



Lecture 9

A bit more about debugging your code,
Some awesome functions for characters,
And starting to look at strings



LAST WEEK...

- Talked a bit more about libraries
- Learnt about 1D arrays
- Looked at 2D arrays (which make up a grid and allow us to do some pretty cool stuff)
- Got introduced to pointers



TODAY...

- Revisit pointers, by solving a problem with pointers
- Look at debugging code including compile time errors and run time errors
- Learn two new functions available to us: `getchar()` and `putchar()`

WHERE IS THE CODE?

**LIVE LECTURE CODE CAN BE
FOUND HERE:**

<https://cgi.cse.unsw.edu.au/~cs1511/21T3/live/Week05/>

PROBLEM TIME

ARRAYS AND POINTERS AND FUNCTIONS - LET'S BRING IT ALL TOGETHER...



Let's see a good use of pointers. Now remember that you can only return one thing back to main and you can't return an array*

The problem is this:

Read in an array of numbers (user will specify how many numbers they plan to read in). Then the first number and the last number in the array will be swapped, and the modified array printed out again.

*So without using pointers, can you have a swapping function that swaps out two things? How would you return both of those things back to the main?

DEBUGGING CODE

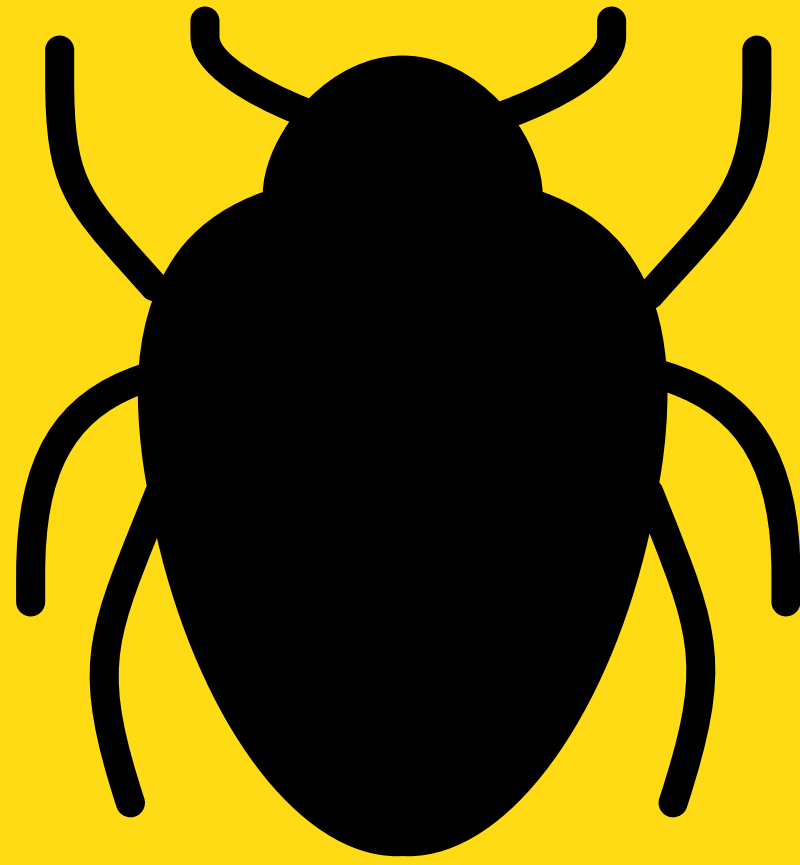
WHAT DOES IT MEAN?



- Debugging – you have probably already spent quite a bit of time writing code, and even more time troubleshooting what you have written. Welcome to the world of debugging!
- Debugging is the process in which we find bugs that cause our code not to work as specified, and remove those bugs

WHAT IS A BUG

CODE ERRORS ARE CALLED BUGS



- Any error that stops your code from working as specified, is referred to as a bug.
- Bugs can occur at:
 - Compile time (syntax issues – easier to fix!)
 - Run time (logic issues – much harder to fix!)

COMPILE TIME ISSUES

BUGS WE GET TOLD ABOUT AT COMPILE TIME

- Usually these types of bugs are syntax associated
- Sometimes the error message is a consequence of the bug itself
- dcc is pretty good at telling you what has gone wrong, and what needs changing



RUN-TIME ISSUES

LOGIC BUGS

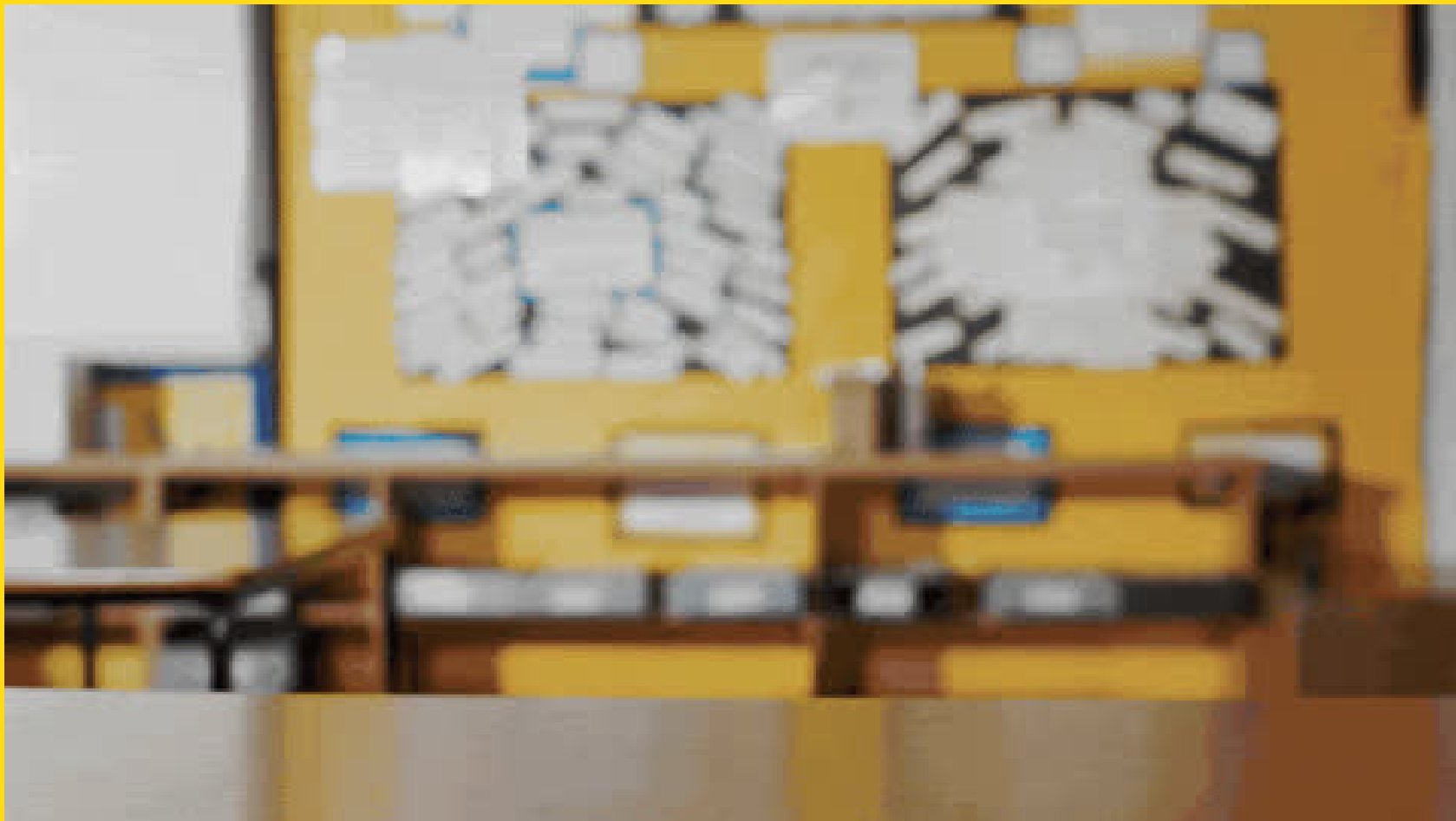
- These are harder to find, because the program might compile successfully but not to what we want it to!
- Can happen if:
 - we have misunderstood the spec
 - Used wrong indexing, wrong comparisons (wrong solution)
- One strategy is to use:
 - external tracing to trace the outputs of your program
 - use printf statements in random places in your code, to see values that your variables take



EXTERNAL TRACING

HAND EXECUTION OF CODE

- External tracing of code is executing a program in one's head or by hand.
- Let's see an example of tracing:
`trace.c`



EXTERNAL TRACING

WEEK04/HIGHEST_NUMBER.C

```
12 #include <stdio.h>
13
14 int highest(int numbers[5]);
15
16 int main (void) {
17     //1. Take numbers in from the user - scanf
18     int numbers[5] = {0}; // {0,0,0,0,0,0,0,0,0,0}
19
20     int i = 0;
21     while (i < 5) {
22         printf("Please enter a number: ");
23         scanf("%d", &numbers[i]);
24         i++;
25     }
26
27     //int highest_number = highest(numbers);
28
29     // Find the highest number
30     printf("The highest number is: %d\n", highest(numbers));
31
32     return 0;
33 }
34
35 // Function finds the highest number in an arrays
36 // Input to function: number
37 // Output to function: int
38 int highest(int numbers[5]) {
39     int highest_number = -1;
40
41     int i = 0;
42     while (i < 5) {
43         if (numbers[i] > highest_number) {
44             highest_number = numbers[i];
45         }
46         i++;
47     }
48
49     return highest_number;
50 }
51
52 }
```

0xF244

0xF240

0xF23C

0xF238

0xF234

0xF230

0xF22C

0xF228

0xF224

0xF220

BREAK TIME (5 MINUTES)

What's the next number in the following sequence?

1

11

21

1211

111221

312211

LET'S DO A QUICK CODE DEMO OF IT

DEBUG! DEBUG! DEBUG!

debug.c

```
011110110110111110111000001111000001000001111100101010000000110000011101010101
1110101110100111010000111111011011101101100111101101110000111101010101000001111
01110111100110111000101111100000101110110101011110000001001011010010100111110101
1111011111111101100100101011001010101001111010101001000000010101000111111111010
1100110101001110101011010011010010100101110011001100001011110101000101010100101
0111011001011011000000110101111010000100010011011111111101110000111101000011110
11111101001100011001110001000110111000010001111011000000001101101001000111000001
1100110110101111110100100100100101110000110011011100110011000111001100010100110
00101010101000000100110101000100010101101011000000101001010001100011001010010001
001001101011011001010101100110101000110111000000001101001001000100000011110011011
01010101100011011011101000101001111110001000000100010010110000101110001110001100
11110010101001011011110011000011110000000110010100111110101100101100001000001110
10001101010100000000001111110010001000000111101000111101000010100100111100011001
11100101111101110011001100001010110001010110001110101110000111001100110101000011
01000000101101101111100000000011111100011101001001111101110110011110000010101001
10010100101011111100011100101101001100000111011001000111000010111011111010110101
1010000001111010110110011000001111100001010000110000111100011000000000010000101
110100100011011001000111001101110100100000101111110010010011111011110111101100000
0111011100001110100000000100100000100000100010011111110111111011101011001000010
1100011000101110100001100111101100010110101000000101011011010011101001000100100
11011001010000101111001000111001011110000010101101101001000110000010101010110010
01000010011010111001011111100110110110011011010011111111000100100101000000011000
00110100100110111011111010000001101011100101110101010100011010100010011001001101
1110011100010100000001010011111101100101010001001100001101011110101110100000110111
11011111111010000100010001101011100001110001110101100110101010111001101110100111
01111001101111011100100001100111111010110011000010110111100000011100000010000000
11110111010011101000010111010111011011101000010000101001110011110111100100010011
11100011010000110001011010111101011001110111100010011010000100001100100010110110
01110101011110111011111111010
```

HELPFUL LIBRARY FUNCTIONS FOR CHARS

GETCHAR()

- `getchar()` is a function that reads a character from input (a single character)
 - Reads one byte of input
 - Usually returns an int (ASCII code of that character that it read)
 - Can return -1 (EOF), which is useful for knowing when to finish input
 - will not get its input until enter is pressed at the end of the line (it keeps filling up a buffer until enter is pressed)

HELPFUL LIBRARY FUNCTIONS FOR CHARS

PUTCHAR()

- putchar() is a function that prints out one character to standard output
- Similar to printf("%c", character);

```
1 #include <stdio.h>
2
3 int main (void) {
4
5     //Declare a variable int called character
6     int character;
7     //Use the getchar() function to read one character at a time
8     //Remember that this function will take char when a new line is entered
9     character = getchar();
10
11     //When you press Ctrl+D to signal EOF (end of file) - the while loop will
12     //be exited
13     while (character != EOF) {
14         printf("You entered the character: ");
15         //Using the function putchar to show output one character at a time
16         putchar(character);
17         printf("\n");
18         //Get the next character from the buffer
19         character = getchar();
20     }
21     return 0;
22 }
```

WHY USE GETCHAR() OVER SCANF()

- scanf() is a formatted way of reading input from terminal, whereas getchar() reads a single character at a time
- scanf() reads a character according to the format specified (%d, %lf, %c), whereas getchar() just reads a single character at a time
- scanf() takes in the format and variable address, whereas getchar() does not take any input.
- So scanf() can do many things and is easy to make mistakes with, if you need one character at a time, it is better to use getchar()



WHY USE PUTCHAR() OVER PRINTF()

- printf() is a formatted way of outputting to terminal, whereas putchar() outputs a single character at a time



SOME OTHER INTERESTING CHARACTER FUNCTIONS

<CTYPE.H> STANDARD LIBRARY

CHECK OUT THE REST OF THE FUNCTIONS:
[HTTPS://WWW.TUTORIALSPPOINT.COM/C_STANDARD_LIBRARY/CTYPE_H.HTM](https://www.tutorialspoint.com/c_standard_library/ctype_h.htm)

Some other useful functions for characters:

- `isalpha()` – will determine if the character is a letter
- `isdigit()` – will determine if the character is a number
- `islower()` – will determine if the character is a lower case letter
- `isupper()` – will determine if the character is an upper case letter
- `tolower()` – will convert the character to a lower case letter
- `toupper()` – will convert the character to an upper case letter

DEMO TIME

LET'S USE SOME OF OUR HELPFUL NEW FUNCTIONS



demo_functions.c

```
1 #include <stdio.h>
2 #include <ctype.h>
3
4 int main (void) {
5
6     //Declare a variable int called character
7     int character;
8
9     printf("Enter your name as an example of getchar() and press Enter: ");
10    //Use the getchar() function to read one character at a time
11    //Remember that this function will take char when a new line is entered
12    character = getchar();
13
14    //When you press Ctrl+D to signal EOF (end of file) - the while loop will
15    //be exited
16    while (character != EOF) {
17        printf("You entered the character: ");
18        //Using the function putchar to show output one character at a time
19        putchar(character);
20        printf("\n");
21        //Check if the character is a lower case letter by using the function
22        //islower() found in <ctype.h> standard library
23        if (islower(character)){
24            //If it is, then convert it to upper case letter by using the
25            //function toupper() found in <ctype.h> standard library
26            character = toupper(character);
27            printf("Your new character is: ");
28            putchar(character);
29            printf("\n");
30        }
31
32        //Get the next character from the buffer
33        character = getchar();
34    }
35    return 0;
36 }
```

FEEDBACK?

**PLEASE LET ME KNOW ANY
FEEDBACK FROM TODAY'S
LECTURE!**

www.menti.com

Code: 8444 4604



WHAT DID WE LEARN TODAY?



POINTERS

the_shuffle.c

DEBUGGING

debug.c

CHARACTER FUNCTIONS

getchar_demo.c

OTHER CHARACTER FUNCTIONS

char_functions.c

ANY QUESTIONS?

**DON'T FORGET YOU CAN
ALWAYS EMAIL US ON
CS1511@CSE.UNSW.EDU.AU
FOR ANY ADMIN QUESTIONS**

**PLEASE ASK IN THE FORUM
FOR CONTENT RELATED
QUESTIONS**

