COMP1511 17s2

— Lecture 4 —

Learn You A Function

Andrew Bennett

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

review: number systems
functions and abstraction

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

Milestones

every two weeks,
a series of short questions for you to give brief responses to,
about your progress in the course since the last milestone

should be quick and easy,
and give you a chance to think about
how you’re going in the course
and to help you stay on top of things

two overall themes:
tracking overall progress in the course,
and craftsmanship

see the Milestone pages
on WebCMS 3

Milestones

Milestone 0 is due

Sunday 13 August, 23:59:59

and is on your progress thus far

Milestones and Blogging

you may find it useful to keep a more regular journal of your work…

Review: Number Systems

we can represent the same numbers
in many different ways

$$28_{10} = 0001\,1100_2 = \texttt{0x1C}_{16}$$

binary is a convenient building block

Review: Number Conversions

Convert these numbers to binary:

53_{10}

$\mathtt{5F3A}_{16}$

$\mathtt{12D}_{16}$

Convert these numbers to hexadecimal:

$10\,101\,111\,011_2$

Review: Number Conversions

Convert these numbers to binary:

$$53_{10} = 110101_2$$

$$\mathtt{5F3A}_{16} = 0101\,1111\,0011\,1010_2$$

$$\mathtt{12D}_{16} = 0001\,0010\,1101_2$$

Convert these numbers to hexadecimal:

$$101\,0111\,1011_2 = \mathtt{57B}_{16}$$

Wouldn’t it be nice if…

… we didn’t have to **copy and paste** blocks of code?

… we could make parts of our code **reusable**?

… make our main function **smaller and simpler**?

… make our programs **nicer to read**?

functions

What is a Function?

you’ve already seen functions outside programming:

$$\cos, \sin, \ldots$$

functions are like a black box.

What is a Function?

you’ve already seen functions *inside* programming!

`printf`, `scanf`

```
int main (int argc, char *argv[]) { ...
```

What is a Function?

functions are way of achieving **abstraction**

Abstraction

“… creating *units* which can be *reused*,
and whose internal details
are *hidden* from outside inspection …”

Abstraction via Functions

Functions allow us to:

separate out, or **encapsulate**
a piece of code serving a single purpose

test and **verify**
a piece of code

reuse
a piece of code

shorten our programs,
making it easier to
modify and **debug**

Anatomy of a Function

return type
function name
parameters
(inside parens, comma separated)
return statement

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

Functions with No Parameters

parameter list: void

```
int getRandomNumber (void) {
    // chosen by fair dice roll...
    // guaranteed to be random
    return 4;
}
```

Functions with No Return Value

return type: void
no return statement necessary

```
void printAsterisks (void) {
    printf ("*****");
}
```

Function Prototypes

every function has a **function prototype**:
tells the compiler that
the function exists,
and the structure it has.

includes **key information**
about the function.

```
int addNumbers (int num1, int num2);
int getRandomNumber (void);
void printAsterisks (void);
```

Noteworthy Features

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

Program Structure

Header comment

`#include`d files

`#define`s

prototypes

`main` function

functions