Loops

Now we add more power to our programming. One of the advantages of a computer is that it can do the same (probably dumb) thing over and over, very fast. This is called looping – like a race car looping around track. In the Indy 500, someone counts out 200 laps and then a checkered flag is waved to indicate the end of the race so that the cars can stop. Similarly the loops in our program will need some mechanism to control the number of "laps". There are "while" loops and "for" loops. We start with while loops. We see later that "for" loops are a special case of "while" loops.

The thought process for the race car driver is something like this:
1. Check the current number of laps completed.

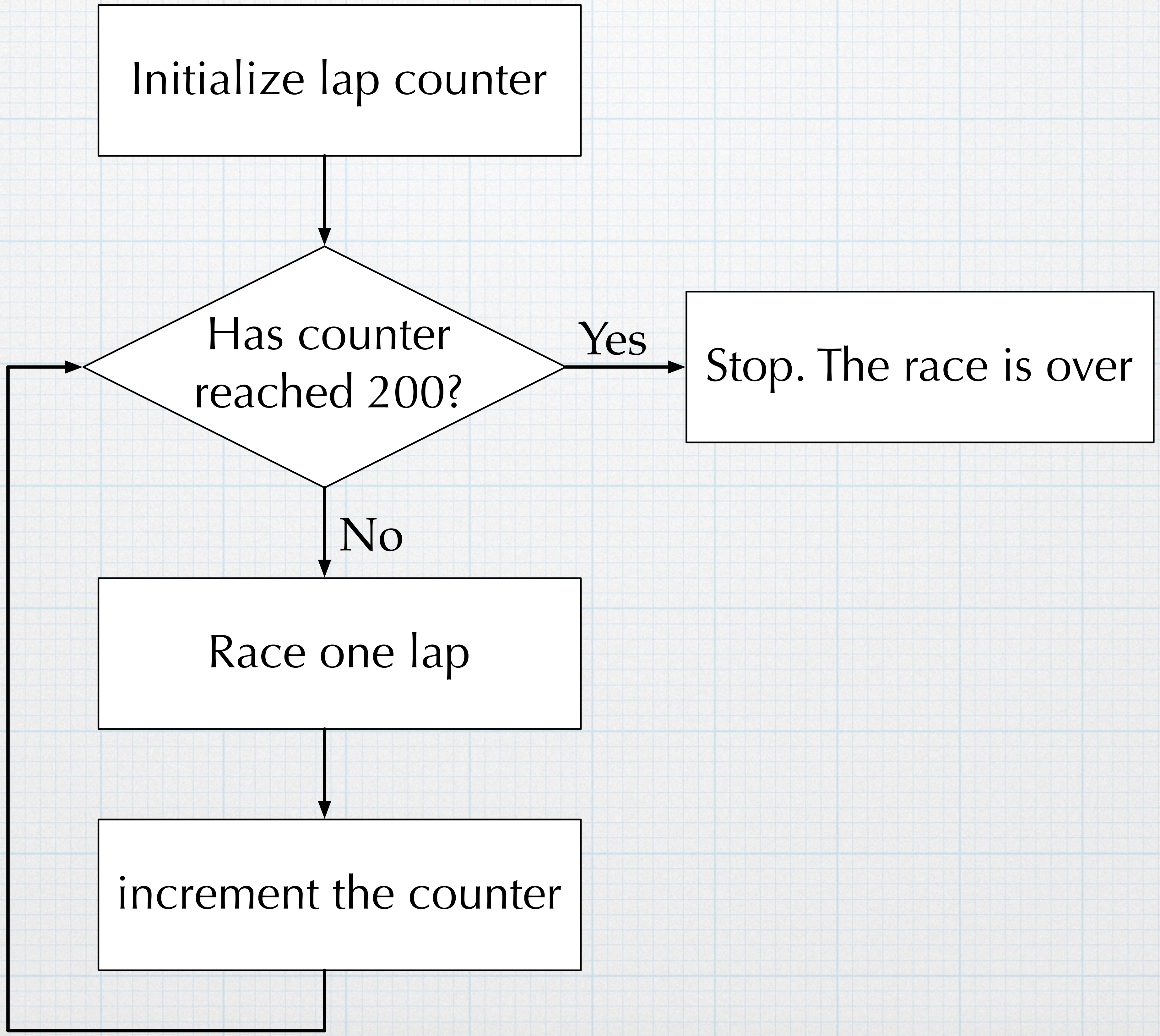
2a. If the number of laps is less than 200, then race one more lap.

3. Increment the counter.

4. Repeat.

2b. If the number of laps = 200, then stop racing and declare a winner.

The process can be visualized with a *flow diagram*. Making a flow diagram is not necessary, but it can be helpful.

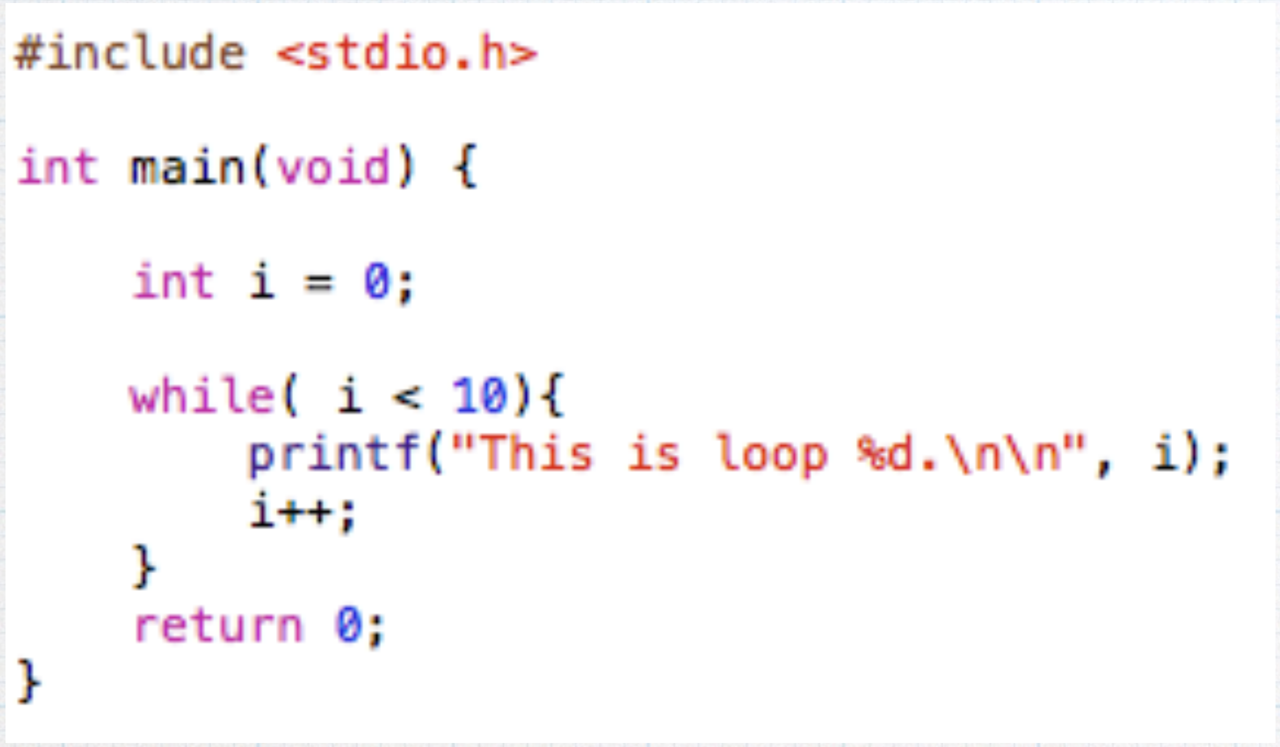


While Loops

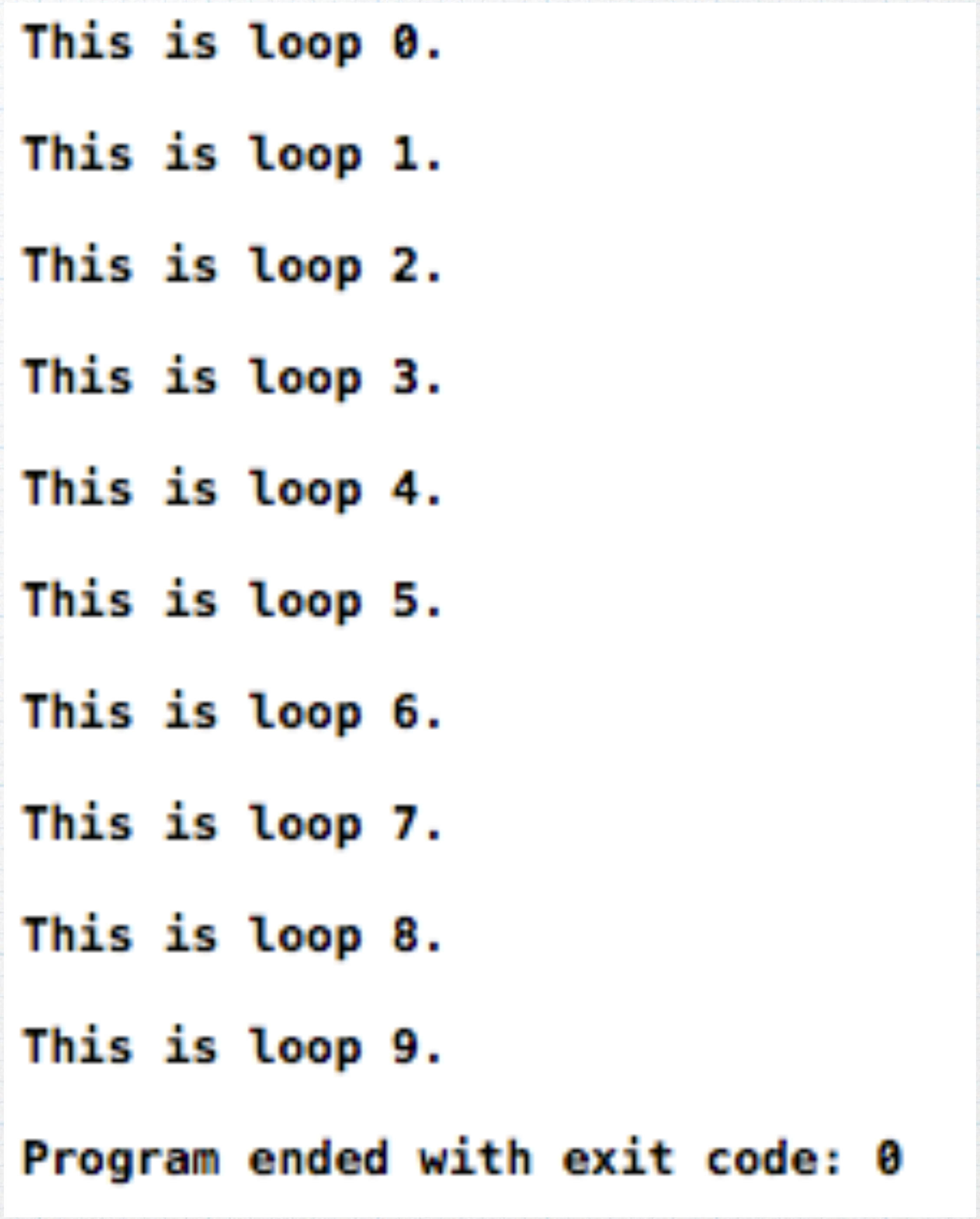
In a computer, the process is exactly the same. There should be a counter to keep track of the loops. There is a condition the must be checked – done inside a "while" block. Depending on the result of the conditional check, either go through the loop again or jump out of the loop and go to the next part of the program.

The basic program structure:

```
int i = 0;  \\ the "counter"
while ( some conditional expression, maybe involving i ) {
    Do something here inside the loop;
    increment i;
}
```



```
#include <stdio.h>

int main(void) {

    int i = 0;

    while( i < 10){
        printf("This is loop %d.\n\n", i);
        i++;
    }
    return 0;
}
```

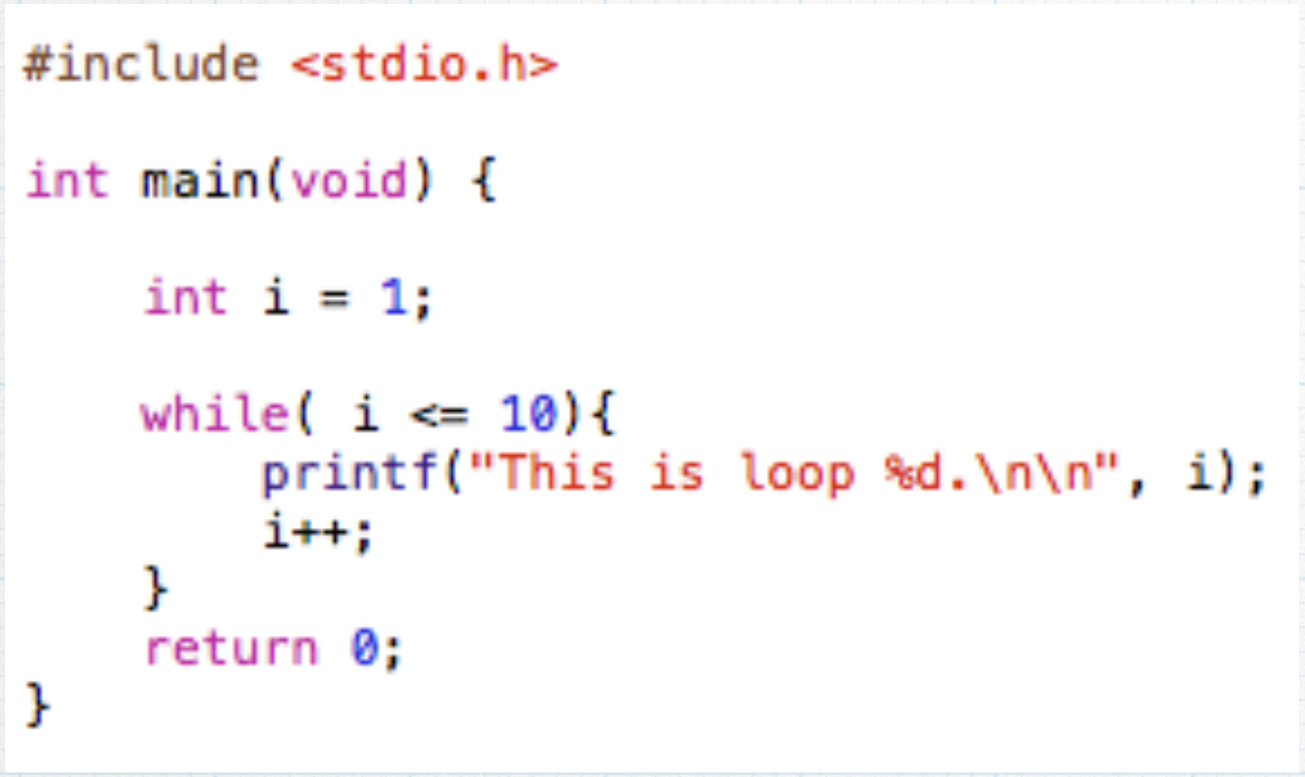


```
This is loop 0.

This is loop 1.

This is loop 2.

This is loop 3.

This is loop 4.

This is loop 5.

This is loop 6.

This is loop 7.

This is loop 8.

This is loop 9.

Program ended with exit code: 0
```

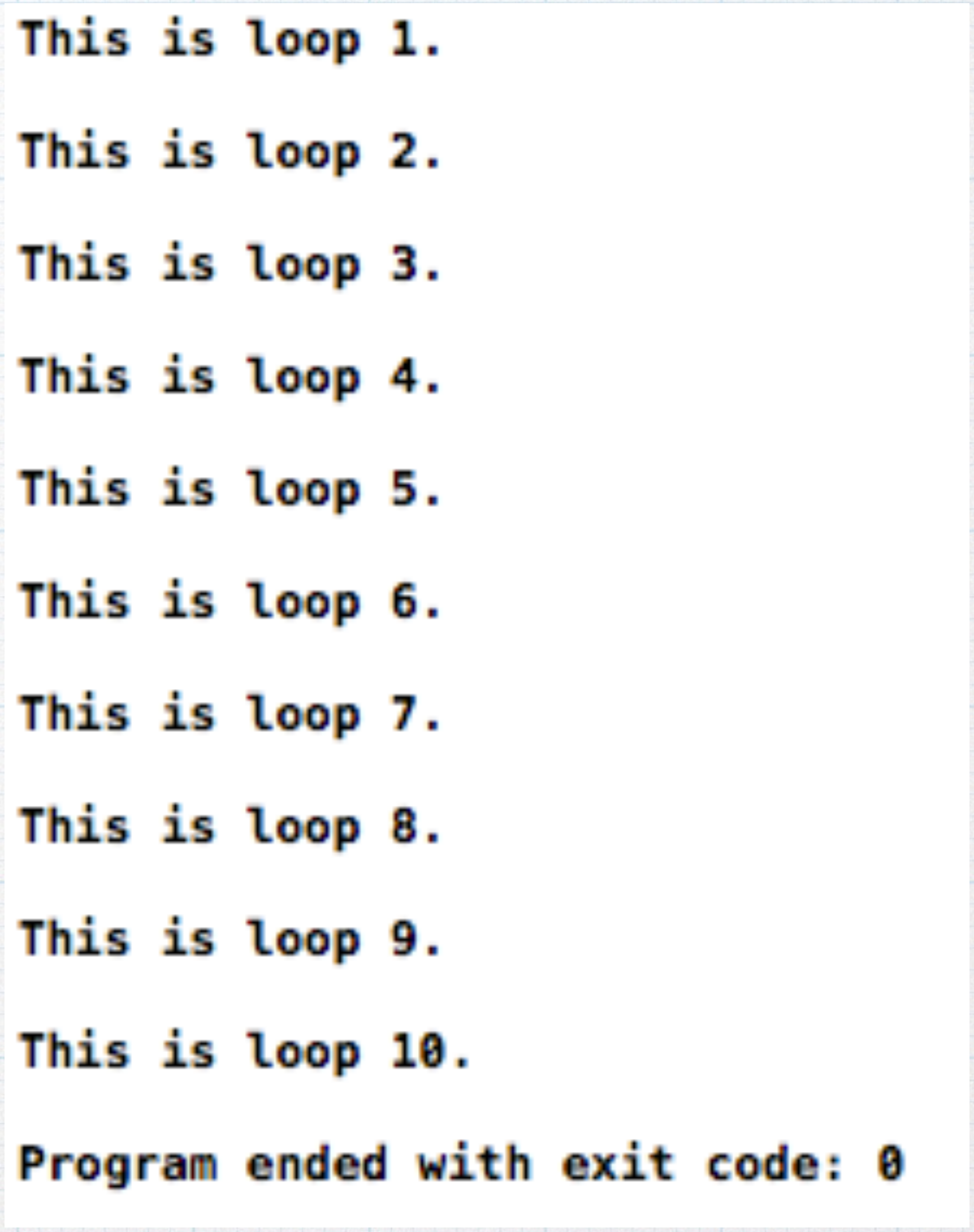
i++; is the “increment” operation.

It has the same effect as i = i + 1;

The sequence of steps: Read the value of the variable i. Then increment by 1 (add 1 to the original value). Store the new value back in the memory location for *i*.



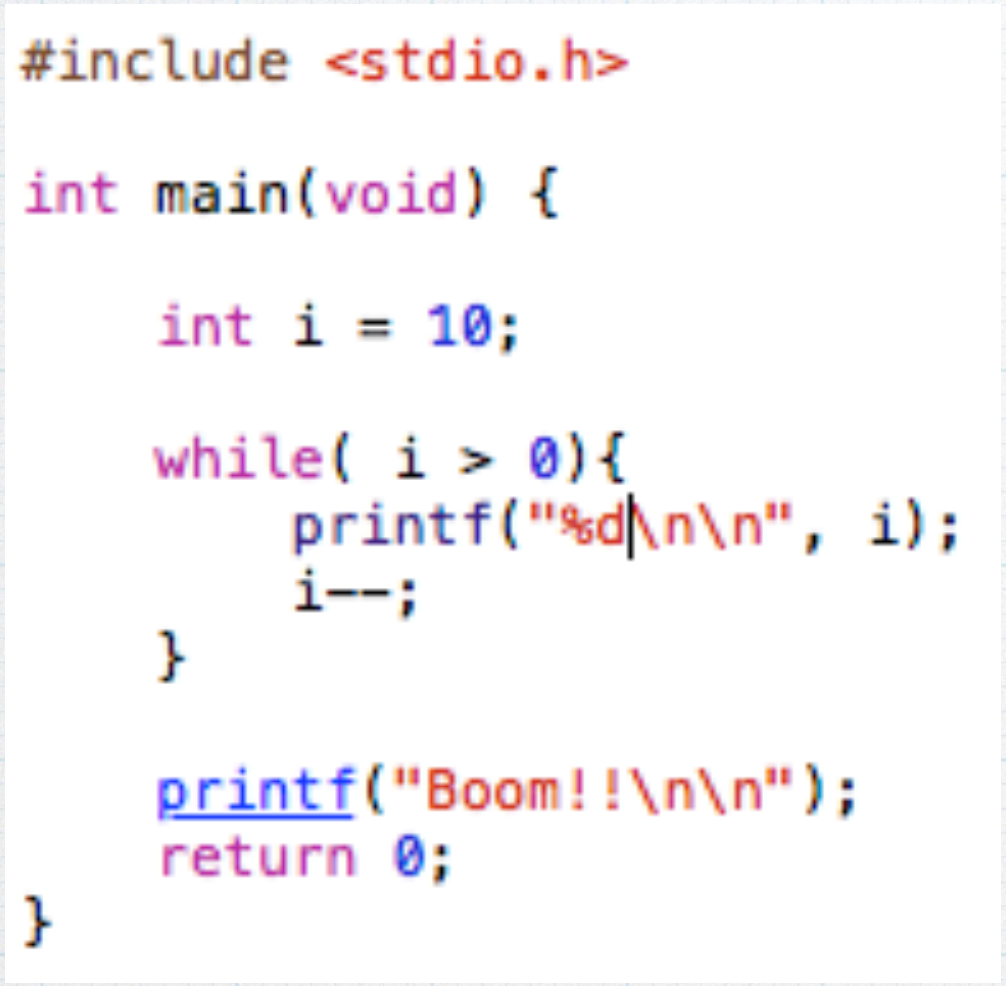
```
#include <stdio.h>

int main(void) {

    int i = 1;

    while( i <= 10){
        printf("This is loop %d.\n\n", i);
        i++;
    }
    return 0;
}
```



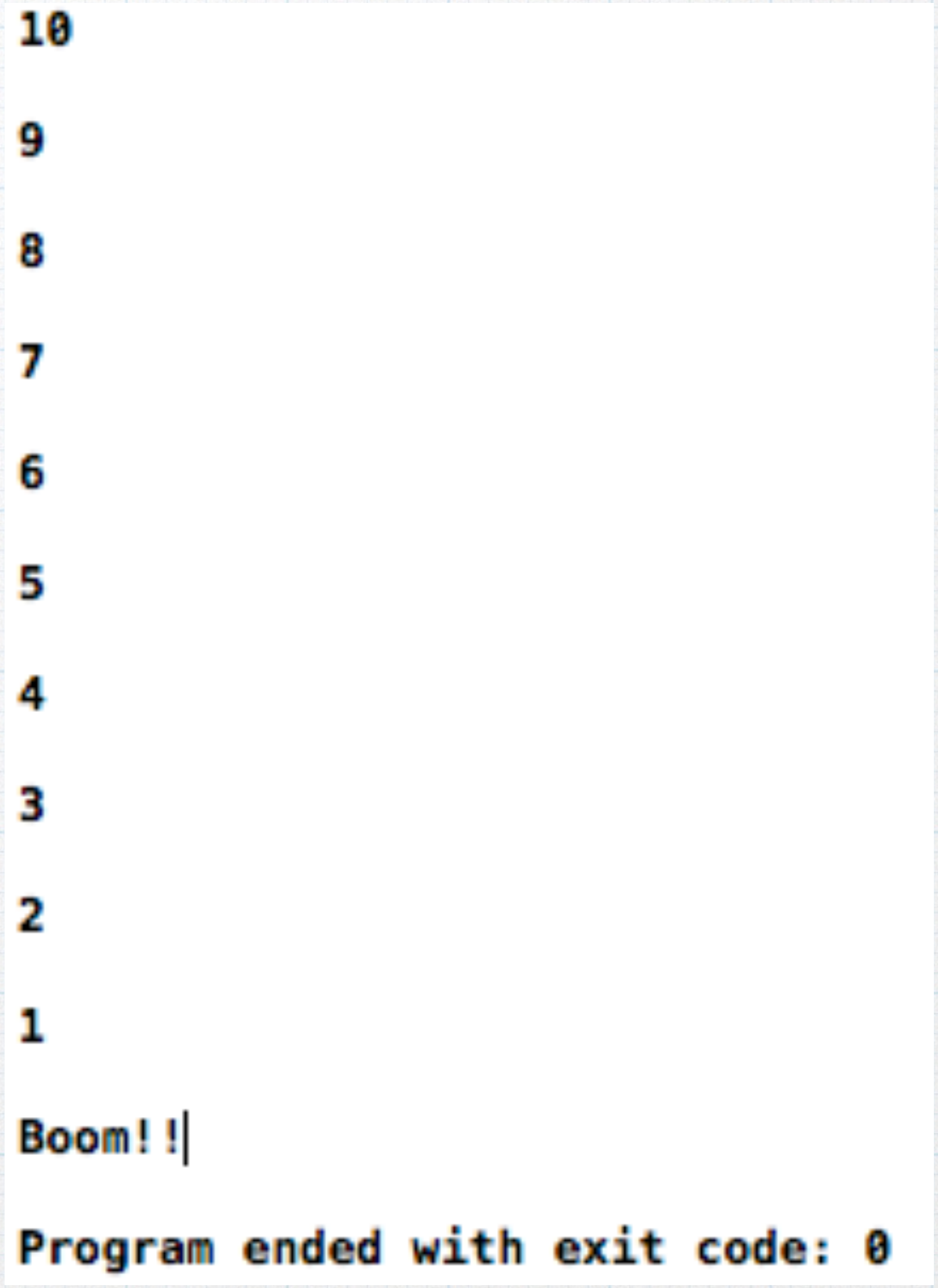
```
This is loop 1.

This is loop 2.

This is loop 3.

This is loop 4.

This is loop 5.

This is loop 6.

This is loop 7.

This is loop 8.

This is loop 9.

This is loop 10.

Program ended with exit code: 0
```

Similar program, but slightly different counting and conditional.

Can count down, too.



```
#include <stdio.h>

int main(void) {

    int i = 10;

    while( i > 0){
        printf("%d\n\n", i);
        i--;
    }

    printf("Boom!!\n\n");
    return 0;
}
```



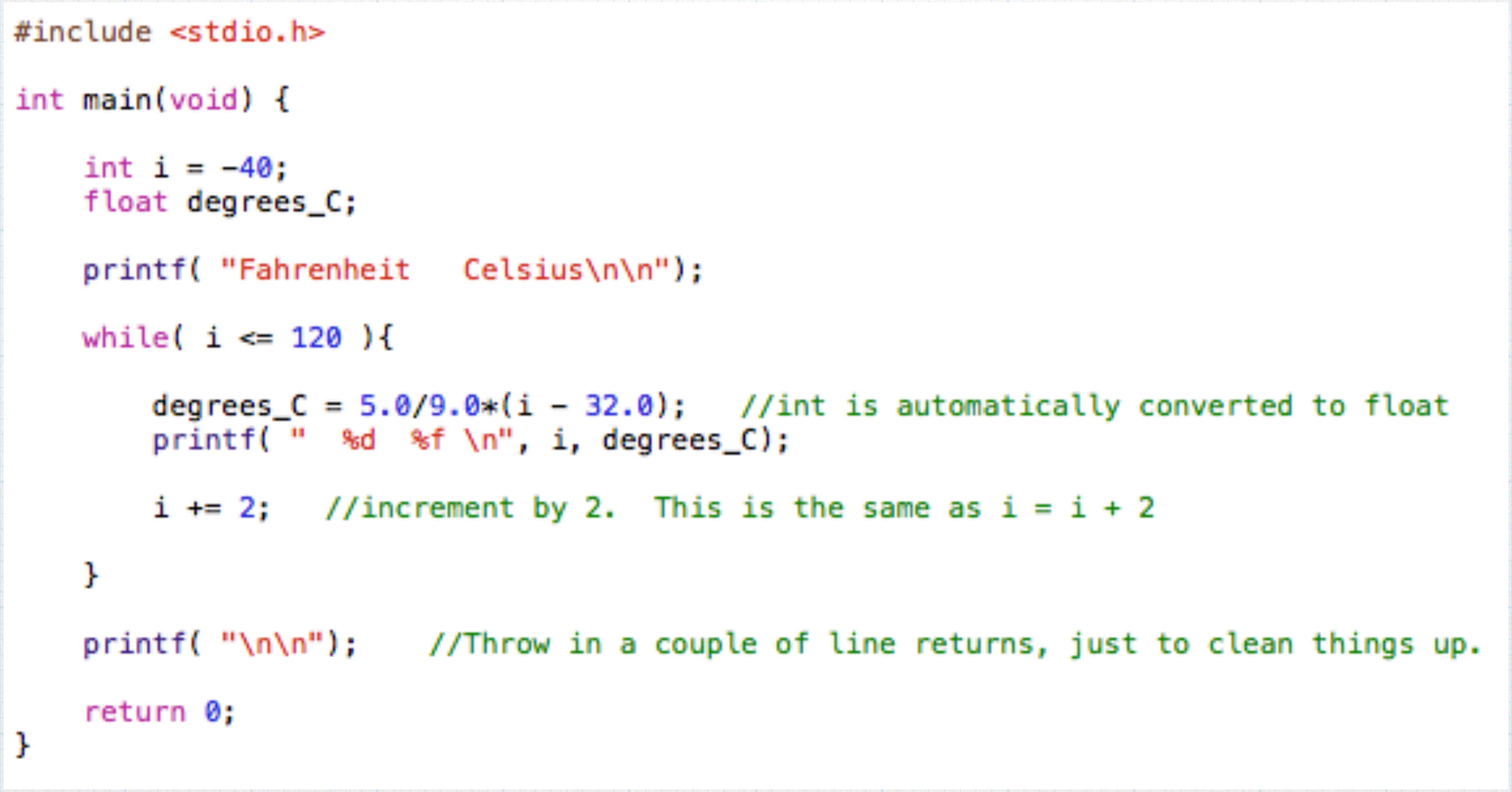
```
10

9

8

7

6

5

4

3

2

1

Boom!!

Program ended with exit code: 0
```

`i--` is the "decrement" operator.

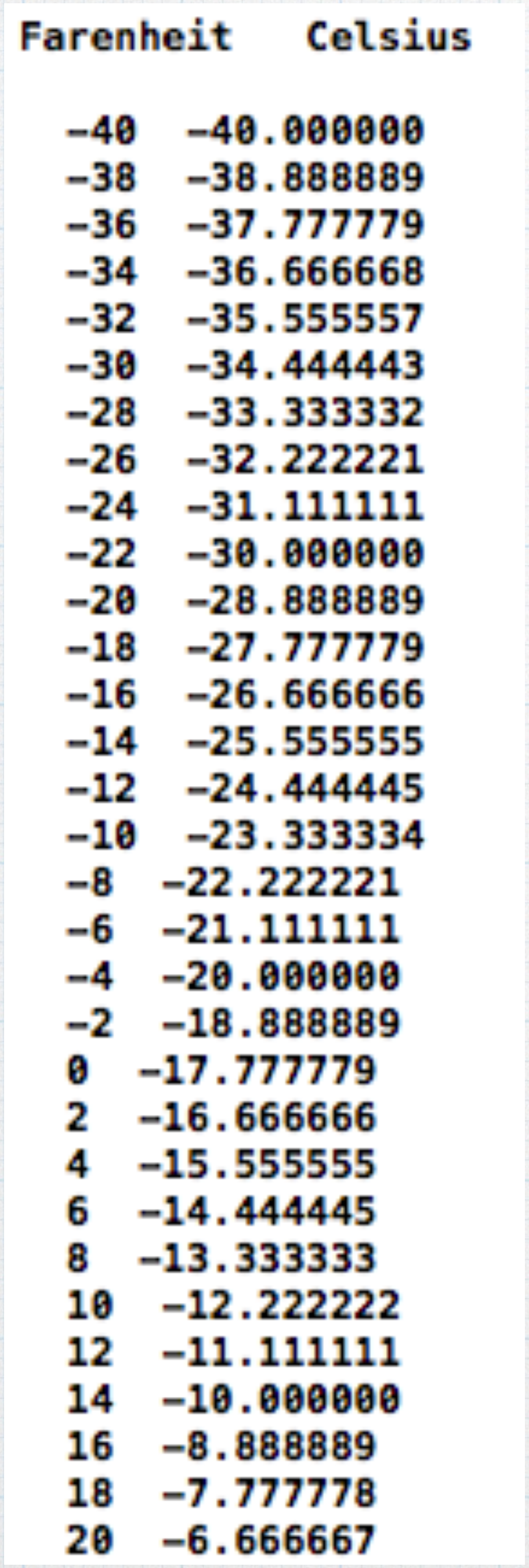
It has the same effect as `i=i-1;`

Something more practical.

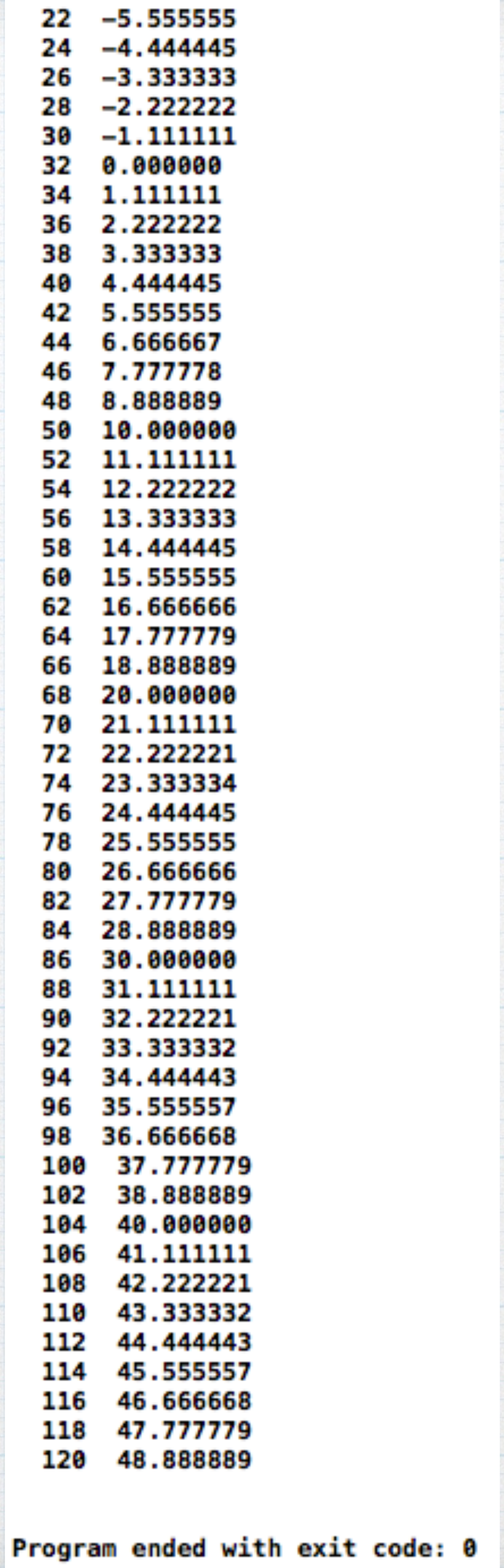
Print out a table of Fahrenheit to Celsius temperature conversions.



```c
#include <stdio.h>

int main(void) {

    int i = -40;
    float degrees_C;

    printf( "Fahrenheit   Celsius\n\n");

    while( i <= 120 ){

        degrees_C = 5.0/9.0*(i - 32.0);   //int is automatically converted to float
        printf( "  %d  %f \n", i, degrees_C);

        i += 2;   //increment by 2.  This is the same as i = i + 2

    }

    printf( "\n\n");    //Throw in a couple of line returns, just to clean things up.

    return 0;
}
```



```
Farenheit   Celsius

  -40   -40.000000
  -38   -38.888889
  -36   -37.777779
  -34   -36.666668
  -32   -35.555557
  -30   -34.444443
  -28   -33.333332
  -26   -32.222221
  -24   -31.111111
  -22   -30.000000
  -20   -28.888889
  -18   -27.777779
  -16   -26.666666
  -14   -25.555555
  -12   -24.444445
  -10   -23.333334
  -8  -22.222221
  -6  -21.111111
  -4  -20.000000
  -2  -18.888889
  0  -17.777779
  2  -16.666666
  4  -15.555555
  6  -14.444445
  8  -13.333333
  10  -12.222222
  12  -11.111111
  14  -10.000000
  16  -8.888889
  18  -7.777778
  20  -6.666667
```



```
22  -5.555555
24  -4.444445
26  -3.333333
28  -2.222222
30  -1.111111
32  0.000000
34  1.111111
36  2.222222
38  3.333333
40  4.444445
42  5.555555
44  6.666667
46  7.777778
48  8.888889
50  10.000000
52  11.111111
54  12.222222
56  13.333333
58  14.444445
60  15.555555
62  16.666666
64  17.777779
66  18.888889
68  20.000000
70  21.111111
72  22.222221
74  23.333334
76  24.444445
78  25.555555
80  26.666666
82  27.777779
84  28.888889
86  30.000000
88  31.111111
90  32.222221
92  33.333332
94  34.444443
96  35.555557
98  36.666668
100  37.777779
102  38.888889
104  40.000000
106  41.111111
108  42.222221
110  43.333332
112  44.444443
114  45.555557
116  46.666668
118  47.777779
120  48.888889


Program ended with exit code: 0
```

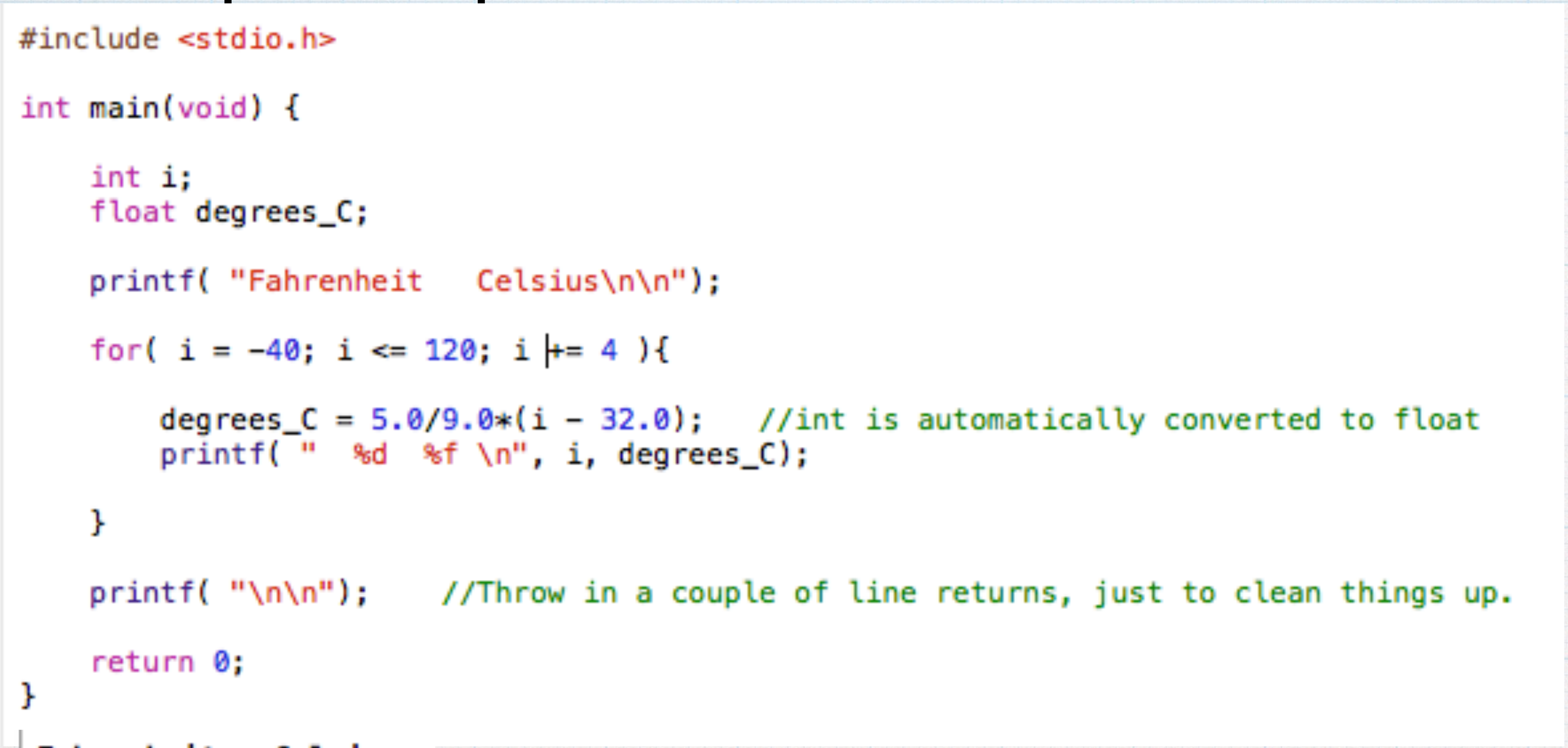
For Loops

While loops are very general, and we could probably do everything we need with them. However, when we are simply counting through a set number of loops, we can use a short-hand notation for the while that combines the counter initialization, the conditional statement, and the counter increment in one statement. This is known as a "For" loop. Using For loops helps cut down on mistakes of forgetting to initialize or increment the counter.

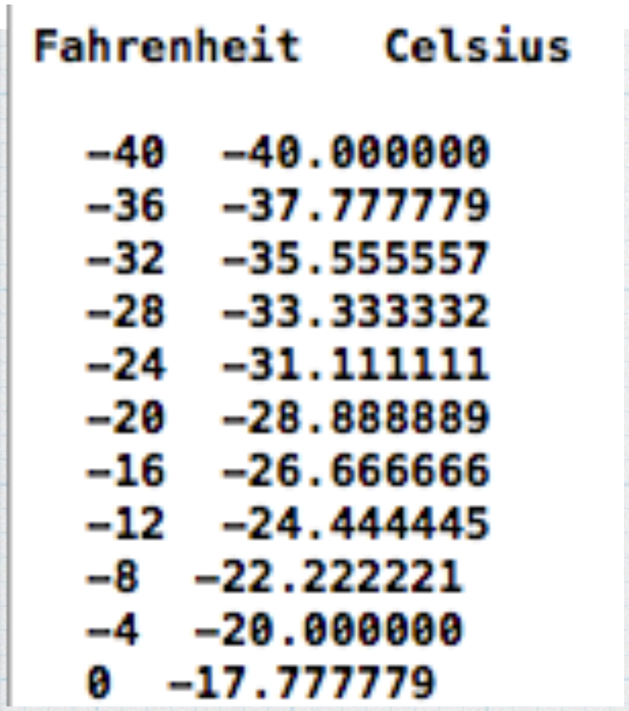
The basic program structure:

```
int i;  \\ the "counter"
For ( initialize i; conditional; incrment i) {
    Do something here inside the loop;
}
```

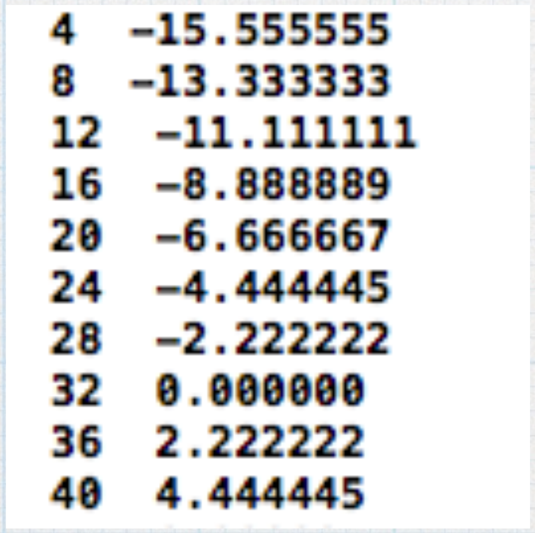
For loop - example



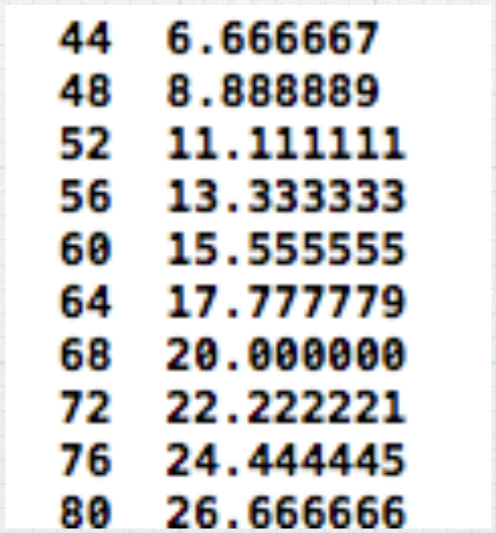
```c
#include <stdio.h>

int main(void) {

    int i;
    float degrees_C;

    printf( "Fahrenheit   Celsius\n\n");

    for( i = -40; i <= 120; i += 4 ){

        degrees_C = 5.0/9.0*(i - 32.0);   //int is automatically converted to float
        printf( "  %d  %f \n", i, degrees_C);

    }

    printf( "\n\n");    //Throw in a couple of line returns, just to clean things up.

    return 0;
}
```



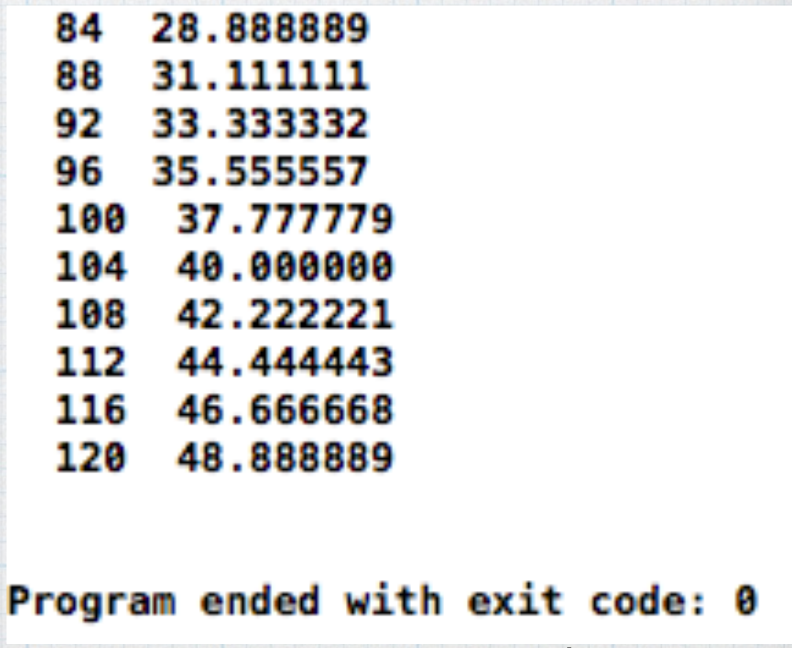
```
Fahrenheit   Celsius

  -40  -40.000000
  -36  -37.777779
  -32  -35.555557
  -28  -33.333332
  -24  -31.111111
  -20  -28.888889
  -16  -26.666666
  -12  -24.444445
  -8  -22.222221
  -4  -20.000000
  0  -17.777779
```



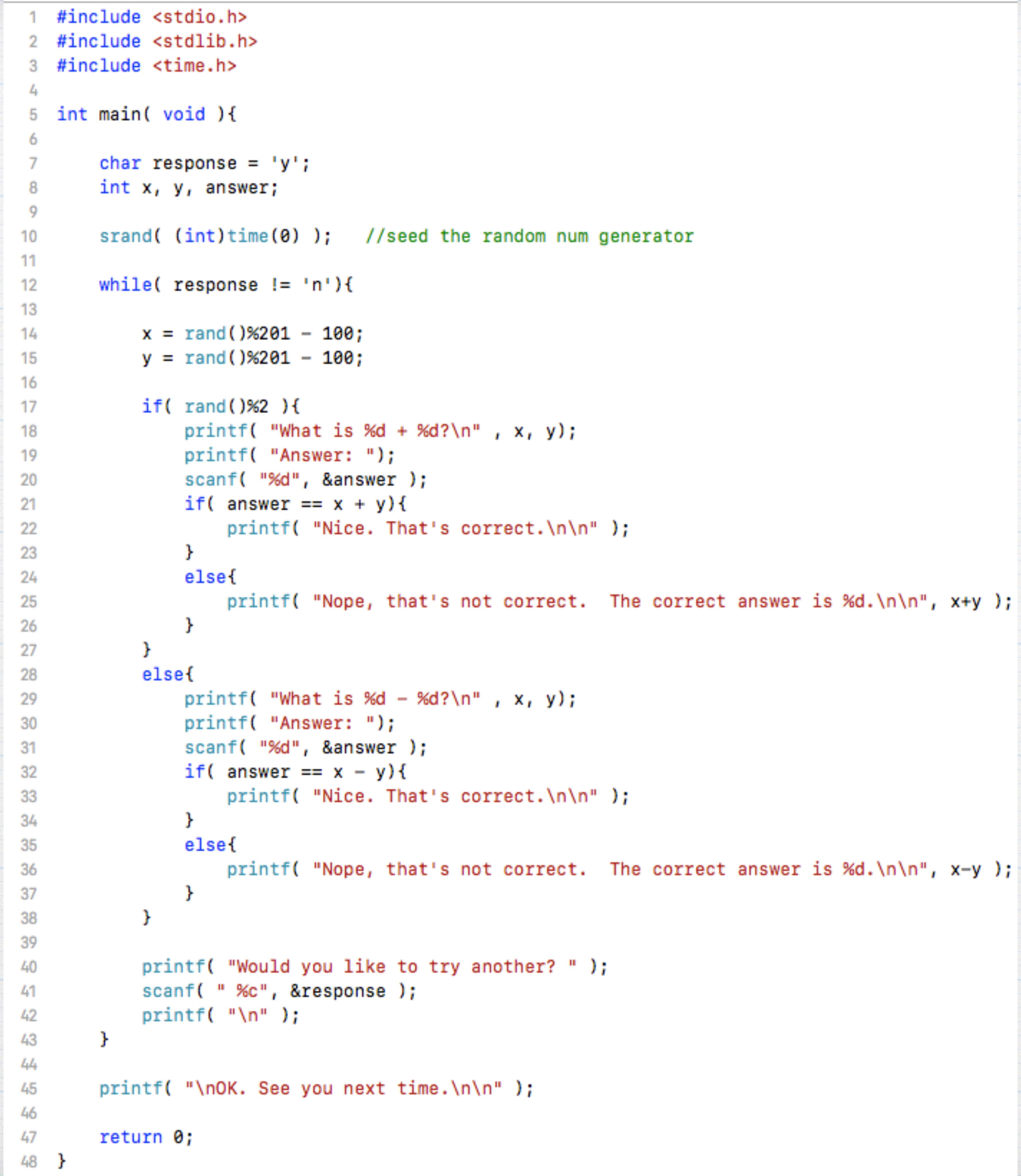
```
  4  -15.555555
  8  -13.333333
  12  -11.111111
  16  -8.888889
  20  -6.666667
  24  -4.444445
  28  -2.222222
  32  0.000000
  36  2.222222
  40  4.444445
```



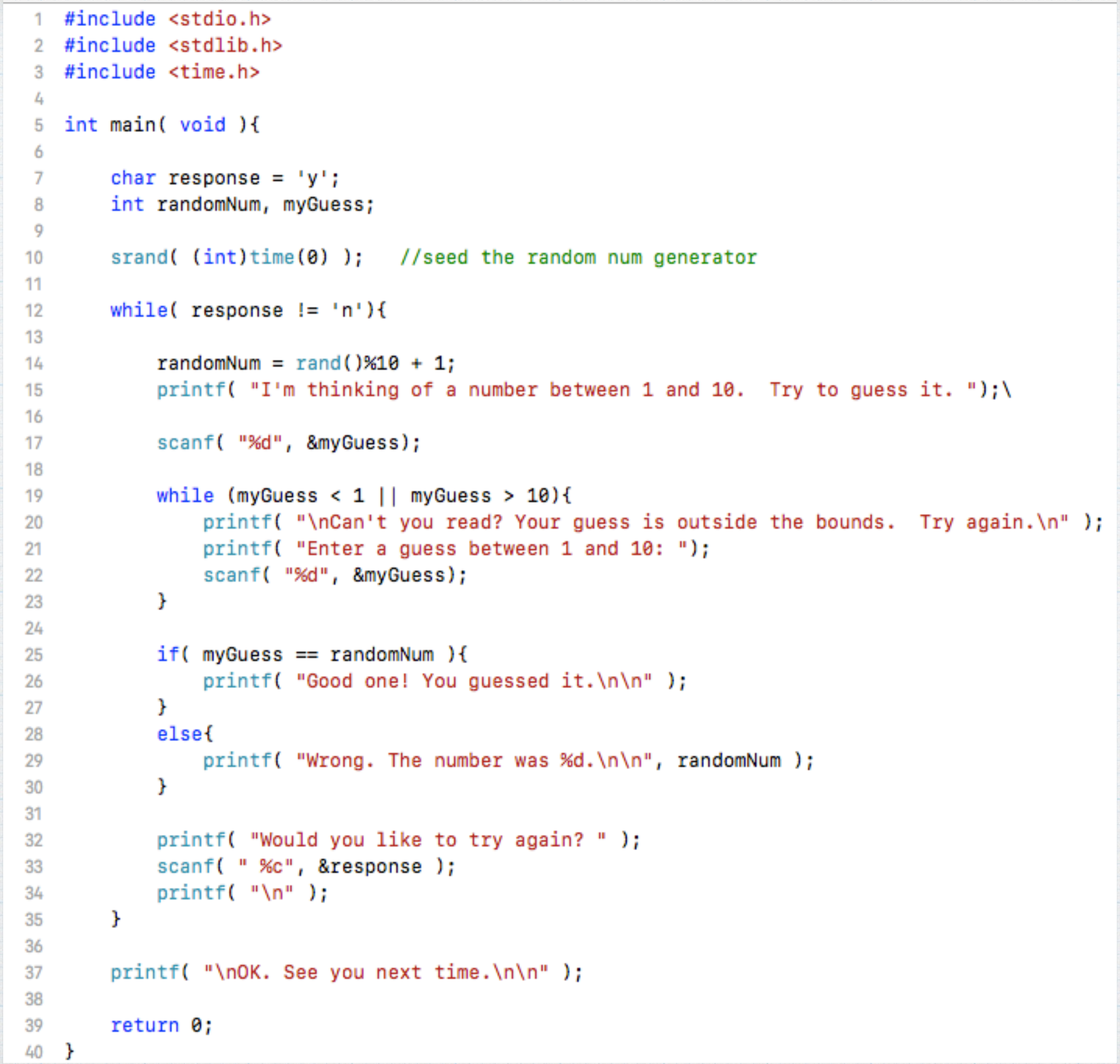
```
  44  6.666667
  48  8.888889
  52  11.111111
  56  13.333333
  60  15.555555
  64  17.777779
  68  20.000000
  72  22.222221
  76  24.444445
  80  26.666666
```



```
  84  28.888889
  88  31.111111
  92  33.333332
  96  35.555557
  100  37.777779
  104  40.000000
  108  42.222221
  112  44.444443
  116  46.666668
  120  48.888889


Program ended with exit code: 0
```



```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main( void ){

    char response = 'y';
    int x, y, answer;

    srand( (int)time(0) );   //seed the random num generator

    while( response != 'n'){

        x = rand()%201 - 100;
        y = rand()%201 - 100;

        if( rand()%2 ){
            printf( "What is %d + %d?\n" , x, y);
            printf( "Answer: ");
            scanf( "%d", &answer );
            if( answer == x + y){
                printf( "Nice. That's correct.\n\n" );
            }
            else{
                printf( "Nope, that's not correct.  The correct answer is %d.\n\n", x+y );
            }
        }
        else{
            printf( "What is %d - %d?\n" , x, y);
            printf( "Answer: ");
            scanf( "%d", &answer );
            if( answer == x - y){
                printf( "Nice. That's correct.\n\n" );
            }
            else{
                printf( "Nope, that's not correct.  The correct answer is %d.\n\n", x-y );
            }
        }

        printf( "Would you like to try another? " );
        scanf( " %c", &response );
        printf( "\n" );
    }

    printf( "\nOK. See you next time.\n\n" );

    return 0;
}
```



```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main( void ){

    char response = 'y';
    int randomNum, myGuess;

    srand( (int)time(0) );   //seed the random num generator

    while( response != 'n'){

        randomNum = rand()%10 + 1;
        printf( "I'm thinking of a number between 1 and 10.  Try to guess it. ");\

        scanf( "%d", &myGuess);

        while (myGuess < 1 || myGuess > 10){
            printf( "\nCan't you read? Your guess is outside the bounds.  Try again.\n" );
            printf( "Enter a guess between 1 and 10: ");
            scanf( "%d", &myGuess);
        }

        if( myGuess == randomNum ){
            printf( "Good one! You guessed it.\n\n" );
        }
        else{
            printf( "Wrong. The number was %d.\n\n", randomNum );
        }

        printf( "Would you like to try again? " );
        scanf( " %c", &response );
        printf( "\n" );
    }

    printf( "\nOK. See you next time.\n\n" );

    return 0;
}
```