COMP1511 17s2

— Lecture 1 —

Numbers In, Numbers Out

Andrew Bennett

<andrew.bennett@unsw.edu.au>

more printf
variables
scanf

While you wait...

Go to the course website, and answer the polls!

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

Admin

tutorials and laboratories start in week 1
(some of you have already had tutes and labs)

lecture recordings are on WebCMS 3

make sure you have home computing set up

(re-)introducing: printf

```
printf ("hello world!\n")
```

prints the text
`"hello world!"`
to the terminal

Printing more than just words

Can we print more than just words?

Printing more than just words

Can we print more than just words?

…

Yes!

`printf`

the *f* stands for “formatted”

Let’s try it out!

```c
// Prints out the sum of two numbers.
// Andrew Bennett <andrew.bennett@unsw.edu.au>
// 2017-07-26

#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[]) {
    // Sum two numbers, and print them out.
    printf ("The sum of 3 and 5 is %d\n", 3 + 5);

    return EXIT_SUCCESS;
}
```

…

We can change the numbers
to add different values together
and print them out!

but that’s boring if it can’t be dynamic,
and it sucks to do it by hand.

Introducing variables!

Variables and Types

Variables are used to store data…
think “boxes”

Each variable has a data type…
this is the size and structure of the “box”

For now, we are using three data types:

char stores characters:
`A`, `e`, `#`

int stores whole numbers:
`2`, `17`, `-5`

float stores “floating-point” numbers:
`3.14159`, `2.71828`

Variables

declare
the first time a variable is mentioned,
we need to specify its type.

initialise
before using a variable we need to assign it a value.

assign
to give a variable a value.

```
int num; // Declare
num = 5; // Initialise (also Assign)
...
num = 27; // Assign
```

Variables

We can also declare and initialise in the same step:

```
int num = 5; // Declare and Initialise
...
num = 27; // Assign
```

Variable Naming (and other identifiers)

must be made up of letters, digits and underscores (_)
the first character must be a letter
are case sensitive (`num1` and `Num1` are different)

Keywords like
`if`, `while`, `do`, `int`, `char`, `float`
cannot be used as identifiers

Printing Variables Out

No variables:

```
printf ("Hello World\n");
```

A single variable:

```
int num = 5;
printf ("num is %d\n", num);
```

Printing Variables Out

More than one variable:

```
int num1 = 5;
int num2 = 17;
printf("num1 is %d and num2 is %d\n", num1, num2);
```

The order of arguments
is the order they will appear:

```
int num1 = 5;
int num2 = 17;
printf ("num2 is %d and num1 is %d\n", num2, num1);
```

`printf`'s placeholders

char uses `%c`
int uses `%d`
float uses `%f`
double uses `%lf`

Try it yourself!
Copy the code from the previous slide
into a C program, and run it.

Change the program so it
declares, initialises and prints
a `char`, a `float` and a `double`.

Numbers in: `scanf`

```
int num = 0;
scanf ("%d", &num);
printf ("num = %d\n", num);
```

Note that the variable is still initialised.
(Not necessary, but good practice.)

Note the & before the variable name.
Don’t forget it!

Reading Variables In

Multiple variables (space separated):

```
int num1 = 0;
int num2 = 0;
scanf ("%d %d", &num1, &num2);
printf ("num1 = %d and num2 = %d\n", num1, num2);
```

Multiple variables (comma separated):

```
int num1 = 0;
int num2 = 0;
scanf ("%d, %d", &num1, &num2);
printf ("num1 = %d and num2 = %d\n", num1, num2);
```

Note the space or comma between the variables.

Try it yourself: `scanf`

Create a C program
using the code from the previous slide.

Using what you know about placeholders
with `printf` and `scanf`,
change it so it scans in
and prints out a *char*.