

Arrays

Chapter 8
page 471

Arrays (8.1)

- One variable that can store a *group of values of the same type*
- Storing a number of related values
 - all grades for one student
 - all temperatures for one month
 - hours worked for each day

10/24/07

CS150 Introduction to Computer Science 1

2

Arrays

```
int age = 42; → [ 42 ]  
  
int ages[3]; → [ 17 | 22 | 21 ]  
ages[0] = 17; →  
ages[1] = 22; →  
ages[2] = 21; // variable_name [ index ]
```

10/24/07

CS150 Introduction to Computer Science 1

3

When would I use this?

- Read in 5 test scores from the user. Calculate the average test score and print out the scores in reverse order.

```
89
90
84
90
100
Average: 90.6
```

You could do this with 5 integers. But what about 100 test scores? Or 1000?

Declaring an Array

- The size of the array must be a *literal* or a `const int`.

```
int size = 99;
const int CONSTSIZE = 1024;

string names[3];           // literal
double tempatures[size];  // illegal!
int tests[CONSTSIZE];     // const int
```

- When the code is compiled, the exact size of the array must be known

Using arrays (8.2)

- The first element in the array is the **0th** element!
- You can use a single element of an array just like any other variable

- The *index* is just an `int`
- Loops are often used to access every element in an array

```
int y, x = 3;
int tests[10];

tests[0] = 2;
tests[x] = 4;
y = tests[0] + 9;
```

Q.1. Practice

- Declare an array to hold the height, in inches, of six trees.
- Set the height of the trees as:
 - 32 inches
 - 45 inches
 - 99 inches
 - 120 inches
 - 500 inches
 - 600 inches

Q.2. Practice (8.3)

- Write a snippet of code to print to the screen every value in this array:

```
const int ARRAYSIZE = 4;
int vals[ARRAYSIZE];
vals[0] = 1;
vals[1] = 2;
vals[2] = 4;
vals[3] = 8;
```

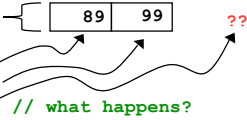
Q.3. Practice

- Write a snippet of code to print to the screen the sum and average of the values in this array:

```
const int ARRAYSIZE = 4;
int vals[ARRAYSIZE];
vals[0] = 1;
vals[1] = 2;
vals[2] = 4;
vals[3] = 8;
```

Danger! Danger! (p 479)

```
int tests[2];  
tests[0] = 89;  
tests[1] = 99;  
tests[4] = 42; // what happens?
```



- C++ does *not* check to make sure the index falls within the array
 - no *bounds checking*
 - this will cause unpredictable results!

10/24/07

CS150 Introduction to Computer Science 1

10

Initialization (8.4)

- How do you set the initial values for the array elements?

- What is the equivalent of:

```
int value = 2;  
int tests[2] =  
string names[3] =
```

- Initialize just a few values:

```
int value[4] =
```

10/24/07

CS150 Introduction to Computer Science 1

11

Implicit array sizing (p 486)

- Set the size of the array by initializing it
- You *must* either specify a size or initialize the array

```
string names[] =
```

```
char letters[] =
```

10/24/07

CS150 Introduction to Computer Science 1

12

Q.4. Using Arrays (8.6)

- Write code that will use arrays to store the names the months and the number of days in each month (assume no leap year!). Print the following to the screen:

```
January 31
February 28
March 31
April 30
May 31
June 30
July 31
August 31
September 30
October 31
November 30
December 31
```

10/24/07

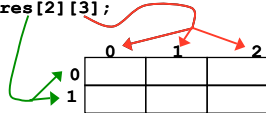
CS150 Introduction to Computer Science 1

13

Two dimensional arrays (8.9)

- A grid of data!

```
int testScores[2][3];
```



```
testScores[0][0] = 99;
testScores[0][1] = 80;
testScores[0][2] = 88;
testScores[1][0] = 89;
testScores[1][1] = 77;
testScores[1][2] = 85;
```

10/24/07

CS150 Introduction to Computer Science 1

14

Why use 2D arrays?

- Hold the scores for each student in one array.

```
const int BOB = 0;
const int ALICE = 1;
const int MIDTERM1 = 0;
const int MIDTERM2 = 1;
const int FINAL = 2;
int testScores[2][3] = { {0, 0, 0},
                        {0, 0, 0} };
testScores[BOB][MIDTERM1] = 99;
testScores[ALICE][FINAL] = 85;
```

- Which values are we setting above?
- How do we set Alice's Midterm2 score?
- What is stored in `testScores[0][1]` ?

10/24/07

CS150 Introduction to Computer Science 1

15

Q.5. Practice

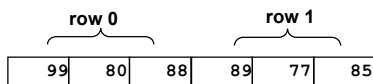
- Using the array below, calculate:
 - the average score on each assignment
 - the average score for each student
 - assume the array already contains data

```
const int NUMSTUDENTS = 100;  
const int NUMASSIGNMENTS = 4;  
  
int testScores[NUMSTUDENTS][NUMASSIGNMENTS];
```

A 2D array in memory

- The 2D array laid out by rows in memory

```
int testScores[2][3]= { {99, 80, 88},  
                       {89, 77, 85}};
```



This is called **row major order**. Some languages use **column major order**.

N-Dimensional Arrays (8.10)

```
// cost of seats in a theatre  
//  
// 4 sections, each section has  
// 20 rows with 30 seats each.  
  
double seats[4][20][30];  
  
seats[0][0][0] = 100.00;  
seats[2][0][0] = seats[1][0][0] / 2;  
seats[3][19][25] = 10.00;  
  
// we can have as many dimensions as  
// necessary in an array
```
