COMP1511 - Programming Fundamentals

Term 2, 2019 - Lecture 7

What did we learn last week?

- **Code Reviews**
- Helping each other with our code!
- **Looping**
- Repeating code
- **Theory of a Computer**
- A processor and some memory
- **Arrays**
- Storing multiple variables together in a collection

What are we covering today?

Arrays

- Recap of what arrays are
- A little bit more information
- Arrays inside arrays

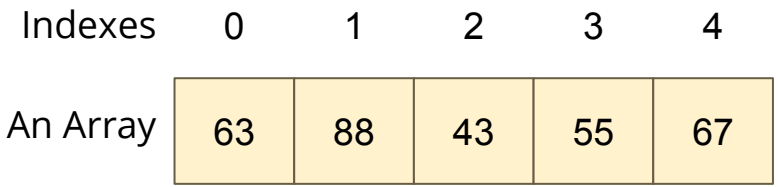
Assignment 1 - CS Paint

- How does the assignment work?
- Hints on getting started

What is an array?

A collection of identical variables

- Contains multiple variables all of the same type
- Declared using a variable type and a size
- Individual variables are accessed using an index



Using Arrays in C

Some example code of an array

```
int main (void) {
    // declare an array of doubles, size 4, initially all 0
    double myArray[4] = {0};

    // assign a value
    myArray[1] = 0.95;
    // test a value
    if (myArray[2] < 1) {
        // print out a value
        printf("Third element is: %lf", myArray[2]);
    }
```

Accessing multiple values at once

Loops and Arrays go together perfectly

- Accessing all members is a reasonably simple while loop

```
int main (void) {
    // declare an array of doubles, size 4, initially all 0
    double myArray[4] = {0};

    // loop through the array and output the elements
    int i = 0;
    while (i < 4) {
        printf("%lf\n", myArray[i]);
        i++;
    }
```

Creating Arrays

Arrays start at an exact size and don't change

- When we create an array, we give it a size and a type
- Both of those are fixed and won't change

```
int main (void) {
    // declare an array of doubles,
    // size 4
    double myArray[4] = {0};

}
```

```
int main (void) {
    // This declaration is not
    // possible!
    int arraySize = 4;
    double myArray[arraySize] = {0};

}
```

We can't declare an array with a variable size like this!

Using Constants for Array Sizes

If we do want to be able to change the size in code . . .

- We can use a constant to set the size
- Unlike a variable, this cannot change after it is compiled
- It does make our lives much easier if we need a change mid-project

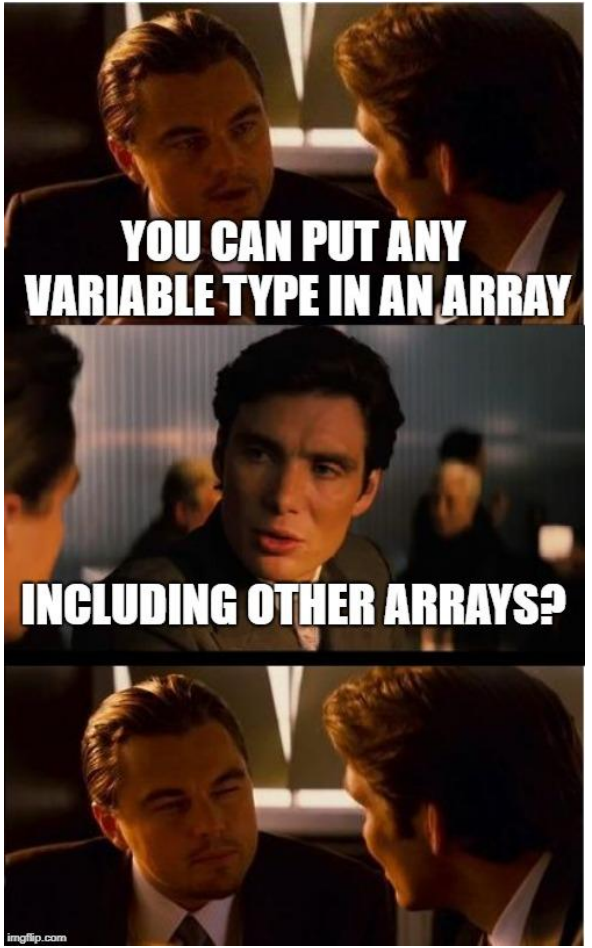
```
#define ARRAY_SIZE 4

int main (void) {
    // This declaration allows us to change the
    // array size while coding
    double myArray[ARRAY_SIZE] = {0};
}
```

Arrays inside Arrays

An Array is a type of variable

An Array can contain any type of variable

- Arrays can be put inside other arrays!
- We call these multi-dimensional arrays
- Think of them as a grid, two or more dimensions



Two Dimensional Arrays

Arrays inside arrays

- Can be thought of like a grid
- The outer array contains arrays
- Each array is a row of the grid
- Addressed using a pair of integers like coordinates
- All inner arrays are of the same type

Indexes	0	1	2	3	4
0	63	88	43	55	67
1	54	52	91	21	32
2	77	58	1	61	79

A 2D Array

Two Dimensional Arrays in Code

```
int main (void) {
    // declare a 2D Array
    int grid[4][4] = {0};

    // assign a value
    grid[1][3] = 3;
    // test a value
    if (grid[2][0] < 1) {
        // print out a value
        printf("The bottom left square is: %d", grid[3][0]);
    }
```

Let's work with 2D Arrays

I would like to make a simple game called "The Tourist"

- The world is a square grid
- The tourist can move up, down, left or right
- Be able to print out the world, including the location of the tourist
- The tourist likes seeing new things . . .
- Track where they've been
- And lose the game if we revisit somewhere we've been

Starter Code

Working with code that's already got some functionality

- We're going to start with some code called tourist.c
- This file already has the ability to print a 2D array
- It's a bit similar to using starting code in the Assignment

Print Map

This is the first time we've seen a function in our code

```
void printMap(int map[N_ROWS][N_COLS], int posR, int posC) {
    int row = 0;
    while (row < N_ROWS) {
        int col = 0;
        while (col < N_COLS) {
            if (posR == row && posC == col) {
                printf("T ");
            } else {
                printf("%d ", map[row][col]);
            }
            col++;
        }
        row++;
        printf("\n");
    }
}
```

Break the problem down into parts

What do we need to do?

- We need to set up our grid and the tourist's position
- The tourist needs to move one step at a time
- Each time the tourist visits a location, we set it to 1
- We also check each location to make sure it's new

The Square Grid World

Variables for the grid and the tourist's position

```c
#include <stdio.h>

// The dimensions of the map
#define N_ROWS 10
#define N_COLS 10

int main (void) {
    int map[N_ROWS][N_COLS] = {0};
    int posR = 0, posC = 0;
```

Break Time

The Tourist is based on a very famous old puzzle . . .

The Travelling Salesman

- Given a list of cities
- And the distances between any pair of cities
- How do you find the shortest distance to travel that visits all the cities exactly once?
- You may see interesting difficult problems like this in the future

Help Sessions

The course is speeding up a bit now, so feel free to use the help!

Help Sessions have expanded for assignment support

- Monday, 4-6pm Clavier (K14 LG20)
- Tuesday, 12-2pm Clavier (K14 LG20)
- Wednesday, 6-8pm Strings (J17 302)
- Thursday, 6-8pm Strings (J17 302)
- Friday, 4-6pm Mat 103 (bring your own device)

Controlling the Tourist

Next Steps

- Let’s add movement
- Then track where the Tourist has been using the map
- After that, we'll check for places we've already been

Looping

- We can loop repeatedly for “turns” to allow the user to input directions

Movement - this code will loop

```
printf("Please enter a numpad direction or 0 to exit: ");
int input;
scanf("%d", &input);
if (input == 4) {
    posC--;
} else if (input == 8) {
    posR--;
} else if (input == 6) {
    posC++;
} else if (input == 2) {
    posR++;
} else if (input == 0) {
    exit = 1;
} else {
    printf("Input is not a numpad direction, please use 2,4,6 or 8\n");
}
```

Tracking the Tourist using the Map

Set each location we visit to 1

```
// loop and let the user control the Tourist's movement
int exit = 0;
while (!exit) {
    // mark the location as having been visited by incrementing
    map[posR][posC] = 1;

    // show the current status
    printMap(grid, posR, posC);

    printf("Please enter a numpad direction or 0 to exit: ");

    // Movement code from previous slide goes here . . .
```

Have we been here before?

We want the game to end if the tourist revisits a location

- If the location we visit is already 1
- Then we're going to exit the game
- We can add this check after our movement

```
// Check if we've been here before
if (map[posR][posC] == 1) {
    printf("We've already been here! How boring!\n");
    exit = 1;
}
```

The Tourist Game

This is now roughly complete

- We can move the tourist
- We can track where we've been
- We can display where we've been as well as current location
- We can exit if we revisit a location

But how safe is it?

- Try different inputs
- Try moving around a bit

What happens if …

Moving around and seeing what works

- Use the controls to move around the map
- Try entering some integers that aren’t the movement

What issues do we find?

Walking off the edge of the map

Our Tourist can walk outside of the bounds of our arrays!

Let's add some code to check if we're outside the map and stop that movement

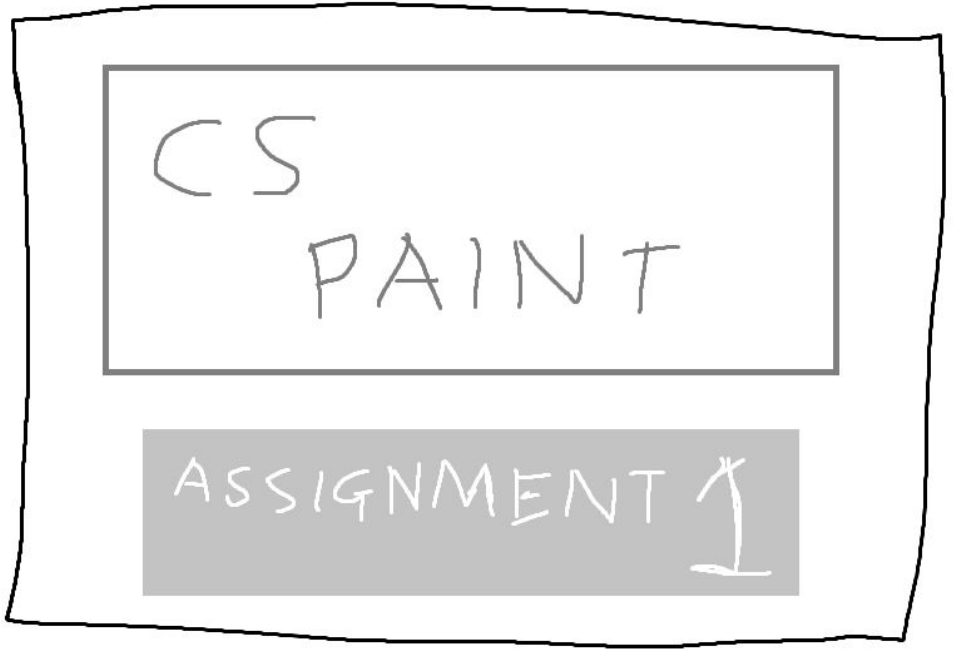
```
// Check if we've walked off the map
if (posR < 0) {
    posR = 0;
} else if (posR >= N_ROWS) {
    posR = N_ROWS - 1;
}
if (posC < 0) {
    posC = 0;
} else if (posC >= N_COLS) {
    posC = N_COLS - 1;
}
```

Where else can we take this code?

What about scoring?

- Could we give the player a score based on the number of places they visited?
- How would we calculate that?
- Also . . .
- Some of this code might be useful in understanding the first assignment

Assignment 1 - CS Paint



What is CS Paint?

It's a program for drawing . . . just like MS Paint, but more CS

- We'll have a 2D array called a canvas of pixels
- We have the ability to read groups of integers from input
- These commands will allow us to draw on the canvas
- The specification is on the Course Website under Assignments

The Specification

It's good to look over this and see what the tasks are

- There's starter code
- Also a set of tests to use
- Programming tasks are separated into ordered stages
- Completing stages in order is highly recommended
- Some helpful information on:
- Running `1511 canvas` for the stylistic look
- Running the reference solution to see what yours should do

What does CS Paint do?

Step by Step

- When it starts running, CS Paint waits for input (there's no prompt)
- Commands are entered as integers
- When the commands are done, it displays a canvas with the results

Your tasks

- Read the input correctly
- Change whichever cells of the canvas need changing

How to get started - learning techniques

We've designed a lot of content to help you get started

- Some of this week's work helps with the Assignment
- The Livestream has video content on reading an unknown number of ints from standard input as well as a brief intro on 2D Arrays
- This lecture exercise has some 2D arrays as well as detecting when we're outside the bounds of a 2D array
- Tomorrow's lecture will look at functions
- This week's lab also reinforces scanf usage and some 2D arrays

How to get started - your own project

Download the Starter Code first!

- One problem at a time. It's safe to ignore Stages 2-4 when you're starting
- Even with Stage 1, work with getting some simple things working first
- Draw a single line with a single command . . . then build up from there

What did we learn today?

Arrays

- More details, use of constants
- Two dimensional arrays

Assignment 1 - CS Paint

- How CS Paint Works
- The Specification
- How to get started