PASCAL: L'ITERAZIONE

TRATTO DA CAMAGNI-NIKOLASSY, CORSO DI INFORMATICA, VOL. 1, HOEPLI

Informatica

Ciclo a condizione iniziale: *while..do*

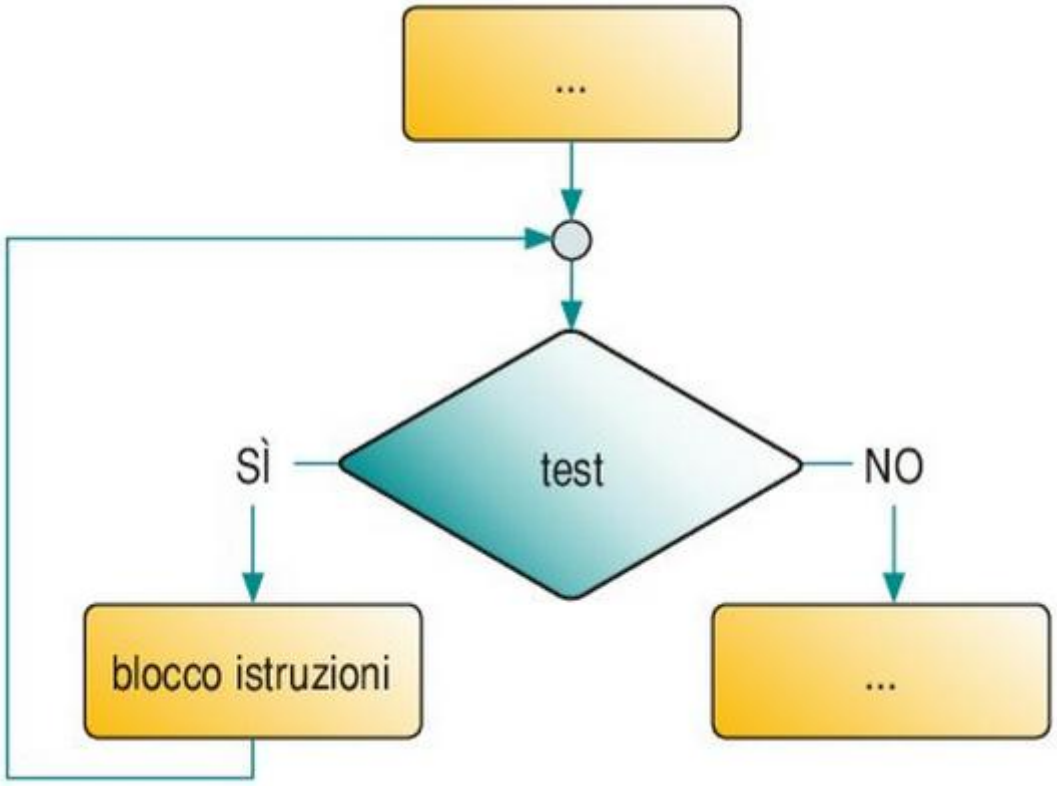
Ciclo a condizione iniziale: *while..do*

- Nella vita quotidiana ci sono delle azioni che vanno ripetute fino a che non si raggiunge un certo obiettivo. Per esempio:

  - versa l'acqua nel bicchiere finché è pieno;
  - aggiungi sale al risotto quanto basta;
  - lava la moto finché è pulita;
  - disponi i libri sulla libreria finché sono ordinati;
  - somma i numeri che leggi finché la somma è minore di 100;
  - … e via di seguito.

Ciclo a condizione iniziale: *while..do*

- Fin tanto che una certa condizione è verificata si esegue un certo blocco di istruzioni

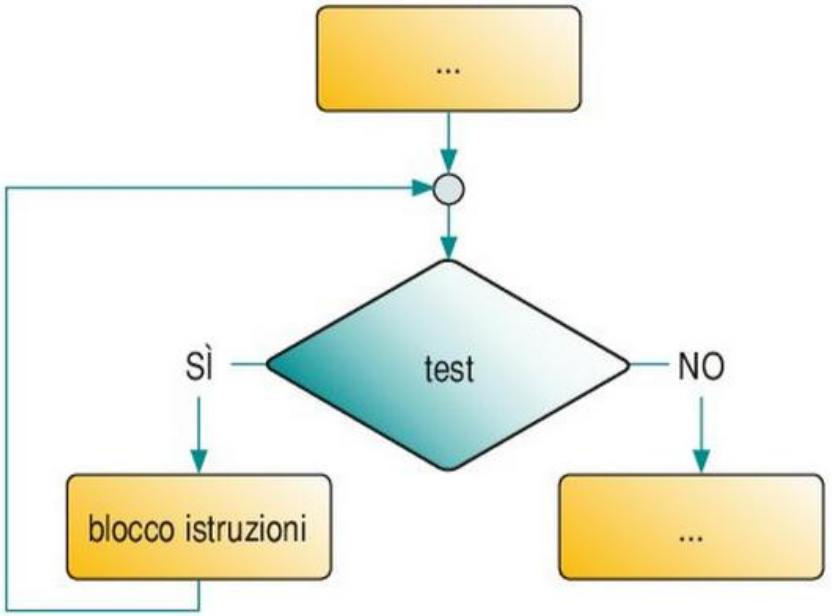


while..do in Pascal

```
while <condizione di ingresso> do
  <blocco di istruzioni>;
```

- viene valutata la condizione di ingresso, che è una operazione di test, e quindi ha risultato VERO o FALSO;
- quando il risultato è VERO si entra nel ciclo e si esegue il blocco di istruzioni (o semplicemente una istruzione);
- al termine dell'esecuzione del blocco si torna indietro, a ripetere il test, cioè si ripete l'esecuzione della condizione logica:
  - se l'esito è ancora VERO si ripete il ciclo;
  - se è FALSO, si esce dall'altro ramo e si prosegue il programma con le successive istruzioni.

while..do: schema a blocchi vs istruzioni



vs

```
while <condizione di ingresso> do
  <blocco di istruzioni>;
```

Esempio: Divisione di un numero per 2

- Letto un numero intero, dividiamolo per 2 mentre è maggiore di 2.

```
0  program divisodue;
1  var
2    numero:integer;
3  begin
4    write('inserisci un numero intero: ');
5    readln(numero);
6    while(numero>2)do
7      numero:=numero div 2;
8    write(valore finale del numero inserito: ');
9    writeln(numero);
10   readln;
11 end.
```

Verifica: Trace tabe

Numero istruzione	Istruzione di test	Numero
5		60
6	VERO	
7		60 `div` 2 = 30
6	VERO	
7		30 `div` 2 = 15
6	VERO	
7		15 `div` 2 = 7
6	VERO	
7		7 `div` 2 = 3
6	VERO	
7		3 `div` 2 = 1
2	FALSO	*(si esce dal ciclo)*
9		**1**

Esercizi 1-7, pag. 352

1. Scrivi un programma che legge un numero `num` e quindi successivamente esegue la somma di `num` numeri inseriti dall'utente.
2. Scrivi un programma che legge un numero `num` e visualizza tutti i numeri pari inferiori a tale numero.
3. Scrivi un programma che esegue la somma di tutti i numeri multipli di 5 compresi tra 10 e 100.
4. Scrivi un programma che legge un numero `num` e visualizza tutti i numeri primi inferiori a tale numero.
5. Scrivi un programma che legge una sequenza di numeri interi positivi terminanti con l'immissione del numero 0 e ne ricerca il valore minimo visualizzandolo sullo schermo.
6. Scrivi un programma che esegue la moltiplicazione di due numeri inseriti da un utente utilizzando il metodo delle somme successive.
7. Scrivi un programma che legge un numero `num` ed esegui il calcolo della somma dei primi `num` numeri interi positivi pari.

Esercizi 8-12, pag. 352

8 Scrivi un programma che legge un numero `num` e visualizza sullo schermo tutti i suoi fattori.

9 Scrivi un programma che legge una serie di numeri interi positivi arrestandosi quando la somma dei numeri immessi supera un valore costante letto come primo numero della sequenza.

10 Scrivi un programma che verifica se l'anno è bisestile: l'utente inserisce un anno e il calcolatore verifica se è bisestile. Se l'utente inserisce un numero minore di 0 il programma termina (senza ovviamente fare alcuna verifica); altrimenti, al termine della verifica, si ricomincia da capo (un anno è bisestile se è divisibile per 4 ma non per 100, oppure se è divisibile per 400).

11 Scrivi un programma che ricerca i primi tre numeri perfetti e li visualizza sullo schermo (un numero è perfetto se è uguale alla somma dei suoi divisori, per esempio il numero 6 è perfetto dato che 6 = 1 + 2 + 3).

12 Scrivi un programma che esegue il calcolo del fattoriale di un numero `num` inserito (il fattoriale di un numero si ottiene moltiplicando il numero per tutti i suoi predecessori: per esempio, il fattoriale di 5 è dato da 5*4*3*2*1 e si indica con 5!).

Conclusioni

ABBIAMO IMPARATO CHE...

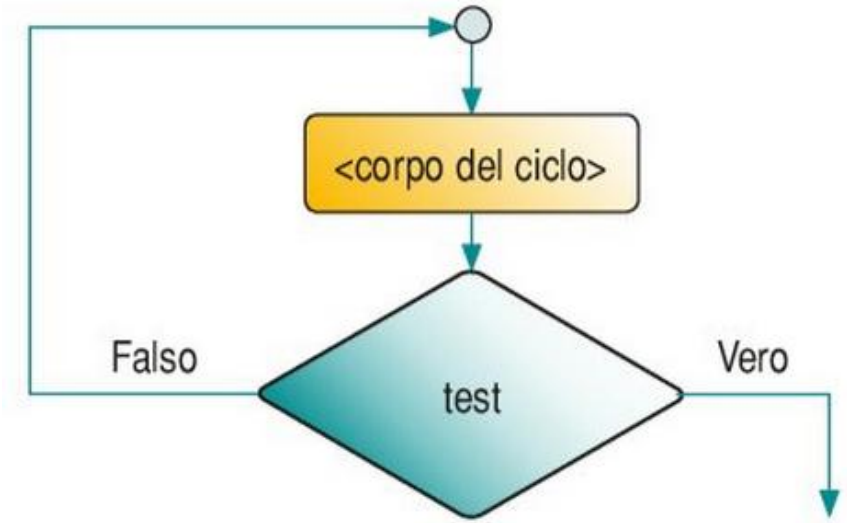
- In alcune situazioni un'azione deve essere eseguita in modo ripetuto, in modo ciclico (o iterato), in base al valore di una istruzione di controllo, chiamata condizione di ingresso al ciclo: l'istruzione prende il nome di **ciclo a condizione iniziale**.
- Questo tipo di ciclo prende anche nome di **iterazione indeterminata**, in quanto non è possibile sapere a priori quante volte dovrà essere ripetuto.
- Una variabile che viene utilizzata per contare quante volte una o più operazioni vengono eseguite prende il nome di **contatore**, che viene incrementata all'interno del ciclo.

Ciclo a condizione finale:

repeat..until

Ciclo a condizione finale: **repeat..until**

- È come il **while..do** con la differenza che:
  - Prima esegue il contenuto del ciclo
  - Poi alla fine verifica la condizione di ripetizione (condizione di uscita)



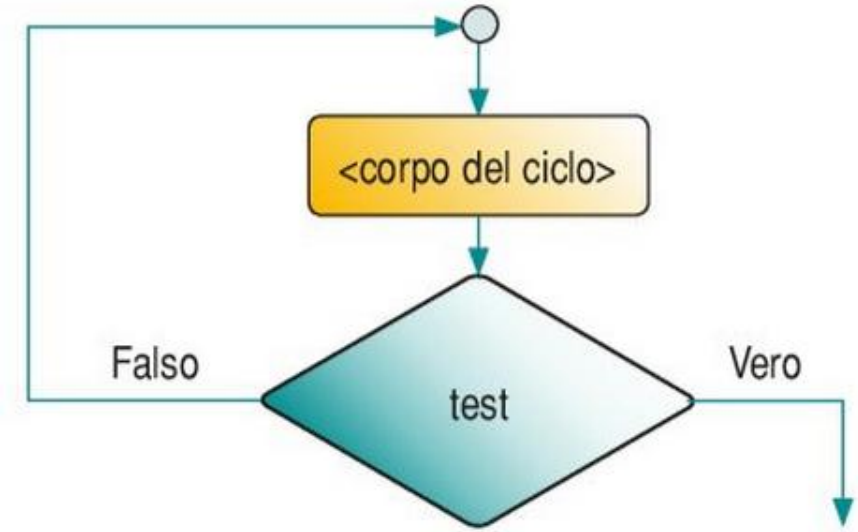
- se la condizione è verificata, il ciclo termina;
- se la condizione non è verificata, il ciclo viene ripetuto.

repeat..until in Pascal

```
repeat
  <blocco di istruzioni>
until <condizione di uscita>;
```

- si entra nel ciclo e si esegue il blocco di istruzioni (o semplicemente una istruzione);
- viene valutata la condizione di uscita, che è una operazione di test, e quindi ha risultato VERO o FALSO;
- se il valore FALSO è si torna indietro a ripetere il corpo del ciclo; se invece è VERO si esce dall'altro ramo e si prosegue il programma con le successive istruzioni.

Repeat..until: schema a blocchi vs istruzioni



vs

```
repeat
  <blocco di istruzioni>
until <condizione di uscita>;
```

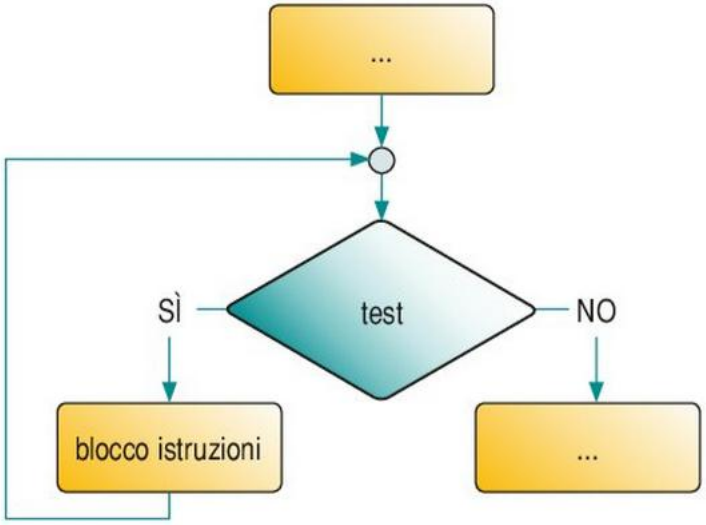
Condizione di uscita (e *di ingresso*)

`until(numero=0)`	Ripete fino a quando il valore di `numero` assume il valore 0, cioè mentre il valore di numero è diverso da 0.
`until(numero<0)`	Ripete fino a quando il `numero` diventa negativo, cioè mentre il valore di numero è positivo o uguale a 0.
`until(numero>0)`	Ripete fino a quando il `numero` diventa positivo, cioè mentre il valore di numero è negativo o uguale a 0.
`until(finito=TRUE)`	Quando il valore di `finito` è `TRUE` esce dal ciclo: ripete invece quando il valore di `finito` diviene uguale a `FALSE`.
`until(numero<>18))`	Quando il valore di `numero` è uguale a 18 esegue il ciclo: ne esce per qualunque altro valore.
`until(scelta<>'S')`	Quando il valore di `scelta` è uguale a S esegue il ciclo: ne esce per qualunque altro valore.

- Le stesse possono essere considerate come valido esempio di condizioni di ingresso nel while..do
  while <condizione> do <blocco>;

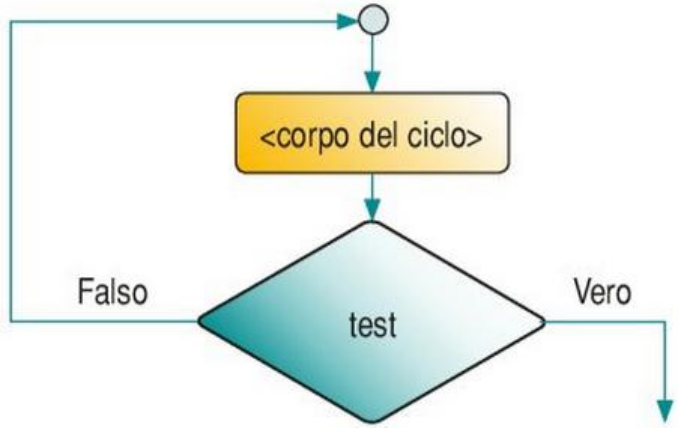
While..do vs Repeat..until

While..do




```
while <condizione di ingresso> do
  <blocco di istruzioni>;
```

Repeat..until




```
repeat
  <blocco di istruzioni>
until <condizione di uscita>;
```

Esempio: Ripetizione di un programma (Pari o dispari)

- Scrivere un programma che letto un valore in input sia in grado di valutare se il numero è pari o dispari.
- Tale operazione va ripetuta fin tanto che l'operatore inserisce nuovi valori.

Pertanto dovrà essere prevista un'istruzione del tipo:

```
writeln('vuoi inserire dei nuovi numeri S/N?')
```

Esempio: Ripetizione di un programma (Pari o dispari)

```
0  program pariDispari;
1  var
2    numero:integer;
3    scelta:char;
4  begin
5    repeat                                          // inizio del ciclo
6      write('inserisci un numero intero:');
7      readln(numero);
8      if(numero mod 2=0)
9        then
10         writeln('il numero inserito è PARI')      // corpo del ciclo
11       else
12         writeln('il numero inserito è DISPARI');
13     write('vuoi inserire un nuovo numero? ');
14     read(scelta)
15   until(scelta<>'S')and(scelta<>'s');             // ripete con scelta = S
16   write('termine elaborazione!');
17   readln;
18 end.
```

Esercizi 1-9, pag. 363

1. Scrivi un programma che legge una sequenza di numeri interi terminanti con uno 0 e visualizza quanti numeri sono stati inseriti.
2. Scrivi un programma che legge una sequenza di numeri interi terminanti con uno 0 e strettamente maggiori di 0 e ne ricerca il valore minimo visualizzandolo sullo schermo.
3. Scrivi un programma che legge un numero num e quindi successivamente esegue la somma di num numeri inseriti dall'utente.
4. Scrivi un programma che legge un numero num e visualizza tutti i numeri pari inferiori a tale numero.
5. Scrivi un programma Pascal che legge da tastiera una sequenza di numeri interi terminante con un numero negativo e al termine stampa a video il numero dei numeri letti che sono maggiori di zero, di quelli che sono minori di zero e di quelli nulli.
6. Scrivi un programma che esegue la somma di tutti i numeri multipli di 5 compresi tra 10 e 100.
7. Scrivi un programma che effettua il conto alla rovescia a partire da un valore minore di 20 inserito dall'utente.
8. Scrivi un programma che effettua random il lancio di due dadi e visualizza i risultati.
9. Scrivi un programma che genera num numeri random (num inserito dall'utente) e visualizza quanti numeri pari e quanti numeri dispari sono stati generati.

Esercizi 10-18, pag. 363

10 Scrivi un programma che effettua la somma dei numeri inseriti dall'utente fino a raggiungere il numero 1000 e indica quanti numeri sono stati sommati.

11 Scrivi un programma che effettua il calcolo della media dei voti della pagella, inserendoli uno alla volta e chiedendo a ogni inserimento di un numero se i voti da inserire sono terminati accettando come risposta S oppure N.

12 Scrivi un programma che effettua il calcolo della media dei voti della pagella, inserendoli uno alla volta e terminando con l'inserimento del numero 0.

13 Scrivi un programma che effettua il prodotto tra due numeri utilizzando il metodo delle somme successive dopo aver controllato l'input e accettato solo valori maggiori di 0.

14 Scrivi un programma che, leggendo due numeri, sottrae il minore dal maggiore finché la loro differenza diventa inferiore a 3 unità visualizzando sullo schermo il risultato di ogni iterazione.

15 Scrivi un programma che, leggendo due numeri, ne esegue la moltiplicazione mediante somme successive visualizzando sullo schermo il risultato di ogni iterazione.

16 Scrivi un programma che, leggendo due numeri, ne esegue la divisione mediante sottrazioni successive visualizzando sullo schermo il risultato di ogni iterazione.

17 Scrivi un programma che legge un numero intero in ingresso e lo converte in numero binario mediante successiva divisione per 2.

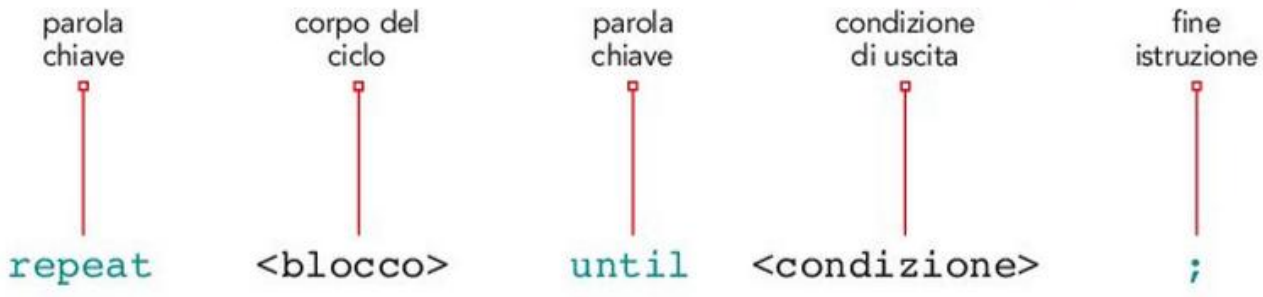
18 Scrivi un programma che gestisce il gioco dei fiammiferi: da un insieme di N fiammiferi, a turno due giocatori ne tolgono un numero qualunque che vada da 1 a K (dove N e K sono definiti a priori, per esempio N = 21 e K = 5).

Perde chi toglie l'ultimo fiammifero.

Conclusioni

ABBIAMO IMPARATO CHE...

- Il Pascal mette a disposizione una istruzione che permette di ripetere più volte un'azione in base al valore di una istruzione di controllo chiamata **condizione di uscita** dal ciclo.
- Questa istruzione è il **ciclo a condizione finale**, e ha la seguente struttura:



- Anche il ciclo a condizione finale e a iterazione **indeterminata**, in quanto non è possibile sapere a priori quante volte dovrà essere ripetuto, ma sicuramente viene eseguito una volta.

Ciclo a conteggio:
for..to..do

Ciclo a conteggio: **for..to..do**

- Nelle istruzioni iterative viste fino a ora non sapevano in anticipo il numero delle volte che veniva ripetuto il ciclo
- Esistono tuttavia delle azioni o dei compiti caratterizzati dal numero di volte che devono essere eseguiti.
  Per esempio:

Fai 10 giorni di ferie.
Leggi 20 pagine.
Per 10 volte scrivi "devo studiare di più".
Attacca 30 figurine sull'album.
Invia 100 sms alla zia.
Imbianca con 3 mani la parete.
Per 210 giorni devi andare a scuola.

for..to..do in Pascal

```
for <val.iniziale> to <val.finale> do
  <blocco di istruzioni>;
```

- La condizione di <inizio> si realizza con l'inizializzazione di una variabile al valore iniziale
- Tale variabile che viene automaticamente incrementata ad ogni ciclo fino al valore finale indicato in <fine>
- In quanto variabile va necessariamente dichiarata

Esempio: Scriviamo 10 volte una frase

```
0  program conteggio;
1  var
2    conta:integer;                    // variabile che esegue il conteggio
3  begin
4  (* scrittura ripetitiva di una frase *)
5    for conta:=1 to 10 do
6      writeln('speriamo che la scuola finisca presto');
7    readln;
8  end.
```

- il ciclo utilizza la variabile conta per effettuare il conteggio delle volte che viene eseguito incominciando a contare dal valore 1 e terminando quando viene raggiunto il valore 10;
- ogni volta viene eseguita l'istruzione di stampa.

Come hai già visto per il ciclo `while`, se è necessario eseguire più istruzioni devi racchiuderle tra `begin ... end`.

Esempio2: conteggio con estremi variabili

- **Problema**: scrivere un programma che sia in grado di contare da un numero iniziale fino a un numero finale, entrambi letti in input

- Per risolvere tale problema bisogna sapere che:
  - Il *valore iniziale* e il *valore finale* del ciclo for possono essere anche due variabili
  - Quando il blocco di istruzioni nel for presenta più di una riga bisogna inserire *begin..end;*

Esempio2: risoluzione

```
0  program conteggio2;
1  var
2     conta,inizio,fine:integer;          // variabili che eseguono il conteggio
3  begin
4  (* enumerazione definita dall'utente *)
5     write('da che numero inizio il conteggio?');
6     readln(inizio);
7     write('a che numero termino il conteggio?');
8     readln(fine);
9     writeln('inizio a contare da ',inizio);
10    for conta:=inizio to fine do
11      begin
12        write('aggiungo 1 e ottengo ');
13        writeln(conta);
14      end;
15   write('finisco di contare a ',fine);
16   readln;
17 end.
```

Esempio3: Tabellina pitagorica parziale

- **Problema:** scrivere un programma che legge in input un numero compreso tra 2 e 10 e visualizza la corrispondente tabellina

Esempio3: Tabellina pitagorica parziale

```
0  program tabellina;
1  var
2    conta,tabellina:integer;                    // variabili che eseguono il conteggio
3  begin
4  (* calcolo della tabellina *)
5    repeat
6      write('che tabellina vuoi visualizzare?');
7      readln(tabellina);
8    until(tabellina>1)and(tabellina<11);
9    for conta:=1 to 10 do
10     writeln(conta *tabellina);
11   readln;
12 end.
```

Cicli *for* annidati

Nell'esempio precedente è stata visualizzata una singola colonna della tabellina pitagorica: vogliamo ora visualizzare l'intera tabellina pitagorica 10 *10, cioè composta da 10 righe e 10 colonne.

È necessario, a questo fine, utilizzare due cicli, uno dentro l'altro: il primo ciclo scrive una riga, cioè la tabellina di un numero, il secondo scrive nella riga successiva la tabellina del numero seguente, e così via partendo da 1 fino a 10.

Esempio:Tabellina pitagorica completa

- Scriviamo un programma che visualizza a tabellina pitagorica 10x10, cioè composta da 10 righe e 10 colonne.

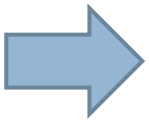
```
0  program tabellina2;
1  (* visualizzazione della tabellina pitagorica *)
2  var
3    riga,colonna:integer;                    // variabili che eseguono il conteggio
4  begin
5    writeln('visualizzazione della tabellina pitagorica');
6    for riga :=1 to 10 do                    // ciclo esterno
7      begin
8        for colonna:=1 to 10 do              // ciclo interno
9          write(' ',colonna *riga);
10       writeln;
11     end;
12   readln;
13 end.
```

Ciclo con passo negativo

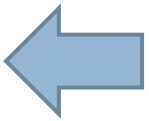
- Con il ciclo *for* si va ad eseguire un blocco di istruzioni fino a che incrementando la variabile contatore non raggiunge il valore max

```
for <val.iniziale> to <val.finale> do
  <blocco di istruzioni>;
```

- È possibile tuttavia modificare la sintassi del *for* in modo da decrementare il contatore fino a un valore finale



```
for < val.iniziale> downto < val.finale> do
  <blocco di istruzioni>;
```



Esempio: Count-down per il lancio di un missile

- Scriviamo un programma che effettua il count-down per il lancio di un missile nello spazio.

```
0  program countdown;
1  var
2    conta:integer;                    // variabile che esegue il conteggio
3  begin
4  (* calcolo del count-down per il lancio del missile *)
5    writeln('conto alla rovescia per il lancio di Apollo 77');
6    for conta:=10 downto 1 do
7      writeln('meno .. ',conta);
8    writeln('0: missile partito!');
9    readln;
10 end.
```

Esercizi 1-9, pag. 378

Risolvi gli esercizi utilizzando solo il ciclo a conteggio.

1. Scrivi un programma che legge un numero **NUM** e quindi successivamente esegue la somma di **NUM** numeri inseriti dall'utente.
2. Scrivi un programma che legge un numero **NUM** e visualizza tutti i numeri pari inferiori a tale numero.
3. Scrivi un programma che esegue la somma di tutti i numeri multipli di 5 compresi tra 10 e 100.
4. Scrivi un programma che legge una sequenza **NUM** di numeri interi positivi strettamente maggiori di 0 e ne ricerca il valore minimo visualizzandolo sullo schermo.
5. Scrivi un programma che effettua il conto alla rovescia a partire da un valore compreso tra 10 e 20 inserito dall'utente.
6. Leggi da tastiera un numero intero **NUM** compreso tra 1 e 20 e visualizza la tabella pitagorica dei numeri tra 1 e **NUM**.
7. Verifica che il quadrato di un numero n è uguale alla somma dei primi n numeri dispari (ad esempio, per **NUM** = 5, è 5 x 5 = 25 = 1 + 3 + 5 + 7 + 9) su una sequenza di numeri inseriti dall'utente e terminanti con uno 0.
8. Leggi un numero **NUM** indicato da un utente e calcola e stampa la somma di tutti i numeri compresi tra 0 e NUM.
9. Leggi un numero **NUM** e un numero **EXP** e calcola la potenza **EXP**-esima di un numero.

Esercizi 10-18, pag. 378

10 Leggi da tastiera un numero **NUM** e tre caratteri C1, C2 e C3; visualizza **NUM** volte la sequenza dei tre caratteri inseriti.

11 Scrivi un programma che calcola quante volte il numero 7 compare nei numeri compresi tra 1 e 100.

12 Scrivi un programma che effettua il prodotto tra due numeri utilizzando il metodo delle somme successive dopo aver controllato l'input e accettato solo valori maggiori di 0.

13 Scrivi un programma che disegna sullo schermo 3 cornici quadrate concentriche costituite dal carattere "*" rispettivamente di lato 12, 8, e 4.

14 Scrivi un programma che visualizza sullo schermo il triangolo di Tartaglia utilizzando due cicli a conteggio innestati spaziando opportunamente le cifre in modo da disporle ben incolonnate.

15 Scrivi un programma che visualizzi i primi 100 numeri dispari a gruppi di 5.

16 Scrivi un programma che visualizzi i primi 12 multipli di un numero *n* inserito da tastiera usando un ciclo for.

17 Scrivi un programma che, dato un numero n, calcoli e visualizzi la media dei suoi divisori.

18 Esegui il calcolo della radice quadrata di un numero utilizzando il metodo degli antichi **Babilonesi** approssimato leggendo in input il numero di iterazioni da eseguire.
Ogni iterazione viene effettuata nel seguente modo:

- si definisce la radice "per eccesso" del numero considerato eccesso1 ← num
- si definisce la radice "per difetto" del numero considerato difetto1 ← 2/eccesso1
- si prende la media aritmetica come nuovo valore di eccesso (eccesso2 ← difetto1+eccesso1)/2