EECS 161
Intro to Programming I
2-D arrays & Command Line Args
Chap. 5, 9.1, and 10
C-string Library

• #include <cstring>

• Can’t do this- astring = “Howdy!”; //Why?

• Need to do this- strcopy(astring, “Hello”);

• cstring is a library of standard operations on c-strings
Predefined C-string Functions

strncpy(tgt, src) – copies target to source
strncpy(tgt, src, n) – copies only n characters
strcat(tgt, src) – concatenates src onto target
strcpy(tgt, src,n) – concatenates only n chars
strlen(src) – returns the number of chars
strcmp(s1, s2) – returns 0 if s1 and s2 are equal,
                returns <0 if s1 < s2, and >0 if s1 > s2
strncmp(s1, s2, n) - only n characters are compared

See table on page 375-376
Command Line arguments

• Pass information in from the command line of your program

• int main(int argc, char *argv[])

• argc is the number of items on the command line
• The program name is included so argc is always at least = 1
• argv is an array of c-strings where each element is a “word” from the command line
Command Line arguments

- Inside foo.cpp when called as
  - foo duck drop roll

```c
int main(int argc, char *argv[])
Then
argc = 4
argv[0] = "foo"
argv[1] = "duck"
argv[2] = "drop"
argv[3] = "roll"
```
String Class

```c
#include "string"
string astring;
cin >> astring;
cout << astring;
```

• Member functions rather than library functions
• So `astring.length()` rather than `strlen(s1)`

• Please read pages 401-405, particularly box on page 404. `=` and `==` behave differently for string class and c-strings
Converting String <> Cstring

char aCString[] = "This is my C-string.";
    string stringVariable;
    stringVariable = aCString;

strcpy(aCString, stringVariable.c_str());

See page 408 for more specifics
Multidimensional Arrays

• data_type array_name[rows][cols];
  – int array[2][3];
  – int array[4][2][3];
  – int array[2][4][2][3];

• What are examples of these?
  – 2-D – Matrices, Spreadsheet, Minesweeper, Battleship, etc.
  – 3-D – Multiple Spreadsheets, (x, y, z) system
  – 4-D – (x, y, z, time) system
Initializing 2-D Arrays

• **Declaration:** int array[2][3] = {{0,0,0},{0,0,0}};

• **Individual elements:** array[0][0]=0; array[0][1]=0; array[0][2]=0; array[1][0]=0; array[1][1]=0; array[1][2]=0;

• **Loop:**
  
  for(i = 0; i < 2; i++)
      
      for(j = 0; j < 3; j++)
          
          array[i][j]=0;

• **Why do we need multiple brackets?**
Reading/Printing 2-D Arrays

• **Reading Array Values**
  
  ```
  for(i = 0; i < 2; i++)
    for(j = 0; j < 3; j++) {
      cout << “Enter a value for ” << i << “, ” << j << “: ”;
      cin >> array[i][j];
    }
  ```

• **Printing Array Values**
  
  ```
  for(i = 0; i < 2; i++)
    for(j = 0; j < 3; j++)
      cout << “Array: ” << array[i][j] << endl;
  ```