COMP1511/1911 Programming Fundamentals

Week 2
Lecture 1

Previously On...

- Welcome and introduction
- An intro to C
- Compiling and running code
- printf and scanf
- Variables
  - ints
  - chars
  - doubles
- Maths

Today

- if statements
- Logical operators
- Chaining if and else
- Loop and loop and loop and loop and loop
  - while loops

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/

Recap Input and Output

- printf for output
- scanf for input
- Different symbols for different data types
  - %d decimal (int)
  - %c character (char)
  - %lf long float (double)
    - %.1lf (double to 1 d.p.)

Why Bother?

- We want our programs to make decisions
  - If our programs can’t change based on circumstance, they’re not very useful
- We also don’t want to repeat ourselves
  - Why write many line when few line work?

Decision Time

- We want to make decisions based on the information we have
  - We let our program branch between sets of instructions
- In C this is an if statement
  - The building block towards algorithms

What Problems Do We Solve?

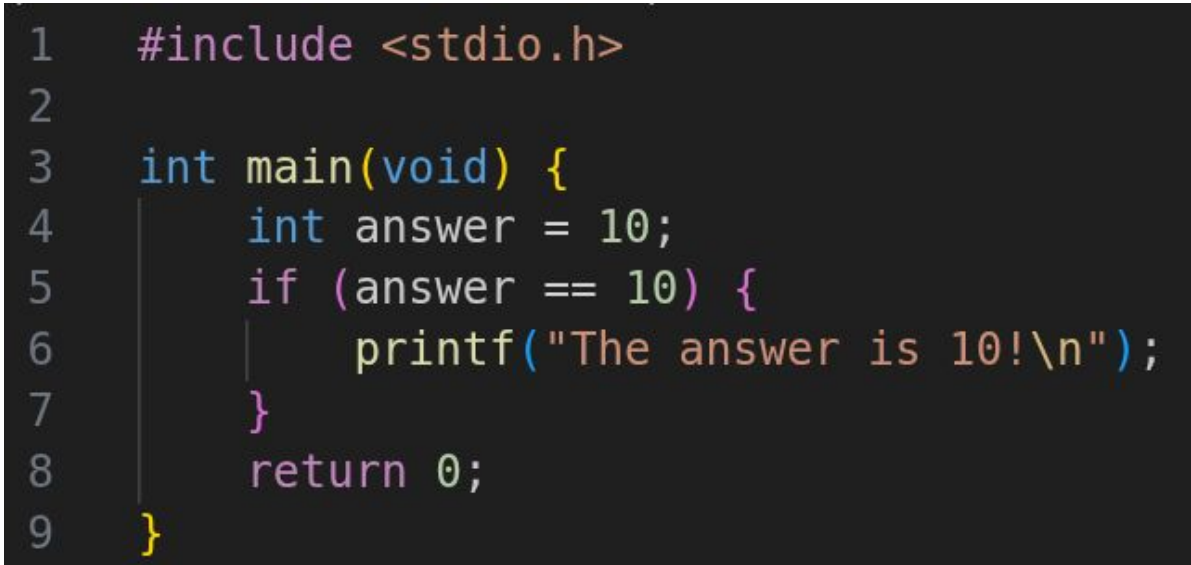
- World peace?
  - No
- Decision problems?
  - Yes

Decision Problems

- A question with two possible answers
  - YES
  - NO
- We can use if statements to answer these questions
  - Is a number evern?
  - Is a number larger than 10?
  - Is a number prime?
- The ability to answer these questions is extremely powerful

If Statements

- These are both the question and the answer
- First we ask the question
  - If the answer is YES, we run the code in the curly braces



```c
#include <stdio.h>

int main(void) {
    int answer = 10;
    if (answer == 10) {
        printf("The answer is 10!\n");
    }
    return 0;
}
```

What If the Answer is NO?

- If the answer to our question (condition) is NO, we can add an else statement to let the computer know what to run instead

```c
#include <stdio.h>

int main(void) {
    int answer = 12;
    if (answer == 10) {
        printf("The answer is 10!\n");
    }
    else {
        printf("The answer is not 10!\n");
    }
    return 0;
}
```

Even More Conditions

- We can combine else and if statements to make else if statements
  - Asking another question to the program

```c
#include <stdio.h>

int main(void) {
    int answer = 12;
    if (answer == 10) {
        printf("The answer is 10!\n");
    }
    else if (answer == 12) {
        printf("The answer is 12!\n");
    }
    else {
        printf("The answer is not 10!\n");
    }
    return 0;
}
```

How to Ask Questions?

- Relational operators
  - < less than
  - > greater than
  - <= less than or equal to
  - >= greater than or equal to
  - == equals
  - != not equal to
- All of these result in 0 when false, and 1 when true

Questions Within Questions?

- Any code can go inside the if statement
  - Even, another if statement!

```c
#include <stdio.h>

int main(void) {
    int answer = 12;
    int answer2 = 10;
    if (answer == 12) {
        if (answer2 == 10) {
            printf("Two ifs were true\n");
        }
        else {
            printf("One if was true\n");
        }
    }
    else {
        printf("No ifs were true\n");
    }
    return 0;
}
```

Asking Two Questions at Once?

- Logical operators
  - && means and
    - Both questions must be true for the whole statement to be true
  - || means or
    - If either questions is true, the whole statement is true
  - ! means not
    - Reverses the result of the question
- && and || go between questions
- ! goes in front of a question

Let's Code Some If Statements

- if statements with logical operators
- if statements with char
- Chaining if and else together

Let's Solve a Problem!

- You are on the hit TV show Survival, and want to guess how many coconuts are left on the island. If you guess right, you win immunity. You get told if your guess is more than, less than, or the winning guess!
- Extra extension
  - If you are within 5 of the correct number, you get told "So close, yet so far!".

Breaking Down the Problem

---

1. The scenario
    a. You are guessing the number of coconuts on the island
2. Logic
    a. Exact match - “Congratulations! You win immunity!”
    b. Within 5 - “So close, yet so far!”
    c. Too high - “Not even close, way less than that.”
    d. Too low - “Not even close, way more than that.”

Breaking Down the Problem

---

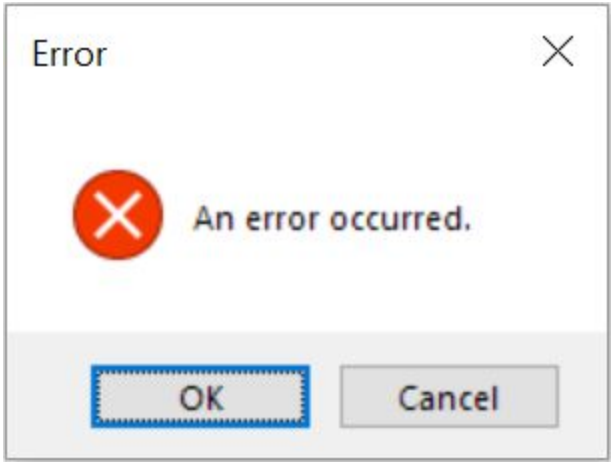
1. Define constants
    - Which ones?
2. Core logic
    a. Ask the castaway to enter their guess
    b. If the guess is equal to the winning number
        i. Print “Congratulations! You win immunity!”
    c. Else if the number is greater than or less than the winning number +-5
        i. Print “So close, yet so far!”
    d. else if greater than the winning number
        i. Print “Not even close, way less than that.”
    e. else if less than the winning number
        i. Print “Not even close, way more than that.”

What Should We Consider?

- Does it matter the order we check the guess?

Breaking Things

- When programming, try to think about what may cause your program to break
  - Try and counter these breaks!
- Make sure you have good error messages
  - Tells the user what went wrong
  - How they can fix it
  - Why it’s happening



Let's Program with scanf

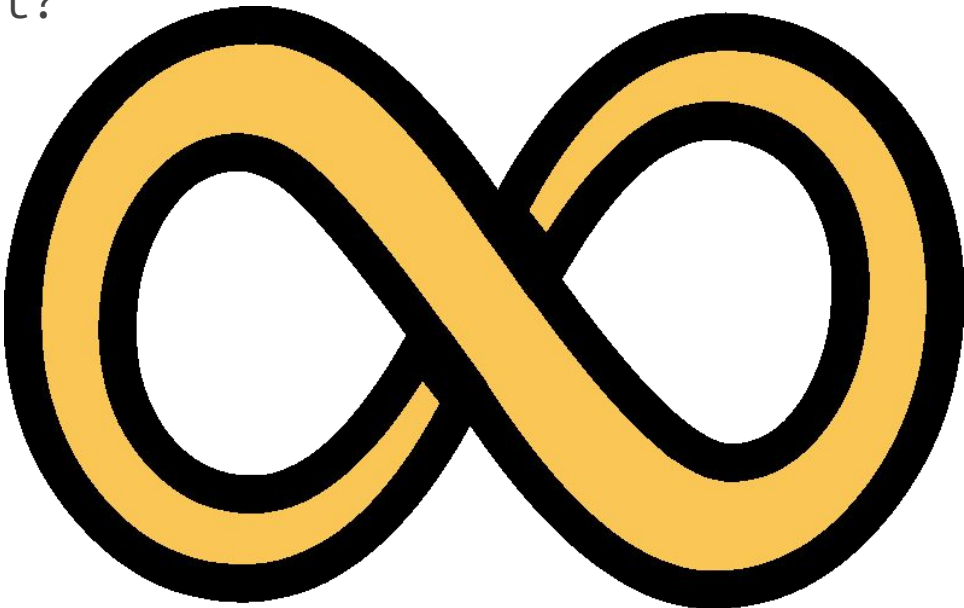
- Gives us the ability to scan stuff from the terminal
- We have to tell the computer what we expect to scanf()
  - int? double? char?
- But since scanf() is a function does it return something?
  - Yes, scanf() returns the number of input values that are scanned
  - If there is some input failure or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

Break Time

...probably...

Repeating Ourselves

- What if we wanted to play the game multiple times?
  - e.g The player gets 5 guesses, or guesses until correct?
- How could we accomplish that?



Repeating Ourselves

- C executes code one line at a time
  - if statements let us turn code on and off
  - But now we want to repeat code
- Let’s copy and paste code over and over again!
  - (do not do this)

While Loops

- While something is true, do that repeatedly
- Three main types
  - Count loop
  - Sentinel loop
  - Conditional loop

```c
#include <stdio.h>

int main(void) {
    int answer = 12;
    // When will this loop end? HINT - Never
    while (answer == 12) {
        printf("This will take a while\n");
    }
    return 0;
}
```

Infinite Loops

- It’s very easy to make a program that goes on forever
  - Your loop control variable must update, otherwise your loop will never end
- What’s the loop control variable of this program?

```c
#include <stdio.h>

int main(void) {
    int answer = 12;
    // When will this loop end? HINT - Never
    while (answer == 12) {
        printf("This will take a while\n");
    }
    return 0;
}
```

Count Loops

- Use a variable to count how many times a loop runs
  - int declared before the loop
- The “termination condition” can be checked in the while loop
- It will be updated inside the loop

```c
#include <stdio.h>

int main(void) {
    int counter = 1;
    while (counter < 5) {
        printf("Printed %d times\n", counter);
        counter = counter + 1
    }
    return 0;
}
```

Sentinel Loops

- Count loops depend on us knowing how many times we want something to happen
  - What if we want something to happen until a certain event?
    - E.g keep prompting for input until an odd number is entered
      - We don’t know how long this will take, but we will know when it’s happened
- A sentinel is a flag value
  - It tells the loop when to stop

Sentinel Loops

- Sentinel loops can also use a variable to decide to exit the loop at any time
- We call this variable a “sentinel”
  - Like an on/off switch for the loop
  - Declared and set outside the loop
  - Its termination condition can be checked in the while condition
  - It will be updated inside the loop
    - Often attached to a decision statement

Sentinel Loops

```c
#include <stdio.h>

int main(void) {
    int exit_condition = 0;
    int coconuts_harvested = 0;
    int total_coconuts = 0;
    while (exit_condition == 0) {
        printf("How many coconuts will you collect? ");
        scanf("%d", &coconuts_harvested);
        total_coconuts = total_coconuts + coconuts_harvested;
        if (total_coconuts > 40) {
            printf("We have enough food to survive! Yay!!\n");
            exit_condition = 1;
        }
        else {
            printf("Let's keep harvesting!\n");
        }
    }
    return 0;
}
```

Conditional Loops

- Use a condition to decide when the exit the loop
  - We also don’t know how many times something will repeat, but we are slowly making progress towards the condition
- We will terminate as a result of some kind of calculation

Conditional Loops

```c
#include <stdio.h>

int main(void) {
    int sum = 0;
    int adder = 0;
    while (sum < 100) {
        printf("Please enter a number: ");
        scanf("%d", &adder);
        sum = sum + adder;
    }
    return 0;
}
```

Action Time

- Let’s play around with some more while loops!

What We Learnt

- if statements
  - Controlling the flow of the program
- while loops
  - Repeating code without repeating code

Reach Out

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

Thank You

Questions?