double a[SIZE] = { 1.2, 3.4,
                       5.6, 7.8,
                       9.0 };
    double max = 0.0;
    /* Find max. elements of array*/
    for(i = 0; i < SIZE; i++)
         if (a[i] > max)
                max = a[i];
    printf("max = %.2lf\n", max);
    return 0;
}
```

```
max = 9.00
```

Example: Sum of Two Array

```
#include <stdio.h>

int main(void) {
   int i, N, A[100], B[100], C[100];
   printf("Enter size of array:\n");
   scanf("%d", &N);
   for(i = 0; i < N; i++){  /* Read array elements*/
        printf("A[%d]=",i);
        scanf("%d",&A[i]);
   }
   for(i = 0; i < N; i++){  /* Read array elements*/
        printf("B[%d]=",i);
        scanf("%d",&B[i]);
   }
   for(i = 0; i < N; i++){  /* Print out sum of array */
        C[i] = A[i] + B[i];
        printf("C[%d]=%d\n", i, C[i]);
   }
   return 0;
}
```

Example: Mean and Standard Deviation of an Array

```
#include <stdio.h>
#include <math.h>
#define N 10
int main(){
   int   i;
   float x[N], sum = 0.0, mean, std_dev = 0.0;
   /* mean calculation */
   for(i=0; i<N; i++){
      printf("%d. number: ",i+1);
      scanf("%f", &x[i]);
      sum += x[i];
   }
   mean = sum/N;
   /* standard deviation calculation */
   for(sum = 0.0, i=0; i<N; i++)
       sum += pow(x[i]-mean, 2.0);
   std_dev = sqrt( sum/(N-1) );
   printf("Mean        = %f\n", mean);
   printf("Standard deviation = %f\n", std_dev);
   return 0;
}
```

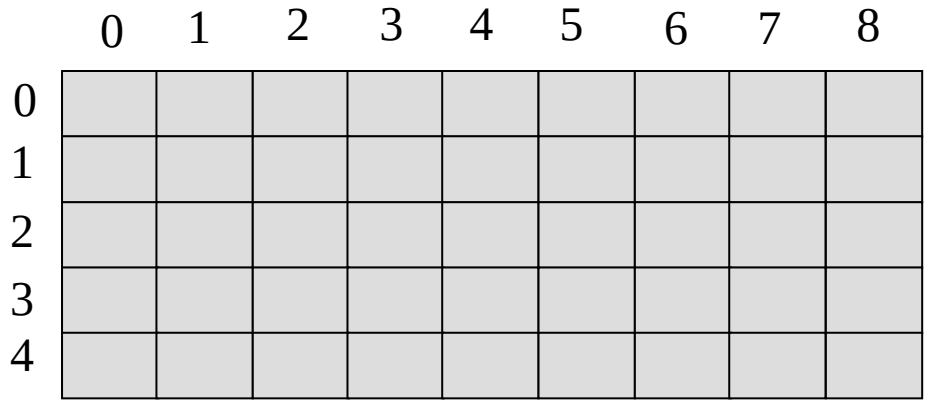
Example: Random Number Generator

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
  int c, n;
  // Initialization, should only be called once
  srand(time(NULL));

  printf("Ten random numbers in [1,100]\n");

  for (c = 1; c <= 10; c++) {
    n = rand() % 100 + 1;
    printf("%d\n", n);
  }
  return 0;
}
```

rand(); //Returns a pseudo-random integer between 0 and RAND_MAX.

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.



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 use 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;
```

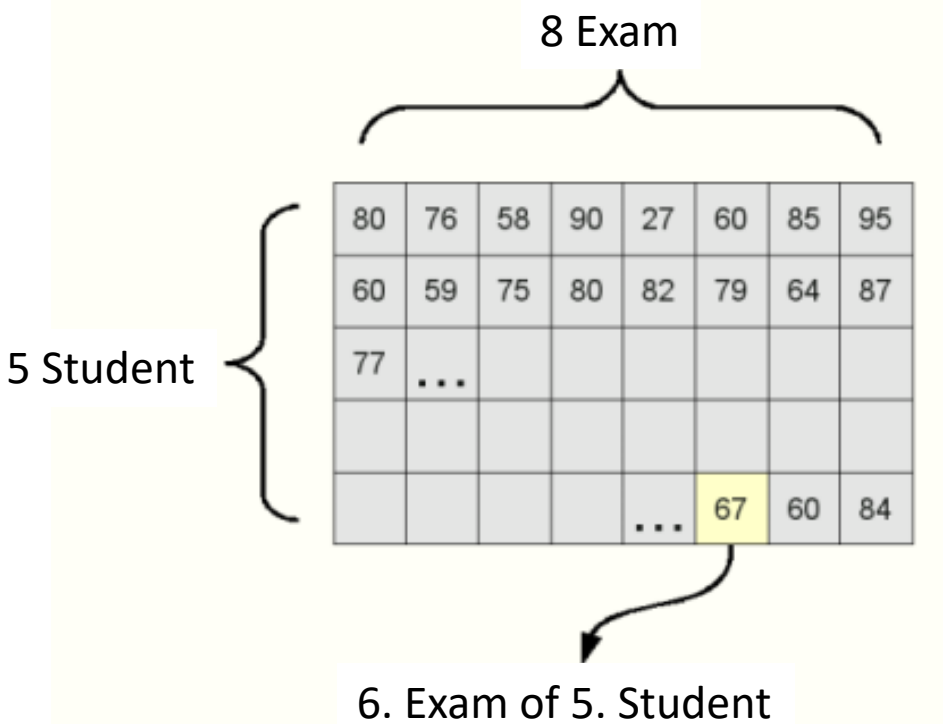
Multidimensional Arrays

- 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:



- 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 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

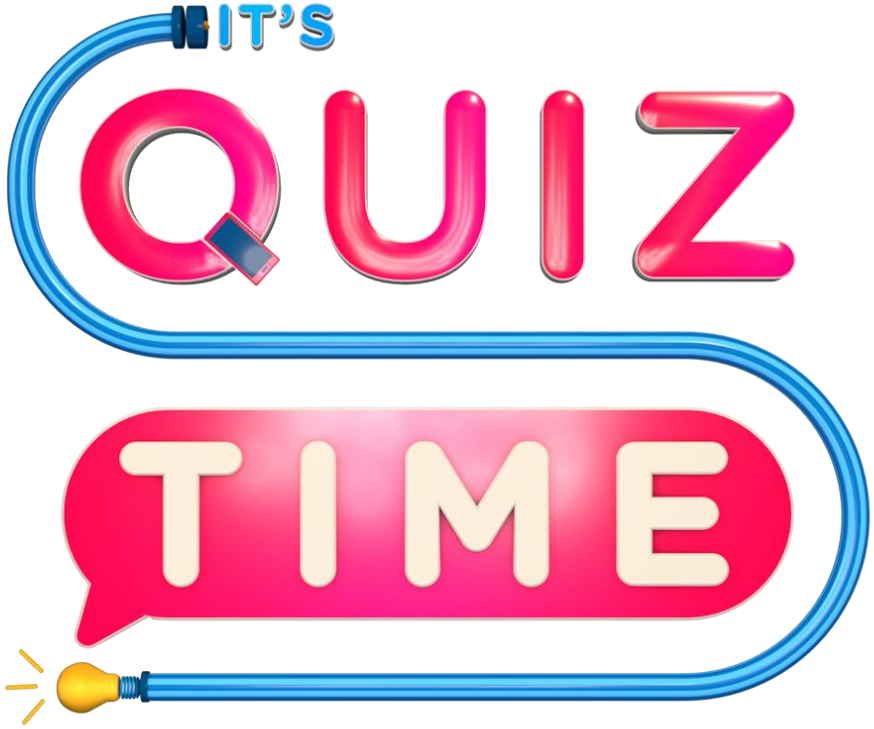
```
#include <stdio.h>
#include <stdlib.h>
 int main(){
    int dizi[100];
    int a[100][100];
    int i, j, n, x, sat, sut;
    printf(«How many elements does your array contain? > ");
    scanf("%d", &n);
     for(x=0; x<n; x++){
        printf(«Enter [%d]. element > ",x+1);
        scanf("%d",&dizi[x]);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

Example: 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