CS150 Intro to CS I

Fall 2015

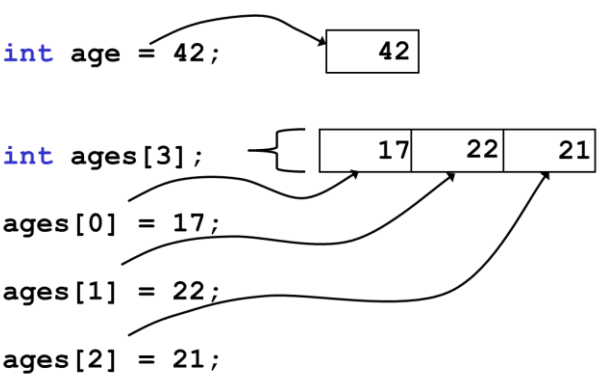
Chapter 7
Arrays

- Reading: pp.375-418
- Good Problems to Work: p.387 [7.2, 7.3, 7.4, 7.5, 7.6]; p. 406 [7.8, 7.9, 7.10, 7.11]; pp. 417-418 [7.14, 7.16, 7.18]

Arrays

- One variable that can store a group of values of the same type
  - Each value is called an *element* of the array
- Stores a number of related values
  - all grades for one student
  - all temperatures for one month
  - hours worked for each day

Simple Variables versus Arrays



Array Declaration

```
int ages[3]; // datatype variableName[size];

const int CLASS_SIZE = 24;
string names[CLASS_SIZE];
```

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

Using Arrays

- The first element in the array is the 0^{th} element
- The *index* is an **`int`**

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

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

Practice

- Declare an array to hold the height, in inches, of six trees.
- Set the height of the first two trees as:
  - 32.6 inches
  - 45.0 inches
- What are the values of the other four elements of the array?

Practice

- Write a snippet of code to read in 10 numbers from the user and put them in this array:

```
const int ARRAY_SIZE = 10;
int numbers[ARRAY_SIZE];
```

Practice

- An unknown number (but at most 100) of exam scores exists in a file (exams.txt). The sentinel value is -99.0.
- Write a loop that reads in each exam score into an array **examScores**.
- In another loop, find the maximum score in the array

Out of Bounds

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

Initialization

- What is the equivalent of:

```
int value = 2; // initialize the variable

int tests[2] =

string names[3] =
```

- Initialize just a few values:

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


Implicit Array Sizing

- Array size is determined by:
  - explicitly specifying a size
  - initializing an array which leads to an implicit size

```
char letters[] = {'a', 'b'};

versus

char letters[10];
```

Parallel Arrays

- Write a program to read the file below into *two arrays*. There are at most 100 students listed.
- Print the PUNetIDs of students who have a score between 88 and 100.

Grades.txt

```
PUNetID  FinalAverage
AAAA1234 90.2
will4614 85.4
…
sentinel 0.0
```


Parallel Arrays

- Add 10 bonus points to AAAA1234
- Print out the overall class average

Arrays as Function Arguments

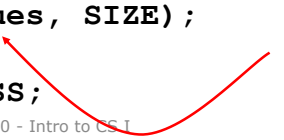
- You can pass an array as an argument to a function

```
void printIntArray (int [], int);

int main()
{
 const int SIZE = 7;
 int values[] = {5, 6, 0, 1, 2, 3, 4};

 printIntArray(values, SIZE);

 return EXIT_SUCCESS;
}
```



Arrays as Function Arguments

```
void printIntArray (int arr[], int size)
{
  for (int i = 0; i < size; i++)
  {
    cout << arr[i] << endl;
  }
}
```

Passing Single Array Elements

- Passing single array elements is like passing in a single variable

```
void showValue (int num);

int main()
{
 const int SIZE = 7;
 int values[] = {5, 6, 0, 1, 2, 3, 4};
 for (int i = 0; i < SIZE; i++)
 {
  showValue(values[i]);
 }
 return EXIT_SUCCESS;
}
```


Passing Entire Arrays into Functions

- Passing entire arrays to a function are passed by reference
- What does this mean?

Practice

- Write a program that will allow a professor to enter up to 20 homework scores. In the main function, ask the professor how many homework scores they would like to enter then read those in. The program must then calculate and display the average after dropping the lowest score. You must use the following functions:

Practice

```
void inputScores (double scores[], int numScores);

double getTotal (double scores[], int size);

double getLowest (double scores[], int size);
```