C Programming & Lab

8. Arrays

Sejong University

Outline

1) **Arrays?**

2) **Arrays Initialization**

3) **Arrays Examples**

4) **Arrays: Real Number**

5) **Arrays: Character**

6) **2 Dimensional Arrays**

7) **Arrays with Functions**

8) **3 Dimensional Arrays**

1) Arrays?

- **Suppose that we need to use several variables.**
- **Read 5 variables from a user and compute the sum of the variables.**

- **It may look like:**
- **int x0, x1, x2, x3, x4;**
- **scanf("%d%d%d%d%d", &x0, &x1, &x2, &x3, &x4);**

1) Arrays?

- **Declaration**

```
#include <stdio.h>
int main(void){

    int x[5];



    return 0;
}
```

Declaration of an array

An array of int type, size is 5

Name of an array is x

1) Arrays?

- **In arrays, the name of variables?**

- **x[0], x[1], x[2], x[3], x[4]**

Numbers in [ ] distinguish variables

Number in [] is called an index

Be cautious about the range of indices

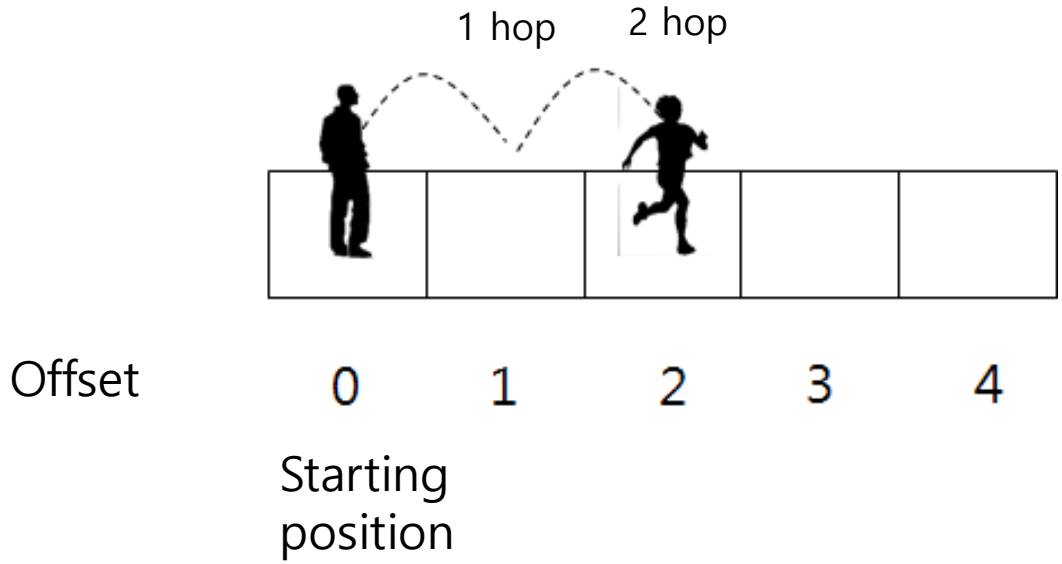
In this case, between 0 and 4

Index of an array always starts from 0

Why it starts from 0, not 1?

1) Arrays?

- **Find a value stored in an array of size 5**
- **Computer uses two sorts of information to locate the value**
  - **① start position of an array**
  - **② Offset: distance from the beginning of the array**



- **Use an offset as an index**
- **So, an index starts from 0**

1) Arrays?

- **Store values in an array and print them out**

```
#include <stdio.h>
int main(void){
   int x[5];
   x[0]=0; x[1]=1; x[2]=2; x[3]=3; x[4]=4;
   printf("%d %d %d %d %d\n", x[0], x[1], x[2], x[3], x[4]);
   return 0;
}
```

```
[Results]
0 1 2 3 4
```

1) Arrays?

- **Arrays and Loops**
  - An index increases by 1, can combine it with loops (for).

```c
#include <stdio.h>
int main(void){
   int x[5];

   x[0]=0; x[1]=1; x[2]=2;
   x[3]=3; x[4]=4;

   printf("%d ", x[0]);
   printf("%d ", x[1]);
   printf("%d ", x[2]);
   printf("%d ", x[3]);
   printf("%d ", x[4]);
   printf("\n");

   return 0;
}
```

```c
#include <stdio.h>
int main(void){
   int x[5];

   x[0]=0; x[1]=1; x[2]=2;
   x[3]=3; x[4]=4;

   for( i=0 ; i<5 ; i++ ){
       printf("%d ", x[i]);
   }
   printf("\n");

   return 0;
}
```

1) Arrays?

[Practice1]

- Store the scores below in an array x of size 7

  80, 71, 91, 95, 77, 79, 88

- Use for loops to print the index and score when the score is >80.

Example

```
0 80
2 91
3 95
6 88
```

[Practice2]

- Declare an array x of size 9
- Store the results of the multiplication table for 3 in the array x
- Print them out as shown on the right side

Example

```
3
6
9
.
.
.
27
```

2) Initialization

- **Without initialization, a particular value will be assigned**
  - Symbol # attaches 0x which indicates hexadecimal number

```
#include <stdio.h>
int main(void){
  int i=0, x[5];
  for(i=0;i<5;i++) {
    printf("%d   %#x \n", x[i], x[i]);
  }
  return 0;
}
```

```
[Results]
-858993460   0xcccccccc
-858993460   0xcccccccc
-858993460   0xcccccccc
-858993460   0xcccccccc
-858993460   0xcccccccc
```

2) Initialization

- **Initialize the entire array**
  - Use { } to initialize an array

```
#include <stdio.h>
int main(void){
  int i=0, x[5] = {0, 1, 2, 3, 4} ;

  for(i=0;i<5;i++) {
    printf("%d ", x[i]);
  }
  return 0;
}
```

```
[Results]
0 1 2 3 4
```

2) Initialization

- **The number of initial values is smaller than the size of an array? assign 0 to the rest of the array.**

```
#include <stdio.h>
int main(void){
  int i=0, x[5] = {0, 1, 2} ;

  for(i=0;i<5;i++) {
    printf("%d ", x[i]);
  }
  return 0;
}
```

```
[Results]
0 1 2 0 0
```

2) Initialization

- **Initialize the entire array with 0**

```
#include <stdio.h>
int main(void){
  int i=0, x[5] = {0} ;

  for(i=0;i<5;i++) {
    printf("%d ", x[i]);
  }
  return 0;
}
```

```
[Results]
0 0 0 0 0
```

2) Initialization

- **Do not specify the size of an array?**
  - sizeof( ) function shows the size of a variable or data type in Byte

```
#include <stdio.h>
int main(void){
  int i=0, x[ ] = {0, 1, 2, 3, 4} ;

  for(i=0;i<5;i++) {
    printf("%d ", x[i]);
  }
  printf("\nSize of an array = %d \n", sizeof(x)/sizeof(int));
  return 0;
}
```

```
[Results]
0 1 2 3 4
Size of an array = 5
```

2) Initialization

- **What will be the output?**

```
#include <stdio.h>
int main(void){
   int x[]={10, 5, 4, 3, 20};
   printf("%d %d₩n", x[2], x[4]);
   return 0;
}
```

3) Examples

- **Declare an array of size 5**
- **Initialize it with 10, 20, 30, 40, 50**
- **Print it out**

Approach1

```
#include <stdio.h>
int main(void){
   int x[]={10, 20, 30, 40, 50};
   printf("%d %d %d %d %d ₩n", x[0], x[1], x[2], x[3], x[4]);
   return 0;
}
```

Approach2

```
#include <stdio.h>
int main(void){
   int x[]={10, 20, 30, 40, 50};
   for(i=0; i<5; i++) printf("%d ", x[i]);
   return 0;
}
```

3) Examples

- **Declare an array of size 5**
- **Initialize it with 3, 4, 5, 1, 3**

```
int x[5]={3, 4, 5, 1, 3};
```

- **Compute the sum of the elements in an array**

Approach1

```
int x[5]={3, 4, 5, 1, 3};
int sum=0;
sum=x[0]+x[1]+x[2]+x[3]+x[4];
```

Approach2

```
int x[5]={3, 4, 5, 1, 3};
int i=0, sum=0;
for(i=0; i<5; i++) sum = sum + x[i];
```

3) Examples

- **What will be the output?**

```
int i=0, sum=0, x[5]={3, 4, 5, 1, 3};
for(i=0; i<5; i++) sum = sum + x[i];
printf("%d %d \n", sum, x[0]+x[1]+x[2]+x[3]+x[4]/5);
printf("%d %d \n", sum, (x[0]+x[1]+x[2]+x[3]+x[4])/5);
printf("%d %d \n", sum, sum/5);
printf("%d %d \n", sum, sum/5.0);
printf("%d %f \n", sum, sum/5.0);
```

3) Examples

- **Compute sum and average**

```
[Example]
16 13
16 3
16 3
16 -1717986918
16 3.200000
```

(Practice 3) Let's check the code in the previous slide and check if the results are the same with the ones above

3) Examples

- **Declare an array of size 5, initialize it with 0**
- **Read 5 integers from a user**

Approach1

```
int x[5]={0};
scanf("%d%d%d%d%d", &x[0], &x[1], &x[2], &x[3], &x[4]);
```

Approach2

```
int i=0, x[5]={0};
for(i=0; i<5; i++) scanf("%d", & x[i]);
```

3) Examples

- **Read 5 integers from a user**

```
#include <stdio.h>
int main(void){
    int i=0, sum=0, x[5]={0};
    printf(“Enter 5 integers: ");
    for(i=0;i<5;i++) scanf("%d", &x[i]);
    for(i=0;i<5;i++) sum+=x[i];
    printf("%d %f \n", sum, sum/5.0);
    return 0;
}
```

```
[Results]
Enter 5 integers: 3 4 5 1 3
16 3.200000
```

3) Practice4

- **Declare an array of size 10, initialize it with 0**
- **Read 10 integers from a user**
- **Compute sum of even numbers and sum of odd numbers, print them out**

Input Example 1

```
1 2 3 4 5 6 7 8 9 10
```

Output Example 1

```
30
25
```

Input Example 2

```
2 4 6 8 10 12 14 16 18 20
```

Output Example 2

```
110
0
```

4) Arrays: Real Number

- **Need several variables for real numbers**
- **Same with integer arrays**

- **Declaration**

```
#include <stdio.h>
int main(void){
    double x[10];


    return 0;
}
```

```
Array of size 10
Real numbers
```

4) Arrays: Real Number

- **Initialization(1)**

```
#include <stdio.h>
int main(void){
    double x[5]={1.0, 2.0, 3.0, 4.0, 5.0};
    int i=0;
    for(i=0;i<5;i++) printf("%.1f ", x[i]);
    printf("\n");
    return 0;
}
```

```
[Results]

1.0 2.0 3.0 4.0 5.0
```

4) Arrays: Real Number

- **Initialization(2)**

```
#include <stdio.h>
int main(void){
    double x[]={1.0, 2.0, 3.0, 4.0, 5.0};
    int i=0;
    for(i=0;i<5;i++) printf("%.1f ", x[i]);
    printf("\n");
    return 0;
}
```

4) Arrays: Real Number

- **Initialization(3)**

```
#include <stdio.h>
int main(void){
    double x[]={1.0, 2.0, 3.0};
    int i=0;
    for(i=0;i<5;i++) printf("%.1f ", x[i]);
    printf("\n");
    return 0;
}
```

```
[Results]

1.0 2.0 3.0 0.0 0.0
```

4) Arrays: Real Number

- **Initialization(4)**

```
#include <stdio.h>
int main(void){
    double x[]={0.0};
    int i=0;
    for(i=0;i<5;i++) printf("%.1f ", x[i]);
    printf("\n");
    return 0;
}
```

```
[Results]

0.0 0.0 0.0 0.0 0.0
```

4) Arrays: Real Number

- **Read 5 real numbers from a user and compute the sum and average**

```
#include <stdio.h>
int main(void){
    double x[5]={0.0}, sum=0.0;
    int i=0;
    printf(“Enter 5 real numbers: “);
    for(i=0;i<5;i++) scanf("%lf", &x[i]);
    for(i=0;i<5;i++) sum=sum+x[i];
    printf(“Sum = %f, Average = %f \n", sum, sum/5);
    return 0;
}
```

```
[Results]

Enter 5 real numbers: 1.0 2.0 3.0 4.0 5.0

Sum =  15.000000, Average = 3.000000
```

Caution: use %lf to read double type in scanf()

4) Arrays: Real Number

- **Compute the maximum value?**

```
#include <stdio.h>
int main(void){
    double x[5]={0.0}, max=0.0;
    int i=0;
    printf("Enter 5 real numbers: ");
    for(i=0;i<5;i++) scanf("%lf", &x[i]);
    max=x[0];
    for(i=1;i<5;i++) if(max<x[i]) max=x[i];
    printf("max = %f \n", max);
    return 0;
}
```

```
[Results]

Enter 5 real numbers: 7.0 8.0 1.0 9.0 2.0

max =  9.000000
```

4) Arrays: Real Number

- **Compute Maximum (1)**

x[0]	x[1]	x[2]	x[3]	x[4]

```
max = x[0];
if(max < x[1]) max = x[1];
if(max < x[2]) max = x[2];
if(max < x[3]) max = x[3];
if(max < x[4]) max = x[4];
```

4) Arrays: Real Number

- **Compute Maximum (2)**

7.0	8.0	1.0	9.0	2.0
x[0]	x[1]	x[2]	x[3]	x[4]

```
max = 7.0;
if(max < x[1]) max = 8.0;
if(max < x[2]) max = 8.0;
if(max < x[3]) max = 9.0;
if(max < x[4]) max = 9.0;
```

4) Arrays: Real Number (Practice 5)

- **Read 5 real numbers and print them in a reverse order**

Input Example

```
1.1 2.2 3.0 4.0 5.0
```

Output Example

```
5.000000
4.000000
3.000000
2.200000
1.100000
```

4) Arrays: Real Number (Practice 6)

- **Read 5 real numbers smaller than 100, print the minimum value**

Input Example

```
1.1 2.2 3.0 4.0 5.0
```

Output Example

```
1.100000
```

4) Arrays: Real Number (Practice 7)

- **Read 5 real numbers smaller than 100, print the minimum value and its index**

Input Example

```
3.0 2.2 1.1 4.0 5.0
```

Output Example

```
1.100000
2
```

5) Arrays: Character

- **Initialization**
- **Convert to upper-case letters**

```
#include <stdio.h>
int main(void){
    int i;
    char ar[4]={'d','u','c','k'};
    ar[0]='D', ar[1]='U', ar[2]='C', ar[3]='K';

    for(i=0;i<4;i++) printf("%c ",ar[i]);
    printf("\n");
    return 0;
}
```

```
D U C K
```

5) Arrays: Character (Example)

- **Print an character array**

```
#include <stdio.h>
int main(void){
    int i;
    char ar[9]={'d','u','c','k',' ','p','o','n','d'};

    for(i=0;i<9;i++) printf("%c ",ar[i]);
    printf("\n");
    for(i=0;i<4;i++) printf("%c ",ar[i]);
    printf("\n");
    for(i=5;i<9;i++) printf("%c ",ar[i]);
    printf("\n");
}
```

```
d u c k   p o n d
d u c k
p o n d
```

6) 2 Dimensional Arrays

- **Arrays: useful when to use several variables**
- **1 Dimensional Array: Use one index**
- **2 Dimensional Array: Use two indices**

```
int a[100]; // 1 dimensional array declaration
int b[10][10]; // create 100 variables but use two indices
```

6) 2 Dimensional Arrays

- **Why two indices?**

Declare an array to store students' scores
5 students' C programming scores

```
int score[5]={78, 93, 20, 44, 88};
```

Store Physics scores

Store 10 values

```
int score[10]={78, 89, 93, 100, 20, 30, 44, 55, 88, 12};
```

First student's scores

To compute the sum of C programming scores, need to figure out which ones are C programming scores

6) 2 Dimensional Arrays

- **How to handle it?**

```
int score[5][2]={{78, 89}, {93, 100}, {20, 30}, {44, 55}, {88, 12}};
```

`score[0][0]` First student's C programming score

`score[0][1]` First student's Physics score

- **First index : refer to Student**
- **Second index : refer to Subject**
  (C programming: 0, Physics: 1)
- **Easier to handle the values**
  - Students
  - Subjects

6) 2 Dimensional Arrays

- **Declaration**

```
#include <stdio.h>
int main(void){
    int a[5][2];
    int b[10][10];
    return 0;
}
```

In total, 10 variables

In total, 100 variables

6) 2 Dimensional Arrays

- **Print**

```
#include <stdio.h>
int main(void){
    int a[2][3]={{10,20,50}, {20,30,40}}; // {C,Physics,Math}
    int i=0, j=0;
    for(i=0;i<2;i++) {
        for(j=0;j<3;j++) {
            printf("a[%d][%d] = %d \n", i, j, a[i][j]);
        }
    }
    return 0;
}
```

```
[Results]
a[0][0] = 10
a[0][1] = 20
a[0][2] = 50
a[1][0] = 20
a[1][1] = 30
a[1][2] = 40
```

6) 2 Dimensional Arrays

- **Read scores and print them out**

```
#include <stdio.h>
int main(void){
    int a[2][3]={ 0 }; // {C,Physics,Math}
    int i=0, j=0;
    for (i = 0; i<2; i++) {
        for (j = 0; j<3; j++) {
            scanf("%d", &a[i][j]);
        }
    }
    for(i=0;i<2;i++) {
        for(j=0;j<3;j++) {
            printf("a[%d][%d] = %d \n", i, j, a[i][j]);
        }
    }
    return 0;
}
```

```
[Results]
a[0][0] = 10
a[0][1] = 20
a[0][2] = 50
a[1][0] = 20
a[1][1] = 30
a[1][2] = 40
```

```
[Input Example]
10 20 50 20 30 40
```

6) 2 Dimensional Arrays (Example)

	C programming	Physics
Student A	20	100
Student B	70	36
Student C	30	50

- **3 students' C programming and Physics scores**
- **Declare a 2 dimensional array and initialize it**
- **Compute the average of C programming and Physics scores**

6) 2 Dimensional Arrays

```
#include <stdio.h>
int main(void) {
    int i=0;
    int score[3][2]= { {20, 100},  {70, 36},  {30, 50} };
    int c_sum=0, p_sum=0;
    for(i=0; i<3; i++) {
        c_sum = c_sum + score[i][0];
        p_sum = p_sum + score[i][1];
    }
    printf(“C programming = %f, Physics = %f\n", c_sum/3.0,
p_sum/3.0);
    return 0;
}
```

```
[Results]

C programming = 40.000000, Physics = 62.000000
```

6) 2 Dimensional Arrays

- **Initialization (1)**

```
#include <stdio.h>
int main(void){
    int a[5][2]={{78, 89}, {93, 100}, {20, 30}, {44, 55}, {88, 12}};
    return 0;
}
```

a[0][0]=78	a[0][1]=89
a[1][0]=93	a[1][1]=78
a[2][0]=20	a[2][1]=78
a[3][0]=44	a[3][1]=78
a[4][0]=88	a[4][1]=78

6) 2 Dimensional Arrays

- **Initialization (2)**

```
#include <stdio.h>
int main(void){
    int a[2][3]={{10,20,50},{20,30,40}};
    return 0;
}
```

a[0][0]=10	a[0][1]=20	a[0][2]=50
a[1][0]=20	a[1][1]=30	a[1][2]=40

- **Initialization with 0 (3)**

```
    int a[2][3]={0};
```

6) 2 Dimensional Arrays

- **Should specify the size of an array?**

```
#include <stdio.h>
int main(void){
    int score[][]={{20,100},{70,35},{30,70},{80,80},{90,25}};
    return 0;
}
```

```
[Compilation Error]
error C2087:‘score’: missing subscript
error C2078: too many initializers
```

6) 2 Dimensional Arrays

- **Can omit the first index**

```
#include <stdio.h>
int main(void){
    int score[ ][2]={{20,100},{70,35},{30,70},{80,80},{90,25}};
    return 0;
}
```

6) 2 Dimensional Arrays (Practice8)

- **Read 9 numbers from a user, store it in a 3x3 array**
- **Print them in a 3X3 form as shown below**

Input Example 1

```
9 8 7 6 5 4 3 2 1
```

Output Example 1

```
9 8 7
6 5 4
3 2 1
```

6) 2 Dimensional Arrays (Practice9)

1) **Declare a 3X3 array, initialize it with 0**
2) **Print it in a 3X3 form**
3) **Read 3 integers from a user, indicating a row, column, and new value**
4) **Replace the value of 3x3 array with the given value at the specified location**
5) **Print the 3X3 array again**

Input Example 1

```
1 2 7
```

Output Example 1

```
0 0 0
0 0 0
0 0 0
0 7 0
0 0 0
0 0 0
```

7) 3 Dimensional Arrays

- **3 students' C programming and Physics scores in two classes**

Class 1

	C programming	Physics
Student A	20	100
Student B	70	36
Student C	30	50

Class 2

	C programming	Physics
Student A'	30	100
Student B'	80	40
Student C'	40	60

7) 3 Dimensional Arrays

- **Print C programming scores in each class**

```
#include <stdio.h>
int main(void){
    int a[2][3][2]={{{20,100}, {70,36},{30,50}},
                    {{30,100}, {80,40},{40,60}}};
    int i=0, j=0, k=0;
    for(i=0;i<2;i++){
        printf(“Class %d C programming\n", i+1);
        for(j=0;j<3;j++) {
            printf("a[%d][%d][%d] = %d \n",
                i, j, 0, a[i][j][0]);
        }
    }
    return 0;
}
```

```
Class 1 C programming
a[0][0][0] = 20
a[0][1][0] = 70
a[0][2][0] = 30
Class 2 C programming
a[1][0][0] = 30
a[1][1][0] = 80
a[1][2][0] = 40
```

8) Arrays with Functions

- **In functional calls, elements of arrays are like other variables**

```
#include <stdio.h>
void max(int a, int b){
      if (a > b) printf("%d ", a);
      else printf("%d ", b);
      printf("\n");
}

int main(void){
      int score[3]={70, 80, 90};
      max(score[0], score[1]);
      return 0;
}
```

```
80
```

- **Caution: Do not use the element of an array outside of the index range (here 0~2), Do not pass them to a function**
  - ✓ Wrong) `max(score[3], score[4]) ;`

8) Arrays with Functions

- **Use the name of an array as an argument of a function**

```
#include <stdio.h>
void print(int ar[3]){
      int i;
      for(i=0;i<3;i++) printf("%d ", ar[i]);
      printf("\n");
}

int main(void){
      int score[3]={70, 80, 90};
      print(score);
      return 0;
}
```

```
70 80 90
```

8) Arrays with Functions

- **Can omit the size of an array**

```
#include <stdio.h>
void print(int ar[ ]){
      int i;
      for(i=0;i<3;i++) printf("%d ", ar[i]);
      printf("\n");
}

int main(void){
      int score[3]={70, 80, 90};
      print(score);
      return 0;
}
```

8) Arrays with Functions

- **Can pass the size of an array to a function**

```
#include <stdio.h>
void print(int ar[ ], int size){
      int i;
      for(i=0; i<size; i++) printf("%d ", ar[i]);
      printf("\n");
}

int main(void){
      int score[3]={70, 80, 90};
      print(score, 3);
      return 0;
}
```

8) Arrays with Functions

- **Can compute the size of an array and pass it to a function**

```
#include <stdio.h>
void print(int ar[ ], int size){
      int i;
      for(i=0;i<size;i++) printf("%d ", ar[i]);
      printf("\n");
}

int main(void){
      int score[3]={70, 80, 90};
      print(score, sizeof(score)/sizeof(int));
      return 0;
}
```

8) Arrays with Functions

- **Can use the entire array as an argument, but different from other variables**
  - In change function, modify **the element of an array ar,** also modify **the element of an array score**
  - Why? will learn later Ch. 9 Pointers

```
#include <stdio.h>
void change(int ar[ ], int size){
      ar[1] = 0;
}

int main(void){
      int score[3]={70, 80, 90};
      change(score, 3);
      printf("%d", score[1]);
      return 0;
}
```

```
0
```

8) Arrays with Functions

- **(Comparison) Pass the element of an array?**
  - Do not modify the array score in main function
  - In function calls, 80 in score[1] is assigned to element (in change function)
  - score[1] differs from element

```
#include <stdio.h>
void change(int element){
      element = 0;
}

int main(void){
      int score[3]={70, 80, 90};
      change(score[1]);
      printf("%d", score[1]);
      return 0;
}
```

```
80
```