COMP1511 - Programming Fundamentals

Term 2, 2019 - Lecture 9

What did we cover last week?

Arrays

- Two dimensional arrays

Functions

- Separating code for reusability and readability

CS Paint

- An explanation the first Assignment

What are we covering today?

Assignment 1

- Some details on Assessment

Functions and Libraries

- Including files and libraries

Professionalism

- What it means to be able to work effectively

Recap - Arrays

Storage for sets of identical variables

- Declared at a specific size using square brackets []
- A single name for the group of variables
- Individual elements of the array are accessed by index (an integer)

Two Dimensional Arrays

- We can declare arrays of arrays, which allows us to make grids of variables
- We usually use rows and columns to index them

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]);
    }
```

Assignment 1

From a practical perspective . . .

- You will write a C program called CS Paint
- It will all be in a single file called `paint.c`
- Submission is through the `give` system

A recap of the basics

CS Paint

- A 2D array as the canvas
- Your program will receive integers as standard input that are drawing commands
- You will interpret those commands and then make changes to the canvas
- CS Paint is already capable of writing the canvas to output, so you don't need to worry about that

Sequence of Commands

Commands will always be in a particular sequence:

- First integer is the type of command
- Other integers are the extra information that command needs
- Your program will receive one or more commands
- You will process each command in turn
- It's reasonably easy to process the entire command before moving onto the next one (rather than trying to process them all at once)
- When the commands are all finished, you will then print the canvas once to standard output (with the function we've provided)

Submit early, submit often

Using “give” will record your submission and back up your work

- It’s much harder to lose your assignment code if we have it!
- If things go bad, you can roll back to previous versions
- You can access your previous versions using our git repository
- The following link is also available in the assignment page:

`https://gitlab.cse.unsw.edu.au/z5555555/19T2-comp1511-ass1_cs_paint/commits/master`

How will your code be tested?

Your program will be run with a series of test cases

- These tests will not be exactly the same as our autotests
- Remember to check all possible inputs you can think of
- Writing your own test files is potentially very useful

Marking

How do you earn marks in this assignment?

- **Close to a pass (40-50%)**
  - A solid attempt at stage one
  - Being able to draw some lines, but not all possible cases
  - Not necessarily dealing with multiple commands
- **Pass (50-64%)**
  - Code runs without errors
  - Able to draw vertical and horizontal lines
  - A serious attempt has been made at the assignment
  - A higher mark will be given for filling rectangles and/or dealing with multiple commands

Marking Continued

- **Credit (65-74%)**
  - Successfully implements all of Stage 1
  - Some effort on Stage 2 will push marks higher
  - Code is reasonably readable
- **Distinction (75-84%)**
  - Successfully implements both Stage 1 and 2
  - Any effort on later stages will award more marks
  - Code is easy to understand and readable

Marking Continued

- **High Distinction (85%+)**
  - Successfully implements Stages 1-3
  - Stage 4 completion will push marks closer to 100%
  - Code is perfectly explained and elegant to read

Free Marks!!!

Yep . . . get them right here!

Make your code understandable and readable!

- Follow the Style Guide
- This means correct indentation and consistent use of bracketing
- Use variable names that are understandable to a reader
- Have clear comments explaining your intentions (even if the code is not functional)
- Structure your code file so that different sections are clear
- Use functions to separate repetitive code

Questions?

Feel free to ask any questions now!

- Help Sessions have been expanded for one on one consultation if you need help with problems

Recap of Functions

Code outside of our main that we can use (and reuse)

- Has a name that we use to call it
- Has an output type and input parameters
- Has a body of code that runs when it is called
- Uses return to exit and give back its output

Functions in Code

```
// a function declaration
int add (int a, int b);

int main (void) {
    int firstNumber = 4;
    int secondNumber = 6;
    // use the function here
    int total = add(firstNumber, secondNumber);
    return 0;
}

// the function is defined here
int add (int a, int b) {
    return a + b;
}
```

Why use functions?

Why do we separate code into functions?

Saves us from repeating code

- Instead of replicating code, we can write it once
- This also makes the code much easier to modify

Easier to organise code

- Complex functionality can be hidden inside a function
- The flow of the program can be read easily with clear function names

C Libraries

We've already used stdio.h several times

- C has other standard libraries that we can make use of
- The simple C reference in the Weekly Tests has some information
- math.h is a useful library of common maths functions
- `stdlib.h` has some useful functions
- Look through the references (including **`man`** manuals in linux)
- Don't worry if you don't understand the functions yet, some of them have no context in the programming we've done so far

Using Libraries

```
// include some libraries
#include <math.h>
#include <stdlib.h>
#include <stdio.h>

int main (void) {
    int firstNumber = -4;
    int secondNumber = 6;

    // change a number to its absolute value
    firstNumber = abs(firstNumber);

    // calculate a square root
    int squareRoot = sqrt(firstnumber);
    printf("The final number is: %d", squareRoot);
    return 0;
}
```

Break Time

CS Paint Hall of Fame

- If you're a fan of Challenge Exercises that award bonus Marcs (not in any way related to course marks)
- We have some advanced challenges that are not necessarily related to programming, but are related to CS Paint
- Check out the spec updates on the course website to see more info

What does it mean to be a programmer?

Marc's four pillars of being a professional:

1. Communication
2. Teamwork
3. Resilience
4. Technical Skills

Communication

Making sure everyone understands what you're doing

- Problem solving in teams involves shared understanding
- In order to solve human problems, we must understand what people need and how we can help them
- The more we communicate with computers the more risk we have of treating people like machines
- The ability to explain our code is important to keep us on track
- It's especially important to be able to explain your code to non-programmers

Teamwork

Code is very rarely created alone

- Teamwork involves sharing and compromise
- Can you work with other people's ideas?
- Can you follow someone else's style and structure?
- Can you adapt your structure so that other people can use it?
- Can you provide support to your teammates?
- Teams made of people who get along are usually more successful than teams made of very skilled individuals!

Resilience

Work is hard.

- We need to look after ourselves
- If a job is so hard you can only survive it for a year, it’s not a good job
- We will sometimes be stuck in “impossible” situations
- Can you deliver your best work, even while knowing that it is not enough?
- Failure is inevitable, what counts is how you recover, not whether you fail

Technical Skills

How's your programming?

- Yes, this comes last in the list
- It's considered the easiest of the four to learn
- Still, we have the majority of COMP1511 to learn technical programming

More about Resilience and Surviving

You have an assignment due soon

- Success isn't about getting everything done
- It's about prioritising your effort so you don't have to do as much work!

Priorities:

- What gets you the most marks with the least amount of time?
- Code Style?
- Legal Play?

Don’t Panic!

Surviving is about acting rationally in panicky situations

- Take a moment to assess where you’re up to
- Figure out what your options are
- Break everything down into small bits
- Complete small pieces one at a time
- Aim for whatever gets you the highest marks

In Practical Terms

Get the most marks from your time

- Clear style, following the style guide
- Aim for basic functionality
  - Stages in order, one at a time
  - Test things with a lot of different inputs
  - Remember, completely working 60% is better than aiming for 100% and not being able to complete any parts
- Know how the marking and late penalties work

What did we learn today?

Assignment 1

- A recap and assessment

Functions and Libraries

- Accessing C libraries and their functions

Professionalism

- It's not all about coding skills and C
- Communication and working with people is very important