Introduction to Arrays in C++
Review for Midterm #2

CS 16: Solving Problems with Computers I
Lecture #12

Ziad Matni
Dept. of Computer Science, UCSB

Announcements

MIDTERM #2 on THURSDAY

- **Homework #11 due today**
- NO HOMEWORK THIS WEEK!
- NO NEW LAB THIS WEEK:
  – Use lab time to ask your TA questions for midterm

Outline

Chapter 8 (8.1, 8.2) in textbook

- Strings

Chapter 7 in textbook

- Arrays

- Midterm Review

Built-In String Manipulators

- Search functions
  - **find, rfind, find_first_of, find_first_not_of**
- Descriptor functions
  - **length, size**
- Content changers
  - **substr, replace, append, insert, erase**

Search Functions 1

- You can search for a the ***first occurrence*** of a string in a string with the **.find** function

```
string str = “With a banjo on my knee and ban the bomb!”;
int position = str.find(“ban”);
cout << position;     // Will display the number 7
```

- You can also search for a the ***first occurrence*** of a string in a string, starting at position ***n***

```
string str = “With a banjo on my knee and ban the bomb!”;
int position = str.find(“ban”, 12);
cout << position;     // Will display the number 24
```

Search Functions 2

- You can use the **find** function to make sure a substring is ***NOT*** in the target string
  - **string::npos** is returned if no position exists

```
if (str.find("piano") == string::npos) {
    do something here…    }
    // This will happen if "piano" isn't in the string str
```

- You can search for a the ***last occurrence*** of a string in a string with the **.rfind** function

```
string str = "With a banjo on my knee and ban the bomb!";
int rposition = str.rfind("ban");
cout << rposition;    // Will display the number 28
```

Search Functions 3

- **find_first_of**
  - Finds 1st occurrence of ***any*** of the characters included in the specified string
- **find_first_not_of**
  - Finds 1st occurrence of a character that is ***not any*** of the characters included in the specified string
- Example:

See demo file:
non_numbers.cpp

Descriptor Functions

- The **length** function returns the length of the string
- The member function **size** is the same exact thing…
  - So, if `string str1 = “Mama Mia!”`,
    then `str1.length() = 9`
    and `str1.size() = 9` also

Example – what will this code do?:

```
string name = “Bubba Smith”;
for (int i = name.length(); i > 0; i--)
    cout << name[i-1];
```

Content Changers 1
append

- Use function **append** to append one string to another

```
string name1 = " Max";
string name2 = " Powers";
cout << name1.append(name2);  // Displays " Max Powers"
```

- Does the same thing as: **name1 + name2**

Content Changers 2

erase

- Use function **erase** to clear a string to an empty string

- One use is:
  name1.erase() -- Does the same thing as: **name1 = “”**

- Another use is:
  name1.erase(*start position, how many chars to erase***)**
  - Erases only part of the string
  - Example:

```
string s = “Hello!”;
cout << s.erase(2, 2);  // Displays “Heo!”
```

Content Changers 3

replace, insert

- Use function **replace** to replace part of a string with another
  - Popular Usage:
    string.replace(*start position,*
    places after start position to replace, *replacement string*)

- Use function **insert** to insert a substring into a string
  - Popular Usage:
    string.insert(*start position, insertion string*)

Example:

```
string country = “USA”;
cout << country.replace(2, 1, “ of A”); // Displays “US of A”
cout << country.insert(7, “BC”);        // Displays “US of ABC”
```

Content Changers 4

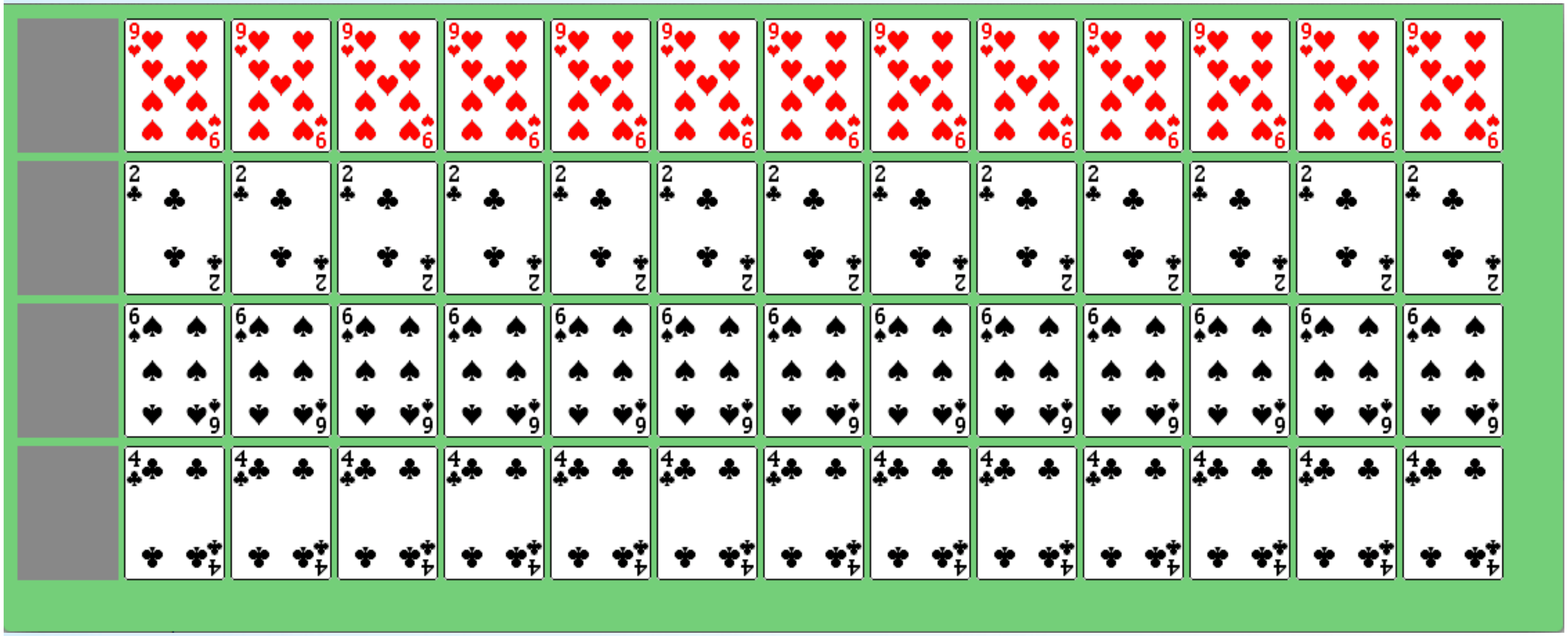
substr

- Use function **substr** (short for “substring”) to extract and return a substring of the **string** object
  - Popular Usage:
    string.substr(*start position, places after start position*)

Example:

```
string city = “Santa Barbara”;
cout << city.substr(3, 5)
               // Displays “ta Ba”
```

ARRAYS



Introduction to Arrays

- An array is used to process a collection of data of the same type
  - Examples: A list of names
    A list of temperatures
- Why do we need arrays?
  - Imagine keeping track of 1000 test scores in memory!
    - How would you name all the variables?
    - How would you process each of the variables?

Declaring an Array

- An array, named **score**, containing five variables of type **int** can be declared as

```
int score[5];
```

- This is like declaring 5 variables of type int:

```
int score[0], score[1], … , score[4]
```

- The value in [brackets] is called: ***a subscript*** or ***an index***

- Note the **size** of the array is the **highest index value + 1**
  - Because indexing in C++ starts at 0, not 1

Array Variable Types

- An array can have indexed variables of ***any type*** – they just all have to be the **SAME** type
- Use an indexed variable the same way an “ordinary” variable of the base type would be
- The square brackets [ ] hold the index
  - Can only be an integer number between 0 and (size – 1)
    - Can also be a variable that represents an integer number

Indexed Variable Assignment

- To assign a value to an indexed variable, use the assignment operator (just like with other variables):

```
int n = 2;
score[n + 1] = 99;
```

  – In this example, variable score[3] is assigned 99

Loops And Arrays

- for-loops are commonly used to step through arrays

<u>Example</u>:

First index is 0

Last index is (size – 1)

```
int max = 9;
for (i = 0; i < 5; i++)
cout << score[i] << " off by "
        << (max – score[i]) << endl;
```

could display the difference between each score and the maximum score stored in an array

Declaring An Array

- When you declare an array, you **MUST** declare its **size** as well!

```
int MyArray[5];
//Array has 5 non-initialized elements

int MyArray[] = {1, 2, 5, 7, 0};
// Array has 5 initialized elements

int MyArray[5] = {1, 2, 5, 7, 0};
// This is ok too!
```

Constants and Arrays

- You can use **constants** (but <u>not</u> variables) to ***declare*** size of an array
  - Allows your code to be easily altered for use on a smaller or larger set of data

<u>Example:</u>

```
const int  NUMBER_OF_STUDENTS = 50; // can change this later
int score[NUMBER_OF_STUDENTS];
       …
for ( int i = 0; i < NUMBER_OF_STUDENTS;  i++)
    cout << score[i] << endl;
```

  - To make this code work for any number of students, simply change the value of the constant in the 1st line…

Variables and Declarations

- Most compilers ***do not allow*** the use of a ***variable*** to **declare** the size of an array

Example:
```
cout << "Enter number of students: ";
cin >> number;
int score[number];
```

- This code is illegal on many C++ compilers
- Later we will take a look at **dynamic arrays** which do support this concept (but using *pointers*)

Arrays and Computer Memory

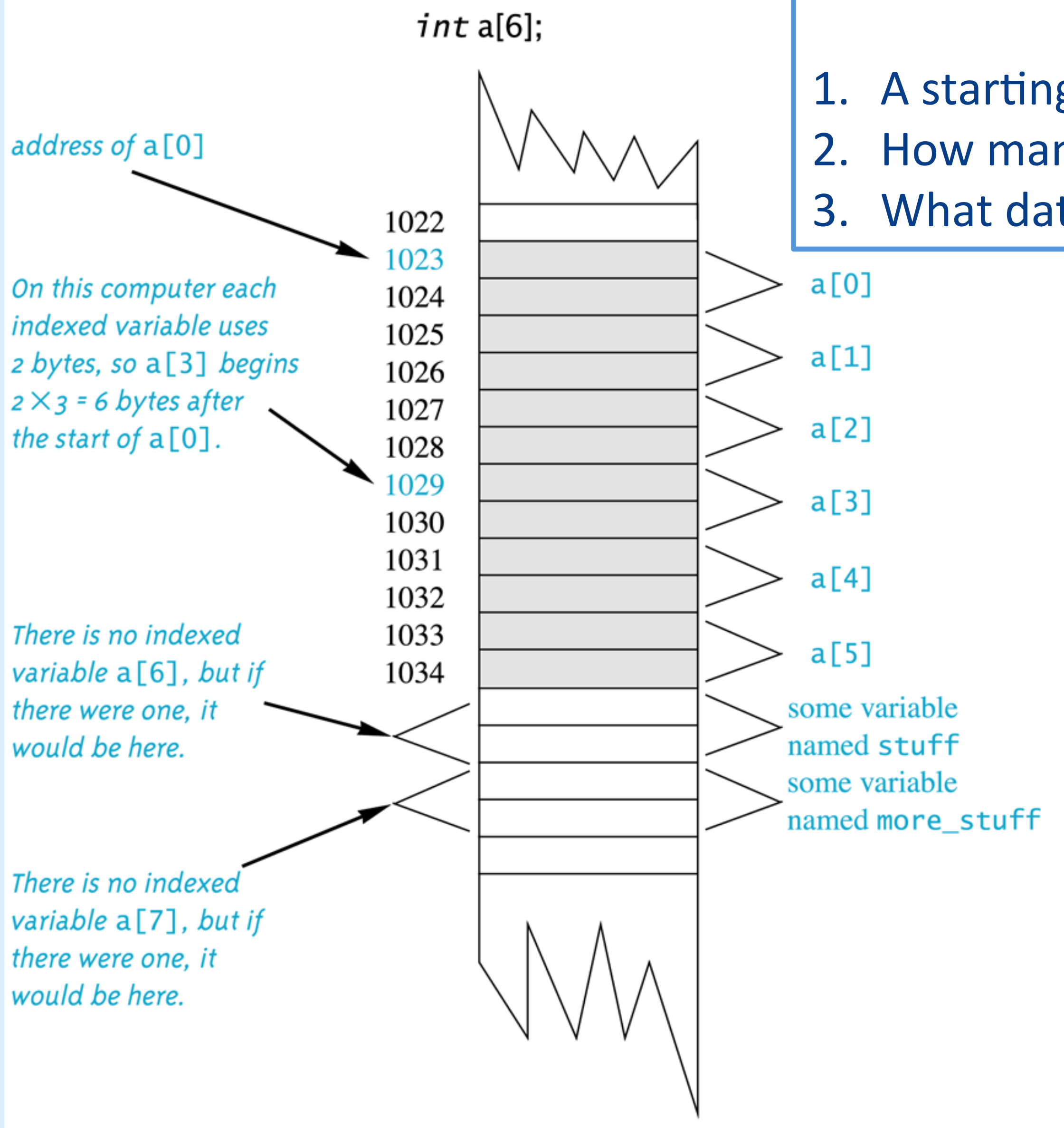
- When you declare the array `int a[6]`, the compiler…
  - Reserves memory for six variables of type **int** starting at some memory address (that the compiler picks)
  - The variables are stored one after another (adjacent in memory)
  - The address of **a[0]** is remembered
    - The addresses of the other indexed variables is not remembered (b/c there's no need to!)

- If the compiler needs to determine the address of **a[3]**
    - It starts at **a[0]** *(it knows this address!)*
    - It counts past enough memory for three integers to find **a[3]**

When reserving memory space for an array in C++, the compiler needs to know:

1. A starting address (location)
2. How many elements in array
3. What data type the array elements are

An Array in Memory



int a[6];
address of a[0]
On this computer each indexed variable uses 2 bytes, so a[3] begins 2 × 3 = 6 bytes after the start of a[0].
There is no indexed variable a[6], but if there were one, it would be here.
There is no indexed variable a[7], but if there were one, it would be here.
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
some variable named stuff
some variable named more_stuff

Array Index Out of Range

- A common error is using a nonexistent index
  - Index values for **int a[6]** are the values 0 through 5
  - An index value that's not allowed by the array declaration is ***out of range***
  - Using an out of range index value does not produce an error message by the compiler!
    - It produces a WARNING, but the program will often (but NOT always) give a run-time error

Out of Range Problems

- If an array is declared as: `int a[6];`
  and an integer is declared as: `int i = 7;`
- Executing the statement: `a[i] = 238;`
  causes…
  - The computer to calculate the address of the illegal a[7]
    - This address could be where some other variable is stored
  - The value 238 is stored at the address calculated for a[7]
  - You could get run-time errors OR YOU MIGHT NOT!!!
- *This is bad practice! Keep track of your arrays!*

Initializing Arrays

- To initialize an array when it is declared
  - The values for the indexed variables are enclosed in braces and separated by commas

- Example: `int children[3] = {2, 12, 1};`
  Is equivalent to:

```
int children[3];
children[0] = 2;
children[1] = 12;
children[2] = 1;
```

Midterm #2

<u>EVERYTHING FROM LECTURES 7 thru 12</u>

- Functions
  - How to use them, declare them, define them
  - *void* functions
  - Call-by-reference vs. Call-by-value
  - Overloading functions
- Design and Debug of Programs
  - Designing loops concepts
  - Tracing, testing functions, stubs
- Numerical conversions
  - Binary, hex, decimal

- File I/O
  - How to open/close, read/write
  - How to check on bad/non-existent files
  - How to anticipate the end of a file
- Strings and Characters in C++
  - Manipulators and member functions
  - Esp. *get()* and *getline()* and their uses with file I/O
- Introduction to Arrays

Example Question 1

If string `s = “California Dreaming”`, then what are:

a) s.erase(4,13) **`“Caling”`**

b) s.find(“or”) **`5`**

c) s.rfind(“a”) **`14`**

`“California Gleaming”`

d) s.substr(0,11) + “G” + s[2] + s.substr(13,6)

Example Question 2

Convert the binary number 10011 into decimal

10011 = 1 + 2 + 0(4) + 0(8) + 16 = 19

Convert the hexadecimal number F2 into binary

F2 = 11110010

Convert the decimal number 22 into binary

22 / 2 = 11 R 0

11 / 2 = 5 R 1

5 / 2 = 2 R 1

2/ 2 = 1 R 0

1 / 2 = 0 R 1

ANS: 10110

Example Question 3

What is the outcome of this code?

```
void DoesIt(int& x1, string op) {
    cout << “Commencing operation: ” << op << endl;
    for (int i=1; i < 4; i++) {
        cout << “Iteration #” << i << endl;
        x1 *= 2;
    }
}

int j = 2;
string o = “Gaucho”;
DoesIt(j, o);
cout << j << “;” << o << endl;
```

Example Question 4

This code should search for a word inside of a text file and then print that line if it finds the word in it. Complete the missing parts.

```
ifstream infile;
infile.open  (“MyTextFile.txt”)

string word, line;
cout << “Enter word to search:”;
cin >> word ;

getline  (infile, line);
while ( !infile.eof( )  ) {
    if ( line.find(word) != string::npos  )
        cout << line << endl;
    getline (infile, line);
}
```

</LECTURE>