Welcome!

COMP1511 18s1

Programming Fundamentals

COMP1511 18s1

— Lecture 5 —

More Loops

Andrew Bennett

<andrew.bennett@unsw.edu.au>

`while` loops

loops inside loops

stopping 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…

understand the basics of **while loops**

understand the basics of **nested while loops**

write programs using **while loops** to solve simple problems

know about the course **style guide**

(**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, it takes consistent and regular practice.)

Admin

Don’t panic!

lecture recordings are on WebCMS3

Echo360 was sad last night :(

weekly tests start this week

don’t be scared!

course **style guide** published

Loops

what if we want to do something multiple times?

Use a loop!

*keep doing this **while** this condition is true*

Anatomy of a Loop

initialisation

.

condition

.

statements

.

update

.

Anatomy of a Loop

initialisation

set up our variables

condition

.

statements

.

update

.

Anatomy of a Loop

initialisation

set up our variables

condition

while “something”…

statements

.

update

.

Anatomy of a Loop

initialisation

set up our variables

condition

while “something”…

statements

things we do inside our loop

update

.

Anatomy of a Loop

initialisation

set up our variables

condition

while “something”…

statements

things we do inside our loop

update

move along to the next iteration

Aside: Definitions

iterate

perform repeatedly

iteration

the repetition of a process

A Counting Loop

"Do this thing `n` different times"

sometimes, it's explicit:
e.g. print out 'hello world!' 10 times

sometimes, it's not:
e.g. print out the numbers from 1-10
e.g. calculate the power of a number (e.g., 2^3)

A Counting Loop

“Do this thing `n` times”

use a **loop counter**

… a variable that we use in our loop

to count how many times we’ve done something

A Counting Loop

do something until we've done it `n` times

e.g. print out 'hello world!' 10 times

counter starts at 0

print "hello world!"; increase counter to 1 (we've done it once)

print "hello world!"; increase counter to 2 (we've done it twice)

print "hello world!"; increase counter to 3 (we've done it three times)

...

print "hello world!"; increase counter to 9 (we've done it 9 times)

print "hello world!"; increase counter to 10 (we've done it 10 times)

now stop, because we've done it 10 times.

A Counting Loop

how would we code this?

start our counter at 0

print “hello world!”
while counter is less than 10,
increase our counter by 1

Anatomy of a Loop

initialisation

set up our variables

condition

while “something”…

statements

things we do inside our loop

update

move along to the next iteration

```
????
while (?????) {
    ????
    ????
}
```

initialisation

```
// set up our loop counter, start at 0
while (?????) {
    ????
    ????
}
```

initialisation

condition

```
// set up our loop counter, start at 0
while (something) {
    ????
    ????
}
```

initialisation

condition

statements

```
// set up our loop counter, start at 0
while (something) {
    // do something
    ????
}
```

initialisation

condition

statements

update

```
// set up our loop counter, start at 0
while (something) {
    // do something
    // move to the next iteration of the loop
}
```

initialisation

condition

statements

update

```
int i = 0;
while (something) {
    // do something
    // move to the next iteration of the loop
}
```

initialisation

condition

statements

update

```
int i = 0;
while (i < 10) {
    // do something
    // move to the next iteration of the loop
}
```

initialisation

condition

statements

update

```
int i = 0;
while (i < 10) {
    printf ("hello, world!\n");
    // move to the next iteration of the loop
}
```

initialisation

condition

statements

update

```
int i = 0;
while (i < 10) {
    printf ("hello, world!\n");
    i = i + 1;
}
```

how do we know when to **stop**?

Loop Counters

```
int i = 0;
while (i < 10) {
    printf ("hello, world!\n");
    i = i + 1;
}
```

Loop Counters

```c
// Print out "hello, world!" n times,
// where n is chosen by the user.

int num;
printf ("Enter a number: ");
scanf ("%d", &num);

int i = 0;
while (i < num) {
    printf ("hello, world!\n");
    i = i + 1;
}
```

Sentinel Value (Flag)

```
int finished = 0;
while (!finished) {
    printf ("hello, world!\n");
    finished = 1;
}
```

Sentinel Value (Flag)

```c
// Print out the number that the user entered
// Stop when they type 0

int n = 1;
while (n != 0) {
    printf ("You entered: %d\n", n);
    scanf("%d", &n);
}
```

what is a **style guide**?

Style Guide

https://cgi.cse.unsw.edu.au/~cs1511/resources/style_guide.html

linked from WebCMS3

nested loops

loops **inside** loops

nested loops

```
while (something) {
    while (somethingElse) {

    }
}
```

Demo: Printing a Square

scan in a number: **width**

print out a square of **width** * **width** stars.

e.g. for width = 4:

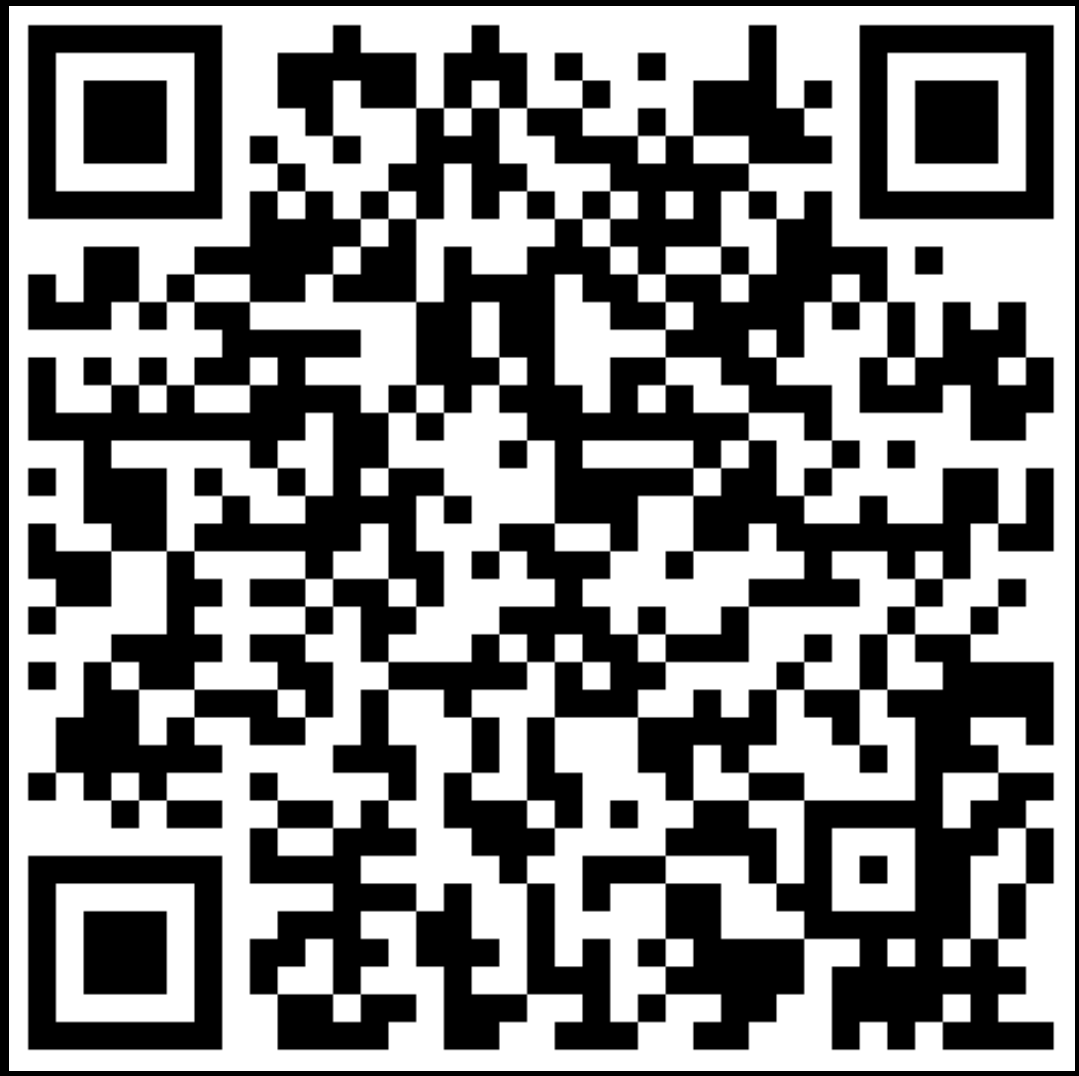
```
* * * *
* * * *
* * * *
* * * *
```

challenge: *can you just print the outside?*

```
* * * *
*     *
*     *
* * * *
```

Feedback?

bit.do/comp1511-feedback-week3



alternate link: https://andrewb3.typeform.com/to/KuVZP4