

## Definite Looping II:

This is a supplementary section of for-loop.

Real Flow Chart of for-loop:

```
For i := a to b do
statement1; { Loop Body}
```

```
For i := a to b do
begin
statement1; { Loop Body}
end;
```

In a **for-to-do** loop, the computer will do the followings.

- 1) Check whether  $a \leq b$ , if not, ignore the loop.
- 2) Then memorize the value of  $a$  and  $b$ , Assign  $a$  to  $i$ ,
- 3) Execute the whole Loop Body
- 4) Afterwards, if there is no any **Branching Statements** in the body, check whether  $i = b$ , if so the looping is finished.
- 5) Otherwise, increment  $i$  by 1 {i.e.  $i:=i+1$ ;} and go back to step (3)

In a **for-downto-do** loop, step (1) is check whether  $a \geq b$ , and step (5) decrement by 1.

p.s. Branching Statements will be taught in Intermediate Class.

Try to dry-run the following programs and write down the value of each variables after the execution.

```
program exe1;
var a,b,c,i,j:integer;
begin
a:=1; b:=5; c:=1;
j:=a*b;
for i:=a to b do begin
c:=c*b;
b:=b-1;
j:=j div i;
end;
end.
```

a	
b	
c	
i	
j	

```
program exe2;
var i, j, a, b:integer;
begin
a := 1;
for i:=1 to 4 do begin
b := 0;
for j:=1 to 3 do b := b + a;
a := b
end;
end.
```

a	
b	
i	
j	

```
program exe3;
var i:integer;
begin
for i := 3 to 9 do
for i := 4 downto 6 do
writeln('Hi!');
end.
```

i	
---	--

## Concept of Scope (範圍的概念):

So far, you have learnt conditional statements, iterative statements and sequence control. Some of them, like **while..do** , **for-to-do** , **if-then-else** , can only affect one statement. Unlike **repeat..until** , which can affect all statements bounded by the words **repeat** and **until**.

Scope is the word that describes the affecting power of those statements. In Pascal, all control structures, i.e. conditional, iterative... , must affect either one or all statements.

*Always remember that a statement is every thing between two semi-colons.*

One	Whole
If-then If-then-else Case-of-end While-do For-to-do	Begin-end; Begin-end. Repeat-until Case-of-else Procedure Function Var Type Record-end

Always use **begin-end;** to transfer one to whole.

```
If 1=1 then writeln(1); writeln(2);
```

The **if-then** statement has effect on writeln(1) only.

```
If 1=1 then begin writeln(1); writeln(2);end;
```

This **if-then** statement has effect on both writeln(1) and writeln(2).

```
If 1 <> 1 then writeln(1) else writeln(2);writeln(3);
```

The **then** statement affects writeln(1) and **else** affects writeln(2);  
But, writeln(3) is independent.

```
Repeat writeln(1);writeln(2);until 1 <> 1;
```

This **repeat-until** statement has effects on all statements between the word **repeat** and **until**.

## Multi-dimensional array:

In previous lesson, a new data structure called array is introduced.

In fact, it is a 1d array.

To declare 2d , 3d ... Nd array, we treat them as **array of array**.

### Syntax:

```
Var identifier : array [low1...up2] of array [low2...up2] of type;
```

Example :

```
Var a : array [1..5] of array [1..5] of integer;
```

This declare *a* as 2d array of integer, having 25 elements which are integers.

Like.


### Another syntax:

```
Var identifier: array [low1...up2, low2...up2] of type;
```

### Access an element:

```
A[1,2]
```

```
A[1][2]
```

*Notes:* In Pascal , “[“ is identical to “(.” While “]” is identical to “.)” .

**Exercise I:**

{Without using computer}

00hse.doc : 2-5 , 11 , 15 , 17 , 19 , 21 ,22 , 28 , 37

01hse.doc : 10 , 33-35 , 38

**Exercise II:**

{Check your answer with TPW if you find difficulties}

00hse.doc : 19 , 21 , 22 , 28

**Exercise III:**

{Use TPW to write a program to complete the task}

Ignore the lines

Input file : INPUT . TXT Output file : OUTPUT . TXT
--

Input: Screen input

Output: Screen output

<http://www.hkoi.org/2003/lightout.html>

**Exercise IV:**

{Use TPW to write a program to complete the task}

03fje.doc #4 Bridge