COMP1511 PROGRAMMING FUNDAMENTALS

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...

the_shuffler.c

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?

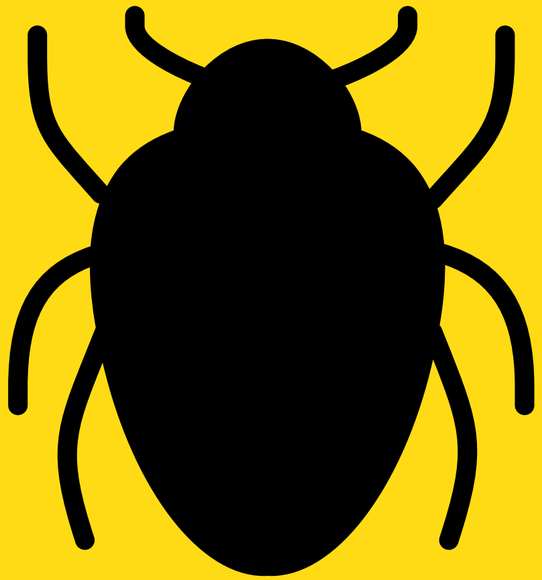
DEBUGGING CODE

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

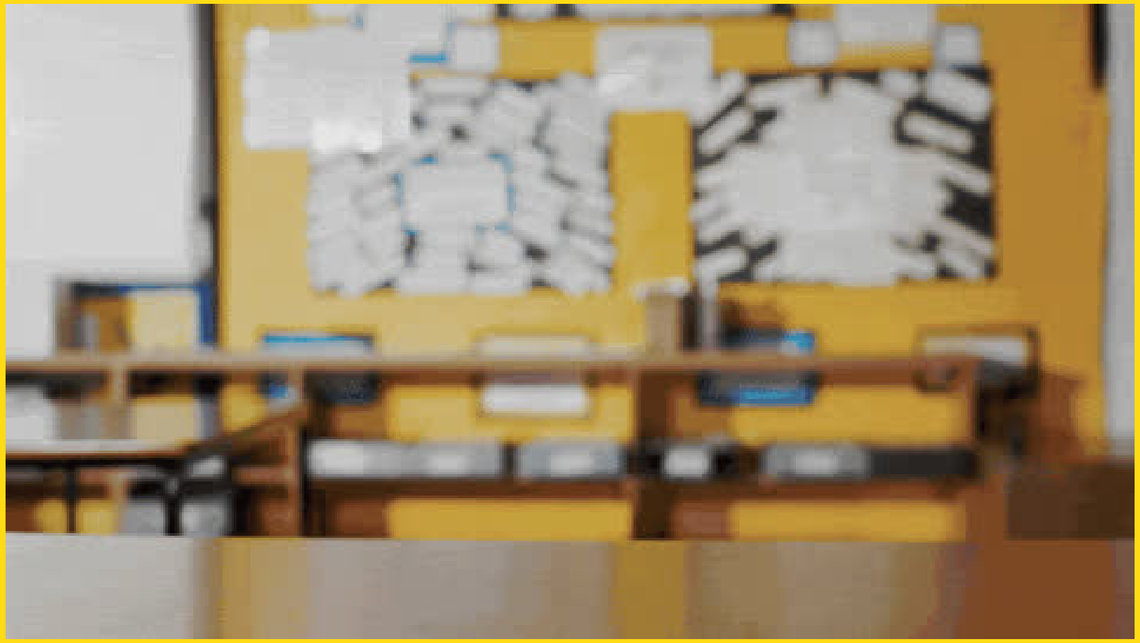
RUN-TIME ISSUES

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

```c
#include <stdio.h>

int highest(int numbers[5]);

int main (void) {

    //1. Take numbers in from the user - scanf
    int numbers[5] = {0}; // {0,0,0,0,0,0,0,0,0,0}

    int i = 0;
    while (i < 5) {
        printf("Please enter a number: ");
        scanf("%d", &numbers[i]);
        i++;
    }

    //int highest_number = highest(numbers);

    // Find the highest number
    printf("The highest number is: %d\n", highest(numbers));

    return 0;
}

// Function finds the highest number in an arrays
// Input to function: number
// Output to function: int
int highest(int numbers[5]) {

    int highest_number = -1;

    int i = 0;
    while (i < 5) {
        if (numbers[i] > highest_number) {
            highest_number = numbers[i];
        }
        i++;
    }

    return highest_number;
}
```

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

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);

```c
#include <stdio.h>

int main (void) {

    //Declare a variable int called character
    int character;
    //Use the getchar() function to read one character at a time
    //Remember that this function will take char when a new line is entered
    character = getchar();

    //When you press Ctrl+D to signal EOF (end of file) - the while loop will
    //be exited
    while (character != EOF) {
        printf("You entered the character: ");
        //Using the function putchar to show output one character at a time
        putchar(character);
        printf("\n");
        //Get the next character from the buffer
        character = getchar();
    }
    return 0;
}
```

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.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

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

int main (void) {

//Declare a variable int called character
    int character;

    printf("Enter your name as an example of getchar() and press Enter: ");
    //Use the getchar() function to read one character at a time
    //Remember that this function will take char when a new line is entered
    character = getchar();

    //When you press Ctrl+D to signal EOF (end of file) - the while loop will
    //be exited
    while (character != EOF) {
        printf("You entered the character: ");
        //Using the function putchar to show output one character at a time
        putchar(character);
        printf("\n");
        //Check if the character is a lower case letter by using the function
        //islower() found in <ctype.h> standard library
        if (islower(character)){
            //If it is, then convert it to upper case letter by using the
            //function toupper() found in <ctype.h> standard library
            character = toupper(character);
            printf("Your new character is: ");
            putchar(character);
            printf("\n");
        }

        //Get the next character from the buffer
        character = getchar();
    }
    return 0;
}
```

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