COMP1511 17s2

— Lecture 5 —

A Distant Memory

Andrew Bennett

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

review: functions, abstraction
memory, types
scope

While you wait...

Go to the course website, and answer the polls!

`webcms3.cse.unsw.edu.au/COMP1511/17s2`

Admin

Don’t panic!

milestones

blogging

assignment 0

Review: Functions

building blocks in our programs

self-contained, reusable pieces of code

abstraction

Review: 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;
}
```

Review: 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

parameters received by the function,
and local variables created by the function,
are all **discarded** when the function returns

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

Touching the `void`

most functions return a value…
but `void` functions don’t return anything

functions may have “side effects”
like changing the state of the system,
by printing things out

Using Functions

we’ve seen how to call a function:
printf, *scanf*

but don’t show the types, just the name of it

Variables and Values

variables vs values...
what is a value?

a number: `5`, `3.14159265`

a letter: `'a'`

a word: `"hello"`, `"Andrew"`

a series of words: `"hi there how are you?"`

Variables and Values

variables vs values...
what is a variable?

something that holds a value

`int i = 5` ...
`i` is the variable, `5` is the value

Variables: How?

variables stored in *memory*

variables are like **boxes**

memory is *lots* of boxes, all lined up in order

Variables: Where?

where are variables stored in our program?

locations in memory have *addresses*
we can find the address of things in memory with `&`

you've used this before, with *scanf*:

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

instead of giving *scanf* `num`,
we're giving it
"where `num` is"

Program Flow

the code in our programs gets executed line-by-line

Scope

functions can’t change anything outside of themselves

passing values into functions?
… we pass *copies* of values.

every time we call a function,
it gets its own set of boxes to store things in