Flow control

Recall where we left our temperature conversion program.

conversion.h

```c
int ctof(int celsius);
int ftoc(int fahrenheit);
```

ctof.c

```c
#include "conversion.h"

int ctof(int celsius)
{
	return 9 * celsius / 5 + 32;
}
```

ftoc.c

```c
#include "conversion.h"

int ftoc(int fahrenheit)
{
	return 5 * (fahrenheit - 32) / 9;
}
```

conversion.c

```c
#include <stdio.h>
#include "conversion.h"

int main(int argc, char **argv)
{
	int celsius;
	int fahrenheit;

	celsius = 100;
	fahrenheit = ctof(celsius);
	printf("%d Celsius is %d Fahrenheit.\n", celsius, fahrenheit);
	printf("And %d converted back.\n", ftoc(fahrenheit));

	return 0;
}
```

Makefile

```
CFLAGS = -Wall -g

all: conversion conversion2 conversion3 conversion4 conversion5

conversion: conversion.o ctof.o ftoc.o
conversion2: conversion2.o ctof.o ftoc.o
conversion3: conversion3.o ctof.o ftoc.o
conversion4: conversion4.o ctof.o ftoc.o
conversion5: conversion5.o ctof.o ftoc.o
conversion.o: conversion.c conversion.h
ctof.o: ctof.c conversion.h
ftoc.o: ftoc.c conversion.h

.PHONY: clean
clean:
	rm -f conversion conversion2 conversion3 conversion4 conversion5 \
	      conversion.o conversion2.o conversion3.o conversion4.o \
	      conversion5.o ctof.o ftoc.o
```

When compiled and run, it converts one temperature.

```
$ make
cc -Wall -g   -c -o conversion.o conversion.c
cc -Wall -g   -c -o ctof.o ctof.c
cc -Wall -g   -c -o ftoc.o ftoc.c
cc   conversion.o ctof.o ftoc.o   -o conversion
$ ./conversion
100 Celsius is 212 Fahrenheit.
And 100 converted back.
```

Let us now change the `main` function to convert multiple temperatures, introducing the `while` loop.

conversion2.c

```c
#include <stdio.h>
#include "conversion.h"

int main(int argc, char **argv)
{
        int celsius;
        int fahrenheit;
        int start = -50;
        int end = 100;
        int step = 10;

        celsius = start;
        while (celsius <= end) {
                fahrenheit = ctof(celsius);
                printf("%d Celsius is %d Fahrenheit.\n", celsius, fahrenheit);
                printf("And %d converted back.\n", ftoc(fahrenheit));
                celsius += step;
        }

        return 0;
}
```

The output of this program is:

```
-50 Celsius is -58 Fahrenheit.
And -50 converted back.
-40 Celsius is -40 Fahrenheit.
And -40 converted back.
-30 Celsius is -22 Fahrenheit.
And -30 converted back.
-20 Celsius is -4 Fahrenheit.
And -20 converted back.
-10 Celsius is 14 Fahrenheit.
And -10 converted back.
0 Celsius is 32 Fahrenheit.
And 0 converted back.
10 Celsius is 50 Fahrenheit.
And 10 converted back.
20 Celsius is 68 Fahrenheit.
And 20 converted back.
30 Celsius is 86 Fahrenheit.
And 30 converted back.
40 Celsius is 104 Fahrenheit.
And 40 converted back.
50 Celsius is 122 Fahrenheit.
And 50 converted back.
60 Celsius is 140 Fahrenheit.
And 60 converted back.
70 Celsius is 158 Fahrenheit.
```

```
And 70 converted back.
80 Celsius is 176 Fahrenheit.
And 80 converted back.
90 Celsius is 194 Fahrenheit.
And 90 converted back.
100 Celsius is 212 Fahrenheit.
And 100 converted back.
```

This code can be slightly simplified by using a `for` loop, which is syntactic sugar for just this pattern of iteration:

conversion3.c

```c
#include <stdio.h>
#include "conversion.h"

int main(int argc, char **argv)
{
        int celsius;
        int fahrenheit;
        int start = -50;
        int end = 100;
        int step = 10;

        for (celsius = start; celsius <= end; celsius += step) {
                fahrenheit = ctof(celsius);
                printf("%d Celsius is %d Fahrenheit.\n", celsius, fahrenheit);
                printf("And %d converted back.\n", ftoc(fahrenheit));
        }

        return 0;
}
```

The two iterative programs are identical, and even produce identical machine code.

Let us now insert a message if one of the iterations catches our interest by demonstrating that the two temperature scales are equivalent. This will employ the `if` statement, which evaluates an expression and executes its statement if and only if the value of the test is nonzero. The `==` operator evaluates to 1 if its operands are equivalent, and 0 otherwise. It should not be confused, ever, with `=`, which is the assignment operator and evaluates to the value assigned.

conversion4.c

```
#include <stdio.h>
#include "conversion.h"

int main(int argc, char **argv)
{
        int celsius;
        int fahrenheit;
        int start = -50;
        int end = 100;
        int step = 10;

        for (celsius = start; celsius <= end; celsius += step) {
                fahrenheit = ctof(celsius);
                printf("%d Celsius is %d Fahrenheit.\n", celsius, fahrenheit);
                printf("And %d converted back.\n", ftoc(fahrenheit));
                if (fahrenheit == celsius)
                        printf("Celsius equals Fahrenheit here!\n");
        }

        return 0;
}
```

The output of this program is mostly the same once again, except on the relevant iteration:

```
-40 Celsius is -40 Fahrenheit.
And -40 converted back.
Celsius equals Fahrenheit here!
```

Perhaps, however, we want completely different behavior whether or not the test is true. The `if` statement can also have an `else` branch, which executes if the test is zero. Thus, either the `if` branch or the `else` branch will execute, as determined by the test, but obviously never both.

conversion5.c

```
#include <stdio.h>
#include "conversion.h"

int main(int argc, char **argv)
{
        int celsius;
        int fahrenheit;
        int start = -50;
        int end = 100;
        int step = 10;

        for (celsius = start; celsius <= end; celsius += step) {
                fahrenheit = ctof(celsius);
                if (fahrenheit == celsius) {
                        printf("%d Celsius equals %d Fahrenheit!\n", celsius,
                               fahrenheit);
                } else {
                        printf("%d Celsius is %d Fahrenheit.\n", celsius,
                               fahrenheit);
                        printf("And %d converted back.\n", ftoc(fahrenheit));
                }
        }

        return 0;
}
```

Now the usual text is not printed at all on the special iteration, only the message:

```
-40 Celsius equals -40 Fahrenheit!
```