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

- Review reading strings from the keyboard
- npos revisited
- Formatted output
- Multidimensional arrays
- Midterm review

READING STRINGS 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"

NPOS REVISITED

npos is a static member constant value with the greatest possible value for an element of type size_t.

When used in some *pos* parameters that allow for out-of-range values, npos indicates the end of the string.

As a return value it is usually used to indicate failure.

This constant is actually defined with a value of -1, which because size_t is an unsigned integral type, becomes the largest possible representable value for this type.

FORMATTED OUTPUT

- part of iostream library
- cout.setf() defines the type of output field
- cout.precision() sets the decimal precision (for real numbers)
- cout.width() sets the width of the output field
- example:

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

 int main() {
   cout << "here's a table with lined-up columns:\n";
   cout.width( 10 );
   cout << "monday";
   cout.width( 10 );
   cout << "tuesday";
   cout.width( 10 );
   cout << "wednesday";
   cout << endl;

   cout.width( 10 );
   cout << "1";
   cout.width( 10 );
   cout << "2";
   cout.width( 10 );
   cout << "3";
   cout << endl;

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

- output:

```
here's a table with lined-up columns
```

• another example:

```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
        const int A = 5;
        const double B = 3.4568;
        doble C;
        cout << "output using fixed precision, 2 decimal places:\n";
        cout.setf( ios::fixed );
        cout.precision( 2 );
        cout << "B=" << B << endl;
        cout << "output using width=10, left justified:\n";
        cout.setf( ios::left );
        cout.width( 10 );
        cout << "B=" << B << endl;
        cout << "output using width=10, right justified:\n";
        cout.setf( ios::right );
```

.
.
.

```
    cout.width( 10 );
    cout << "B=" << B << endl;
    cout << "you have to repeat the formatting if you "
         << " want the same thing again:\n";
    C = sin( B );
    cout.setf( ios::right );
    cout.width( 10 );
    cout << "C=" << C << endl;

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

MULTIDIMENSIONAL ARRAYS

- so far, we have only talked about one-dimensional arrays, where you can think of the data as being stored in EITHER a single row OR a single column in spreadsheet

- but we can also define multi-dimensional arrays (arrays with more than one dimension)

- today we'll talk about two-dimensional arrays.

- note that you can define multi-dimensional arrays (i.e., more than 2 dimensions), but that is beyond the scope of this class

- With two-dimensional arrays, you can think of it like data being stored in BOTH rows and columns.

- you define a two-dimensional array by adding another set of square brackets, with a value in between indicating
the size of the dimension, for example:
int myArray[3][4]; which declares a two-dimensional array with 3 rows and 4 columns

- we will refer to the two dimensions as rows and columns so that you can visualize a spreadsheet, like this:

There are 3 rows, going horizontally There are 4 columns, going vertically

- C++ defines arrays as "row major'', which means that the row dimension comes first, then the column dimension

- example on next slide:

```
#include <iostream>
using namespace std;
#include <time.h>
#include <stdlib.h>

// declare global variables
const int LENGTH = 3;
const int WIDTH = 4;
int main() {
    int myArray[LENGTH][WIDTH];
    srand( time( NULL ));

    //initialize array
    for ( int y=0; y<length; y++ ) {
        for ( int x=0; x<width; x++ ) {
             myArray[y][x] =  ( rand() % 10 ) + 1;
        } // end for x
    } // end for y

    // print array
    for ( int y=0;  y<length; y++ ) {
        for ( int x=0; x<width; x++ ) {
             cout << "(" << x << "," << y << ") = "
                  << myArray[y][x] << " ";
        } // end for x
        cout << endl;
    } // end for y

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

midterm review

unit I: introduction; output

- software development cycle
- what is an IDE
- compiling simple C++ programs • the use of cout
- “hello world” program
- what is the ASCII table

unit II: simple data types

- storing data
- rules for creating and using variables
- primitive data types: int, double, char bool
- assignment operators
- mathematical operators
- floating point versus integer division
- outputting variable values
- logical operators
- “truth” tables

unit III: more data types

- random numbers
- for loops
- arrays
- C++ style strings
- C++ string functions
- C style strings
- two-dimensional arrays
- constants

unit IV: input; more on variables

- keyboard input
- formatted output
- shortcut operators
- base conversion
- hexadecimal and octal constants

about the midterm

- 10% of term grade
- you may bring one index card of notes
- you may NOT use any electronic device
- you will be asked to find compiler errors in a program
- you will be asked to show what the output is of different programs
- you will be asked to write a short program from scratch
- you will be asked to convert numbers between base 2, base 8, base 10 and base 16