Loops In Pascal

In this section of notes you will learn how to rerun parts of your program without having to duplicate your code.

Basic Structure Of Loops

1) Initialize the control
   a) Control – typically a variable that determines whether or not the loop executes or not.
2) Testing the control against a condition
3) Executing the body of the loop
4) Update the value of the control

The Need For Repetition (Loops)

Writing out a simple counting program (1 – 3).

```
program counting (output);
begin
   writeln('1');
   writeln('2');
   writeln('3');
end.
```

Types Of Loops

Pre-test loops

1. Initialize control
2. Check if a condition is met (using the control in some Boolean expression)
   a) If the condition has been met then continue on with the loop (go to step 3)
   b) If the condition is not met then break out of the loop (loop ends)
3. Execute the body of the loop
4. Update the value of the control
5. Repeat step 2

General characteristics

- The body of the loop executes zero or more times
- Execute body only if the condition is true (stop executing when it becomes false)
- Examples: while-do, for

The Need For Repetition (2)

Simple program but what if changes need to be made?

- The source code must be re-edited and re-compiled each time that a change is needed.

What if you need the program to count many times?

Types Of Loops (2)

Post-test loops

1. Initialize control (often this step is unneeded)
2. Execute the body of the loop
3. Update the value of the control
4. Check if a condition is met (using the control in some Boolean expression)
   a) If the condition has been met then break out of loop (loop ends)
   b) If the condition hasn't been met then continue on with loop (go to step 2)

General characteristics

- The body of the loop executes one or more times
- Execute body only if condition is false (stop executing when it's true)
- Examples: repeat-until

Pre-Test Loop: While-Do

Can be used if the number of times that the loop must execute is not known in advance.

Format:

```
while (Boolean expression) do
    body
```

Example (The full program can be found in Unix under /home/231/examples/repetition/whileDo.p)

```
  i: = 1;
  while (i <= 5) do
  begin
      writeln('i = ', i);
      i := i + 1;
end; (* while *)
```

Pre-Test Loop: While-Do

Can be used if the number of times that the loop executes is not known in advance.

Format:

```
while (Boolean expression) do
    body
```

Example (The full program can be found in Unix under /home/231/examples/repetition/whileDo.p)

```
  i: = 1;                  <----  1) Initialize control
  while (i <= 5) do        <----  2) Check condition
  begin
      writeln('i = ', i);  }
      i := i + 1;          } <--  3) Execute body
end; (* while *)
                                  4) Update control  (i := i + 1)
```

Tracing The While Loop

Variables	Execution
i	./a.out

Pre-Test Loop: For

Typically used when it is known in advance how many times that the loop will execute (counting loops).

Format (counting up):

```
for initialize control to final value do
    body
```

Format (counting down):

```
for initialize control downto final value do
    body
```

First For Loop Example

Example one (The full program can be found in Unix under /home/231/examples/repetition/forLoopUp.p)

```
begin
  var i     : integer;
  var total : integer;
  total := 0;
  for i := 1 to 5 do
  begin
    total := total + i;
    writeln('i=', i, 'total=', total);
  end; (* for *)
end.
```

First For Loop Example

Example one (The full program can be found in Unix under /home/231/examples/repetition/forLoopUp.p)

```
begin                                     1) Initialize control  (i := 1)
  var i     :  integer;                   3) Update control      (to)
  var total : integer;
  total := 0;
  for i := 1 to 5 do
  begin                                   2) Test condition      (5)
    total := total + i;                 }
    writeln('i=', i, ' total=', total); } <-- 4) Execute body
  end; (* for *)
end.
```

Tracing The First For Loop Example

```
        Variables                        Execution
i                 total                   ./a.out
```

Second For Loop Example

Example one (The full program can be found in Unix under /home/231/examples/repetition/forLoopDown.p)

```
begin
  var i       : integer;
  var total : integer;
  total := 0;
  for i := 5 downto 1 do
  begin
    total := total + i;
    writeln('i=', i, ' total=',total);
  end; (* for *)
end.
```

Tracing The Second For Loop Example

```
        Variables                        Execution
i                 total                   ./a.out
```

Post Test Loops: Repeat-Until

Used instead of a while-do loop if you need the loop to execute the loop at least once.

Format:

```
repeat
   body
until (Boolean expression);
```

Repeat-Until: An Example

The full version can be found in Unix under: /home/231/examples/repetition/guzzlingGame.p

Repeat-Until: An Example (2)

```
repeat
   answer := random(10);
   write('Enter your guess: ');
   readln(guess);
   if (guess = answer) then
     writeln('You guessed correctly!')
   else
     writeln('You guessed incorrectly');
   writeln('Number was ', answer, ', your guess was ', guess);
   write('Play again?  Enter "N" or "n" to quit or anything else to ');
   writeln('continue');
   write('Choice: ');
   readln(choice);
   writeln;
until (choice = 'N') OR (choice = 'n');
```

Repeat-Until: An Example (2)

```
repeat
    answer := Random(10);
    write('Enter your guess: ');
    readln(guess);
    if (guess = answer) then
       writeln('You guessed correctly!')
    else
       writeln('You guessed incorrectly');
    writeln('Number was ', answer, ', your guess was ', guess);
    write('Play again?  Enter "N" or "n" to quit or anything else to ');
    writeln('continue');
    write('Choice: ');
    readln(choice);
    writeln;
until (choice = 'N') OR (choice = 'n');
```

1) Execute body

2) Update control

3) Test condition

Infinite Loops

Infinite loops never end (the stopping condition is never met).

They can be caused by logical errors:

- The loop control is never updated (Example 1 – below).
- The updating of the loop control never brings it closer to the stopping condition (Example 2 – next slide).

Example 1 (The full version can be found in Unix under /home/231/examples/repetition/infinite1.p)

```
i := 1;
while (i <=10) do
    writeln('i=', i);
i := i + 1;
```

To stop a program with an infinite loop in Unix simultaneously press the <ctrl> and the <c> keys

Infinite Loops (2)

Example 2 (The full version can be found in Unix under /home/231/examples/repetition/infinite2.p)

```
i := 10;
while (i > 0) do
begin
     writeln('i = ', i);
     i := i + 1;
end;
```

To stop a program with an infinite loop in Unix simultaneously press the <ctrl> and the <c> keys

Nested Loops

One loop executes inside of another loop(s).

Example structure:

```
Outer loop (runs n times)
    Inner loop (runs m times)
        Body of inner loop (runs n x m times)
```

Example program (the full program can be found in Unix under: /home/231/examples/repetition/nested.p)

```
for i := 1 to 2 do
    for j := 1 to 3 do
        writeln('i=', i, ' j=', j);
  writeln('All done!');
```

Testing Loops

Make sure that the loop executes the proper number of times.

Test conditions:

1) Loop does not run
2) Loop runs exactly once
3) Loop runs exactly "n" times

Testing Loops: An Example

```
program testLoops (input, output);
begin
  var sum  : integer;
  var i      : integer;
  var last   : integer;
  sum := 0;
  i := 1;
  write('Enter the last number in the sequence to sum : ');
  readln(last);
```

Testing Loops: An Example (2)

```
while (i <= last) do
  begin
    sum := sum + i;
    writeln('i=', i);
    i := i + 1;
  end;
  writeln('sum=', sum);
end.
```

You Should Now Know

When and why are loops used in computer programs

What is the difference between pre-test loops and post-test loops

How to trace the execution of pre and post-test loops

How to write the code for a loop in a program

What are nested loops and how do you trace their execution

How to test the execution of loop