COMP1511 PROGRAMMING FUNDAMENTALS

LECTURE 9

2D Arrays
Command Line Arguments
Multi File Projects (if time, but I am really cooking here)

LAST WEEK...

- 2D Arrays
- Strings
- Command Line Arguments

TODAY...

- Hope you have had a great weekend and have gotten started on your assignment (maybe?)
- Recap 2D arrays
- A bigger 2D array problem (like the assignment!) - I am desperate for churros!
- Command Line Arguments

WHERE IS THE CODE?

Live lecture code can be found here:

HTTPS://CGI.CSE.UNSW.EDU.AU/~CS1511/26T1/CODE/WEEK_4/

PROBLEM TIME

SIMILAR TO YOUR ASSN 1

This lecture runs through lunchtime, I am hungry (always) and I am craving churros! I will need to navigate a grid-based map of the university with a mission to collect every churro hidden on the map.

1. Setup Phase
   - The map is an 8×8 grid.
   - The user first inputs the starting coordinates of the player.
   - The user then inputs the number of walls, followed by their coordinates.
   - The user then inputs the number of churros, followed by their coordinates.

PROBLEM TIME

SIMILAR TO YOUR ASSN 1

2) Gameplay Phase

- The map is printed after every move.
- The player uses w, a, s, and d to move.
- Collision: If a player tries to move into a wall (#) or out of bounds, the move is rejected.
- Collection: If a player moves onto a churro (C), that tile becomes EMPTY and the churro is collected.
- Winning: The game ends when all churros are collected.
- Quitting: The user can press CTRL+D at any time to end the search early (after all, you still need to get through your assignment without a sugar crash!)

PROBLEM TIME

SIMILAR TO YOUR ASSN 1

3) Side quest - actually a sugar rush is a great idea!

If a player steps on a TRAP tile, we calculate their new position by ”flipping” their coordinates.

- If they are at (row, col), we send them to (MAP_ROWS - 1 - row, MAP_COLUMNS - 1 - col).

PROBLEM TIME

SIMILAR TO YOUR ASSN 1

You are going to get some starter code:

1) initialise_map function
2) print_map function

```
void initialise_map(struct location map[MAP_ROWS][MAP_COLUMNS]);
void print_map(struct location map[MAP_ROWS][MAP_COLUMNS]);
```

PROBLEM TIME

SIMILAR TO YOUR ASSN 1

Your enum in this problem:

```
enum tile {
    EMPTY,
    WALL,
    CHURRO,
    TRAP,
    PLAYER
};
```

PROBLEM TIME

SIMILAR TO YOUR ASSN 1

And then a struct location, made up of a tile and potential to add more things as you go...

```
struct location {
    enum tile tile;
};
```

PROBLEM TIME

SIMILAR TO YOUR ASSN 1

map[4][0].tile

So that means, your map is an array of structs, with a tile entity at each grid point:

```
struct location map[MAP_ROWS][MAP_COLUMNS];
```

[0][0]	[0][1]	[0][2]	[0][3]	[0][4]	[0][5]	[0][6]	[0][7]
[1][0]	[1][1]	[1][2]	[1][3]	[1][4]	[1][5]	[1][6]	[1][7]
[2][0]	[2][1]	[2][2]	[2][3]	[2][4]	[2][5]	[2][6]	[2][7]
[3][0]	[3][1]	[3][2]	[3][3]	[3][4]	[3][5]	[3][6]	[3][7]
[4][0]	[4][1]	[4][2]	[4][3]	[4][4]	[4][5]	[4][6]	[4][7]
[5][0]	[5][1]	[5][2]	[5][3]	[5][4]	[5][5]	[5][6]	[5][7]
[6][0]	[6][1]	[6][2]	[6][3]	[6][4]	[6][5]	[6][6]	[6][7]
[7][0]	[7][1]	[7][2]	[7][3]	[7][4]	[7][5]	[7][6]	[7][7]

PROBLEM TIME

SIMILAR TO YOUR ASSN 1

So, each one (for example the cell at row 4 and col 0 initialised with empty entity and clean space):

```
struct location map[MAP_ROWS][MAP_COLUMNS];
```

```
struct location {
    enum tile tile;
};
```

```
enum tile {
    EMPTY,
    WALL,
    CHURRO,
    TRAP,
    PLAYER
};
```

map[4][0].tile == EMPTY

PROBLEM TIME

SIMILAR TO YOUR ASSN 1

So it looks something like this once initialised:

```
Welcome to the Churro Quest!
Enter player starting row and col: 0 0
How many walls? 2
Wall 0 (r c): 1 1
Wall 1 (r c): 2 9
Invalid spot! Try again: 2 7
How many churros? 2
Churro 0 (r c): 4 4
Churro 1 (r c): 5 5

--- Let the Search for Churros Begin! ---
P  .  .  .  .  .  .  .
.  #  .  .  .  .  .  .
.  .  .  .  .  .  .  #
.  .  . (!) .  .  .  .
.  .  .  .  C  .  .  .
.  .  .  .  .  C  .  .
.  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .
Move (wasd):
```

map[1][1].tile == WALL
map[2][7].tile == WALL

map[0][0].tile == PLAYER

map[3][3].tile == TRAP

map[4][4].tile == CHURRO
map[5][5].tile == CHURRO

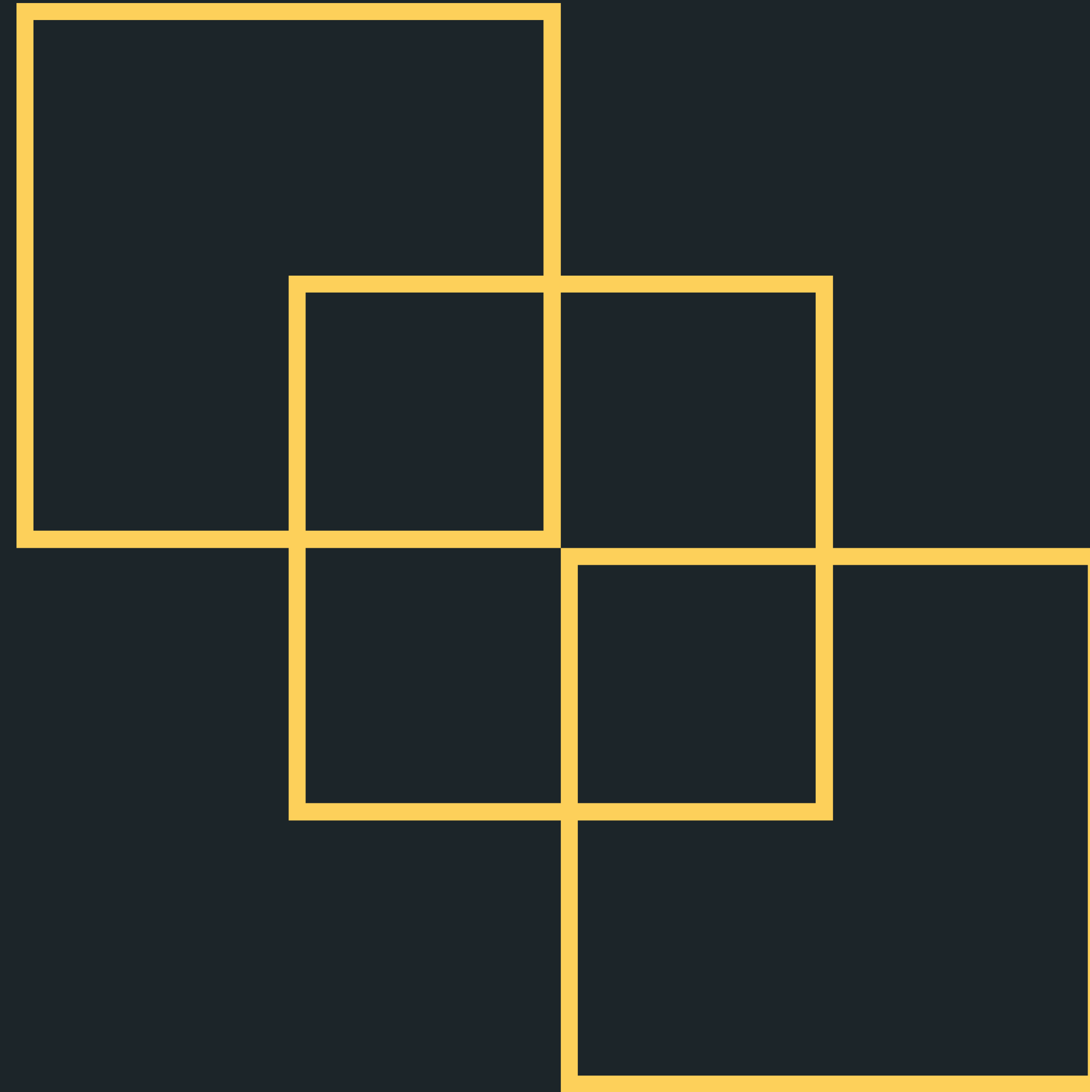
ASSN1

STYLE TIPS

Follow the style guide, but some simple things to watch out for:

- Functions
- #defines for magic numbers
- comments
- line length

BREAK TIME



Can you reproduce this figure using just one line, without lifting the pen and without going back over an already drawn line?

COMMAND LINE ARGUMENTS

WHAT ARE THEY?

- So far, we have only given input to our program after we have started running that program (using scanf())
- This means our `int main(void) {}` function has always been void as input
- Command line arguments allow us to give inputs to our program at the time that we start running it! So for example:

```
avas605@vx5:~$ dcc test6.c -o test6
avas605@vx5:~$ ./test6 argument2 argument3 argument4
```

TIME TO CHANGE THAT VOID

LET'S GET OUR MAIN FUNCTION TO ACCEPT SOME INPUT PARAMETERS

- In order to change your main function to accept command line arguments on first running, you need to change the void input:

```
int main(int argc, char *argv[]) {}
```

- int argc = is a counter for how many command line arguments you have (including the program name)
- char *argv[] = is an array of the different command line arguments (separated by a spaces). Each command line argument is a string (an array of char)

AN EXAMPLE

```c
1 #include <stdio.h>
2
3 int main (int argc, char *argv[]) {
4     printf("There are %d command line arguments in this program\n", argc);
5
6     //argv[0] is always the program name
7     printf("The program name is %s (argv[0])\n", argv[0]);
8
9     // What about the other command line arguments? Let's loop through
10    // the array and print them all out!
11    for (int i = 0; i < argc; i++) {
12        printf("The command line argument at index %d"
13                "argv[%d] is %s\n", i, i, argv[i]);
14    }
15
16    return 0;
17 }
```

```
avas605@vx02:~$ dcc argv_demo.c -o argv_demo
avas605@vx02:~$ ./argv_demo We are almost half way through this term!
There are 9 command line arguments in this program
The program name is ./argv_demo (argv[0])
The command line argument at index 0argv[0] is ./argv_demo
The command line argument at index 1argv[1] is We
The command line argument at index 2argv[2] is are
The command line argument at index 3argv[3] is almost
The command line argument at index 4argv[4] is half
The command line argument at index 5argv[5] is way
The command line argument at index 6argv[6] is through
The command line argument at index 7argv[7] is this
The command line argument at index 8argv[8] is term!
```

WHAT IF YOU WANT NUMBERS AND NOT STRINGS?

REMEMBER THAT EACH COMMAND LINE ARGUMENT IS A STRING

- You want numbers, if you want to use your command line arguments to perform calculations
- There is a useful function that converts your strings to numbers:

  `atoi()` in the standard library: <stdlib.h>

WHAT IF YOU WANT NUMBERS AND NOT STRINGS?

REMEMBER THAT EACH COMMAND LINE ARGUMENT IS A STRING

```c
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[]) {
    // Remember that the command line arguments are all strings, so if you
    // need to do mathematical operations, you will need to convert them
    // to numbers
    // You can do this with a really handy function atoi() in the stdlib.h library!

    // Let's print out all the command line arguments given and then add
    // them together to give the sum of the command line arguments

    int sum = 0;
    for (int i = 1; i < argc; i++) {
        printf("The command line argument at index %d (argv[%d]) is %d\n",
                i, i, atoi(argv[i]));
        sum = sum + atoi(argv[i]);
    }
    printf("The sum of the arguments is %d\n", sum);

    return 0;
}
```

```
avas605@vx02:~$ dcc atoi_demo.c -o atoi_demo
avas605@vx02:~$ ./atoi_demo 3 4 5 6 7
The command line argument at index 1 (argv[1]) is 3
The command line argument at index 2 (argv[2]) is 4
The command line argument at index 3 (argv[3]) is 5
The command line argument at index 4 (argv[4]) is 6
The command line argument at index 5 (argv[5]) is 7
The sum of the arguments is 25
```

Feedback please!

I value your feedback and use to pace the lectures and improve your overall learning experience. If you have any feedback from today's lecture, please follow the link below. Please remember to keep your feedback constructive, so I can action it and improve the learning experience.

https://forms.office.com/r/Z8WRbaF5ug

WHAT DID WE LEARN TODAY?

COMMAND LINE ARGUMENTS

2D_array.c

REHASH 2D ARRAYS

2D_array.c

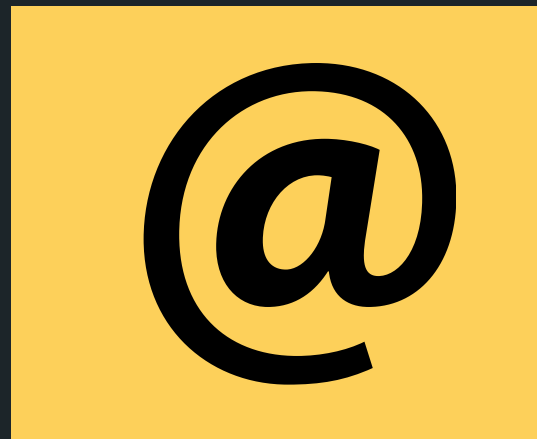
LECTURE PROGRAM

cs_churros.c

REACH OUT

CONTENT RELATED QUESTIONS

Check out the forum



ADMIN QUESTIONS

cs1511@unsw.edu.au