COMP1511/1911 Programming Fundamentals

Week 5
Lecture 1

Assignment 1

- It’s out
  - Did you do it over the weekend?
- Please start earlier
  - Utilise help sessions
    - Everyone has the exact same plan of using the help sessions at the last minute to finish their assignment before the deadline

Previously On...

- 2D arrays
  - Powerful abstraction tool
  - Essential for Assignment 1
- Command line arguments
  - Get input from the user before the program runs

Today

- Bigger 2D array problem
  - Like the assignment
    - Not actually the assignment
  - Honestly this may take the whole lecture :)
- Command line arguments

Where is the Code?

- Lecture code is updated every hour on the hour
- Live lecture code can be found here
  - https://cgi.cse.unsw.edu.au/~cs1511/26T2/

Problem Time (Similar to Assignment 1)

- This lecture requires caffeine to get through, and the only way I can get it is through Diet Coke. I will need to navigate a grid-based map of the university to find every Diet Coke hidden on the map.

Setup Phase

- The map is an 8x8 grid
- The user first inputs starting coordinates of the player
- The user then inputs the number of walls, followed by their coordinates
- The user then inputs the number of Diet Cokes, followed by their coordinates

Gameplay Phase

- The map is printed after every move
- The player uses w, a, s, and d to move
- Collision: If the player tries to move into a wall (#), or out of bounds, the move is rejected
- Collection: If the player moves onto a Diet Coke (D), that tile becomes EMPTY and the Diet Coke is collected
- Winning: The game ends when all Diet Cokes are collected
- Quitting: The user can press CTRL + D at any time to end the game, we don’t want a caffeine rush when doing the assignment!

Side Quest

- Actually a caffeine rush sounds cool as heck
- If a player steps on a TRAP tile, we calculate their new position by “flipping” their coordinates
  - If they are at (row, column), we send them to MAP_ROWS - 1 - row, MAP_COLUMNS - 1 - col)

Starter Code

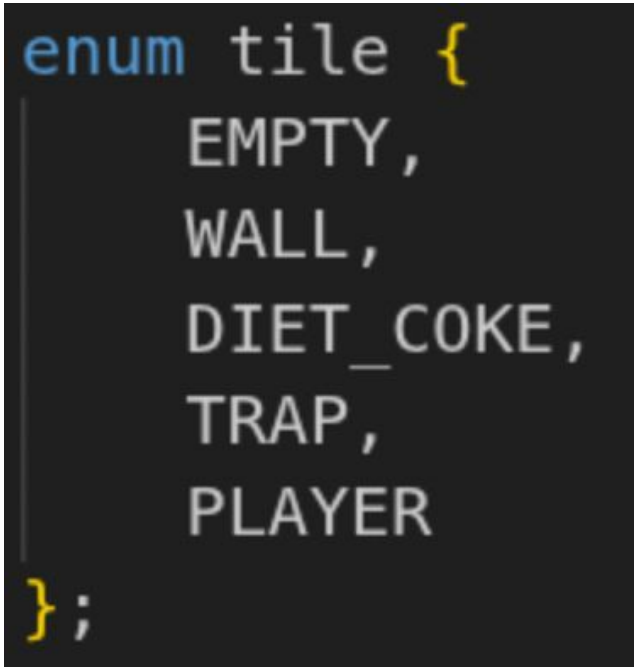
- This problem comes with some starter code
  - Similar to your assignment

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]);
```

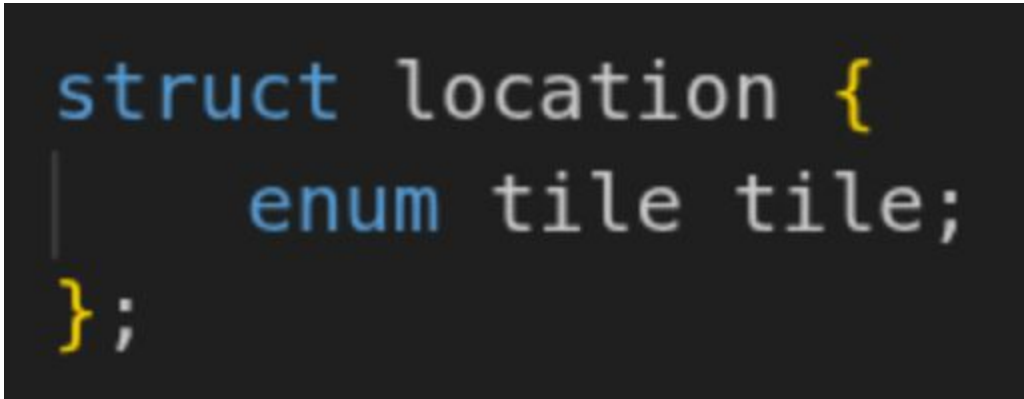
Your Enum

- You will have an enum for this problem
  - Similar to your assignment



Your Struct

- You will have a struct location made up of tiles
  - SIMILAR TO YOUR ASSIGNMENT



Understanding the Map

- So the map is an array of arrays of structs, with a tile entity at each grid point

[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]	[0][7]

map[4][0].tile

Understanding the Map

- So each cell is initialised with empty entity and clean space

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

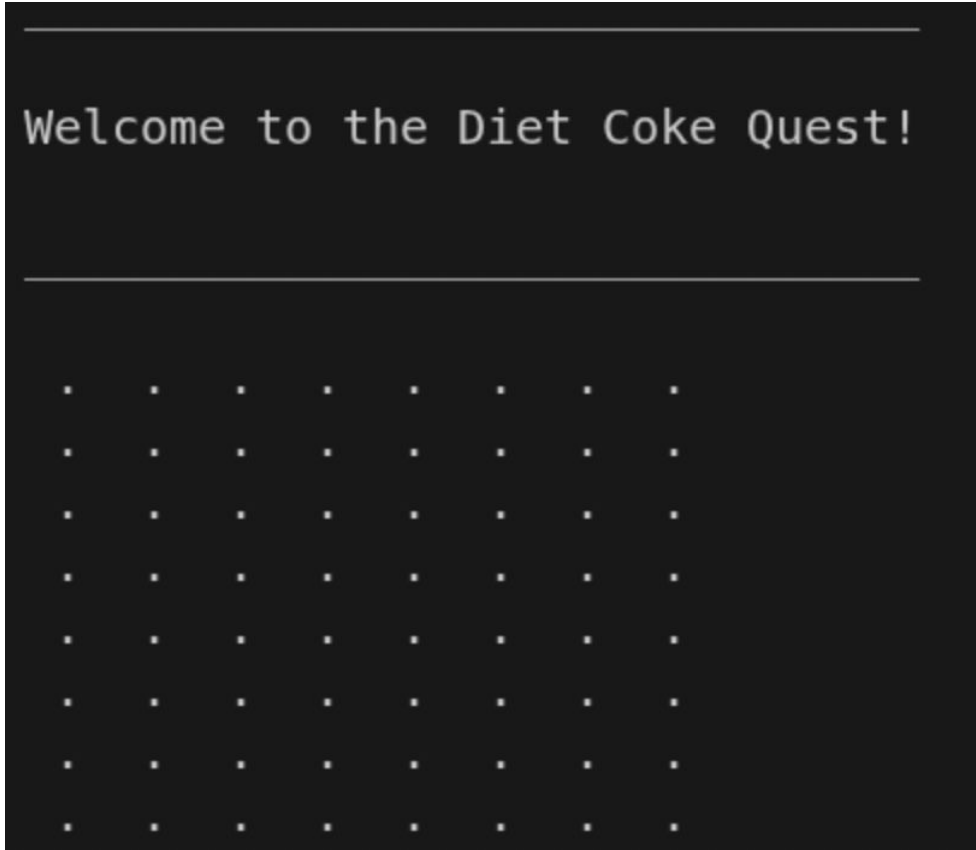
```
enum tile {
    EMPTY,
    WALL,
    DIET_COKE,
    TRAP,
    PLAYER
};
```

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

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

The Map When Initialised

- Looking a little empty…
  - Let’s start solving!



Assignment 1 Style Tips

- Follow the style guide, but some simple things to watch out for:
  - Functions
  - #defines for magic numbers
  - Comments
  - line length

Break Time!

...probably

Command Line Arguments

- So far we have only given input to our program after we run it
  - Therefore our main has taken no inputs
    - int main(void)
- Command line arguments allow us to give inputs to our program at the time that we start running it!

Command Line Arguments

- For example:

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("The command line arguemnt at index 0 (argv[0] is %s\n", argv[0]);
    printf("The command line arguemnt at index 1 (argv[1] is %s\n", argv[1]);
    printf("The command line arguemnt at index 2 (argv[2] is %s\n", argv[2]);
    printf("The command line arguemnt at index 3 (argv[3] is %s\n", argv[3]);
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

z3548950@vx16:~/public_html/COMP151126T2/code/week4$ ./prog survivor is cool
The command line arguemnt at index 0 (argv[0] is ./prog
The command line arguemnt at index 1 (argv[1] is survivor
The command line arguemnt at index 2 (argv[2] is is
The command line arguemnt at index 3 (argv[3] is cool
```

Breaking It Down

- `int argc`
  - Counter for how many command line arguments you have
    - Including your program name
- `char *argv[]`
  - Array of the different command line arguments (separated by a spaces)
    - Each command line argument is a string (an array of char)

Numbers Instead of Strings

- If you want numbers, you'll need to convert strings to integers
  - atoi() is a useful stdlib function to do this
    - Remember to #include <stdlib.h>!

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

int main(int argc, char *argv[]) {
    int num;
    num = atoi(argv[1]);
    printf("%d\n", num);
    return 0;
}
```

Numbers instead of Strings

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

int main(int argc, char *argv[]) {
    int sum = 0;
    for (int i = 1; i < argc; i++) {
        printf("The command line arguemnt at argv[%d] is %d.\n", i, atoi(argv[i]));
        sum += atoi(argv[i]);
    }
    printf("The sum is %d.\n", sum);
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
z3548950@vx16:~/public_html/COMP151126T2/code/week4$ ./prog 3 67 58
The command line arguemnt at argv[1] is 3.
The command line arguemnt at argv[2] is 67.
The command line arguemnt at argv[3] is 58.
The sum is 128.
```

What We Learnt

- Recap 2D arrays
  - Using them practically
- Recap strings
  - fgets
  - fputs
- Command line arguments
  - Getting input from the user

Reach Out

- Check the course forum for questions!
  - Post there too!
- Admin questions
  - cs1511@unsw.edu.au

Thank You

Questions?