

# COMP1511 17s2

## — Lecture 6 —

### Do It Again

Andrew Bennett

`<andrew.bennett@unsw.edu.au>`

review: functions, abstraction

review: types, scope

loops and repetition

# While you wait...

1

Go to the course website, and answer the polls!  
[webcms3.cse.unsw.edu.au/COMP1511/17s2](http://webcms3.cse.unsw.edu.au/COMP1511/17s2)

**Don't panic!**

milestone 1

assignment 0

# Review: Functions

3

using functions

# Review: Functions

4

the *call stack*

# Review: Functions

5

function parameters

# Review: Functions

6

function return values

# Remember if statements?

```
int main (int argc, char *argv[]) {  
    printf ("Enter a number: ");  
  
    int num;  
    scanf ("%d", &num);  
  
    if (num < 10) {  
        printf ("Hello!\n");  
    }  
  
    return EXIT_SUCCESS;  
}
```

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

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

```
int main (int argc, char *argv[]) {  
    printf ("Enter a number: ");  
  
    int num;  
    scanf ("%d", &num);  
  
    while (num < 10) {  
        printf ("Hello!\n");  
    }  
  
    return EXIT_SUCCESS;  
}
```

What if we wanted to do something more than once?

```
int main (int argc, char *argv[]) {  
    printf ("Enter a number: ");  
  
    int num;  
    scanf ("%d", &num);  
  
    while (num < 10) {  
        printf ("Hello!\n");  
        num++;  
    }  
  
    return EXIT_SUCCESS;  
}
```

# Anatomy of a Loop

initialisation  
condition  
statements  
update

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

# Aside: Shorthand

11

```
i = i + 1;  
i += 1;  
i++;
```

and

```
i = i - 1;  
i -= 1;  
i--;
```