COMP1511/1911 Programming Fundamentals

Week 4
Lecture 1

Assignment 1

- Released today!
- Aims of the assignment
  - Apply arrays and two-dimensional arrays in problem solving
  - Apply good style to your code
  - Apply the use of functions in code
  - Practice skills in debugging code
    - and skills in hunting for missing semicolons
      - Hopefully my lectures have taught you well

Assignment 1 - Quick Rundown

- 4 stages
  - Each stage ramps up in difficulty
- Go through the stages chronologically
  - Don’t skip stages
- A quick rundown of the setup will be shown in a short video

This Sounds a bit Scary...

- If only there were some sort of session...where I could revise...or get help...that’d be super cool...

Do You Need Help?

- Help sessions exist!
  - https://discourse01.cse.unsw.edu.au/26T2/COMP1511/t/help-sessions-starting-next-week/42
- Just drop in via timetable!
  - No booking required!!

Do You Need Help?

- Revision sessions coming soon!
  - https://www.tickettailor.com/events/comp1511unsw/2265522
- Running in
  - Week 4
  - Week 6
  - Week 8
  - Week 11
- Great if you've fallen behind, or just want to consolidate what you've learnt so far!
  - A bit more structured than help sessions
  - Goes through fun new revision content!

Even More Help!!! (we're so cool)

- Each week you've been getting a check-off understanding rating
  - Hopefully this isn't news
- You can now click on the "Check-off" button (on the home page) to see your very own personalised support dashboard based on your understanding!

Even More Even More Help!!! (we're so very cool)

- Email cs1511@unsw.edu.au
  - You could win (book) a 30 minute private tutoring session
    - This would be very epic and help you a lot (not clickbait)

Previously On...

- Discovered functions
  - Ways to break up our code
- Discussed style
  - Ensuring our code is readable
- Talked about arrays
  - Homogeneous collections
- Briefly touched on strings
  - Arrays of characters with a special character at the end

Today

- Recap basic arrays
- Recap strings
- Array of structs
- Array of arrays

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/

Why Bother?

- Persistent practice procures perfection
- Coding hard?
  - Code lots!
    - Coding easier!
- Collections of collections are powerful representations

Arrays Recap

- Collection of the same type
- Declared using a type, name, and size of array
- Elements accessible via index
  - Index starts at 0, ends at size - 1
- Pairs well with loops
  - while and for

Arrays Recap Problem

- A user enters 10 numbers, we want to count how many times they enter 6 or 7
- A user enters 10 numbers, we want to return the sum of only the odd numbers

Strings Recap

- A collection (array) of characters (chars)
- Finish with a null-terminating character
  - ‘\0’
  - The array must be big enough to accommodate this character
  - Not displayed as part of the string
  - Indicates that this array of characters is a string
    - Instead of just an array of characters
  - It is very useful to know when our string has come to an end

Strings Recap

- Can be declared in a couple of ways

```c
#include <stdio.h>

int main(void) {
    char greeting[] = "Hello!";
    printf("%s\n", greeting);
    return 0;
}
```

```c
#include <stdio.h>

int main(void) {
    char good_tv[] = {'S', 'u', 'r', 'v', 'i', 'v', 'o', 'r', '\0'};
    printf("%s\n", good_tv);
    return 0;
}
```

Reading in Strings

- Reading in strings is NOT like reading in other data types
  - ints and floats?
    - Use scanf
  - strings?
    - Use fgets

fgets

- fgets takes three arguments
  - The array the string will be stored in
  - The number of characters that will be read in
  - Where the string is coming from
    - In our case, always stdin

```c
#include <stdio.h>

#define MAX_SIZE 64

int main(void) {
    char array[MAX_SIZE];
    fgets(array, MAX_SIZE, stdin);
    printf("%s", array);
    return 0;
}

```

Reading in Repeatedly

- Reads in input until ctrl+d is entered

```c
#include <stdio.h>

#define MAX_SIZE 64

int main(void) {
    char array[MAX_SIZE];
    while (fgets(array, MAX_SIZE, stdin) != NULL) {
        printf("%s", array);
    }
    return 0;
}
```

string.h

- We’re using a new library!
  - #include <string.h>
- Gives us access to more functions for strings
  - Similar to how stdio.h gives us functions for input and output
- Read more here:
  - https://www.w3schools.com/c/c_ref_string.php

Other String Functions

- strlen()
  - Gives us the length of a string
- strcpy()
  - Copies the contents of one string to another
- strcat()
  - Attaches a string to the end of another (concatenate)
- strcmp()
  - Compares two strings
- strchr()
  - Finds the first or last occurrence of a character

Example

- Using some string.h functions!

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

int main(void) {
    char array[10];
    //Copy Survivor into our array
    strcpy(array, "Survivor");
    printf("%s\n", array);

    //Checks length of our string
    int length = strlen(array);
    printf("%d\n", length);

    //Compares two strings
    int compare_string = strcmp("Henry", "Epic");
    printf("%d\n", compare_string);

    int new_compare_string = strcmp("Survivor", array);
    printf("%d\n", new_compare_string);

    return 0;
}
```

Break Time!

...probably

Arrays of Anything

- Arrays are simply containers that hold variables of a certain type
  - Those types can be anything
  - Including user-defined types!

Arrays of Structs

- An array of structs may look like this
  - What is stored in each array index?
  - How do we access x and y of each index?

```c
#include <stdio.h>

struct coordinate {
    int x;
    int y;
};

int main(void) {
    struct coordinate maps[3];
    return 0;
}
```

Array of Arrays

- An array of arrays is essentially a grid
  - Each array within the array of arrays must contain the same data type
- The first number is how many rows we have, the second is how many columns

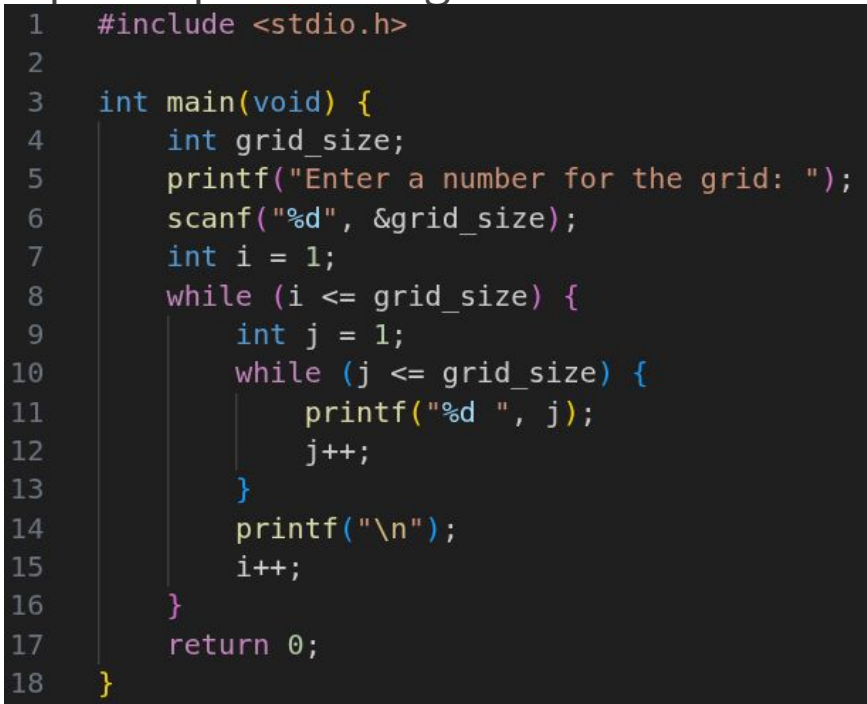
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

```c
#include <stdio.h>

int main(void) {
    int month[4][7];
    return 0;
}
```

Looping Through Arrays of Arrays

- A while ago, we used a while loop to print a grid of numbers
  - How can we transfer this knowledge to printing arrays of arrays?



```c
#include <stdio.h>

int main(void) {
    int grid_size;
    printf("Enter a number for the grid: ");
    scanf("%d", &grid_size);
    int i = 1;
    while (i <= grid_size) {
        int j = 1;
        while (j <= grid_size) {
            printf("%d ", j);
            j++;
        }
        printf("\n");
        i++;
    }
    return 0;
}
```

Transfer to an Array

```c
#include <stdio.h>

int main(void) {
    int month[4][7] = {};
    int row = 0;
    while (row < 4) {
        int col = 0;
        while (col < 7) {
            printf("Printing Row %d Column %d. Value %d\n", row, col, month[row][col]);
            col++;
        }
        row++;
    }
    return 0;
}
```

Transfer to Array

---

row = 0

col = 0

x						

```c
#include <stdio.h>

int main(void) {
    int month[4][7] = {};
    int row = 0;
    while (row < 4) {
        int col = 0;
        while (col < 7) {
            printf("%d", month[row][col]);
            col++;
        }
        row++;
    }
    return 0;
}
```

Transfer to Array

---

row = 0

col = 1

	x					

```c
#include <stdio.h>

int main(void) {
    int month[4][7] = {};
    int row = 0;
    while (row < 4) {
        int col = 0;
        while (col < 7) {
            printf("%d", month[row][col]);
            col++;
        }
        row++;
    }
    return 0;
}
```

Transfer to Array

---

row = 0

col = 2

		x				

```c
#include <stdio.h>

int main(void) {
    int month[4][7] = {};
    int row = 0;
    while (row < 4) {
        int col = 0;
        while (col < 7) {
            printf("%d", month[row][col]);
            col++;
        }
        row++;
    }
    return 0;
}
```

Transfer to Array

---

row = 0

col = 3

			x			

```c
#include <stdio.h>

int main(void) {
    int month[4][7] = {};
    int row = 0;
    while (row < 4) {
        int col = 0;
        while (col < 7) {
            printf("%d", month[row][col]);
            col++;
        }
        row++;
    }
    return 0;
}
```

Transfer to Array

---

row = 0

col = 4

				x		

```c
#include <stdio.h>

int main(void) {
    int month[4][7] = {};
    int row = 0;
    while (row < 4) {
        int col = 0;
        while (col < 7) {
            printf("%d", month[row][col]);
            col++;
        }
        row++;
    }
    return 0;
}
```

Transfer to Array

---

row = 0

col = 5

					x	

```c
#include <stdio.h>

int main(void) {
    int month[4][7] = {};
    int row = 0;
    while (row < 4) {
        int col = 0;
        while (col < 7) {
            printf("%d", month[row][col]);
            col++;
        }
        row++;
    }
    return 0;
}
```

Transfer to Array

---

row = 0

col = 6

						x

```c
#include <stdio.h>

int main(void) {
    int month[4][7] = {};
    int row = 0;
    while (row < 4) {
        int col = 0;
        while (col < 7) {
            printf("%d", month[row][col]);
            col++;
        }
        row++;
    }
    return 0;
}
```

Transfer to Array

---

row = 1
col = 0

x						

```c
#include <stdio.h>

int main(void) {
    int month[4][7] = {};
    int row = 0;
    while (row < 4) {
        int col = 0;
        while (col < 7) {
            printf("%d", month[row][col]);
            col++;
        }
        row++;
    }
    return 0;
}
```

Transfer to Array

---

row = 1
col = 1

	x					

```c
#include <stdio.h>

int main(void) {
    int month[4][7] = {};
    int row = 0;
    while (row < 4) {
        int col = 0;
        while (col < 7) {
            printf("%d", month[row][col]);
            col++;
        }
        row++;
    }
    return 0;
}
```

Transfer to Array

---

row = 1
col = 2 (and so on)

		x				

```c
#include <stdio.h>

int main(void) {
    int month[4][7] = {};
    int row = 0;
    while (row < 4) {
        int col = 0;
        while (col < 7) {
            printf("%d", month[row][col]);
            col++;
        }
        row++;
    }
    return 0;
}
```

Problem Time

- Read in a 3x3 grid of numbers, and print out each row and column.
  - Extended - Print out only even numbers (and where they are on the grid)

Problem Time 2

- Read in Diet Coke drinking data for 5 days each week for 4 weeks. If any day has more than 3 cans, print out the day and week, and say it’s a problem.
  - Extend, if the sum of a week is greater than 30, we have a big problem.
  - Extended Extended, if the sum of the 4 weeks is greater than 100, we have a massive problem

What We Learnt

- Recapping arrays
- Recapping strings
  - How to read in strings
  - Cool string libraries
- Arrays of collections
  - Array of structs
  - Array of arrays

Reach Out

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

Thank You

Questions?