CS 161
Intro to CS I
Continue Arrays
Odds and Ends...

• Last week to demo Assignment #4
• Be working on Assignment #5
```cpp
#include <iostream>
#include <cstring> //string.h

using namespace std;

int main() {
    //char str[20]; //statically allocated can only hold 19 chars entered
    //instead of limiting the number of chars entered by the user, we can
grow our array to hold the right amount of chars
    char s; //read a char at a time
    char *str=new char[1]; //create smallest c-string, which is just null char '\0'
    char *temp; //need to capture old string in pointer

    //add '\0' to smallest c-string with length zero, but with one element
    str[0]= '\0';
    cout << "Length: " << strlen(str) << endl;

    //read one character from input and add to string if it is not the newline/enter
    cin.get(s);
    while(s!='\n'){
        temp=str; //point to old string, so we don't lose it
        str=new char[strlen(temp)+2]; //add space to the array, strlen doesn't count null char + extra
        strcpy(str,temp); //copy old string into new string (destination, source)
        str[strlen(temp)]=s; //add new character where null was in the old string
        str[strlen(temp)+1]= '\0'; //add null character to the last element in array
        delete [] temp; //get rid of old array, so we don't have memory leak
        cin.get(s); //get another character to make sure it isn't ' \n' to stop reading input
    }

    return 0;
}
```
int main() {
    int array[5];
    ...
    pass_1darray(array);
    ...
}
void pass_1darray(int *a) {
    cout << “Array at zero: ” << a[0] << endl;
}
OR
void pass_1darray(int a[]) {  
    cout << “Array at zero: ” << a[0] << endl;
}
Creating Memory in Functions

Advantages to Dynamic Memory

int *i=NULL; //created in main function

create_mem(&i); //call in main void

create_mem(int **m) {
    *m = new int[4];
}

OR

i = create_mem(); //call in main

int * create_mem() {
    return new int[4];
}
What About Memory Leaks?

• What happens here...

...  
int main () {
    int *i=NULL;  //created in main function
    while(1) {
        i = create_mem();  //call in main
    }  
}

int * create_mem() {
    return new int[4];
}
Fixing Memory Leaks...

- What happens here...

...  
int main () {
    int *i=NULL; //created in main function
    while(1) {
        i = create_mem(); //call in main
        delete [] i;  //free memory that i points to, preventing mem leaks
    }
}
int* create_mem(){
    return new int[4];
}
Demo...
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;
Static 2-D arrays...
Demo...
Passing a 2-D Array (Static)

```c
int main() {
    int array[5][5];
    ...
    pass_2darray(array);
    ...
}
void pass_2darray(int a[][5]) {
    cout << "Array at zero: " << a[0][0] << endl;
}
OR
void pass_2darray(int a[5][5]) {
    cout << "Array at zero: " << a[0][0] << endl;
}
```
Command-line Arguments
Static vs. Dynamic 2-D arrays...
Jagged Arrays

```c
int *array[2];
array[0] = new int[3];
array[1] = new int[2];
```
Demo...