

Week 8: Arrays

Gaddis: 7.1-4,6

CS 1428
Fall 2014

Jill Seaman

1

Array Data Type

- Array: a variable that contains multiple values of the same type.
- Values are stored consecutively in memory.
- An array definition in C++:

```
int numbers[5];
```
- This creates an array called numbers which contains 5 integers (ints).

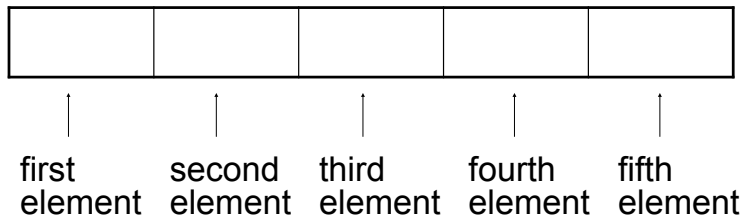
2

Array - Memory Layout

- The definition:

```
int numbers[5];
```

 allocates the following memory:
(values are stored consecutively in memory)



3

Array size in memory

- The size in bytes is: the number of elements times the size of each element.

```
int numbers[5];
```
- An int requires 4 bytes
- numbers array allocates 5 integers:
- 5 integers * 4 bytes = 20 bytes
- ```
double temperatures[100];
```
- 100 doubles of 8 bytes: size is 800 bytes.<sub>4</sub>

## Array Terminology

- Given the following array definition:  
`int numbers[5];`
- `numbers` is the name of the array
- `int` is the data type of the array elements
- `5` is the size declarator:  
the number of elements (values) in the array.

5

## Size Declarator

- The size declarator must be an *integer* and a *constant*.
    - ▶ it must be greater than 0
    - ▶ IT CANNOT BE A VARIABLE!\*
  - It can be a literal or a named constant.
- ```
const int SIZE = 40;
double grades[SIZE];
```
- Named constants ease program maintenance when the size of the array must be changed.

*Unless you are using a special compiler

6

7.2 Accessing Array Elements

- Individual elements of the array have unique subscripts (also called indexes).
- The subscripts are 0-based
 - ▶ the first element has subscript 0
 - ▶ the second element has subscript 1
 - ▶ ...
 - ▶ the last element has subscript (size -1)

the last element's subscript is n-1 where n is the number of elements in the array

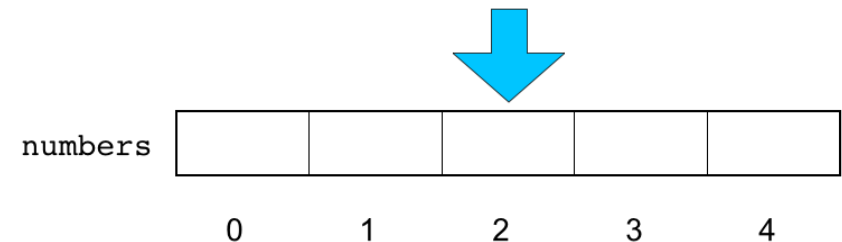
7

Accessing Array Elements

- Syntax to access one element:

```
numbers[2] //the third element of numbers array
```

- Called “numbers at 2” or “numbers sub 2”



8

Array subscripts

- The subscript is ALWAYS an integer
 - regardless of the type of the array elements.
- the subscript can be ANY integer expression
 - literal: 2 numbers[2]
 - variable: i numbers[i]
 - expression: (i+2)/2 numbers[(i+2)/2]

9

Array subscripts

- Given the following array definition:

```
double tests[10];
```

the expression `tests[i]` may be used exactly like **any** variable of type double.

```
tests[0] = 79;
cout << tests[0];
cin >> tests[1];
tests[4] = tests[0] + tests[1];
```

10

Using array elements:

```
double values[3]; //array declaration
values[0] = 22.3; //assignment to array element
values[1] = 11.1;

cout << "Enter a number: ";
cin >> values[2];

double sum = values[0] + values[1] + values[2];
double avg = sum/3.0;

cout << "Values at zero: " << values[0] << endl;

int i=2;
if (values[i] > 32.0)
    cout << "Above freezing" << endl;
```

11

7.4 Array initialization

- You can initialize arrays when they are defined.

```
const int NUM_SCORES = 3;
float scores[NUM_SCORES] = {86.5, 92.1, 77.5};
```

- Values are assigned in order:
 - scores[0] = 86.5
 - scores[1] = 92.1
 - scores[2] = 77.5
- NOTE: uninitialized arrays have unknown values stored in them (not necessarily 0).

12

Implicit array sizing

- When you initialize, you don't need to specify the size declarator.

```
float scores[] = {86.5, 92.1, 77.5};
```

- In this case, the compiler determines the size of the array from the number of elements listed.

13

7.5 Processing Array Contents

- Generally there are NO operations (>>, <<, =, ==, +) that you can perform over **entire** arrays.
- Some operations may appear to work (no errors) but you don't get the desired results.

```
int numbers1[] = {1, 2, 3};
int numbers2[] = {4, 5, 6};

cin >> numbers1;           //input, won't work
cout << numbers1 << endl;  //output, won't work
numbers1 = numbers2;      //assignment, won't work
if (numbers1==numbers2)   //comparison, won't work
...
numbers3 = numbers1 + numbers2; //addition, won't work
```

14

Operations over arrays

- Most array operations must be done one element at a time.
- Input the 7 programming assignment grades for 1 student in CS1428

```
const int NUM_SCORES = 7;
int scores[NUM_SCORES];
cout << "Enter the " << NUM_SCORES
    << " programming assignment scores: " << endl;
cin >> scores[0];
cin >> scores[1];
cin >> scores[2];
cin >> scores[3];
cin >> scores[4];
cin >> scores[5];
cin >> scores[6];
```

- Is there a better way?

15

Array input using a loop

- We can use a for loop to input into the array
- The subscript can be a variable

```
const int NUM_SCORES = 7;
int scores[NUM_SCORES];
cout << "Enter the " << NUM_SCORES
    << " programming assignment scores: " << endl;

for (int i=0; i < NUM_SCORES; i++) {
    cin >> scores[i];
}
```

16

Array output using a loop

- We can use a for loop to **output** the elements of the array

```
const int NUM_SCORES = 7;
int scores[NUM_SCORES];
cout << "Enter the " << NUM_SCORES
    << " programming assignment scores: " << endl;

for (int i=0; i < NUM_SCORES; i++) {
    cin >> scores[i];
}

cout << "You entered these values: ";
for (int i=0; i < NUM_SCORES; i++) {
    cout << scores[i] << " ";
}
cout << endl;
```

17

Summing values in an array

- We can use a for loop to sum the elements of the array (the *running total*)

```
const int NUM_SCORES = 7;
int scores[NUM_SCORES];
cout << "Enter the " << NUM_SCORES
    << " programming assignment scores: " << endl;

for (int i=0; i < NUM_SCORES; i++) {
    cin >> scores[i];
}

int total = 0; //initialize accumulator
for (int i=0; i < NUM_SCORES; i++) {
    total = total + scores[i];
}
```

How do you get the average programming assignment score?

18

Finding the maximum value in an array

- We can use a for loop to find the max value:
- Note: keep track of the maximum value encountered so far (the *running maximum*)

```
const int NUM_SCORES = 7;
int scores[NUM_SCORES];
cout << "Enter the " << NUM_SCORES
    << " programming assignment scores: " << endl;

for (int i=0; i < NUM_SCORES; i++) {
    cin >> scores[i];
}

int maximum = scores[0]; //init max to first elem
for (int i=1; i < NUM_SCORES; i++) { //start i at 1
    if (scores[i] > maximum)
        maximum = scores[i];    // no else needed
}
```

19

Array assignment

- To copy one array to another, you must assign element by element.

```
const int SIZE = 4;

int values1[SIZE] = {100, 200, 300, 400};
int values2[SIZE];

// values2 = values1; WRONG, won't work right

for (int i = 0; i < SIZE; i++) {
    values2[i] = values1[i];
}
```

20

Partially filled arrays

- The programmer does not always know ahead of time how many elements there will be in the array (i.e. reading from a file).
- If it is unknown how much data an array will be holding:
 - ▶ Make the array large enough to hold the largest expected number of elements.
 - ▶ Use a counter variable to keep track of the number of items stored in the array.

21

Partially filled arrays

```
const int MAX_STUDENTS = 100;
int scores[MAX_STUDENTS];

ifstream infile;
infile.open("students.txt");

int count = 0;
while (count < MAX_STUDENTS && infile >> scores[count]){
    count++;
}

int total = 0;
for (int x = 0; x < count; x++) //not MAX_STUDENTS
    total = total + scores[x];
```

22

7.3 C++: No bounds checking

- C++ does not check it to make sure an array subscript is valid (between 0 and size-1)
- If you use a subscript that is beyond the bounds of the array you will not get a warning or error.
- You may unintentionally change memory allocated to other variables.

```
const int SIZE = 3;
int values[SIZE];

for (int i=0; i < 5; i++) {
    values[i] = 100;
}
```

This code defines a three-element array and then writes five values to it.

23

Finding the maximum value in an array and its position

- Keep track of the minimum value, AND what its position is:

```
const int NUM_SCORES = 7;
int scores[NUM_SCORES];
// input code goes here

int indexOfMax = 0; //init indexOfMax to first
int maximum = scores[0]; //init max to first elem
for (int i=1; i < NUM_SCORES; i++) { //start i at 1
    if (scores[i] > maximum) {
        maximum = scores[i];
        indexOfMax = i;
    }
}
cout << "The highest score was " << maximum
     << " and it was assignment " << indexOfMax+1
     << endl;
```

24