

Loops In Pascal

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

James Tam

The Need For Repetition (Loops)

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

```
program counting (output);
```

```
begin
```

```
  writeln('1');
```

```
  writeln('2');
```

```
  writeln('3');
```

```
end.
```

James Tam

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?

James Tam

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

James Tam

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 the body only if the condition is true (stop executing when it becomes false)
- Examples: while-do, for

James Tam

Types Of Loops (2)

Post-test loops

1. Initialize control (often this step is unneeded because the control is set in the body, step 3)
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 the body only if condition is false (stop executing when it's true)
- Examples: repeat-until

James Tam

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/tamj/examples/loops/whileDo.p)

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

James Tam

Pre-Test Loop: While-Do

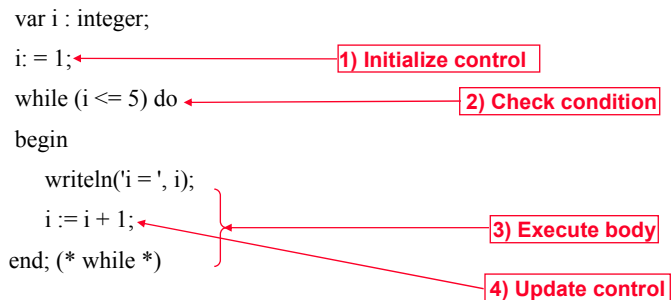
Can be used for almost any stopping condition

Format:

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

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

```
var i : integer;  
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
```



James Tam

Tracing The While Loop

Execution	Variables
>./whileDo	i

James Tam

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
```

James Tam

First For Loop Example

Example one (The full program can be found in Unix under /home/231/tamj/examples/loops/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.
```

James Tam

First For Loop Example

Example one (The full program can be found in Unix under /home/231/tamj/examples/loops/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.
```

The diagram illustrates the four steps of a for loop:

- 1) Initialize control**: Points to the initialization of the loop counter `i := 1`.
- 2) Test condition**: Points to the `to 5` part of the `for` statement, indicating the loop's range.
- 3) Update control**: Points to the `do` keyword, indicating the end of the loop's range.
- 4) Execute body**: Points to the `begin` block of the loop, indicating the code that is executed repeatedly.

James Tam

Tracing The First For Loop Example

Execution

>./ forLoopUp

Variables

i total

James Tam

Second For Loop Example

Example two (The full program can be found in Unix under /home/231/tamj/examples/loops/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.
```

James Tam

Tracing The Second For Loop Example

Execution

> ./forLoopDown

Variables

i total

James Tam

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*);

James Tam

Repeat-Until: An Example

Example:

The full version can be found in Unix under:
/home/231/tamj/examples/loops/repeatUntil.p

James Tam

Repeat-Until: An Example (2)

```
program repeatUntil (output);
begin
  var i : integer;
  i:= 1;
  repeat
  begin
    writeln('i =', i);
    i := i + 1;
  end; (* loop *)
  until (i > 5);
end.
```

James Tam

Repeat-Until: An Example (2)

```
program repeatUntil (output);
begin
  var i : integer;
  i := 1;
  repeat
  begin
    writeln('i = ', i);
    i := i + 1;
  end; (* loop *)
until (i > 5);
end.
```

1) Initialize control

2) Execute body

3) Update control

4) Test condition

James Tam

Tracing Repeat-Until Loop Example

Execution	Variable
>./ repeatUntil	i

James Tam

Solving A Problem Using Loops

Write a program that will execute a game:

- The program will randomly generate a number between one and ten.
- The player will be prompted to enter their guess.
- The program will continue the game until the player indicates that they no longer want to continue.

The full program can be found in UNIX under:
/home/231/examples/loops/guessingGame.p

James Tam

Repeat-Until: An Example (2)

```
var guess : integer;
var answer : integer;
var choice : char;

repeat
  answer := random(10) + 1;
  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" to quit or anything else to continue');
  write('Choice: ');
  readln(choice);
  writeln;
until (choice = 'N') OR (choice = 'n');
```

James Tam

Recap: What Looping Constructs Are Available In Pascal/When To Use Them

Construct	When To Use
Pre-test loops	You want the stopping condition to be checked before the loop body is executed (typically used when you want a loop to execute zero or more times).
• While-do	• The most powerful looping construct: you can write a 'while-do' loop to mimic the behavior of any other type of loop. In general it should be used when you want a pre-test loop which can be used for most any arbitrary stopping condition e.g., execute the loop as long as the user doesn't enter a negative number.
• For	• A 'counting loop': You want a simple loop to count up or down a certain number of times.
Post-test: Repeat-until	You want to execute the body of the loop before checking the stopping condition (typically used to ensure that the body of the loop will execute at least once).

James Tam

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/tamj/examples/loops/infinite1.p)

```
var i : integer;  
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

James Tam

Infinite Loops (2)

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

```
var i : integer;
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

James Tam

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/tamj/examples/loops/nested.p)

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

James Tam

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

James Tam

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);
```

James Tam

Testing Loops: An Example (2)

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

James Tam

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 properly 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

James Tam