

Repetition In Pascal

In this section of notes you will learn how to have a section of code repeated without duplicating the 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 to count many times?

This example indicates the need for a mechanism to allow for parts of the program to repeat an arbitrary number of times (loops).

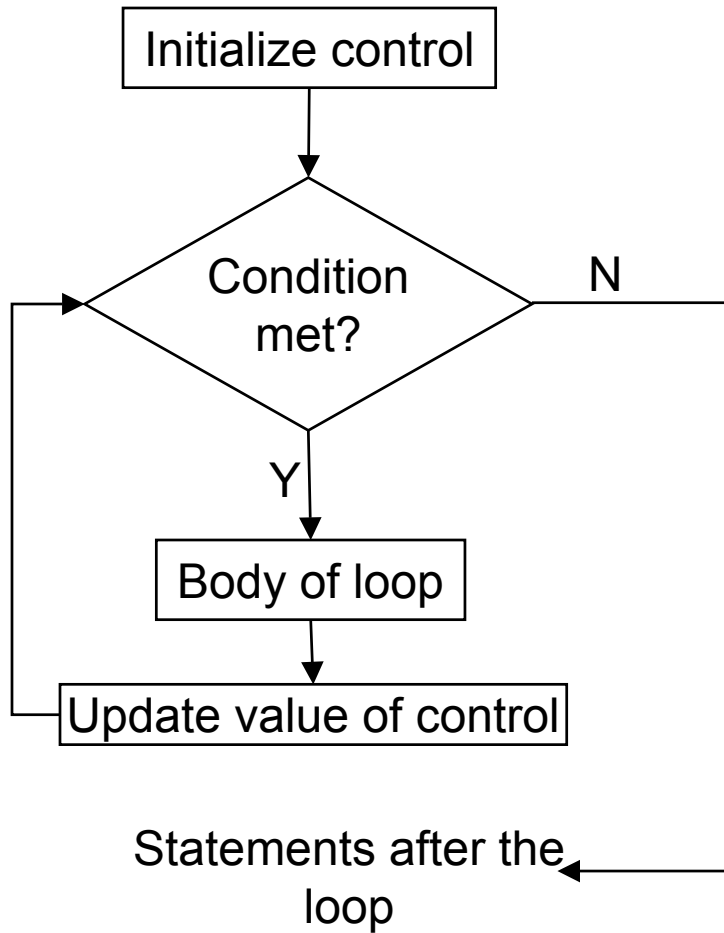
Basic Structure Of Loops

- 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

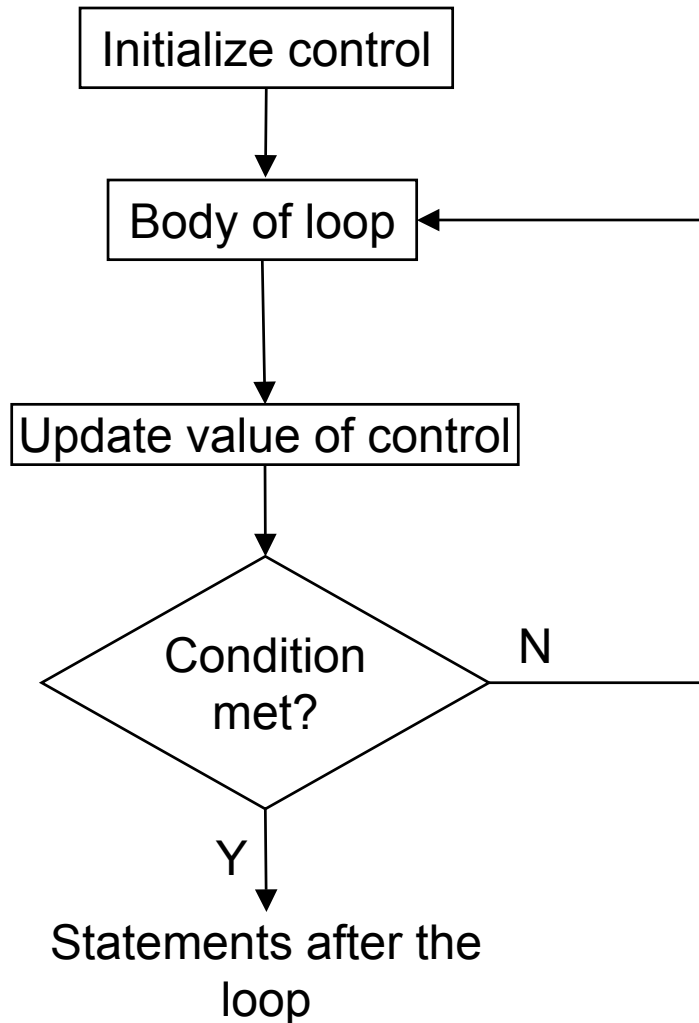
Types Of Loops

- 1) Pre-test
 - a) Initialize control
 - b) Check if a condition is met (using the control in some Boolean expression)
 - c) If the condition has been met then execute the body of the loop, otherwise the loop ends (update control in body – if applicable)
 - d) Repeat step (b)
 - The body of the loop executes zero or more times
 - Execute body only if condition is true
 - Examples: while-do, for
- 2) Post-test
 - a) Initialize control
 - b) Execute the body of the loop (update value of control in body – if applicable)
 - c) If the condition has been met (using the control in some Boolean expression) then skip otherwise the loop repeats.
 - The body of the loop executes one or more times
 - Execute body only if condition is false
 - Example: repeat-until

Pre-Test Loop: Flowchart (While-do, For)



Post-Test Loop: Flowchart (Repeat-Until)



Pre-Test Loop: While-Do

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

Syntax:

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

Example (Full text-only version can be found in Unix under /home/231/examples/repetition/whileDo.p)

```
i := 1; ← Initialize control  
while (i <= 10) do ← Test condition  
begin  
    writeln('i = ', i);  
    i := i + 1; ← Execute body  
end; (* while *) ← Update control
```

The diagram illustrates the execution flow of a while-do loop. Red arrows point from text boxes to specific parts of the code: 'Initialize control' points to 'i := 1;', 'Test condition' points to 'while (i <= 10) do', 'Execute body' points to the loop body (indicated by a bracket), and 'Update control' points to 'i := i + 1;'. The final 'end; (* while *)' line is not annotated.

Tracing The While Loop

Variables

i

Execution

./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)

```
var
  i, total : integer;
begin
  total := 0;
  for i := 1 to 10 do
  begin
    total := total + i;
    writeln('i = ':4, i:2, 'total = ':13, total:2);
  end; (* for *)
end.
```

The diagram illustrates the execution flow of the for loop. Red arrows point from text boxes to specific parts of the code:

- Initialize control** points to `total := 0;`
- Update control** points to `10` in `for i := 1 to 10 do`
- Test condition** points to `do` in `for i := 1 to 10 do`
- Execute body** points to the body of the loop (the `begin` block), indicated by a bracket on the right side of the loop body.

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)

```
var
  i, total : integer;
begin
  total := 0;
  for i := 5 downto 1 do
  begin
    total := total + i;
    writeln('i = ':4, i:2, 'total = ':13, total:2);
  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 := trunc(Random * MAX);  
    write('Enter your guess: ');  
    readln(guess);  
    if (guess = answer) then  
        writeln('You guessed correctly!')  
    else  
        writeln('You guessed incorrectly');  
        writeln('Play again?');  
        writeln('Enter "Y" or "y" to play again');  
        writeln('Enter "N" or "n" otherwise');  
        write('Choice: ');  
        readln(choice);
```

end;

until (choice = 'N') OR (choice = 'n');

Test condition

Execute body

Initialize control

Update control

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 compilable version can be found in Unix under /home/231/examples/repetition/infinite1.p)

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

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

Infinite Loops (2)

Example 2 (a compilable 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 compilable 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 = ':9, i, 'j = ':9, j);
```

```
writeln('All done!');
```

Summary

What is the purpose of loops in a programming language?

What are the different types of loops that are employed in Pascal

- Pre-test: while-do, for
- Post-test: repeat-until

What are nested loops and how do they work