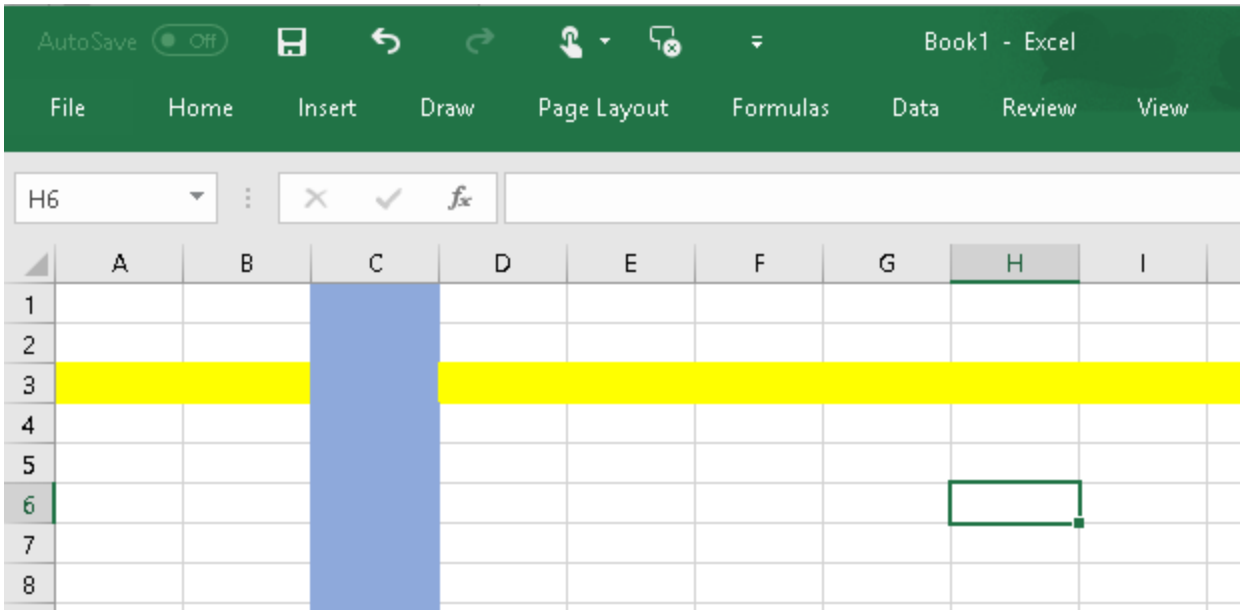
CISC 1600/1610
Computer Science I

Multi-Dimensional Arrays
And
String operations

2D Arrays

- Rows and Columns, just like Excel



2-Dimensional arrays

- Storing a table of data

```
const int NUMSTUDENTS=5, NUMTESTS=3;
char grades[NUMSTUDENTS][NUMTESTS];
grades[2][0]='A';
grades[3][0]='B';
```

grades

	Test1	Test2	Test3
Amy	???	???	???
Rob	???	???	???
David	A	???	???
Bill	B	???	???
Alice	???	???	???

2-Dimensional arrays

- Storing a table of data

```
const int NUMSTUDENTS=5, NUMTESTS=3;
char grades[NUMSTUDENTS][NUMTESTS];
grades[2][0]='A';
grades[3][0]='B';
```

grades

	Test1	Test2	Test3
Amy	???	???	???
Rob	???	???	???
David	A	???	???
Bill	B	???	???
Alice	???	???	???

Column versus Row

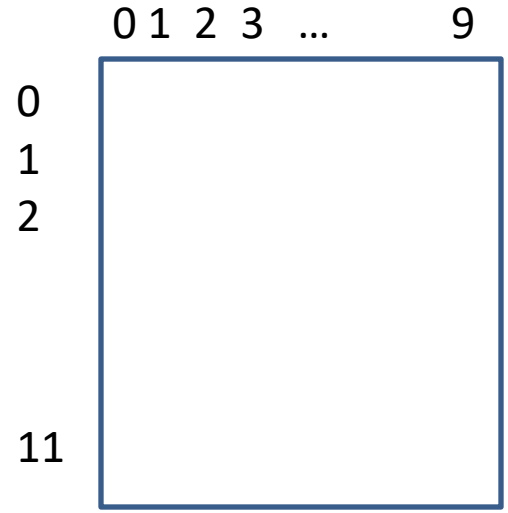
Col 0 Col 1 Col2 Col3 Col 4

Column versus Row

Fill Top to Bottom Array[rowNum][columnNum]

	Col 0	Col 1	Col2	Col3	Col 4
Row#0	87	93	96	89	91
Row#1	78	89	79	92	95
Row#2	58	99	97	90	98

Lab 12: Create the Multiplication Tables



1. Declare 2 constant values:

   ROWSIZE = 12;

   COLSIZE = 10;

2. Define a 12 row by 10 column array

   multArray

Lab 12: 2 Loops to Load the array

```
for (int rindex = 0; rindex < ROWSIZE; rindex++)
//goes through each rowindex in the array
{
        for (int cindex = 0; cindex < COLSIZE; cindex++)
        //goes through each columnindex in the array
        {
                int cval = cindex + 1;
                int rval = rindex + 1;
                myArray[rindex][cindex] = cval*rval;
        }
}
```

Lab 12: Print the Multiplication Tables

Use the double loops to print it

```
#include <iomanip>      // required for std::setw
```

setw to set each value to be up to 6 characters

<u>Extra:</u>

- Print Row and/or Column Headers
- Add border lines

InClass Exercise:
Printing the Multiplication Tables

- Need 2 loops (outer and inner) again…
- Need to move to next row after finishing each

```
for (int rindex = 0; rindex < ROWSIZE; rindex++)
{
    for (int cindex = 0; cindex < COLSIZE; cindex++)
        cout << setw(6) << myArray[rindex][cindex] << "  ";
}
cout << endl;
```

Now place the labels for the Columns…

```
for (int cindex = 0; cindex < COLSIZE; cindex++)

{
        cout << setw(6) << “LABEL " << cindex << "  ";


}
cout << endl;
```

An Object

What is an object?

- A collection of data, called '**members**'
- Member **functions** (also called methods) that operate on them
- Based on **CLASS** definitions, to be discussed later

String – your first object

```
#include <string>
```

String data:

A string is essentially an array of characters, ending with a '\0'

String – data

string name = "hannah";

Really looks like this in memory:

Value	h	a	n	n	a	h	\0
[Index]	[0]	[1]	[2]	[3]	[4]	[5]	[6]

String data:

A string is essentially an array of characters, ending with a '\0'

String Member Function **length()**

h	a	n	n	a	h	\0
[0]	[1]	[2]	[3]	[4]	[5]	[6]

length() - returns the length of the string
- total count of characters

```
string name = "hannah";
cout << name.length(); // actual chars (not \0)
```

6

There are 6 characters, in indices 0 through 5.

http://www.cplusplus.com/reference/string/string/

String function: **getline**

Reads a full line (spaces included, until the end of line) into a string

Example:

```
string wholeLine;
getline(cin, wholeLine);
```

String function: getline from a file

Example:

```
string wholeLine;
ifstream infile;
infile.open("inventory.dat");
getline(infile, wholeLine);
```

String Element Access **at()** & **[]**

Two ways to access elements in a string - like an array:

```
string s = "I like dogs";
cout << s.at(5);  // character at index 5
```

e

```
cout << s[5];   // returns the character at index 5
```

e

Can also use these to modify one character:

s[5] ='q'; OR s.at(8)='q';

substr(index,len) function

You may split a String using substr function , one index and length. Example

```
string line = "We are here and happy";
String word = line.substr(7,4);
cout << word;
```

```
here
```

Note that 2nd number in substr is number of characters you need starting from the index

find() function

You may find a character you need by using a find() function. Example

```
string space = " ";
int  pos  = 0;
string line = "We are here and happy";
word = line.substr(pos,line.find(space));
cout << word;
```

```
We
```

Another Example

```
string space = " ";
int  pos  = 3;
string line = "We are here and happy";
word = line.substr(pos,line.find(space));
cout << word;
```

```
are
```

InClass exercise

Write a program to ask a sentence from the user.

Then, split the sentence into single words and display the split words on the screen

Example:

please enter a sentence

We are here

INFO: If I remove the spaces and split the words

We

are

here

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

int main(){
   string line;
   cout << "please enter a sentence" << endl;
   getline (cin, line);
   cout << "If I remove the spaces and split the words" << endl;
   string space = " ";
   string word;
    int pos  = 0;        // position
     char cursor = '+';   // + means there are more words, q means quit
          while (cursor != 'q')      {
               word = line.substr(pos,line.find(space));
               cout << word << endl;
               pos = pos + word.length()  + 1 ;
                  if (line[pos-1] == '\0')
                     cursor = 'q';
                  else
                    line = line.substr(pos,line.length());
                 pos =0;  //re-initialize position
           }
  cout << endl;
}
```

Writing into a string array

You may have a split function such as:

```
void split (string line, string splitter)
```

Call in a main() as in:

```
int main()
        {   string line;
            cout << "please enter a sentence" << endl;
            getline (cin, line);
            split (line, " ");
        }
```

Displaying all string elements with a for-each loop

You may use a for-each loop for a **string array** defined as **words[i] = word;**

```
for (string a : words)
    cout << a << endl;
```

String Concatenation

```
#include <cstdlib>
```

This is called concatenation (gluing two strings together):

```
string firstname = "Maria";
string lastname = "deSuprima";
string FullName = firstname + " " + lastname;
// Fullname is now "Maria deSuprima"
```

Compare function

```
string s1 = "hello 1";
string s2 = "hello 2";
int x = s1.compare(s2); // returns 0 if equal,
    // < 0 if s1 is 'less', and
    // > 0 if s1 > s2 (length or letter order)
cout << x; // would print -1
```

Usage:

```
    if(s1.compare(s2)  ==  0)
    // they match
```

stoi , stof, stod Functions: Converting string to other types

```
string s1 = "42";
int x = stoi(s1); // converts the string to int

string s2 = "42.5";
float x = stof(s1); // string to float

string s2 = "42.5";
double x = stod(s1); // string to double
```

A case study :Palindrome

- A word or number that reads the same forwards or backwards
- How would we read in letters into an array and determine if the letters form a palindrome?
- Hint: 2 indexes
- Set a variable called isPalindrome to report the answer

Palindrome sentences

- Live not on evil.
- Step on no pets!

- Read using

```
getline(cin,s);
```

- Remove spaces and punctuation
- Then decide if it is a palindrome

Parallel (helper) Arrays

- Create a 3x7 array to hold work hours
- Create an array for each day's 'name'
- Create an array to hold a 3 names

	Sun	Mon	Tues	Wed	Thurs	Fri	Sat
Anna							
Mark							
James							

Parallel Arrays

Now calculate the Average Work Hours (per day)

	Sun	Mon	Tues	Wed	Thurs	Fri	Sat
Anna							
Mark							
James							

Average							

Parallel Arrays

How do you calculate the average of the first day (Sunday) ?

Lab 13: Parallel Arrays. Step 1

In a function, read in up to 50 employee names, one per line. If the name is "quit", stop reading.

Name
Anna
Mark
James

Create a function called
getEmployeeNames
Arguments:
Name array (first and last name)
Maximum size
Return the number of names filled.

Lab 13: Parallel Arrays . Step 2

For each employee, read in their hourly pay rate.

Name	Hourly Pay
Anna	19
Mark	18
James	25

Create a function called
getHourlyPay
Arguments:
NameArray (filled in Step 1)
HourlyPayArray (starts empty)
NumberOfNamesFilled (from Step1)

Last Homework (Steps 3/4/5)

Lab 13: Parallel Arrays. Step 3

For each employee, read in their hours for each day. If it was a day off, a 0 is entered.

Name	Hourly Pay	WorkHours						
Anna	19	3.5	4	3	0	0	3	3.2
Mark	18	0	5	6	2	2	4	0
James	25	3	0	0	4	4.6	0	0

In main(), read in a 2D array of WorkHours

– Not have to be in a function

Assume 7 days per week

Lab 13: Parallel Arrays. Step 4

Calculate the Total Hours for each employee

Name	Hourly Pay	WorkHours							Total Hours
Anna	19	3.5	4	3	0	0	3	3.2	13.2
Mark	18	0	5	6	2	2	4	0	19
James	25	3	0	0	4	4.75	0	0	11.75

In main():

Using WorkHours, calculate the Total Hours, which is stored in a new array called TotalHours

Lab 13: Parallel Arrays. Step 5

Calculate the Gross Pay* in a GrossPay array

* pay before deductions for taxes, benefits, etc.

Name	Hourly Pay	Total Hours	Gross Pay
Anna	19	13.2	250.80
Mark	18	19	342.00
James	25	11.75	293.75

Create a function called CalculateGrossPay.
Arguments:

HourlyPay (previously filled)
TotalHours (previously filled)
GrossPay (empty)
NumberOfNamesFilled (previously calculated)

Lab 13: Parallel Arrays. Step 6

Name
Anna
Mark
James

Total Hours
13.2
19
11.75

Gross Pay
250.80
342.00
293.75

Print on each row:

Employee Name (in a 20 character field)
TotalHours (in 12 chars) and
Gross Pay (in 10 chars)

Lab 13: Parallel Arrays. Step 7

Use a function to determine the highest paid to any employee. Give their name and pay.

Hint: find the INDEX of the highest paid and return that to also print the name.

Use another function to find the lowest paid. Give their pay and full name.

Hint: find the INDEX of the lowest paid and return that to also print the name.

Example:

Highest Pay: $342.00 Mark

Lowest Pay: $250.80 Anna