Loops

Domhnall O'Hanlon

February 24, 2015

Part 3: Loops

Overview

We've already seen a few instances where we wanted to repeatedly print information to the screen. Performing repetitive tasks is one of the things that computers are exceptionally good at doing. In this chapter we'll introduce a variety of different loops that you can use to write better programs.

Looping

If you've never done any programming before loops can be a challenging topic due to their unfamiliarity. If you think about it in more general terms, how would you give a computer instruction to do something over and over again? How would you avoid getting stuck in an *infinite loop*?

Start, Middle, End

When you creating a loop in any programming language you will have tell it when and where to start. Starting a loop is known as **initialising** the loop.

Next you will have some code to run - for example increment at counter, print some text etc. - while the loop is running.

Finally you need an end condition. Once this condition is met or exceeded then the loop should exit gracefully.

While Loops

Syntax

A typical while loop will begin with a test condition:

while(testIfTrue){  
 code to run while true;  
 incrementer;  
}

If the test is true then the code inside the loop (i.e between the braces) will run - keep in mind that each line must end with a semi-colon. Finally, you'll need to have some sort of incrementer that gets updated during each pass through the loop. This is essential so that you don't get stuck in an infinte loop.

Example

Try the previous example with different test conditions. For example using while (counter <= 30) will cause the loop to run an extra time. Similarly you could continue executing the loop while the counter is *not* equal to a certain value. while (counter != 30)

Be careful if you are changing the direction of the inequality!

Visualisation

while loop

Programming Challenge

Here's a simple example you can code using while loops. Which would you prefer, one million euro today or 1 cent, doubled every day for a month (30 days)?

Your loop should run 30 times, doubling the value of your variable each time.

Do While

With a while loop there is always a possibility that the test condition will never be true and that the code within the loop will never run.

A do while loop differs from a while loop in that it will always run at least once.

Do While Syntax

int a = 10;  
  
 /\* do loop execution \*/  
 do  
 {  
 printf("value of a: %d\n", a);  
 a = a + 1;  
 }while( a < 20 );

Visualisation

Do While loop

For Loops

The for loop contains the starting condition, end condition and incrementer all at the beginning of the loop

Syntax

int main(){  
   
 int counter;  
  
 for(counter = 0; counter < 10; counter++){  
 printf("Hello World!");  
 }  
   
 return 0;  
 }

Visualisation

for loop

Example

/\* Print all the even numbers between 0 and 100 in 3 lines of code \*/  
 int main(){  
   
 int i;  
  
 for(i = 0; i <=100; i++){  
 if(i%2 == 0){  
 printf("%d \n", i);  
 }  
 }  
   
 return 0;  
 }

Nesting For loops

A quick challenge to really test your understanding so far.

Create a program that has variables to represent the number of rows and number of colums that a table should have. Then use nested for loop to print a 3 x 3 table to the console.

Break

break; break out of the loop right now.

Continue

stop where you are and go back to the start

Switch

#### Grade example (revisited?)