UNIVERSITY OF MASSACHUSETTS
DARTMOUTH

ECE160: Foundations of Computer Engineering I

Lecture #21 – Array (I)

Instructor: Dr. Liudong Xing
SENG-213C, lxing@umassd.edu
ECE Dept.

Administrative Issues (3/31)

- Lab#9 solution posted
- No more homework for the rest the semester!

- Today's topics
  - Finish Files II (L#20)
  - Then Array I (L#21)

Review of Lectures #18 & 20

- A file is a collection of information/related data treated as a unit
- How to declare a file_pointer (**FILE**)
- How to open a file (**fopen()**)
  - Modes: “r”, “w”, and “a”
- How to read data from a disk file (fscanf())
- How to write output to an external disk file (fprintf())
- How to close a file (**fclose()**)
- *getchar()* and *putchar()* read and write the standard input (keyboard) and output (monitor) streams
- *getc()/fgetc()* and *putc()/fputc()* read and write a file stream specified by the file_pointer

Agenda

- Arrays
  - Concepts
  - Declaration and definition
  - Initialization

Motivation

- Can you write a program to read 30 integers, add them, and then print them as well as the sum?
- How about 300 or 3000 or 30000 integers?
- To process large amounts of data, a powerful data structure such as an array is needed!

Arrays

- An array is a fixed-size, sequenced collection of elements of the same data type.
- Arrays allow to represent a group of elements as a single unit.
  - the elements must be of the same type (e.g., int, float, etc.)
- Before use, an array has to be defined and declared.
  - Reserve memory space for the elements in the array!

Array Declaration and Definition (An Example)

int myarray[30];

- An array that contains 30 integers is declared
- The array as a whole has a name, myarray, but each member can be accessed individually using its index 0 …29 (Note: the first index is 0!)
  - myarray[0] indicates the first element of the array
  - myarray[1] indicates the second element of the array
  - ……
  - the last element of the array is myarray[29]
- The array elements are stored in contiguous and increasing memory locations.

Array Declaration and Definition (Syntax)

```
element_type   array_name [number _of_elements];
```

- element_type specifies the type of the array's elements, e.g., int, float, double **(cannot be void type!)**
- array_name: the name of the array
  - Array names are C identifiers
- number_of_elements: the length/size of the array
  - Must be an integer or integer expression greater than 0
  - Can be defined explicitly:

    ```
    int a[100];
    ```
  - Use a constant defined in a preprocessor directive (L#5):

    ```
    #define N 100
    int a[N];
    ```
  - Can be an integer expression:

    ```
    int a[N+30];
    ```

Identifier Name Rules (Review)

- The first character can not be a digit. It has to be an alphabetic character or underscore.
- The identifier name must consist only of alphabetic characters, digits, or underscores.
- First 31 characters of an identifier are significant/used.
- DO NOT use a C reserved word /keywords (e.g., **int**).

Array Declaration and Definition

- Invalid array declaration examples:

```
int a[-20];
double b[52.7];
```

- You may declare arrays and single variables of the same type in the same line:

```
#define N 100
int a[N], d, e;
```

Exercise (1)

- True or false

____all elements of a given array have the same data type

____ all elements of a given array are placed randomly in computer's memory

____the index of the first element of an array is 1

Exercise (2)

- Find error(s), if any, in the following statements:
  - int a, b(6);
  - float a23b[99], 1cd[77];
  - void city[32], town[73];
  - double temperature[-70];
  - long phone[300]; /*The first and the last array elements in the array just defined are phone[1] and phone[300] */

Array Initialization (Way #1)

It can be done at definition time

```
int myarray[5]={1,2,10,15,0};
```

```
int myarray[] = {1,2,10,15,0};
```

- If an array size is omitted from the definition with an initializer list, the size of the array will be the number of elements in the initializer list
- The array is automatically declared to have a size of 5

Array Initialization (Cont'd)

int myarray[5]={3,7};

This implies that

myarray[0]=3, myarray[1]=7,
myarray[2]=0, myarray[3]=0,
myarray[4]=0

- If # of initial values provided < # of array elements, unassigned elements are filled with 0s

- To initialize an array to all 0s, supply just the first 0:

int myarray[30000] = {0};

Array Initialization (Cont'd)

```
int myarray[5]={3, 7, 23, 6, 9, 21};
```

- Causes a syntax error because there are 6 initializers and only 5 array elements!

- Providing more initializers in an array initializer list than there are elements in the array is a syntax error!

Exercise (3)

- Find error(s), if any, in the following statements:
  - int a[3]: 11, 22, 33;
  - int a={11,22}, b[20];
  - float a[3]={23, 34, 45, 56};
  - double d(4)= (11, 22, 33, 44);
  - a[4] = {11, 22, 33, 44};

Array Initialization (Way #2): Inputting Values into the Array

- Usually done in a for loop
- Example: assume

int myarray[5];

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

Array Initialization (Way #3): Value Assignment

- Usually done in a for loop
- Example: assume

int myarray[5];

```
for(int i=0; i< 5; i++)
{
        myarray[i]=i*2+1;
}
```

- Assign values to individual elements:

```
myarray[3] = 37;
```

Value Assignment (Cont'd)

- Cannot assign one array to another array, even if they match fully in type and size
- Example: assume

  int myarray[25], yourarray[25];

- Copy arrays at the individual element level using loops!

```
for(int i=0; i< 25; i++)
{
        myarray[i]=yourarray[i];
}
```

Example: Printing An Array /Outputting Values

```
#include "stdio.h"
#define array_size 5
#define my_const 70

int main(void)
{
   int myarray[array_size];

   for (int i=0; i < array_size; i++)
   {
            myarray[i] = i*my_const;
            printf("myarray[%d] is :%d\n",i, myarray[i]);
   }

   return 0;
}
```

What is the output of the program?

Exercise (4)

- An array has 10 elements, whose values are read from the keyboard. Write a program that finds and prints the maximum of the array.

What is the limitation of this program?

Run the program using the following input: -111, -112, -113, -114, -115, -116, -117, -118, -119, -120

Revise the solution to remove the limitation.

```
#include "stdio.h"
#define array_size 10
void main(void)
{
        int myarray[array_size];
        int max = -100;
        int i = 0;

        for (i = 0; i < array_size; i++)
        {
                printf("Enter array member  %d\n", i);
                scanf_s("%d", &myarray[i]);
                if (myarray[i] > max)
                max = myarray[i];
        }
        printf("The largest array member is: %d\n", max);
}
```

Summary of Lectures #21

- Arrays allow to represent a group of elements of the same type as a single unit.
- Like variables, before use, an array has to be defined and declared
- Three ways to initialize an array
  - At the definition time
  - Inputting values form the keyboard
  - Assigning values

Things To Do

- Review the lecture
- Run and test the programs in Exercise (4) on Slides 21-22

Next Topics

- Arrays (Cont'd)