

# Computer Science & Engineering 150A

## Problem Solving Using Computers – Laboratory

### Lecture 13 - Array

Shuai Xie

(Adapted from Derrick Stolee, Lin Liu & Shuai Xie )

Spring 2010

- The 7th CodeLab assignment

"CodeLab Assignment-7"

Due: 23:59 Apr 23

- Simple data types use a single memory cell to store a variable.
- Sometimes, we want to *group* information together.
- For example: a program processes exam scores for a class could:

- 1 Declare 54 variables:

```
double student1, student2, student3, ... student54;
```

- 2 Store all of the scores together in continuous memory cells  
Access them as a group  
Reference them through the same variable name

```
double student[54];
```

# Declaring and Referencing Arrays

CSCE150A

Assignment

Arrays

Using arrays

- An array is a single variable that references several data elements in a row.
- To define, declare the *name of the array* and the *number of cells* associated with it.

**Type ArrayName[Size];**

- Example:      `double x[8];`

Creates 8 memory cells of type `double` with the name `x`;

These memory cells will be adjacent to each other in memory.

- Access individual data elements with [index] brackets.
- By specifying the *array name* and identifying the element desired, we can access a particular value.
  - $x[0]$  (read as  $x$  sub zero) references the 0th element of the array  $x$ ;
  - $x[1]$  is the next element in the array, followed by  $x[2]$ , ...,  $x[n-1]$ .
- The array subscript value must be in the range from zero to one less than the number of memory cells in the array.  
double student[54];  
index: 0, 1, 2, ..., 53  
Element "student[54]" is invalid and may cause serious memory problem.

# Parallel Arrays

CSCE150A

Assignment

Arrays

Using arrays

- We can define two arrays and have the  $i$ th element in each array correspond to the same thing, such as the following:

```
int id[50];  
double gpa[50];
```

- `id[i]` and `gpa[i]` correspond to the  $i^{th}$  student.

# Array declaration and Initialization

CSCE150A

Assignment

Arrays

Using arrays

- Array name can not be the same as the variable name.  
int a;  
double a[10];  
**wrong!**
- When declare array, array size should be a fixed number, not a variable.  
int n=5,  
int a[n];  
**wrong!**
- You can declare multiple arrays along with regular variables.  
double student[200], president, professor[10];

# Array declaration and Initialization

CSCE150A

Assignment

Arrays

Using arrays

- We can initialize a simple variable when we declare it:

```
int sum = 0;
```

- Same with arrays:

```
#define SIZE 10  
...  
int array[SIZE];  
for(i=0; i < SIZE; i++)  
{  
    array[i] = 0;  
}
```



# Array Initialization

CSCE150A

Assignment

Arrays

Using arrays

- We can also initialize an array in its declaration.

```
int prime100[] = {  
    2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37,  
    41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83,  
    89, 97 }
```

# Array Subscripts

CSCE150A

Assignment

Arrays

Using arrays

- Subscripts denote the *number of places away from the start*.
- Any expression of type `int` can be used as an array subscript.
  - For Example: `array[2*i+1]`
  - To create a valid reference, the value of this subscript must lie between 0 and one less than the declared size of the array.

- To use `scanf` or `printf`.  
`scanf("%d", &x[i]);`  
`printf("%d\n", x[i]);`

# Using for Loops for Sequential Access

CSCE150A

Assignment

Arrays

Using arrays

- Usually, we iterate through element in order from 0 ...  $\text{SIZE}-1$ .
- Use a for loop with index variable, starting at 0 and bounded by SIZE.

```
for(i=0; i < SIZE; i++)  
{  
    printf("%d ",array[i]);  
}
```

- Using the loop counter as an array index (subscript) gives access to each array element in turn.
- Can not output the whole array by one printf.

```
printf("%d",array);
```

**Wrong!**