Loops In Pascal

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

The Need For Repetition

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

The full text-only program can be found in Unix under /home/231/examples/repetition/counting.p:

```
program counting (output);

begin

  writeln('1');

  writeln('2');

  writeln('3');

end.
```

The Need For Repetition (2)

Simple program but what if changes need to be made?

- Need to re-edit source code and re-compile program?

What if you need the program to count many times?

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

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 hasn't been 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 condition is true
- Examples: while-do, for

Types Of Loops (2)

Post-test loops

1. Initialize control
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
- Examples: repeat-until

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 (Full text-only version 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 (Full text-only version 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 *)
```

1) Initialize control

2) Check condition

3) Execute body

4) Update control

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).

Syntax (counting up):

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

Syntax (counting down):

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

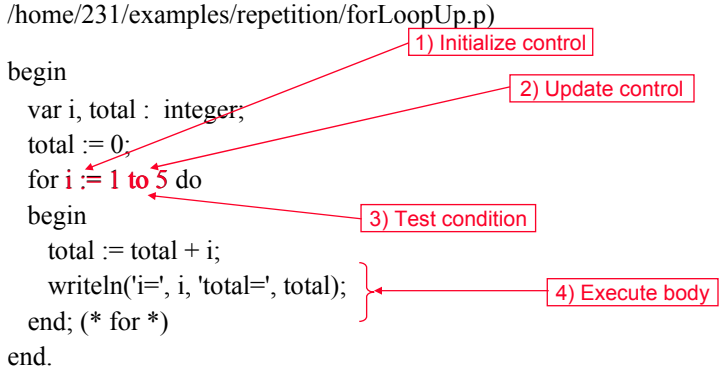
First For Loop Example

Example one (A compilable text-only version can be found in Unix under /home/231/examples/repetition/forLoopUp.p)

```
begin
  var i, 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 (A compilable text-only version can be found in Unix under /home/231/examples/repetition/forLoopUp.p)




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

Tracing The First For Loop Example

Variables		Execution
i	total	./a.out

Second For Loop Example

Example one (A compilable text-only version can be found in Unix under /home/231/examples/repetition/forLoopDown.p)

```
begin
  var i, 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 at least once.

Syntax:

repeat

    body

until (*Boolean expression*);

Repeat-Until: An Example

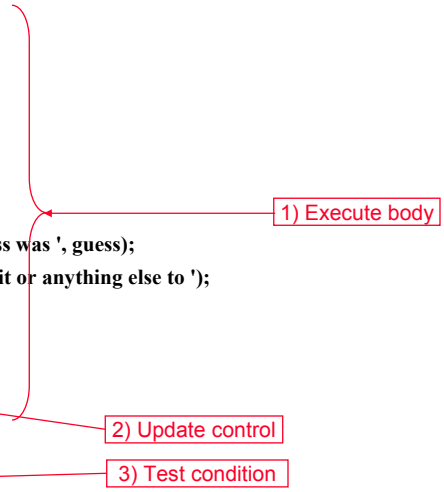
A compilable version of this example can be found in Unix under: /home/231/examples/repetition/guzzlingGame.p

Repeat-Until: An Example (2)

```
repeat
begin
   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;
end;
until (choice = 'N') OR (choice = 'n');
```

Repeat-Until: An Example (2)

```
repeat
begin
   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;
end;
until (choice = 'N') OR (choice = 'n');
```



Infinite Loops

Loops that never end (the stopping condition is never met).

Infinite loops can be caused by logical errors:

•The loop control is never updated (Example 1).

•The updating of the loop control never brings it closer to the stopping condition (Example 2).

Example 1 (a text-only 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 (a text-only 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 (complete text-only 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!');
```

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

What are nested loops and how do you trace their execution