ENGG1003 - Lab 2

Brenton Schulz

1 C Summary

This section will be included in all future lab documents and lists a summary of C language features taught prior to the lab session. It will grow each week.

1.1 Basic Structure

```c
#include <stdio.h>
int main() {
   // Your program starts here
   return 0;
}
```

1.2 Comments

```c
// This is a comment to end of line

/* this is a block comment
   which could span
   multiple
   lines */
```

1.3 Operators

Operation	C Symbol
Addition	+
Subtraction	-
Multiplication	*
Division	/
Increment	++
Decrement	--
Less than	<
Less than or equal to	<=
Greater than	>
Greater than or equal to	>=
Equal to	==
Not equal to	!=

Table 1: Arithmetic operators in C

1.4 Standard i/o

Read a single variable from `stdin` with `scanf();` `scanf("format specifier", &variable);`

Write a single variable to `stdout` with `printf();` `printf("format specifier", variable);`

You can use `printf();` *without* a newline (`\n`) to create an input prompt:

```c
printf("Enter a number: ");
scanf("%d", &variable);
```

This prints:

```
Enter a number:  _
```

where _ indicates the terminal prompt (ie: where typed characters will appear).

NB: Pressing `enter` after typing a value will produce a new line.

1.5 Format Specifiers

The following table is woefully incomplete. The compiler *may* generate warnings if `%d` is given something other than `int` and `%f` is given something other than `float`. If `printf();` output is wrong apply an explicit data type cast.

Data Type	Format Specifier
Integers	`%d`
Floating point	`%f`
Float with `n` decimal places	`%.nf`

Table 2: Basic format specifiers

1.6 Type Casting

Placing the syntax `(type)` before a variable name performs a type cast (ie: data type conversion).

eg: convert `a` to an `int` prior to using its value.

```c
(int)a
```

NB: This does *not* modify the original variable.

This is often done automatically by the compiler but sometimes it is required. Adding it unnecessarily doesn't have any negative impact.