CISC 1110 (CIS 1.5)
Introduction to Programming Using C++

Spring 2012

Instructor : K. Auyeung

Email Address: kenny@sci.brooklyn.cuny.edu
Course Page: http://www.sci.brooklyn.cuny.edu/~kenny/cisc1110
Class Hours: MW 2:40 – 4:45PM 4428N

Agenda

- Midterm A
- Constant Values
- Arrays of Strings
- Two-Dimensional Arrays

Constants

• constants are types of data values that are defined in programs and do NOT change while the program runs

• these are similar to variables because they have a name, data type and value

• BUT they are DIFFERENT from variables because the value DO NOT CHANGE

• some libraries define constants as well as functions

• you can also define your own constants

- to define a constant, use the keyword const
- for example:

```
#include <iostream>
using namespace std;

int main() {
   const int NORTH = 0;
   const int WEST = 1;
   const int SOUTH = 2;
   const int EAST = 3;
   cout << "the robot is moving " << EAST << "\n";

   return 0;
} // end of main()
```

• constants are handy for defining the length of an array

```
#include <iostream>
#include <time.h>
#include <stdlib.h>
using namespace std; int main() {
   // declare constant
   const int MAX = 100;
   // declare variables
   int a[MAX];
   int i;
   // initialize random number generator
   srand( time( NULL ));
   // set entries in array to random numbers
   for ( i=0; i<MAX; i++ ) {
     a[i] = rand();
   }
   // output array entries

   for ( i=0; i<MAX; i++ )
      cout << a[i] << endl;

   return 0;
} // end of main()
```

What are Strings

a string in C++ is one of a special kind of complex data type called a class

we will talk more about classes in detail at the end of the term

a class is a compound data type, unlike the simple, native data types we've already discussed (e.g., int, char, bool, double and float)

a class has members:
it has data fields and functions

Reading string from the keyboard

there are two ways to read input values from the keyboard into a string variable:\\

(1) Using cin >>

(2) using the getline() function

the first way, using the >> operator, will only read until the first whitespace character is read (the term ``whitespace'' refers to characters like blank spaces, tabs and newlines); this means that the input will stop as soon as the first whitespace character is read

for example:

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  string s;
  cout << "please enter your name:";
  cin >> s;
  cout << "s = " << s << endl;

  return 0;
} // end main()
```

if the user enters david ortiz when the program asks
please enter your name:
then the value of s will be "david”

HOWEVER, when reading a string variable using the getline() function, the input will stop as soon as the first newline character is read (i.e., the user hits the ENTER key on the keyboard), e.g.:

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  string s;
  cout << "please enter your name:";
  getline( cin, s );
  cout << "s = " << s << endl;

  return 0;
} // end of main()
```

here, if the user enters david ortiz when the program asks please enter your name: then the value of s will be "david ortiz"

Strings: Declaring and Initializing

•strings are declared like this:

string s; where s is a variable whose data type is a string

•you can set the value of the string using the assignment operator and double quotes ("):

s = "hello";

•NOTE that you use single quotes for char values and double quotes for string values:

char c = 'A';
string s = "hello";

•ALSO NOTE that when you use the string class, you also need to include the string header file, in addition to the one(s) you've already been using:

```
#include <iostream>
#include <string>
using namespace std;
```

Strings: Concatenation Operator

The plus sign (+) is the concatenation operator, e.g.:

```
string s1, s2, s3;

s1 = "david ";

s2 = "ortiz";

s3 = s1 + s2;
```

After the above code fragment, the value of s3 will be ”david ortiz”

Strings: Indexes

• a string is like an array of char

•so you can use the index of the individual characters of the string just like you can use the indexes of the individual elements of an array

•if you have:
string s = "ortiz";
then:

s[0] is assigned the value o (the letter “oh”)
s[1] is assigned the value r
s[2] is assigned the value t
s[3] is assigned the value i
s[4] is assigned the value z

Strings: length

• if you have:
string s = "ortiz";
then the length of the string is 5

• there are two member functions of the string class that will tell you the length of a string: length() and size() (they do the same thing)
you call them like this:

```
string s1;
int n1, n2;
s1 = "ortiz";
n1 = s1.length();
n2 = s1.size();
```

After this code fragment, the value of n1 will be 5 and so will the value of n2

Strings: searching

• the find() member function is used to locate a substring within a primary string

• the function returns the value of the index in the primary string at which the substring starts, if the substring exists in the primary string;
or else the function returns the constant string::npos

• for example:

```
string s1 = "david ortiz";
int n1, n2;
n1 = s1.find( "avid" );
n2 = s1.find( "ask", 0 );
```

After the above code fragment:
the value of n1 will be 1
the value of n2 will be string::npos or -1
the first argument to the find() function is the substring to search for
the second argument (which is OPTIONAL) to the find() function is the index in the primary string at which to start searching; 0 means to start searching at the beginning.

Strings: editing

- there are three editing member functions that are part of the string class:

– insert()
– replace()
– erase()

• the insert() function inserts a substring into the primary string

syntax:
mystring.insert(<pos1>, <str>);

inserts the entire string str into mystring, starting at position pos1 in mystring for example:

if mystring = “hello”, then mystring.insert(0, "goodbye");
will change the value of mystring to “goodbyehello”

if mystring = “hello”, then mystring.insert(1, "goodbye");
will change the value of mystring to “hgoodbyeello”

if mystring = “hello”, then mystring.insert(5, "goodbye");
will change the value of mystring to “hellogoodbye”

• the replace() function replaces a substring in one string with another string

syntax:
mystring.replace(<pos1>, <pos2>, <str>);
replaces the section of the string mystring between position <pos1> and <pos2> with string str

for example:
if mystring = “hello”, then mystring.replace(0, 3, "goodbye");
will change the value of mystring to “goodbyelo”

• the erase() function erases a number of characters from a string

syntax:
mystring.erase(<pos>, <num>);
erases <num> characters from the string mystring starting at position <pos>

for example:
if mystring = “hello”, then mystring.erase(0, 3);
will change the value of mystring to “lo”

if mystring = “hello”, then mystring.erase(2, 3);
will change the value of mystring to “he”

• complete example:

```
#include <iostream>
#include <string>
using namespace std;

int main() {
   string s = "ortiz";
   cout << "first, s=" << s << endl;
   s.insert( 0, "david " );
   cout << "second, s=" << s << endl;
   s.replace( 0, 1, "D" );
   s.replace( 6, 1,"O" );
   cout << "third, s=" << s << endl;
   s.erase( 1, 4 );
   cout << "fourth, s=" << s << endl;

} // end main()
```

The output of the above program will be:

```
first, s=ortiz
second, s=david ortiz
third, s=David Ortiz
fourth, s=D Ortiz
```

Strings: parsing

• the substr() member function is used to extract a substring from within a primary string

• example:

```
#include <iostream>
#include <string>
using namespace std;

int main() {
   string s1 = "D Ortiz";
   string s2;

   cout << "s1=" << s1 << endl;
   s2 = s1.substr( 2, 7 );
   cout << "s2=" << s2 << endl;

} // end main()
```

The output of the above program will be:

```
s1=D Ortiz
s2=Ortiz
```

Arrays of Strings

• because a string is a special kind of data type (called an object), you can also define arrays of strings, for example:

```
string myArray[10];
```

• an array of strings is handled basically just like an array of a simple data type (like an array of ints)

• example:

```
#include <iostream>
#include <string>
using namespace std;

int main() {
     const int MAX = 8;
     string myArray[MAX] = { "Last night I had the strangest dream",
                             "I ever dreamed before",
                             "I dreamed the world had all agreed",
                             "To put an end to war",
                             "I dreamed I saw a mighty room",
                             "The room was filled with men",
                             "And the paper they were signing said",
                             "They'd never fight again"  };

     cout << "here is your song: ";
     for ( int i=0; i<MAX; i++ ) {
     cout << i << "-th line = " << myArray[i] << endl; } // end for i

     return 0;
} // end of main()
```

Escape Sequences

- '\' escape character
- Tells the computer to interpret the character following the escape character
- Allows us to print some unprintable characters
- i.e. \n

Common Escape Sequences

\n	new line
\t	tab
\b	backspace
\”	double quote
\\	backslash
\%	percent sign
\0	null character
\a	BEL (bell)
\f	form feed (new page on printer)
\r	carriage return