

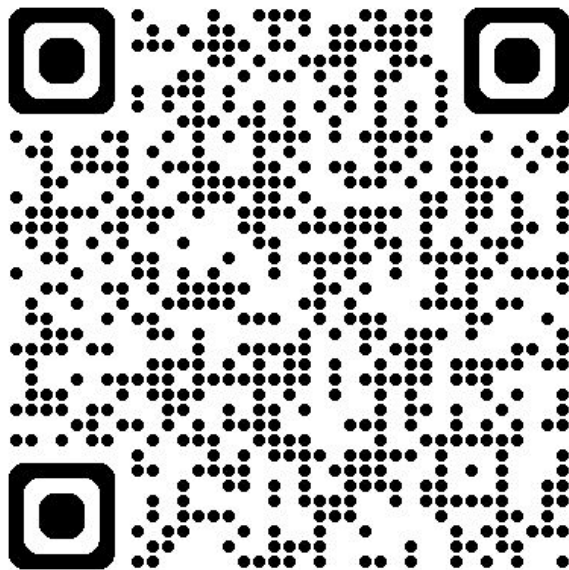
COMP1511/1911 Programming Fundamentals

Week 5 Lecture 1

Lecture Program 2D Arrays of structs Command Line Arguments

Link to Week 5 Live Lecture Code

https://cgi.cse.unsw.edu.au/~cs1511/26T3/code/week_5/



Last Week

- 2D Arrays
- Strings (pre-recorded video)
- Arrays of strings (pre-recorded video)

Today's Lecture

- Revision: A bigger 2D array of structs with enums program!
 - mud_and_bones.c
 - Putting together concepts needed in assn1
 - Style tips for assn 1
- Strings and Array of strings recap Kahoot
- Strings coding exercise
- Command line args

Array Lecture Program

All the key concepts in Practice

Mud and Bones



We have the following “game”.

- A dog (the player) is moving around on a map
- The locations on the map contain either grass or mud.
- They may also contain a bone.
- The dog can move around the map to collect bones
- If he steps in mud he will spread that mud to the next location he goes to.
- If the dog steps in mud too many times he has to go home for a bath

Mud and Bones



The **game ends** when:

The dog finds all the bones!



The dog steps in mud 3 times and has to go home for a bath



The player presses Ctrl^D!

Mud and Bones

Important types and constants given to you for this code

```
#define MAP_ROWS 8
#define MAP_COLUMNS 8
enum ground_type {
    GRASS,
    MUD
};
```

```
enum item_type {
    EMPTY,
    BONE
};

struct location {
    enum item_type item;
    enum ground_type ground;
};
```

The Map: 8x8 2D array of struct location

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

	Col 0	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7
Row 0								
Row 1								
Row 2								
Row 3								
Row 4								
Row 5								
Row 6								
Row 7								

The Map: 8x8 2D array of struct location

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

	Col 0	Col 1	Col 2
Row 0	EMPTY GRASS	BONE MUD	EMPTY GRASS
Row 1	EMPTY MUD	BONE GRASS	EMPTY MUD
Row 2	EMPTY GRASS	EMPTY GRASS	BONE GRASS
Row 3	EMPTY GRASS	EMPTY GRASS	EMPTY GRASS

If we zoom into a section of the map, we can see each one is a struct location with an item type and a ground type

The Map: 8x8 2D array of struct location

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

	Col 0	Col 1	Col 2
Row 0	EMPTY GRASS	BONE MUD	EMPTY GRASS
Row 1	EMPTY GRASS	BONE GRASS	EMPTY GRASS
Row 2	EMPTY GRASS	EMPTY GRASS	EMPTY GRASS

In this example

map[0][1].item
has the value BONE

map[0][1].ground has
the value MUD

Mud and Bones

Provided Function Prototypes

```
void initialise_map(struct location map[MAP_ROWS][MAP_COLUMNS]);  
void print_map(  
    struct location map[MAP_ROWS][MAP_COLUMNS],  
    int dog_row,  
    int dog_col,  
    int bone_count,  
    int mud_count  
);
```

Mud and Bones Starter Code

- Creates a map variable
- Calls initialise on the map
- Calls print_board, passing in INVALID_INDEX for dog_row and dog_col and 0 for bone_count (0) and mud_count(0)

Mud and Bones Stage 1

- Initialise dog starting position
 - Scan in co-ordinates from the user and set the dog's starting position.
 - If illegal, set to (0, 0)
 - Assume that user will at least type in valid integers
 - Update the board and mud and bone counts accordingly
 - Increment the bone count and remove bones from the map once found
 - Update the mud_count.
 - Print the board!

Mud and Bones Stage 2

In a loop that ends with Ctrl-D (no winning yet)

- Allow the user to enter 'w' 'a' 's' 'd' to move the dog around the map.
- Increment the bone count and remove bones from the map once found.
- Update the mud count if the dog steps in mud. (Don't worry about spreading mud to other locations yet).
- Print the map after each move
- Note: we are assuming the user types in valid input. If they don't then the game may crash!!!

Mud and Bones Stage 3

- Implement
 - Winning the game by finding all the bones
 - Losing the game by stepping in mud 3 times
- Implement mud spreading

Assignment 1 Style Tips

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

- Functions
- `#defines` constants for magic numbers including 'w' etc
- Comments
- line length

Get feedback from

- Style checker
- Checking the style guide
- Asking your tutor or a help session tutor to give feedback

Quick Break. Back Soon...



Strings Revision

- Strings Kahoot
 - Recap string basics
- `word_guess.c`
 - Put it all together into some code

What are Command Line Arguments?

Command Line Arguments

- So far, we have only given input to our program after we have started running that program (using `scanf()` or `fgets()`)
- Our main function prototype has always been `int main(void) ;`
- Command line arguments allow us to give inputs to our program at the time that we start running it! E.g.

```
$ gcc prog.c -o prog  
$ ./prog argument1 argument2 argument3 argument4  
$ ./prog 123 hello
```

Command Line Arguments

- To use command line arguments you need to change your main function prototype to

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

- **argc**
 - a counter for how many command line arguments you have (including the program name)
- **char *argv[]**
 - an array of the different command line arguments
 - each command line argument is a string (an array of char)

Command Line Arguments

- If we ran our program as follows:

```
$ ./prog 123 dog "hello world"
```

- argc would be equal to 4
- argv would be an array of strings we can visualise as follows:

“./prog”	“123”	“dog”	“hello world”
0	1	2	3

Command Line Arguments

```
int main(int argc, char *argv[]) {  
    printf("There are %d command line arguments\n", argc);  
  
    // argv[0] is always the program name  
    printf("This program name is %s\n", argv[0]);  
  
    // print out all arguments in the argv array  
    for (int i = 0; i < argc; i++) {  
        printf("Argument at index %d is %s\n", i, argv[i]);  
    }  
    return 0;  
}
```

Command Line Arguments

```
$ gcc -o command_line_args command_line_args.c
$ ./command_line_args 123 dog "Hello World" COMP1511
This program has 5 command line arguments
This program name is ./command_line_args
Argument at index 0 is ./command_line_args
Argument at index 1 is 123
Argument at index 2 is dog
Argument at index 3 is Hello World
Argument at index 4 is COMP1511
```

Converting Strings to Integers: atoi

- You may want to use your command line arguments to perform calculations, but they are strings!
- There is a function that converts strings to integers:
 - `atoi()` in the standard library: `<stdlib.h>`
 - E.g. `int x = atoi("952")`
 - Would give us a value of 952 stored in x

Converting Strings to Integers: atoi

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

Command Line Arguments

- `command_line_args.c`
- `atoi_demo.c`

What did we learn today?

- 2D array of structs with enums coding example
 - `mud_and_bones.c`
- String and array of strings recap Kahoot
- Strings exercise `word_guess.c`
- Command Line Arguments
 - `command_line_args.c` `atoi_demo.c`

What will we learn tomorrow?

- Pointers!!!
 - Learn how to ask variables for their addresses!!
 - Learn how to allow functions access to modify your int, double, char, enum, struct variables!

Feedback Please!

Your feedback is valuable!

If you have any feedback from today's lecture, please follow the link below or use the QR Code.

Please remember to keep your feedback constructive, so I can action it and improve your learning experience.



<https://forms.office.com/r/QD15dbp6KD>

Reach Out

Content Related Questions:

[Course Forum](#)

Admin related Questions email:

cs1511@unsw.edu.au



Student Support - I Need Help With...

My Feelings and Mental Health
Feeling depressed, overwhelmed
or not your usual self?



Mental Health Connect
student.unsw.edu.au/mhc



TalkCampus
student.unsw.edu.au/Talkcampus



UNSW (24/7) Mental Health Support: [+61\(2\) 9385 5418](tel:+61293855418)
Text support After Hours: [0485 826 595](tel:0485826595)



Offshore
Call 24-hour Medibank Hotline: [+61 2 8905 0307](tel:+61289050307)

Uni and Life in Australia
Stress, financial, visas,
Accommodation & More



Student Support
Indigenous Student Support

student.unsw.edu.au/advisors
nura-gili-centre-indigenous-programs

Reporting Sexual Assault/Harassment



Equity Diversity & Inclusion (EDI)

edi.unsw.edu.au/sexual-misconduct

Educational Adjustments
To manage my studies and
disability / health condition



Equitable Learning Services (ELS)

student.unsw.edu.au/els

Academic and Study Skills



Academic Skills

student.unsw.edu.au/skills

Special Consideration
Because life impacts our
studies and exams



Special Consideration

student.unsw.edu.au/special-consideration

Physical Health
Doctors visits, Vaccinations



Health Service

student.unsw.edu.au/hsu