CME111 Programming Languages I

Week 10

Arrays

Assist. Prof. Dr. Caner ÖZCAN
Arrays

► In processing conducted with the help of computers, entering a large number of data and sorting entered data according to a systematic processing can be needed.

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

► Therefore, to process multiple data in the computer program "array" is used as a so-called sequential data fields.

► Data fields called single-name are placed consecutively memory in general.
Arrays

- We would need more than one variable of the same type for the same purpose.

- For example, a class of 100 people take "Programming Languages" course and have grades. Rather than to use individual variables (100-variable name required), these notes can be transferred to an array.

- In this way many variable name and area are not used.

- Information is kept below a specific order under one name and processed quickly.
**Arrays**

- Data structure that holds multiple variables of the same type together.
- The simplest type is a one dimensional.
  - Elements of a one dimensional array is regarded as arranged one after another in a row.

**Code:**

```c
#define N 10
...
int A[N];
```

- First index = 0
- Last index = N-1 = 9
**Arrays**

- n. element of array is indicated by c[n-1].

- Array elements are like normal variables.
  - c[0] = 3;
  - printf("\%d", c[0]);

- Operations can be performed on 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.

```c
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.

```c
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 the first data, we leave out the size of the array.

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

- Zero assignments to all elements of the array.

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

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

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

► We need to use every element of the array index to reach each element.
► Index components are expressed its position in the array.
► Elements of the array are listed in succession. (By the way no spaces between elements of array).
► Each element of the array is defined, respectively, and this ranking starts from 0.
Using Array

Example

```c
#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;
```
Using Array

► Warning!
  ▪ C does not check indexes about proper range.

```c
#define MAX_STD_NUMBER 5
...
int grades MAX_STD Numero];
...
grades[53] = 98;
grades[5] = 98;
```
Using Array

► While reaching the index of the elements, loops are used typically and the index in each iteration of the loop is working on an element.

► The most used loop is the for loop. Because in the loop statement, both first value assignment and index variable can also be used as clearly.

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

```c
#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;
}
```

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.20</td>
<td>3.40</td>
<td>5.60</td>
<td>7.80</td>
<td>9.00</td>
</tr>
</tbody>
</table>

Example: Maximum Element

```c
#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;
}
```

Example: Maximum Element

max = 9.00
Example: Sum of Two Array

```c
#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: Sum of Two Array
Example: Mean and Standard Deviation of an Array

```c
#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;
}
```
```c
#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.
```
References


► Paul J. Deitel, “C How to Program”, Harvey Deitel.

► Bayram AKGÜL, C Programlama Ders notları