

# Welcome!

## COMP1511 18s1

### Programming Fundamentals

# COMP1511 18s1

## — Lecture 4 —

# More Functions + Loops

Andrew Bennett

<andrew.bennett@unsw.edu.au>

even more functions

`while` loops

# Before we begin...

**introduce** yourself to the person sitting next to you

**why** did they decide to study **computing**?

# Overview

**after this lecture, you should be able to...**

handle **invalid input** to your program

understand **why** we use functions

write **simple functions**

understand the basics of **while loops**

(**note:** you shouldn't be able to do all of these immediately after watching this lecture. however, this lecture should (hopefully!) give you the foundations you need to develop these skills. remember

programming is like learning any other language or skill, it takes consistent and regular practice.)

# Admin

## Don't panic!

these slides are on WebCMS3 ("DRAFT")

lecture recordings are on WebCMS3

make sure you have **home computing** set up

make sure you can send and receive **uni emails**

# A challenge for you

## Guess the Number

computer is thinking of a number  
enter a guess  
program responds “higher” or “lower” or “correct!”

### hint

to start out with:  
have a fixed secret number  
(i.e. `int secret = 5`)  
`scanf` their guess  
rerun the program to guess another number

remember **functions**?

# Functions

building blocks in our programs

self-contained, reusable pieces of code

abstraction

# Anatomy of a Function

## return type

(`void` if no return value)

## function name

## parameters

(inside parens, comma separated;

`void` if no parameters)

## statements

## return statement

```
int addNumbers (int num1, int num2) {  
    int sum = num1 + num2;  
    return sum;  
}
```

# Functions as Building Blocks

for example:

a function that takes a number and multiplies it by 2

we can take our number, and put it into the function, and get it out doubled

```
int x = 5;  
x = doubled (x);
```

## key things:

input (parameters)

output (return value)

functions won't change values

# Why Functions?

Revisiting `license.c`

# Why Functions?

**main** function:

want to know **what** it's doing

don't need to know **how** it's doing it

# Side Note: When `scanf` Goes Wrong

what do we do if somebody enters **invalid input**?

(e.g. enters a word, not a number)

```
int a;  
int b;  
// What happens if they didn't type in two numbers?  
int num = scanf("%d %d", &a, &b);
```

# Side Note: When `scanf` Goes Wrong

`scanf` **returns** the number of things successfully scanned in

e.g.

```
int a;  
int b;  
// num will be 2 if both a and b were scanned successfully  
int num = scanf("%d %d", &a, &b);
```

# Side Note: When `scanf` Goes Wrong

we can wrap this in an `if` statement:

```
int a;
int b;
// num will be 2 if both a and b were scanned successfully
if (scanf("%d %d", &a, &b) != 2) {
    printf("Invalid input!\n");
}
```

# Features of Functions

a function can have zero or more parameter(s)

a function can **only** return zero or one value(s)

\* \* \*

a function stores a local copy of parameters passed to it

the original values of variables remain unaltered

before we get started: extending the challenge

# Extending the challenge

## Guess the Number (v2)

computer is thinking of a number  
enter a guess  
program responds “higher” or “lower” or “correct!”  
then asks again  
and again  
until you guess correctly

### hint

use a loop to run the code multiple times (coming up next!)

and now for something **new**...

# Remember `if` statements?

```
int main (void) {  
    printf ("Enter a number: ");  
  
    int num;  
    scanf ("%d", &num);  
  
    if (num < 10) {  
        printf ("Hello!\n");  
    }  
  
    return 0;  
}
```

**if** the condition is true, **then** do something, **else** do something else.

## What if we wanted to do something more than once?

```
int main (void) {  
    printf ("Enter a number: ");  
  
    int num;  
    scanf ("%d", &num);  
  
    while (num < 10) {  
        printf ("Hello!\n");  
    }  
  
    return 0;  
}
```

## What if we wanted to do something more than once?

```
int main (void) {
    printf ("Enter a number: ");

    int num;
    scanf ("%d", &num);

    while (num < 10) {
        printf ("Hello!\n");
        num++;
    }

    return 0;
}
```

# Anatomy of a Loop

**initialisation**

**condition**

**statements**

**update**

```
int i = 0;
while (i < 10) {
    printf ("Hello (number %d)\n", i);
    i = i + 1;
}
```