CS 161
Intro to CS I

Static vs. Dynamic Multi-Dimensional Arrays and Command-Line Args
Odds and Ends

- Exam II Friday, 2/26
- Assignment #5 due Sunday

```
valgrind exe
```

```
Cannot have a mem leak
```

```
user enters: "ask"
```

```
1. Limit # of chars read by user: 3000, 4 chars
2. Cannot make an array of specific size, 256, 512, 1024
```

```
read character by character
```

```
3000 - 3001 chars
```

```
```

```
```
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

Char morse[26][6];
C++ string morse[26];
C style char* morse[26];
morse[0]='A';
...
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...

C-style string

int array[][3]
rows cols

for all rows? stack,
for all cols?

A row at a time is loaded out in mem.
Demo...

```cpp
#include <iostream>
using namespace std;
#define ROWS 2
#define COLS 3
int main () {
    int a[ROWS][COLS];

    for(int i=0; i<ROWS; i++)
        for(int j=0; j<COLS; j++)
            a[i][j]=100;

    cout << "where initial array pointer lives: " << &a << endl;
    cout << "where 1st row pointer lives: " << &a[0] << endl;
    cout << "where 1st row pointer lives: " << &a[0][0] << endl;
    cout << "where 1st row lives/1st element lives: " << a[0] << endl;
    cout << "where 1st row lives/1st element lives: " << &a[0][0] << endl;
    cout << "1st element contents: " << a[0][0] << endl;
    cout << "where 1st row lives/2nd element lives: " << &a[0][1] << endl;
    cout << "where 1st row lives/3rd element lives: " << &a[0][2] << endl;
    // [0][3] is the same as [1][0] ONLY because static
    cout << "where 2nd row lives/4th element lives: " << &a[0][3] << endl;
    cout << "where 2nd row lives/4th element lives: " << &a[1][0] << endl;
    return 0;
}
```
Passing a 2-D Array (Static)

```c
int main() {
    int array[5][5];
    ...
    pass_2darray(array);
    ...
}
void pass_2darray(int a[5][5]) {
    cout << "Array at zero: " << a[0][0] << endl;
}
OR
void pass_2darray(int a[][5]) {
    cout << "Array at zero: " << a[0][0] << endl;
}
```

- Can modify the contents of array
- \( i.e. \) elements
- Stride is important
- \( \text{cols} \) over \( \text{rows} \)
- 5 more cols from \( C_t \) at \( [0, 0] \)

---

OSU Oregon State University
In-class Exercise #5

• Get into groups of 4-5.
• Draw the picture and pseudo code for creating an int ****p; that points to int i; (no other named memory locations allowed!!!)