COMP1511/1911 Programming Fundamentals

Lecture 2

Previously On...

- Welcome and Introduction
- Course administration
- How COMP1511 works
- How to approach learning programming
- What is programming?
- Linux and how Linux works

Today

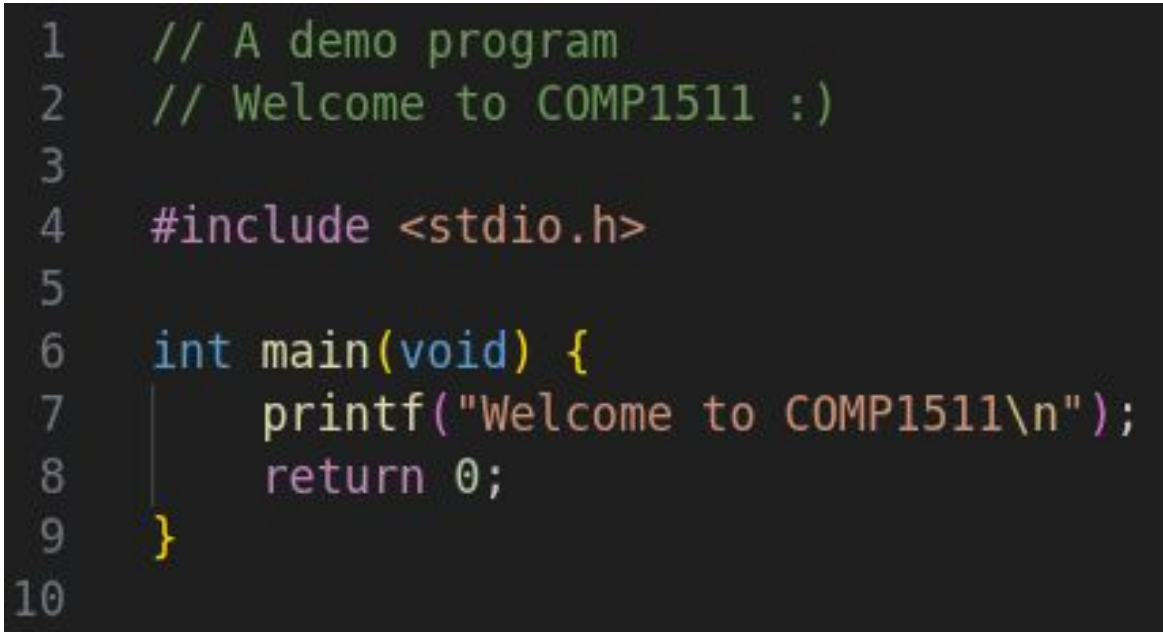
- Variables
  - How we store information
- Constants
- Maths in C

Where is the Code?

- Lecture code is updated every hour on the hour
  - 4pm
  - 5pm
  - 6pm
  - Etc
- Live lecture code can be found here
  - https://cgi.cse.unsw.edu.au/~cs1511/26T2/

Recap

- Our first program
  - What does each bit do?



Why Bother?

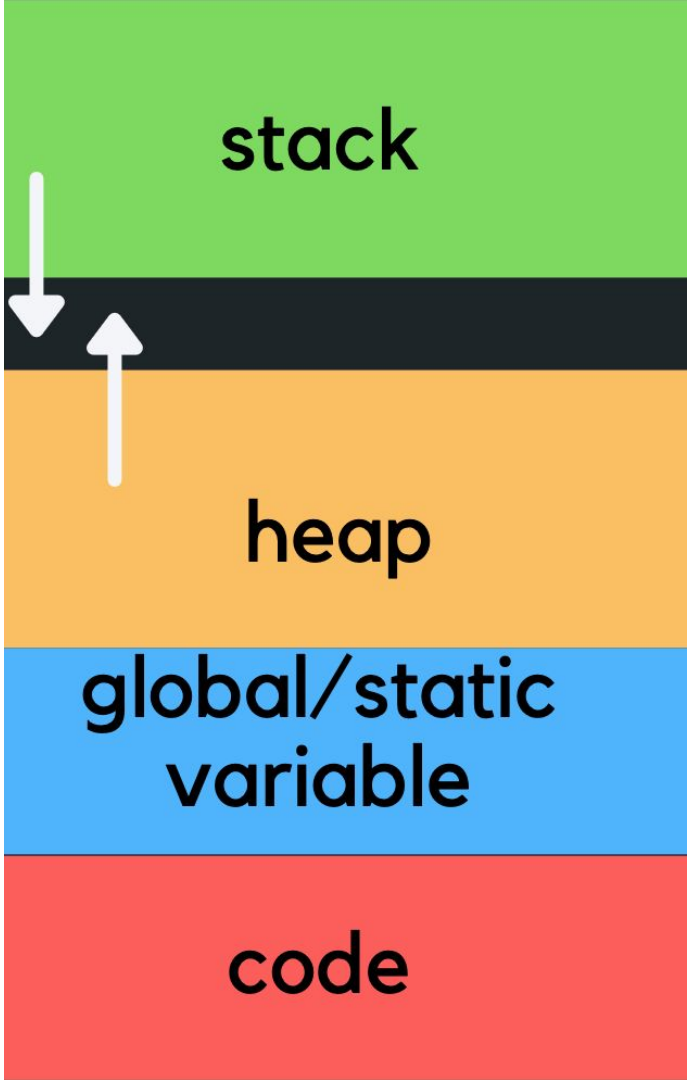
- Our programs currently can’t remember anything
  - If we can’t remember the past, how can we make decisions towards the future?
- We want to store some information for our program
- We want to update that information as things change
- We want our code to be neat and readable

How Does a Computer Remember Things?

- Computer memory is a pile of on and off switches
  - We call these bits
    - The smallest possible unit in computing, a choice between 1 or 0
- We often collect these into bunches of 8 bits
  - Known as a byte

What Does This Look Like?

- The CPU processes instructions and performs basic arithmetic
- RAM keeps track of the data needed in these instructions/operations



What is a Variable?

- Our way of asking the computer to remember something for us
- Called a “variable” because it can change value
- Each variables is a certain number of bits of information
  - Used to represent something
- Made with a specific purpose in mind

What Kinds of Variables Will We Learn Today?

- Three main data types
  - int
    - Whole number
      - 0, 1, 2, etc
  - char
    - Single character
      - ‘A’, ‘B’, ‘h’, etc
  - double
    - Floating point number
      - 8.111, 9.213, 6.7, etc
- Each of these are allocated a different number of bytes in memory when the program runs

Naming Our Variables

- Names are quick descriptions of what variables are
  - answer or diameter instead of a or b
- We always use lowercase letters to start our variable names
- C is case sensitive
  - answer and Answer and ansWer are all different variables
- C reserves some words
  - return, int, and double can’t be used as variable names
    - There are more
- Multiple words?
  - Use snake_case
    - split_words_using_underscores

Naming Our Variables

- We have a style guide
  - Please use it!
  - https://cgi.cse.unsw.edu.au/~cs1511/26T2/resources/style_guide.html
- Remember, someone else has to read your code and understand what you're trying to do!

Integers

- Data type int
- Whole number
  - No fractions or decimals
- Most commonly uses 32 bits
  - 4 bytes
- This gives us 2^{32} possible numbers
  - Large, but not infinite
- Exact ranges from -2,147,483,648 (-2^{31}) to 2,147,483,647 (2^{31} - 1)

Characters

- Data type char
- A single character in C
  - Can also be represented as an int
    - A single character holds as ASCII value (0-127) as opposed to the character itself
- To assign a single character, we use single quotes
  - char initial = ‘H’
- char letter = ‘a’ actually assigns the number 97, but we see it as a character

Doubles

- Data type double
- A double-sized floating point number
  - Decimal value, floating point means the decimal can be anywhere in the number
    - E.g 10.567, 1.0567, 105.67
      - The points are different places in the same digit
- It’s called a double because it uses 64-bits instead of 32 bits
  - Double the size of our integer
  - 8 bytes instead of 4

Let's Try Some Code

- Declare and initialise variables

```c
// Making some awesome variables
// Henry Hickman
// Week 1

#include <stdio.h>

int main(void) {
    int answer = 10;
    // Variables can be changed
    int answer = 42;
    // We can have multiple variables
    int answer_2 = 67;
    return 0;
}
```

Printing to the Terminal

- Not just for specific messages we type in advanced
  - We can also print variables to our display!
- To print out a variable, we use a format specifier
  - % followed by a character to let the compiler know what data type we want to print
  - %d -> int
    - (d is for decimal)
- After the comma, put the name of the variable

Printing to the Terminal

- In this case, because we're printing an integer, we've used %d

```c
// Let's try printing!

#include <stdio.h>

int main(void) {
    int answer = 12;
    printf("The answer is %d!\n", answer);
}
```

Print Out Many Variables

- The variables will match the symbols in the same order they appear
- You can have as many variables as you want, and of different data types!

```c
// Printing multiple variable types

#include <stdio.h>

int main(void) {
    int good_survivor = 28;
    int best_survivor = 17;
    printf("The best season of Survivor is season %d, but season %d is alright too!\n", best_survivor, good_survivor);
}
```

Different Types of Numbers

- The %d and %lf are format specified used in printf statements
  - Let’s the compiler know what types of data we’re printing
    - %d is decimal integer
    - %lf is long floating point number
      - Double
- Remember, the computer only does what we tell it
  - We must be extremely specific about what we do
  - Down to specifying the data types we use

Different Types of Numbers

- %d for our ints
- %lf for our doubles

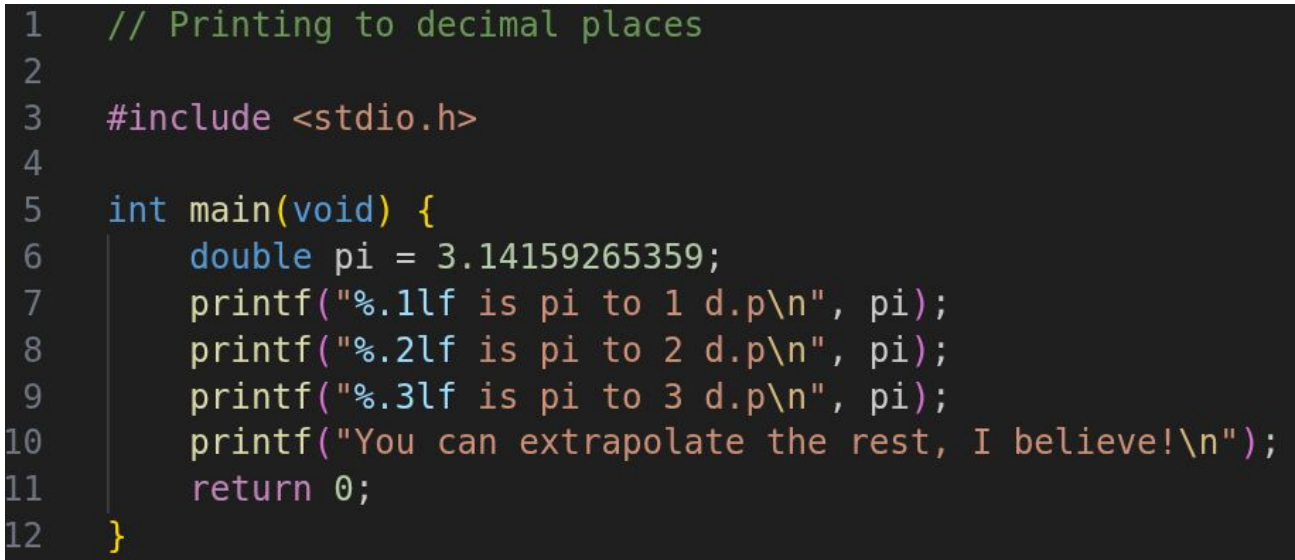
```c
// Printing ints and doubles

#include <stdio.h>

int main(void) {
    int diameter = 5;
    double pi = 3.1415;
    printf("The diameter is %dcm and pi is %lf!\n", diameter, pi);
}
```

Points of Precision

- Sometimes for doubles, we don’t want all the information
- %.1lf prints to 1 decimal place
- %.2lf prints to 2 decimal places



```c
// Printing to decimal places

#include <stdio.h>

int main(void) {
    double pi = 3.14159265359;
    printf("%.1lf is pi to 1 d.p\n", pi);
    printf("%.2lf is pi to 2 d.p\n", pi);
    printf("%.3lf is pi to 3 d.p\n", pi);
    printf("You can extrapolate the rest, I believe!\n");
    return 0;
}
```

What About Chars?

- %c can be used to let the compiler know we want to display a character
  - %c stands for character
- Remember, characters go in single apostrophes
  - ‘a’ not “a”

```c
// Printing a char

#include <stdio.h>

int main(void) {
    char initial = 'H';
    printf("My name starts with %c\n", initial);
    return 0;
}
```

Taking Input

- Programs that produce output are great
  - But for people to use them, they need to be able to interact with them
  - Giving input is a great kind of interaction!
- Let’s use scanf for that!
  - If you know about fgets...no you don’t!

scanf

- Reads input from the user
- Format specifies (%d, %lf, %c) are used to specify the input we expect
- The & symbol tells scanf the address of the variable in memory (where we want to put the data)
  - More on that later in the term

```c
// Let's get input

#include <stdio.h>

int main(void) {
    int answer;
    printf("Tell me your answer: ");
    scanf("%d", &answer);
    printf("Your answer is %d!\n", answer);
}
```

scanf chars

- If you scanf a char, it is still stored as an ASCII value
  - We can flip between %c and %d all we want!

```c
// Scan in and print some chars

#include <stdio.h>

int main(void) {
    char letter;
    printf("Enter a letter: ");
    scanf("%c", &letter);
    printf("The letter is %c and the ASCII code is %d\n", letter, letter);
}
```

Variables that Never Change

- Constants are like variables, except they never change
- To define a constant, we use #define and follow it with the name of the constant and the value

```c
// Constants never change
// War...war never changes

#include <stdio.h>

#define HEIGHT 194
#define MEANING_OF_LIFE 42
#define PI 3.14159

int main(void) {
    printf("Pi is a constant of %lf\n", PI);
}
```

How Does scanf Really Work?

- Gives us the ability to scan stuff from the terminal
- We have to tell the computer what we expect to scanf()
  - int? double? char?
- But since scanf() is a function does it return something?
  - Yes, scanf() returns the number of input values that are scanned
  - If there is some input failure or error then it returns EOF (end-of-file)
    - We will look at this more later on!
  - This can be useful to check for any errors

Newline and scanf

- scanf(“%d”, &number) can ignore newlines, because it is looking for integers
- scanf(“%c”, &character) cannot ignore newlines, because \n is just a character
- We can use scanf(“ %c”, &character) to ignore all whitespace before the character is entered!

Let's Talk About Maths

- We can do maths in C
  - We love maths right? Maths is cool and epic?
- A lot of maths will look very similar
  - `+` is addition
  - `-` is subtraction
  - `*` is multiplication
  - `/` is division
- These will happen in the normal order of operations
  - BEDMASS
- We can use brackets to force precedence

```c
// Doing some maths

#include <stdio.h>

int main(void) {
    int number1 = 12;
    int number2 = 13;
    int result;
    result = (number1 + number2) * number1;
    printf("%d\n", result);
}
```

Maths with chars

- Because chars are just ASCII numbers, we can manipulate them with maths!
  - ‘b’ = ‘a’ + 1
  - ‘a’ = ‘A’ + 32

```c
// Doing maths on chars in C

#include <stdio.h>

int main(void) {
    char letter = 'H';
    printf("%c as lowercase is %c\n", letter, letter + 32);
}
```

The Quirks of Integers

- Because there are limits to computing, integers sometimes cause problems
  - Boeing 787s had to reset every 248 days because of integers
    - https://www.engadget.com/2015-05-01-boeing-787-dreamliner-software-bug.html
- I once did a whole comedy show about these quirks
  - https://www.youtube.com/watch?v=zYHf5nAX9Pk

The Quirks of Integers

- Adding two large integers together can hit the limit, and cause the number to become negative
  - These errors can cause large problems
    - Extreme example - Ariane 5 explosion
    - Less extreme example - Gangnam Style view count
- Ints are not always 32-bits

The Quirks of Doubles

- Doubles have a lot of precision, but they can’t accurately represent every number
- Try to represent ⅓ using binary here for fun later
  - https://www.csfieldguide.org.nz/en/interactives/binary-cards/?digits=10&start=WBBBBBBBBB&offset=-9
    - Note: it’s not possible

Let's Think About Division

- C thinks in data types
  - If either number in the division is doubles, the result will be a double
  - If both numbers are ints, the result will be an int
  - ints do not round nicely, they simply drop the decimal place
    - 5/3 becomes 1, not 2

Remainders

- % is called modulo
  - It tells us the remainder after we do some division
  - 15 % 5 gives us 0, because if we divide 15 by 5, there is no remainder
  - 14 % 5 gives us 4, because if we divide 14 by 5, there is 4 remainder
- Useful for figuring out things like
  - Odd or even
  - Divisibility
  - ...and more!

What We Learnt

- Recap on C
- Variables
- Maths

Reminders

- THERE IS NO LECTURE ON MONDAY
- IT IS AN ONLINE LECTURE ON MONDAY
- HENRY WILL POST A YOUTUBE VIDEO SOME TIME BEFORE MONDAY

Reach Out

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

Thank You

Questions?