Prof. Paolo Molena / scuola media "Quintino di Vona" – Milano

Unità di Apprendimento PROGRAMMI (31/08/05)

area delle Informazioni

2 media

1. Il controllo:.

1.a - che cosa è un controllo

La **struttura di controllo**. Il **passaggio**, il **salto**

1.b - i diagrammi di flusso

L'utilizzo dei diagrammi di flusso per visualizzare i diversi percorsi.

2. Il controllo nei programmi:.

2.a - l'**alternativa** (*se ? allora fai … altrimenti fai …*)

La sua gestione con IF _ THEN _ ELSE

2.b - la **condizione semplice** (*se ? allora fai …*)

In VISUAL BASIC

2.c - l'ordine di **salto** (GOTO)

l'uso dei **connettivi** (AND, OR)

3. la ripetizione del controllo (iterazione, **ciclo**):

Tipi di cicli, i **puntatori** di inizio e di fine ciclo, cicli a condizione **iniziale**, **finale**, **interna**

3.a - Ripetere attendendo un cambiamento

la gestione di un ciclo a condizione iniziale con WHILE _ WEND

la gestione dei cicli con DO _ LOOP

In VISUAL BASIC

- il controllo del **tempo** (TIME$, TIMER) In VISUAL BASIC
- l'**estrazione** (RND e RANDOMIZE TIMER)
- controllo delle **variazioni di un sensore** ( )

3.b - Ripetere contando

la gestione di un ciclo a condizione finale con FOR _ NEXT, FOR _ STEP _ NEXT

- contare per far passare il tempo
- utilizzare il numero della variabile che conta
- contare i caratteri di una variabile alfanumerica

In VISUAL BASIC

3.c – La nidificazione dei cicli

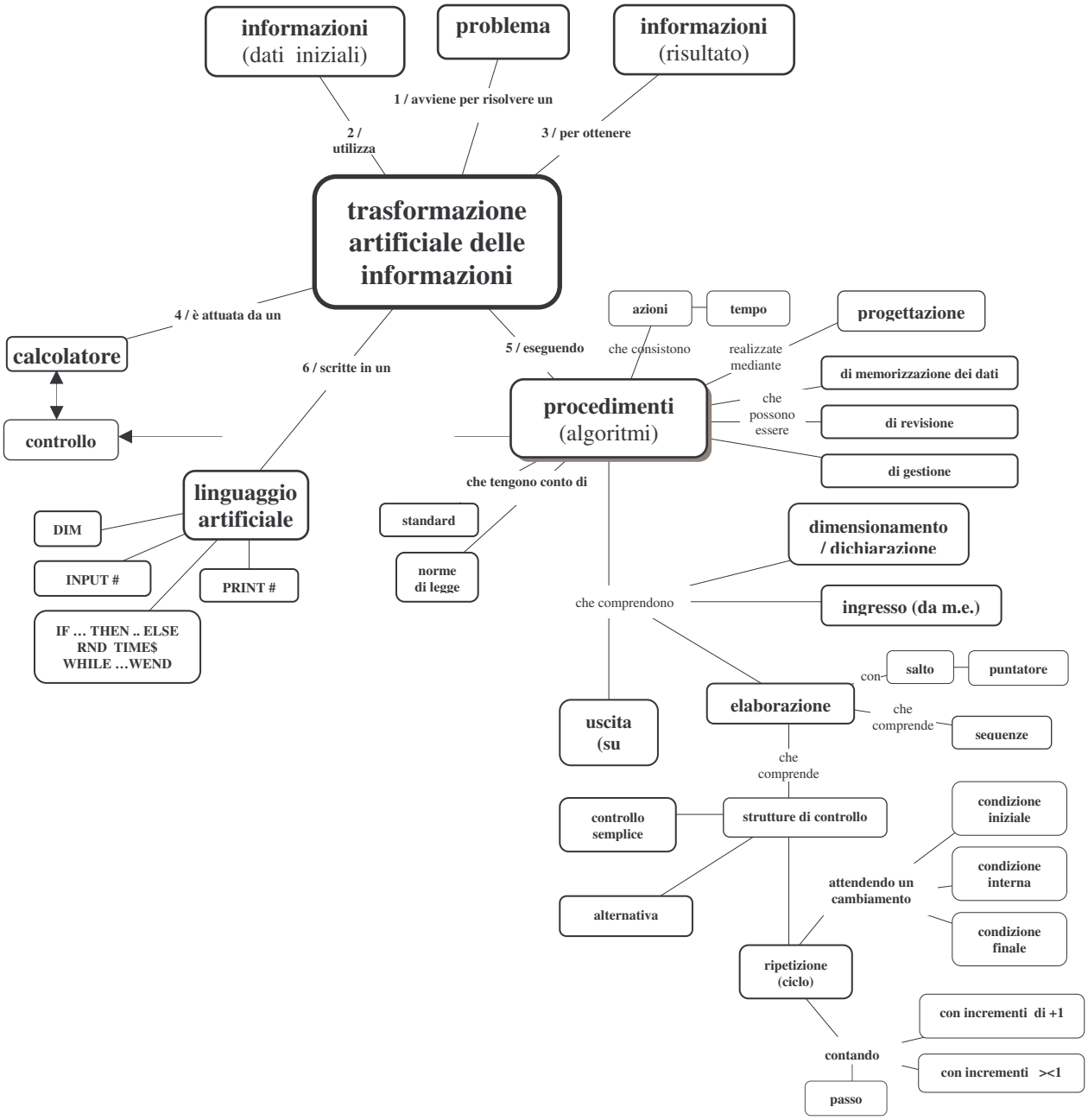
L'inserimento di un ciclo all'interno di un altro. Il ciclo interno. Il ciclo esterno

Un'occhiata all'hardware: l'area degli **stacchi**

- **presentazione**

In questa unità di apprendimento continueremo il percorso iniziato lo scorso anno con l'U.di A. Algoritmi. I procedimenti sui quali abbiamo lavorato erano ***sequenziali***. Tra l'ingresso dei dati (parametri d'entrata) e la scrittura del risultato (risultato del problema o parametro d'uscita) vi era una sequenza di azioni che il calcolatore doveva fare, una dopo l'altra. Questo modo di procedere è molto simile a quello che facciamo quando, a scuola, risolviamo un problema di matematica ma molto lontano da quello che succede nella realtà quando risolviamo un problema concreto (calcolare una busta paga o un affitto, studiare il procedimento per raggiungere una località di vacanza, istruire un robot a muoversi in un ambiente). In tutti questi casi i ***dati d'ingresso dovranno essere controllati*** (confrontati cioè con altri dati) e in base all'esito di questi controlli vi potranno essere diversi percorsi. Dunque la risoluzione di un problema con dei controlli può determinare **diversi processi*** a seconda dell'esito degli stessi controlli.

> * **processo** = sequenza di azioni che viene effettivamente svolta per andare dall'ingresso dei dati sino alla risoluzione del problema


informazioni (dati iniziali)
problema
informazioni (risultato)
1 / avviene per risolvere un
2 / utilizza
3 / per ottenere
trasformazione artificiale delle informazioni
4 / è attuata da un
calcolatore
controllo
6 / scritte in un
linguaggio artificiale
DIM
INPUT #
PRINT #
IF … THEN .. ELSE
RND TIME$
WHILE …WEND
5 / eseguendo
procedimenti (algoritmi)
che consistono
azioni
tempo
realizzate mediante
progettazione
che possono essere
di memorizzazione dei dati
di revisione
di gestione
che tengono conto di
standard
norme di legge
che comprendono
dimensionamento / dichiarazione
ingresso (da m.e.)
uscita (su
elaborazione
con
salto
puntatore
che comprende
sequenze
che comprende
controllo semplice
strutture di controllo
alternativa
attendendo un cambiamento
condizione iniziale
condizione interna
condizione finale
ripetizione (ciclo)
contando
con incrementi di +1
con incrementi ><1
passo

IL CONTROLLO

Il controllo; che cosa é?

Le procedure di cui ci siamo occupati sinora erano formate da una sequenza di azioni da svolgere ordinatamente una dietro l'altra. Sono procedure tipiche della realtà scolastica e in particolare della matematica ma le procedure che noi svolgiamo nella nostra realtà quotidiana sono ben diverse. Tra le varie differenze una consiste certamente nella presenza in queste ultime di frequenti controlli. Vi sono cioè frequenti momenti in cui dobbiamo prendere delle decisioni, e queste vengono prese confrontando delle informazioni presenti nella nostra mente. Vediamo a questo proposito di fare alcuni semplici esempi:

- *stiamo andando a casa di un nostro nuovo amico e ci troviamo di fronte ad un bivio*
- *nostra mamma ci manda a fare la spesa e ci dice di comperare le mele meno care*

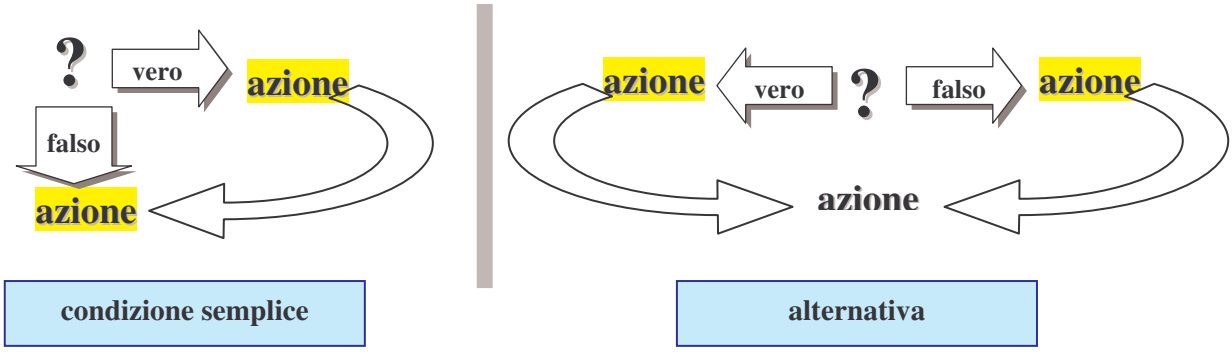
Nel primo caso dovremo confrontare l'immagine del bivio che proviene dai nostri occhi con eventuali immagini precedenti, oppure cercare in quell'immagine dei particolari che, confrontati con ciò che ricordo di eventuali descrizioni dell'amico (che ad es. ci può aver detto che la sua casa è oltre un ponte), mi permettano di effettuare una scelta. Nel secondo caso leggerò i prezzi in successione, confrontandolo il primo con il secondo e tenendo in memoria quello minore, che poi verrà confrontato con quello letto successivamente.

LA STRUTTURA DI CONTROLLO *

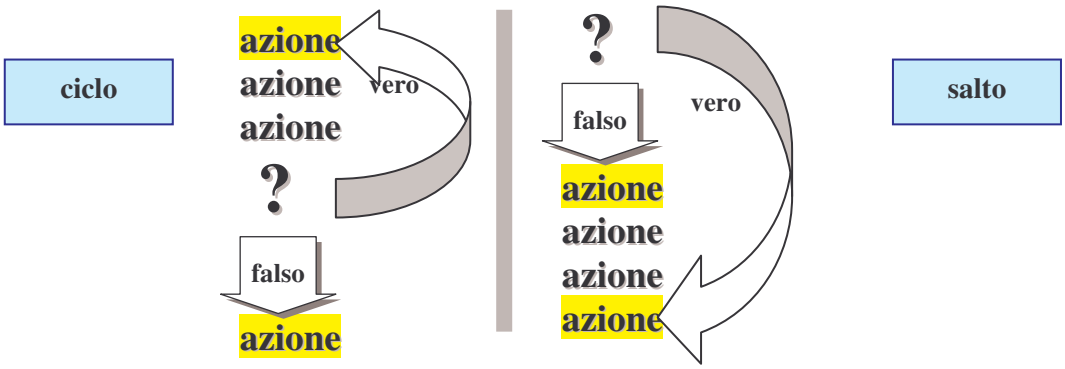
La procedura sequenziale con cui abbiamo lavorato sinora è relativamente semplice da comprendere. Infatti essa consiste in un certo numero di azioni che, al momento dell'esecuzione saranno eseguite una dopo l'altra nell'ordine con cui sono state archiviate nella memoria del calcolatore.
Il controllo introduce grosse novità. Infatti esso prevede in alcuni casi che ***delle azioni possano essere eseguite solo in caso di risposta Vero***, in altri che ***delle azioni vengano eseguite solo in caso di risposta Vero ed altre azioni in caso di risposta Falso***.
In altri casi ancora, una risposta al controllo può portare a ***ripetere azioni già eseguite***, in altri ancora può portare a ***saltare una serie di azioni collocate dopo il controllo***.

* in informatica viene usato il termine **struttura di controllo** per indicare non solo la domanda ma anche tutte quelle azioni che sono condizionate dalla risposta alla domanda.

Esamina le diverse situazioni rappresentate negli schemi collocati di seguito. Si può notare che, mentre alcune azioni verranno eseguite in ogni caso, l'esecuzione di altre è strettamente collegata all'esito del controllo.



condizione semplice **alternativa**



Mentre per l'attività di analisi continueremo ad utilizzare, almeno per ora, lo strumento del **grafo di scomposizione**, l'individuazione del percorso di risoluzione richiederà l'utilizzo di un apposito linguaggio grafico, quello dei **diagrammi di flusso**.

I diagrammi di flusso

I diagrammi di flusso vengono utilizzati per descrivere gli algoritmi in cui l'esistenza di una struttura di controllo diversifica i percorsi. Costituiscono lo strumento ideale per rappresentare la conoscenza procedurale visto che consentono, con una rapida occhiata, di cogliere il susseguirsi delle azioni lungo i diversi itinerari. Oltre a rappresentare con un grafico il flusso delle azioni nel tempo, essi servono anche per classificare le azioni contenute nel procedimento.

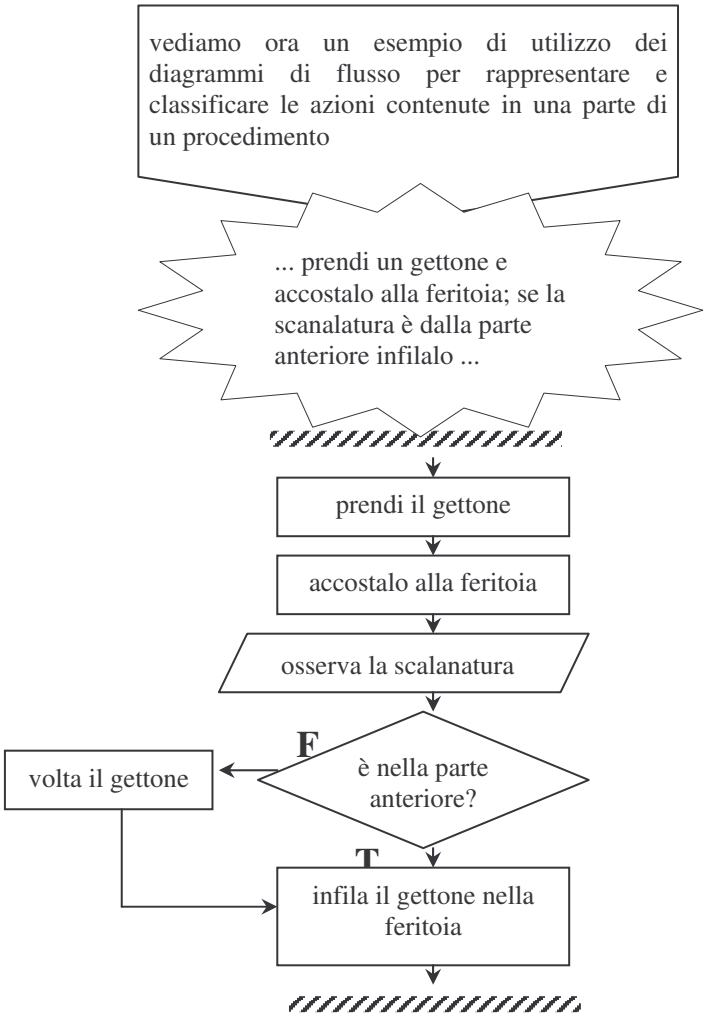
Le azioni vengono classificate in tre tipi:

1) azioni con cui vengono ***elaborate delle informazioni*** (già in possesso dell'esecutore) o con cui viene modificato lo stato delle cose che ci circonda

2) azioni con cui l'esecutore ***richiede dati*** che non possiede o con cui trasmette dati che possiede (ingresso o uscita di informazioni)

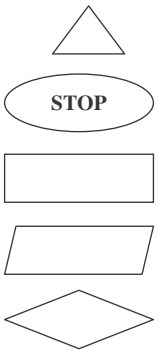
3) ***azioni di controllo***, cioè domande a cui il calcolatore deve rispondere usando i dati in suo possesso
Come abbiamo visto nell'esempio, un'azione di controllo va scritta come domanda alla quale vi sono sempre e solo due risposte (vero = V / falso = F).

Le frecce indicano il ***passaggio*** (se portano all'azione successiva) o il ***salto*** (se portano ad un'azione diversa dalla successiva



Con l'azione di controllo il calcolatore verifica dati che devono già essere in suo possesso; di conseguenza molto spesso l'azione di controllo è preceduta da un'azione con la quale il calcolatore si procura il dato da controllare successivamente.

Segni convenzionali (simboli)



istruzione di inizio di esecuzione

istruzione di arresto dell'esecuzione

istruzione di elaborazione

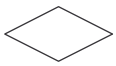
istruzione di trasmissione

istruzione di controllo

istruzione di passaggio o di salto

Sintassi

- nessuna freccia porta a 
- nessuna freccia esce da 

- da ogni  e ogni  esce una sola freccia
- da ogni  escono due frecce
- esiste almeno un cammino da  ad uno **STOP**

Nota Bene: quando la procedura non è completa, l'interruzione viene indicata con: 

i diagrammi di flusso delle ricette da cucina

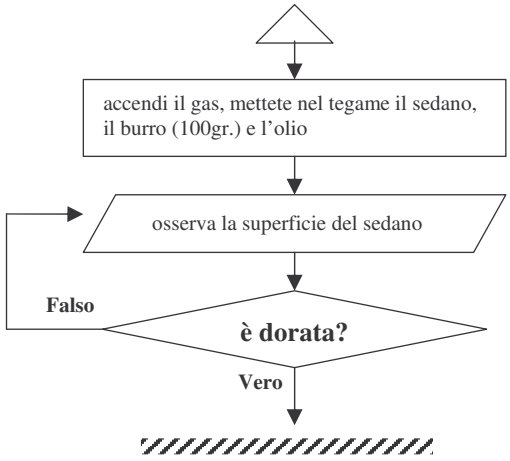
Le ricette di cucina rappresentano un tradizionale "terreno" di utilizzo dei diagrammi di flusso in quanto rappresentano una conoscenza procedurale che deve essere descritta con molta precisione. Realizza su un foglio a parte il diagramma di flusso della seguente ricetta in cui sono evidenziati i controlli (per aiutarti abbiamo realizzato la parte iniziale con il primo controllo) :

RISOTTO CON IL SEDANO DI VERONA (per 6 persone)

burro, 100 gr - burro, una noce - olio d'oliva, 3 cucchiai - un sedano di Verona tagliato a fette sottili - riso, 500 gr - vino bianco secco, un bicchiere - brodo bollente, un litro e mezzo - grana grattugiato, 100 gr

un tegame a bordi alti e di grosso spessore - fornello - posate - una pentola per il brodo

Metti nel tegame il sedano con i 100 gr di burro e l'olio e fallo rosolare. Butta poi il riso e giralo per tre minuti; aggiungi il vino, lascialo evaporare. Continua la cottura mescolando e aggiungendo, a mestoli, il brodo bollente, stando attento a non farlu asciugare. Dopo 15 minuti di questa cottura, prima di togliere il riso dal fuoco, aggiungi la noce di burro e il formaggio.



Ed ora al lavoro!

Osserva la mappa sul fianco e rispondi alle seguenti domande:

In quali parti si dividono le procedure?

_____

_____

Di che cosa si occupa la parte di elaborazione?

_____

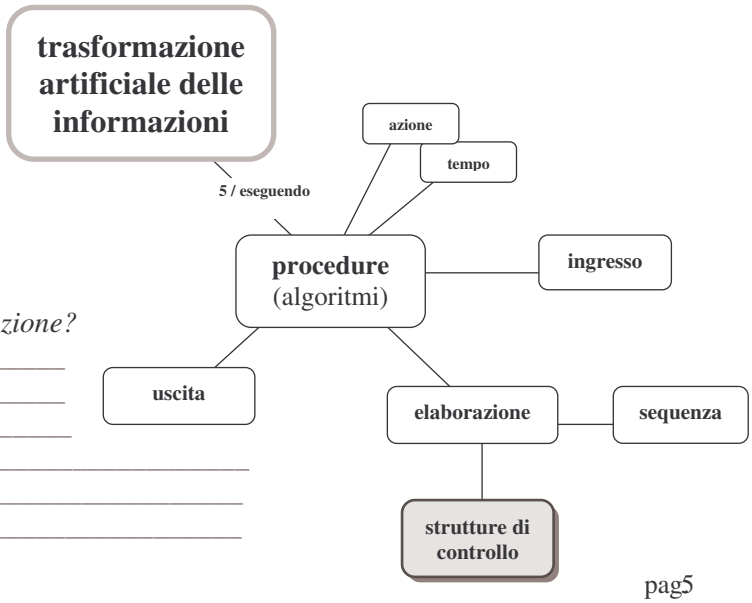
_____

Da quali parti è composta? _____

_____

Che cosa è una struttura di controllo? _____

_____



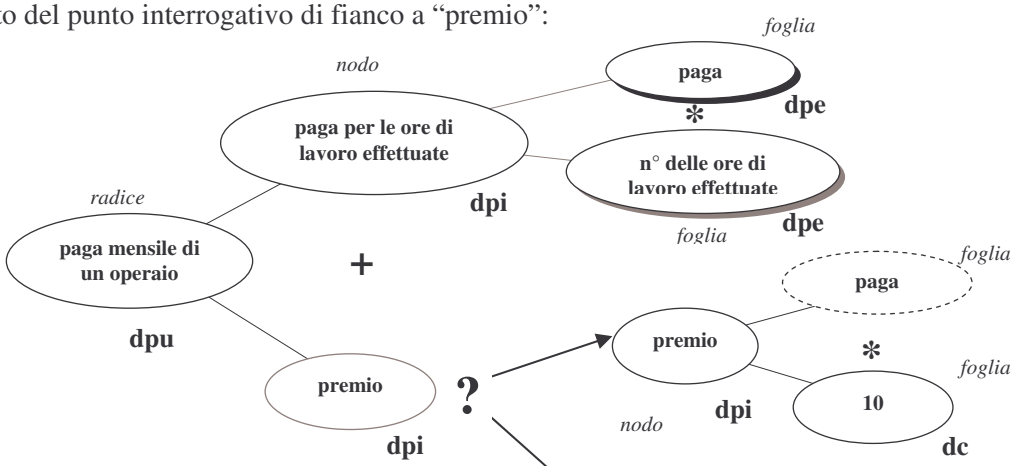
IL CONTROLLO NEI PROGRAMMI

L'alternativa

Prendiamo ora un problema d'informatica che contiene un controllo:

> *Un calcolatore deve calcolare la **paga mensile** di ogni operaio di un impresa edile. La paga mensile di un operaio è formata dalla **paga corrispondente alle ore lavorate**, calcolata in base al **numero di ore di lavoro** che l'operaio ha effettuato nel mese. A fine mese, ad essa, viene aggiunto un **premio** corrispondente alla paga di **10 ore** se durante il mese avrà effettuato più di 160 ore di lavoro altrimenti il premio corrisponderà alla paga **5 ore** di lavoro.*

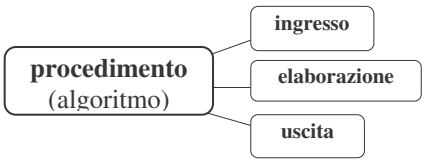
Ne facciamo l'**analisi** utilizzando per l'ultima volta l'albero di scomposizione. Leggiamolo e cerchiamo di capire il significato del punto interrogativo di fianco a "premio":



Le modalità di calcolo del premio saranno subordinate alla risposta che si ottiene alla domanda:

il n° delle ore di lavoro effettuate > 160 ?

ad essa è subordinato il percorso e dunque il calcolo da effettuare per individuare l'entità del premio.



Passiamo ora al procedimento di risoluzione scritto in linguaggio di progetto ricordandoci di dividere le tre fasi

ingresso

- **chiedi** *il numero di ore di lavoro effettuate nel mese*
- **chiedi** _____

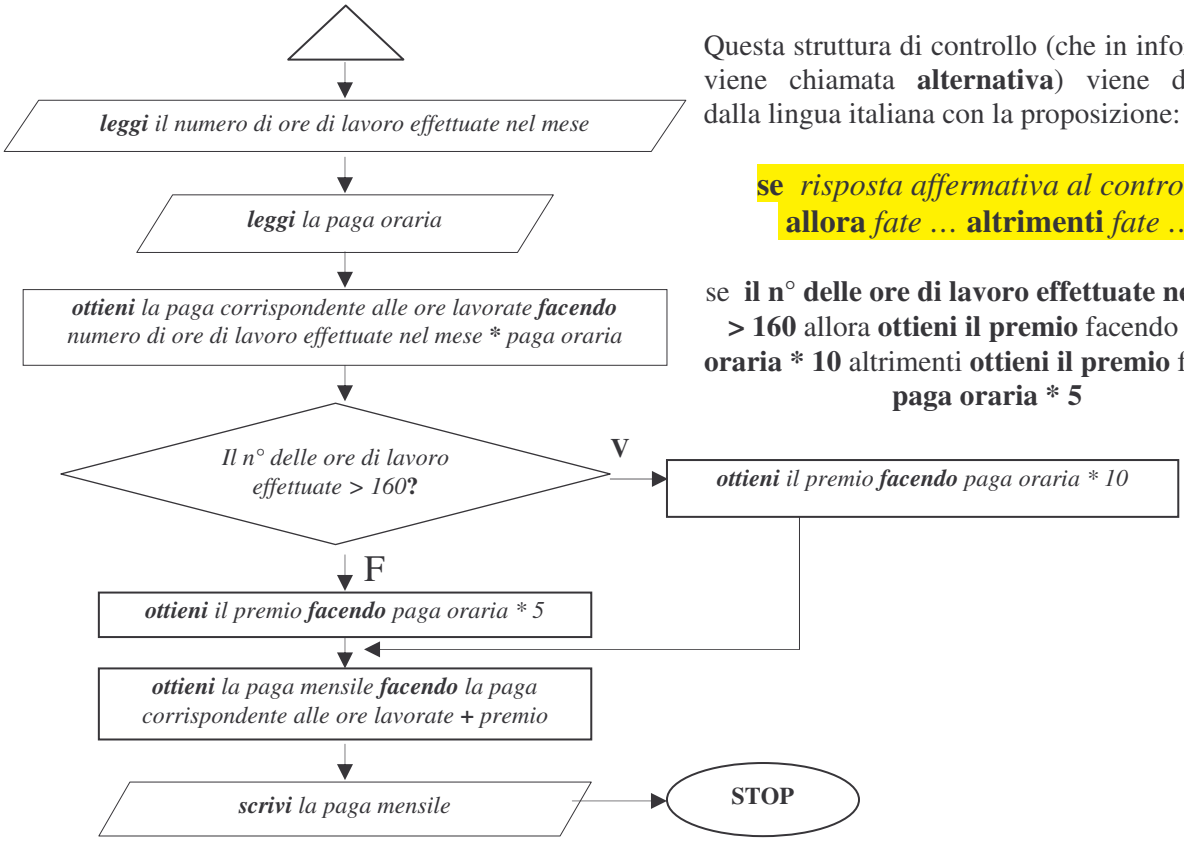
elaborazione

- **ottieni** *la paga per le ore di lavoro effettuate* **facendo** _____
  _____
- **se** *il n° delle ore di lavoro effettuate* **>** *160* **allora** ottieni *il premio* facendo _____
  _____
  altrimenti ottieni *il premio* **facendo** _____
  _____
- **ottieni** *la paga mensile di un operaio* **facendo** _____

uscita

- **scrivi** *la paga mensile di un operaio* _____

Sempre utilizzando il linguaggio di progetto ne abbiamo realizzato il diagramma di flusso.



Questa struttura di controllo (che in informatica viene chiamata **alternativa**) viene descritta dalla lingua italiana con la proposizione:

se *risposta affermativa al controllo*
allora *fate …* **altrimenti** *fate …*

se **il n° delle ore di lavoro effettuate nel mese > 160** allora **ottieni il premio** facendo **paga oraria * 10** altrimenti **ottieni il premio** facendo **paga oraria * 5**

e in BASIC con:

➢ **IF** *controllo* **THEN** *istruzioni* [**ELSE** *istruzioni*]

Dove le parentesi quadre [ ] indicano delle opzioni non obbligatorie. Le parole in grassetto, quali lf, rappresentano delle parole chiave del linguaggio e devono venire scritte tali e quali. Le parti in corsivo, tipo '*istruzioni*', devono venire sostituite dagli elementi specifici del programma.

controllo

Un controllo è un'espressione che può essere vera o falsa.

Operatore	Significato
=	Vero se i due termini sono uguali
<>	Vero se i due termini sono diversi
<	Vero se il primo termine è minore del secondo
<=	Vero se il primo termine è minore o uguale al secondo
>	Vero se il primo termine è maggiore del secondo
>=	Vero se il primo termine è maggiore o uguale al secondo

I dati inseriti nel controllo possono anche essere il risultato di una espressione aritmetica.

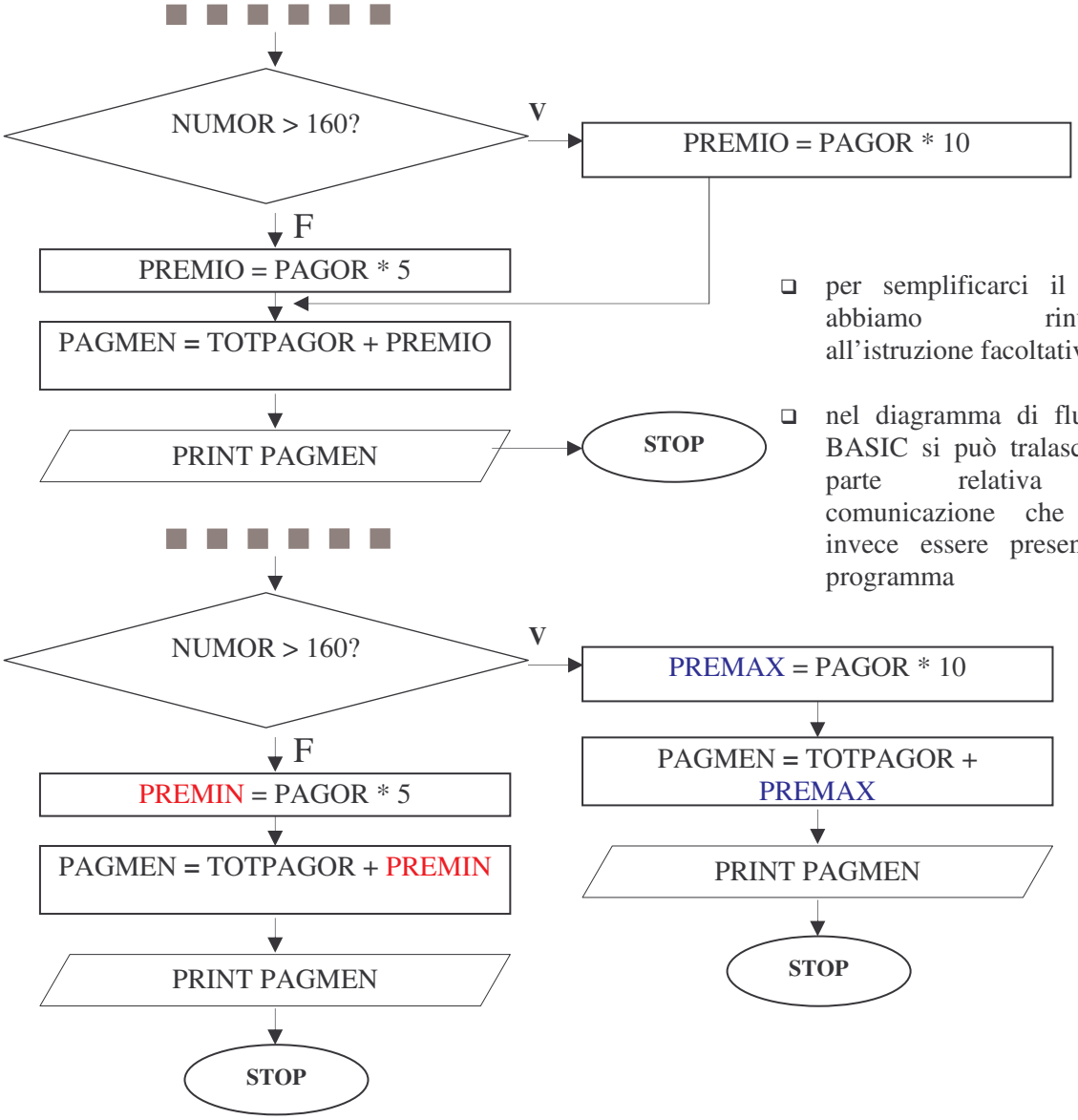
istruzioni

Le istruzioni posizionate dopo **Then** vengono eseguite solo in caso di risposta affermativa al controllo. Può trattarsi di qualunque tipo di istruzione. Si possono inserire più istruzioni su una stessa riga a condizione che siano separate dal simbolo di due punti
Le istruzioni dopo **Else** vengono eseguite solo in caso di risposta negativa al controllo.

Possiamo ora trasformare il procedimento di risoluzione il un programma:

10 **INPUT** "Scrivi il n° di ore effettuate dall'operaio ";NUMOR
20 **INPUT** "Scrivi la paga oraria che spetta all'operaio ";PAGOR
30 **LET** TOTPAGOR = NUMOR * PAGOR
40 **IF** NUMOR > 160 THEN PREMIO = PAGOR * 10 **ELSE** PREMIO = PAGOR * 5
50 **LET** PAGMEN = TOTPAGOR + PREMIO
60 **PRINT** "La paga mensile che spetta all'operaio è di £.";PAGMEN
70 **STOP**

L'aver chiamato con lo stesso nome, in entrambe le soluzioni, la variabile che contiene il valore del premio (PREMIO) permette ai due percorsi un facile ricongiungimento sia in caso di risposta VERO che in caso di risposta FALSO. Se avessimo utilizzato nomi diversi avremmo dovuto tenere separati i percorsi.
Vediamo qua sotto entrambe le soluzioni con i diagrammi di flusso in BASIC.



- per semplificarci il lavoro abbiamo rinunciato all'istruzione facoltativa LET
- nel diagramma di flusso in BASIC si può tralasciare la parte relativa alla comunicazione che dovrà invece essere presente nel programma

La seconda soluzione è sconsigliata perché ci costringerebbe a scrivere una linea di controllo lunga e ripetitiva:

……
40 **IF** NUMOR > 160 THEN PREMAX = PAGOR * 10: PAGMEN = TOTPAGOR + PREMAX: PRINT PAGMEN: STOP **ELSE** PREMIN = PAGOR * 5: PAGMEN = TOTPAGOR + PREMIN: PRINT PAGMEN: STOP
……

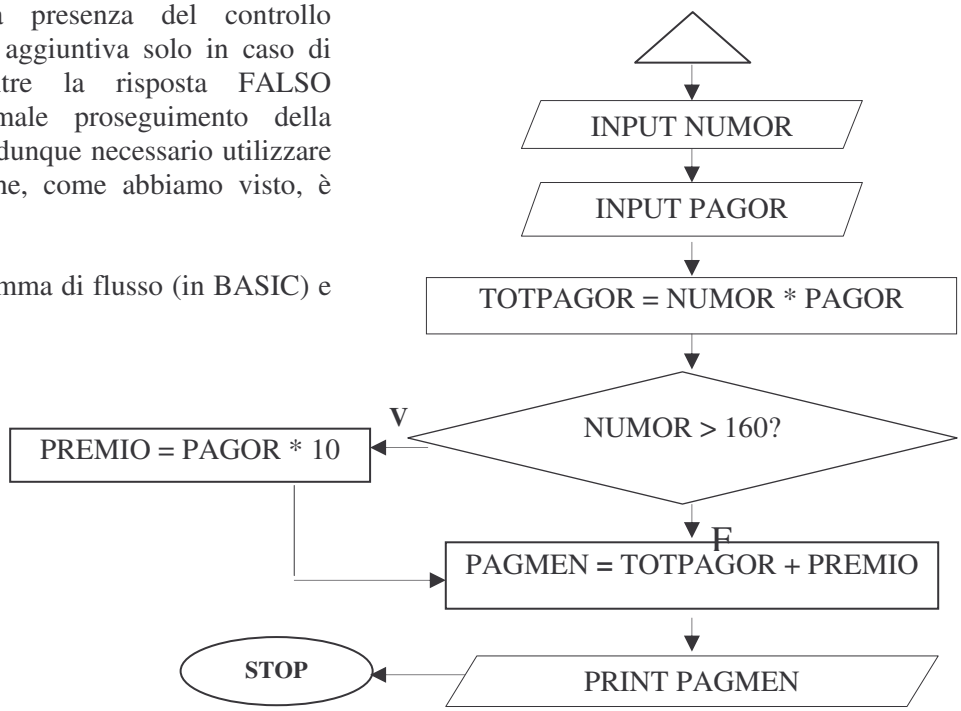
Il controllo semplice

modifichiamo ora il testo del problema:

*Un calcolatore deve calcolare la **paga mensile** di ogni operaio di un impresa edile. La paga mensile di un operaio è formata dalla **paga corrispondente alle ore lavorate**, calcolata in base al **numero di ore di lavoro** che l'operaio ha effettuato nel mese. A fine mese, ad essa, viene aggiunto un **premio** corrispondente alla paga di **10 ore** se durante il mese avrà effettuato più di 160 ore di lavoro.*

In questo caso la presenza del controllo determina un'azione aggiuntiva solo in caso di risposta vero mentre la risposta FALSO determinerà il normale proseguimento della procedura. Non sarà dunque necessario utilizzare l'istruzione ELSE che, come abbiamo visto, è facoltativa.

Osserviamone diagramma di flusso (in BASIC) e programma:




```
10    INPUT "Scrivi il n° di ore effettuate dall'operaio "; NUMOR
20    INPUT "Scrivi la paga oraria che spetta all'operaio"; PAGOR
30    LET TOTPAGOR = NUMOR * PAGOR
40    IF NUMOR > 160 THEN PREMIO = PAGOR * 10
50    LET PAGMEN = TOTPAGOR + PREMIO
60    PRINT "La paga mensile che spetta all'operaio è di £.";PAGMEN
70    STOP
```

rifletti sul programma appena visto e scrivi qui sotto perché la proposizione collocata alla linea 50 è valida sia in caso di risposta vero che in caso di risposta falso.

In VISUAL BASIC

espansioni del linguaggio

Oltre all'uso che abbiamo appena visto, in VB vi è un secondo modo di scrivere l'istruzione IF .. THEN .. ELSE .. molto utile se vi sono molte istruzioni da eseguire.

```
If  controllo Then
istruzioni
….
Else
        istruzioni
….
End If
```

quando vi è una variabile da controllare ma sono da prevedere molti possibili esiti sarà utile utilizzare **Select Case**:

Select Case *variabile da controllare*
Case *esito del controllo*
istruzioni
Case *esito del controllo*
istruzioni
…..
Case Else
Istruzioni
End Select

facciamo un esempio: *nella variabile mansione$ vi è la mansione svolta da un dipendente in una ditta e sono da prevedere diverse mansioni a cui corrispondono differenti calcoli per lo stipendio*

Select Case *mansioni$*
Case "operaio"
istruzioni
Case "impiegato"
istruzioni
Case "dirigente "
istruzioni
Case Else
Istruzioni
End Select

L'uso della scheda di programmazione

La scheda di programmazione è stata adattata alle esigenze che pone questa Unità Didattica. Nel riquadro del problema compare direttamente il testo del problema di informatica e, sul lato, il diagramma di flusso che conterrà le istruzioni scritte in BASIC. Nel programma dovremo ricordare di inserire ciò che è utile per la comunicazione con l'operatore e anche i pro-memoria per il programmatore (REM). Non è più richiesto il percorso in linguaggio di progetto e il grafo di scomposizione anche se si consiglia, almeno all'inizio, di non abbandonare queste fasi di lavoro svolgendole su un foglio a parte o, almeno, mentalmente.
Vediamo come sarà compilata la scheda che riporta il problema appena esaminato.

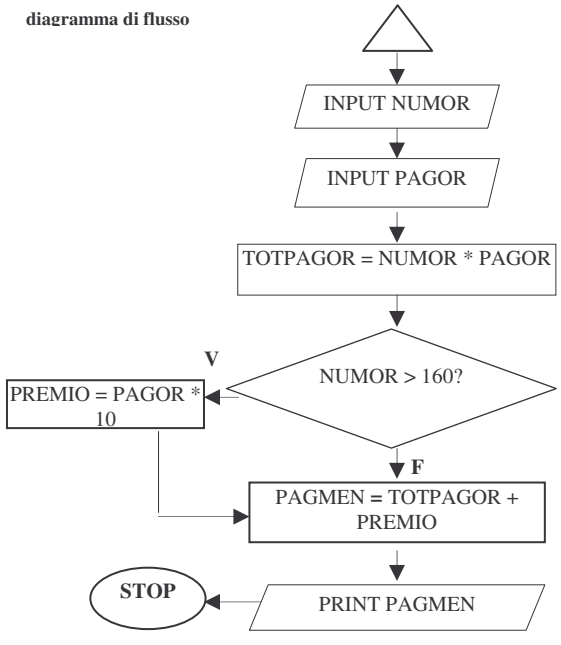
problema
Un calcolatore deve calcolare la paga mensile di ogni operaio di un impresa edile. La paga mensile di un operaio è formata dalla paga corrispondente alle ore lavorate, calcolata in base al numero di ore di lavoro che l'operaio ha effettuato nel mese. A fine mese, ad essa, viene aggiunto un premio corrispondente alla paga di 10 ore se durante il mese avrà effettuato più di 160 ore di lavoro.

Cognome e nome Rossi Aldo **n°** 28 **data** 18/02/00 **classe** 2C

programma

```
10  INPUT "Scrivi il n° di ore effettuate dall'operaio ";  NUMOR
20  INPUT "Scrivi la paga oraria che spetta  all'operaio ";PAGOR
30  LET TOTPAGOR = NUMOR * PAGOR
40  IF NUMOR > 160 THEN PREMIO = NUMOR*10
50  LET PAGMEN = TOTPAGOR + PREMIO
60  PRINT "La paga mensile che spetta all'operaio è di £";PAGMEN
70  STOP
```

diagramma di flusso



Ed ora al lavoro!

Qui di seguito vi sono alcuni problemi di informatica contenenti controlli. Dovrai trascriverne il testo sulle schede di programma date dall'insegnante e poi compiere tutti i passaggi sino ad ottenere il programma per il calcolatore. Dovrai fare correttamente queste operazioni:

- leggere attentamente il testo del problema individuando gli eventuali dati costanti e i dati parametrici che andranno richiesti in fase di richiesta dei dati
- individuare la procedura di risoluzione disegnando il diagramma di flusso
- realizzare il programma curando in particolare la comunicazione diretta all'operatore sia in fase di ingresso dati che in uscita
- compiere, dopo aver istruito il calcolatore, l'esecuzione di prova controllando la correttezza dei risultati.

1) Alla paga base mensile del commesso di un negozio viene aggiunto 1/ 10 del valore delle vendite effettuate durante il mese.
Se il valore delle vendite mensili supera i 5000 euro alla paga verrà aggiunto un'ulteriore premio di 25 euro.
Il calcolatore deve calcolare la paga definitiva.

2) Una società immobiliare vende degli appartamenti il cui valore dipenderà dalla superficie e dalla posizione rispetto al resto della città. Infatti se l'appartamento è in centro città il suo costo aumenterà di 1/3. Il calcolatore, dopo aver chiesto la superficie in mq. e il costo al mq. dovrà calcolare il costo definitivo dell'appartamento.

3) In una regione agricola i terreni producono una certa quantità di chili di frumento per ogni ettaro di terreno. Se il terreno è stato concimato il raccolto aumenterà di 1/3.
Preparare un programma che, data la superficie di un azienda agricola (in ettari) di un azienda calcoli la quantità di frumento prodotto.

L'ordine di salto

Sappiamo che il calcolatore, quando esegue un programma, lavora sequenzialmente; istruzione dopo istruzione, linea dopo linea, passa sempre dalla proposizione precedente a quella successiva. Collegata ad una proposizione di controllo può esserci la necessità di ordinare al calcolatore un **salto**, cioè il passaggio ad una linea diversa da quella successiva. In GWBASIC questo è possibile grazie alla presenza, all'inizio di ogni linea di programma, del numero di linea che dunque può essere utilizzato come **puntatore**. In altri linguaggi, ed anche in altre versioni di BASIC come il VISUAL BASIC, il ruolo di puntatore può essere svolto anche da parole, ed è per questo motivo che in quei linguaggi l'uso del numero di linea non è più obbligatorio.
Il salto, anche in VB, viene ordinato con:

➢ **GOTO** *numero della linea da dove si deve proseguire l'esecuzione*

Sperimentiamone un'applicazione modificando uno dei programmi scritti precedentemente.

linguaggio di progetto	**gwbasic**
leggi la paga oraria *leggi il n° di ore effettuate*	*10 INPUT"Scrivi la paga oraria "; Pagor* *20 INPUT"Scrivi il n° di ore di lavoro effettuate "; Ore*
***se** il n° di ore effettuate **è maggiore di** 160 **allora ottieni** la paga mensile facendo la paga oraria **per** il n° di ore effettuate più 10: scrivi la paga mensile: stop*	*30 IF Ore >160 THEN **GOTO 70***
***ottieni** la paga mensile **facendo** la paga per le ore effettuate*	*40 Pagmen=Pagor*Ore*
***scrivi** la paga mensile*	*50 PRINT"La paga mensile è di lire ";Pagmen*
stop	*60 STOP* ***70** Pagmen = Pagor * (Ore+10)* *80 PRINT"La paga mensile è di lire ";Pagmen* *90 STOP*

Come si vede la procedura scritta il linguaggio di progetto rimane invariata. La necessità di inserire l'ordine di salto ci costringe a rendere differente il percorso del programma e a collocare subito il numero di linea. E' bene limitare l'utilizzo di quest'ordine ai casi in cui è strettamente necessario.

L'uso dei connettivi nei controlli

Nelle nostre conversazioni quotidiane utilizziamo spesso i connettivi "e" e "oppure" per unire due o più domande. In BASIC è possibile combinare più condizioni fra loro per mezzo di operatori logici, i più frequenti dei quali sono AND e OR.
AND (e) dà un esito positivo solo se sono affermativi entrambi i controlli ad esso collegati, mentre OR (o) è affermativo se lo è almeno uno dei controlli. E' anche possibile utilizzare l'operatore logico NOT che inverte l'esito di un controllo:

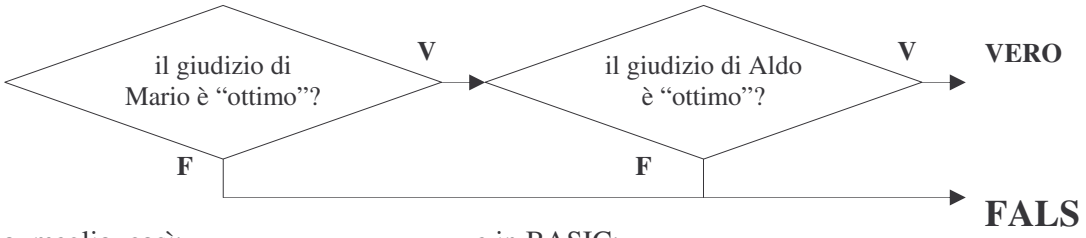
il connettivo "e" viene tradotto con

- **AND**

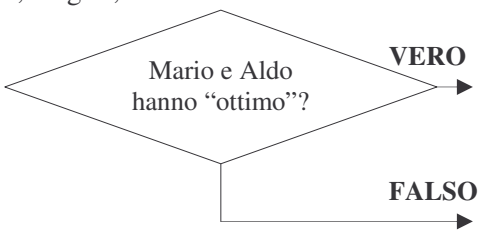
mentre il connettivo "o" viene tradotto con

- **OR**

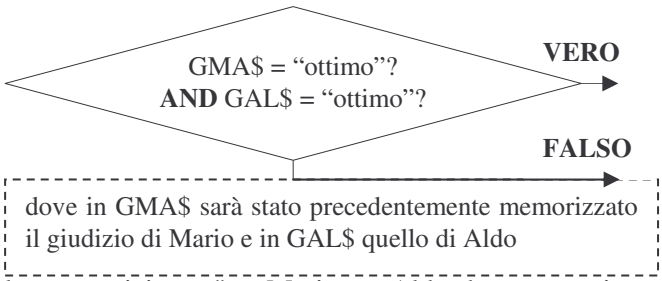
Ad esempio la proposizione "... Mario e Aldo hanno meritato "ottimo" nel compito di informatica?" nel diagramma di flusso può essere riportata così:



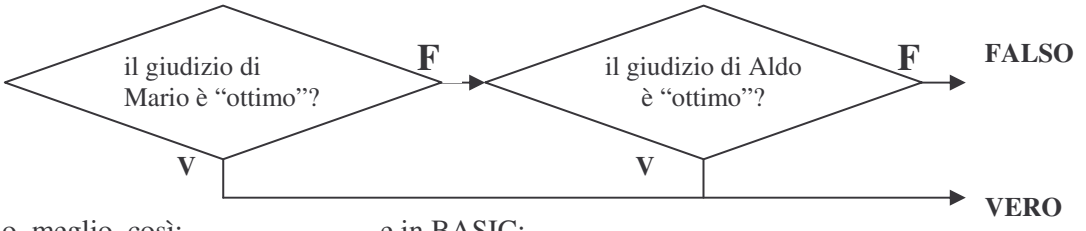
o, meglio, così:



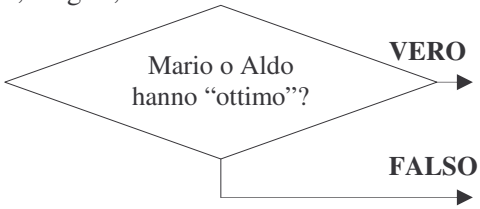
e in BASIC:



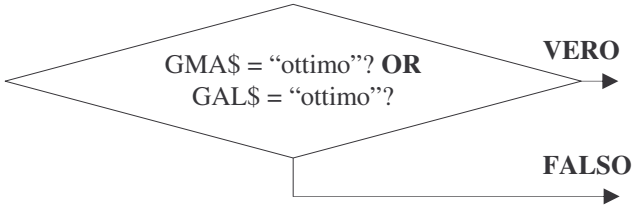
Mentre esempio la proposizione "... Mario o Aldo hanno meritato "ottimo" nel compito di informatica?" nel diagramma di flusso può essere riportata così:



o, meglio, così:



e in BASIC:



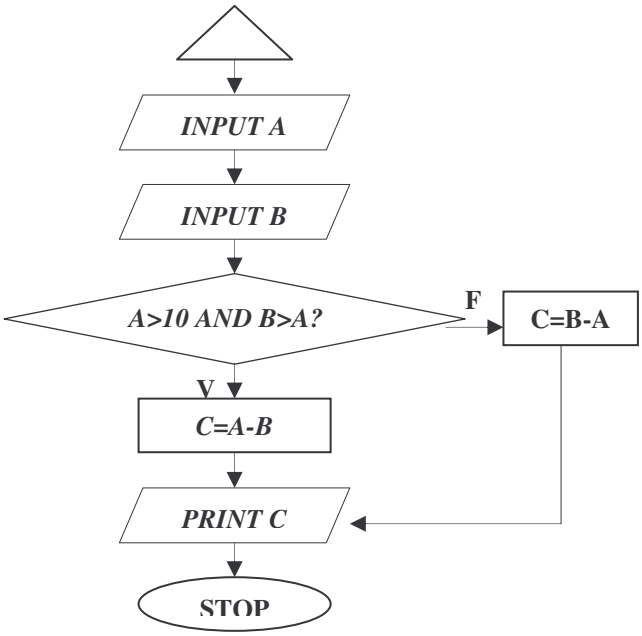
Come si è visto nell'esempio, al contrario di quanto accade nella lingua italiana, in BASIC vanno ripetuti entrambi i termini del controllo.

in italiano: i giudizi di Mario e di Aldo sono "ottimo"?

in BASIC: GMA$="ottimo" AND GAL$="ottimo"?

Esercizi:

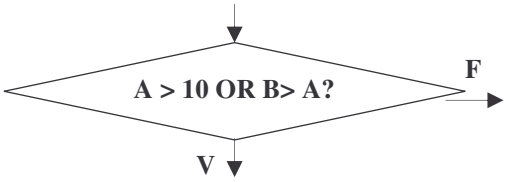
1) Analizza il seguente diagramma di flusso e rispondi alle domande sul fianco provando poi le singole soluzioni al calcolatore:



Se, alla richiesta del calcolatore, memorizzo nella variabile A il numero 6 e nella variabile B il numero 8, alla fine il calcolatore scriverà sullo schermo il numero _____.

Se, alla richiesta del calcolatore, memorizzo nella variabile A il numero 12 e nella variabile B il numero 10, alla fine il calcolatore scriverà sullo schermo il numero _____.

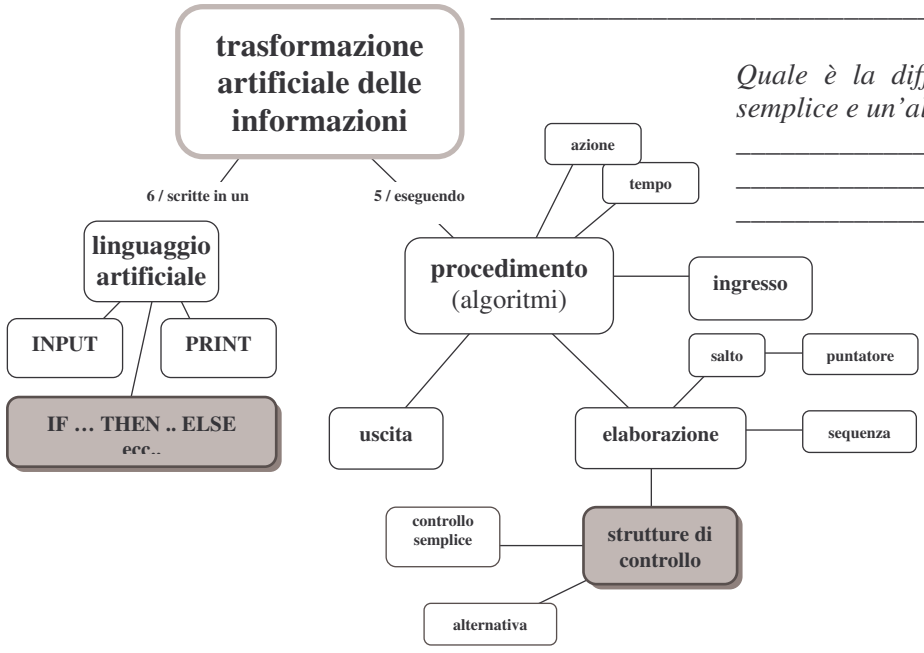
2) Modificando il controllo nel seguente modo:



Se, alla richiesta del calcolatore, memorizzo nella variabile A il numero 6 e nella variabile B il numero 8, alla fine in calcolatore scriverà sullo schermo il numero _____.

3) La mappa sul fianco evidenzia gli argomenti di cui ci siamo occupati e i loro collegamenti.
Quali sono le altre istruzioni BASIC studiate in questa fase e non collocate nel riquadro? _____
_____

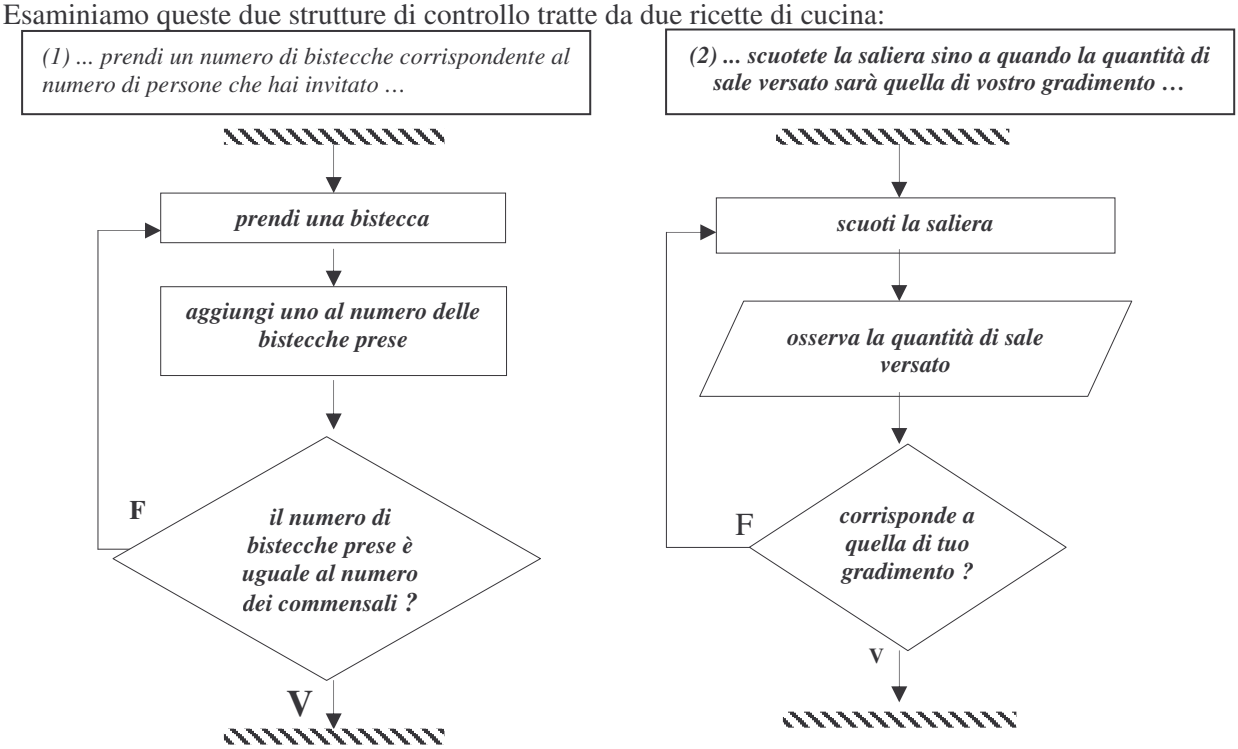
Quale è la differenza tra un controllo semplice e un'alternativa? _____
_____
_____
_____



LA RIPETIZIONE DEL CONTROLLO (ITERAZIONE, **CICLO**)

Tipi di cicli

Quando una o più azioni vengono ripetute sino al cambiamento della risposta ad un controllo abbiamo una ripetizione (o ciclo).
Esaminiamo queste due strutture di controllo tratte da due ricette di cucina:



Questi esempi illustrano due diversi tipi di ripetizione:

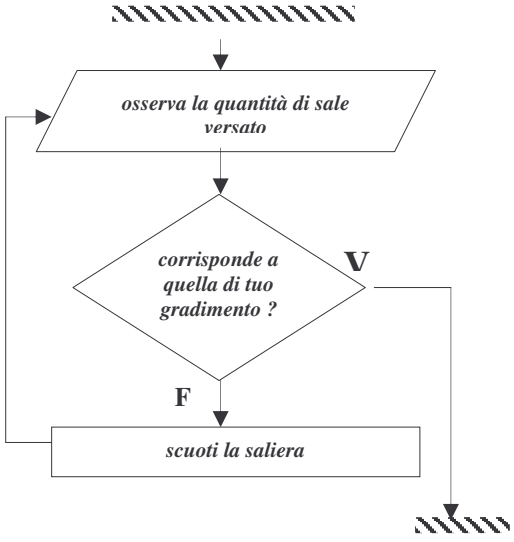
1) - nel primo esempio, ***ripetere contando***, il numero delle ripetizioni è conosciuto a priori perché il *numero delle bistecche da prendere* corrisponde al *numero dei commensali* (dato che si presuppone già presente nella memoria dell'esecutore). Per tenere il conto l'esecutore organizzerà un apposito spazio di memoria che conterrà il numero delle bistecche prese sino a quel momento. Il contenuto di questo spazio di memoria sarà incrementato di un'unità ogni volta che viene presa una bistecca.
2) - nel secondo esempio, ***ripetere attendendo un cambiamento***, non è possibile sapere a priori (cioè prima di entrare nel ciclo) quante ripetizioni saranno effettuate. L'informazione che deve essere controllata (*la quantità di sale versato*) proviene dall'esterno (dagli occhi) e dovrà essere confrontata con l'altra informazione (*la quantità di sale gradita*) presente nella memoria dell'esecutore. Si esce dalla ripetizione quando le due quantità coincideranno e dunque la risposta al controllo sarà Vero.

Il secondo esempio potrà essere rappresentato però anche da un diagramma di flusso molto diverso da quello visto sopra. Confronta i due e rispondi alle seguenti domande:

- Quali sono i due dati che vengono confrontati nel controllo?
  nome del 1° dato > _____
  nome del 2° dato > _____

- Quando sarà opportuno utilizzare il primo diagramma di flusso? _____
  _____

- Quando sarà opportuno utilizzare il diagramma di flusso collocato a destra? _____
  _____



Nel primo caso l'azione che viene ripetuta ("scuoti la saliera") è posta prima del controllo e andrà comunque eseguita almeno una volta. Questa soluzione non prevede dunque che l'esecutore rifiuti il sale. Questo ciclo viene chiamato a **condizione finale**.
Nell'ultimo grafico l'azione che viene ripetuta ("scuoti la saliera") è posta dopo il controllo e potrà anche non essere eseguita. E' una soluzione che prevede anche un esecutore che rifiuti il sale.
Questo ciclo viene chiamato a **condizione iniziale**.
Come vedremo, esiste anche la possibilità che la condizione sia **interna** al ciclo.

Abbiamo dunque visto che esistono due tipi di cicli:

- ***quello dove si ripete contando***
- ***quello dove si ripete attendendo un cambiamento***

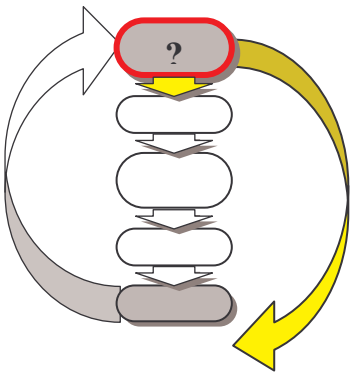
Ognuno di questi due tipi, a seconda di dove è collocato il controllo, può presentare tre situazioni diverse:

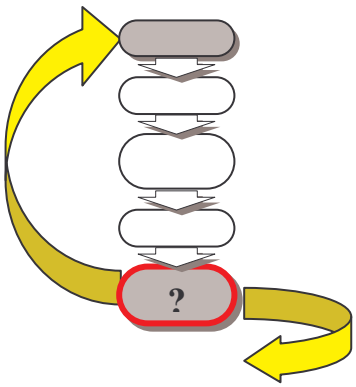
- ciclo a **condizione finale**
- ciclo a **condizione iniziale**
- ciclo a **condizione interna**

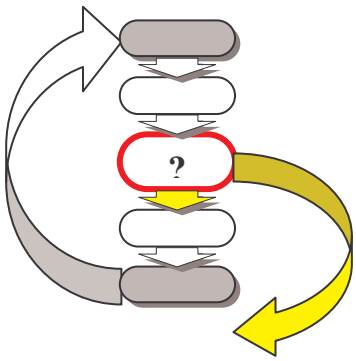
Gli esempi visti sono tratti dai procedimenti svolti per risolvere problemi quotidiani. Ritroveremo situazioni simili anche quando l'esecutore della procedura sarà un calcolatore.
Tutti i cicli possono essere gestiti dalle istruzioni di controllo IF ... THEN e IF ... THEN ... ELSE abbinati all'ordine GOTO. Il linguaggio GWBASIC e, ancora di più il VB, possiedono però istruzioni molto più funzionali studiate apposta per gestire queste strutture di controllo.

…puntatori di inizio, di fine ciclo e tipi di cicli

Per muoversi all'interno di un ciclo è fondamentale sapere dove esso inizia e dove finisce. Nei vecchi linguaggi era il numero di linea a svolgere la funzione di **puntatore di inizio** e di **fine ciclo**. Nel momento del controllo l'istruzione GOTO, abbinata all'indicazione del numero della linea di destinazione, permetteva di continuare la ripetizione sino a quando, il cambiamento di risposta al controllo consentiva l'uscita dal ciclo.
L'affermarsi, nei moderni linguaggi di programmazione, di appositi ordini che permettono sia la delimitazione che il controllo, ha determinato la rinuncia al numero di linea.
Il controllo, che potrà essere abbinato al puntatore di inizio ciclo, al puntatore di fine ciclo o anche essere interno al ciclo, determina sempre l'uscita dal ciclo. Gli ordini per gestire tutte queste tre situazioni sono presenti solo in versioni BASIC più evolute come il VISUAL BASIC.
Esamina i tre seguenti grafici indicando che cosa descrive ognuno di essi e che cosa descrivono le singole parti che lo compongono.







primo > _____

_____

secondo > _____

_____

terzo > _____

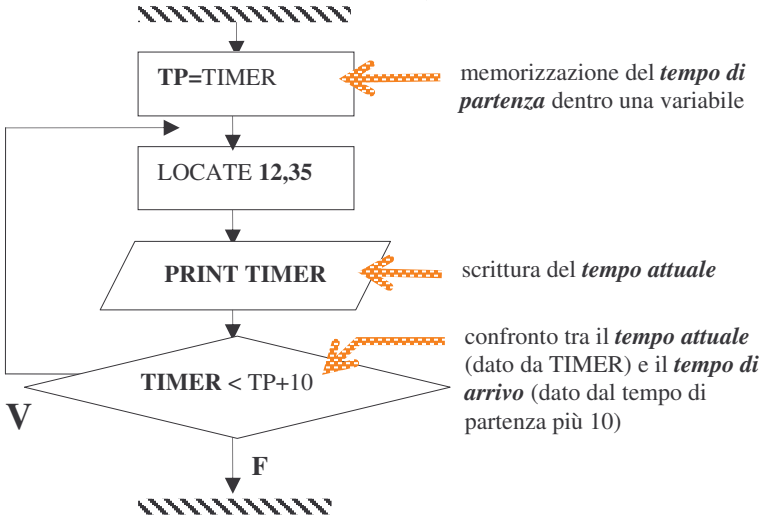
_____

Ripetere attendendo un cambiamento

Proviamo ora ad esaminare un caso di ripetizione in attesa di un evento esterno (il trascorrere di una certa quantità di tempo).

Esempio: Scrivere per 10 secondi in mezzo allo schermo (LOCATE 12,35) il contenuto di TIMER

	gwbasic
10	TP=**TIMER**
20	**LOCATE** 12, 35
30	**PRINT** TIMER
40	**IF** TIMER < TP+10 **GOTO** 20
50	……



Nell'esempio appena visto abbiamo realizzato un ciclo a condizione finale dove il ruolo di puntatore di fine ciclo è svolto dall'ordine IF … THEN, mentre l'inizio del ciclo è indicato dal numero di linea 20, al quale viene rinviata l'esecuzione nel caso di risposta "Vero" al controllo.
Per gestire i cicli di "ripetizione in attesa di un cambiamento" il GWBASIC possiede un apposito ordine che però è adatto a gestire solo cicli a condizione iniziale.

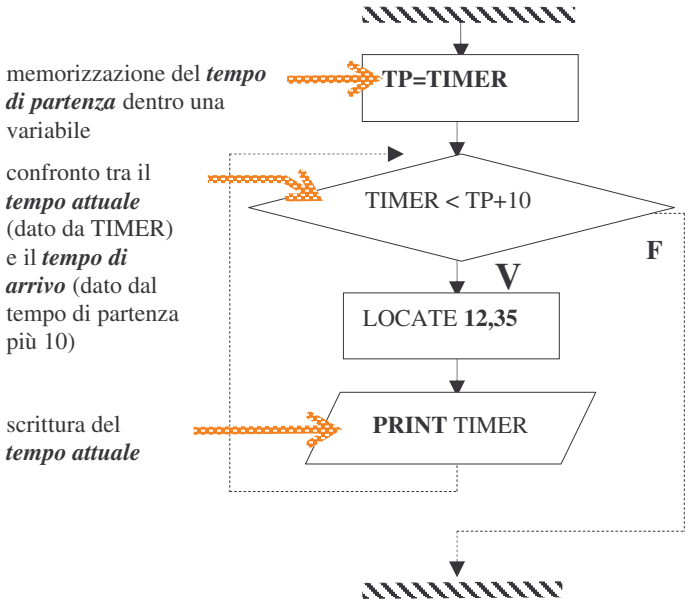
Si tratta di:

- **WHILE** *condizione* (svolge anche la funzione di puntatore di inizio ciclo)
  _____ (azioni che saranno ripetute solo in caso di risposta "vero" al controllo)
- **WEND** (svolge anche la funzione di puntatore di fine ciclo)

che corrisponde in lingua naturale alla proposizione:

mentre *condizione* **fai** *azioni che saranno ripetute solo in caso di risposta "vero"*

Volendo applicare l'ordine appena studiato all'esempio visto in precedenza, dovremo dunque portare all'inizio del ciclo il controllo:



In lingua naturale potremmo dire:
***mentre** il tempo attuale **è inferiore** al tempo di arrivo **scrivi** il tempo attuale*

Che nel programma sarà scritto così:

	gwbasic
10	TP=TIMER
20	**WHILE** TIMER < TP+10
30	**LOCATE** 12,35
40	**PRINT** TIMER
50	**WEND**
60	……

l'uso del tratteggio nella freccia indica un spostamento ad una linea diversa dalla linea seguente controllato da un ordine diverso da GOTO

In VISUAL BASIC

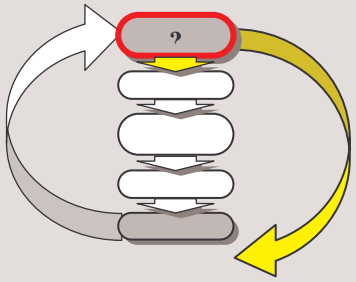
espansioni del linguaggio

Nel tipo di ciclo "ripetere attendendo un cambiamento", VB mette a disposizione istruzioni che permettono di gestire le tre diverse situazioni viste:

il ciclo a condizione iniziale viene gestito dall'istruzione:

Do *condizione*
istruzioni da ripetere
Loop

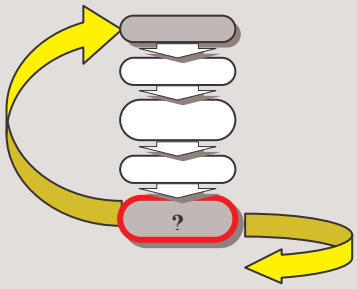
che in italiano corrisponde a:
fai *condizione azioni da ripetere*



il ciclo a condizione finale viene gestito dall'istruzione:

Do
istruzioni da ripetere
Loop condizione

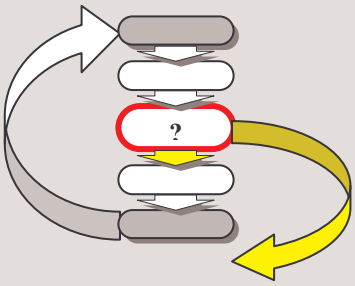
che in italiano corrisponde a:
fai *azioni da ripetere condizione*



il ciclo a condizione interna viene gestito dall'istruzione:

Do
azioni da ripetere
condizione **Exit Do ***
azioni da ripetere
Loop

che in italiano corrisponde a:
fai *azioni da ripetere condizione azioni da ripetere*



*** Exit Do** è in genere preceduto da un controllo con **If** *condizione* **Then**

… diversi utilizzi del ciclo con WHILE … WEND

la lettura del tempo (TIME$)

Il calcolatore possiede un orologio interno, che conta il tempo. In GWBASIC esso viene memorizzato in una variabile alfanumerica chiamata:

- **TIME$**

e composta di 8 caratteri. Due caratteri sono dedicati alle ore, due ai minuti e altri due ai secondi. Come segno di separazione vengono usati i due punti.

se ad esempio do l'ordine in diretta: PRINT TIME$ e il calcolatore scrive 08:12:07 vuol dire che, sull'orologio del calcolatore, sono le ore otto, dodici minuti e sette secondi

La variabile TIME$ può essere riassegnata:

ad esempio con TIME$="00:00:00" posso azzerare l'orologio interno.

Come con tutte le variabili alfanumeriche, è possibile effettuare dello slicing

ad esempio con MID$(TIME$,7,2) il calcolatore mi fornirà solo i secondi.

il controllo del tempo (TIMER)

Con la variabile numerica

- **TIMER**

il calcolatore conta i secondi trascorsi dalla mezzanotte. Essa non è riassegnabile.
Se si scrive PRINT TIMER il calcolatore scrive il tempo trascorso dalla mezzanotte aggiornato ai centesimi di secondo (ad esempio 68354.18 dove l'otto in coda mi indica i centesimi di secondo). Per far trascorrere un certo tempo (*tempo di attesa*) è necessario prima memorizzare il *tempo iniziale*, compresi i centesimi di secondo, in una variabile (ad esempio **tp=timer**).
Seguirà il controllo del tempo trascorso dato dall'espressione:
tempo attuale = tempo iniziale + tempo di attesa
Ad esempio se si fa un controllo **timer=tp+20** il calcolatore va a leggere il tempo attuale (compresi i centesimi di secondo) e lo va a confrontare con il tempo iniziale (compresi i centesimi di secondo) più 20 secondi. Se ad esempio il tempo iniziale era ***68354.18***, per dare risposta vero alla tua domanda il calcolatore, nel momento in cui legge TIMER dovrebbe trovare esattamente ***68374.18*** cosa che, vista la velocità a cui procede TIMER è quasi impossibile. In genere viene un numero minore o maggiore del numero a cui si deve arrivare.
Per questo motivo nei controlli del tempo non si usa mai il segno = (uguale) ma si usano i segni > (maggiore) o < (minore).
Per un attesa di 20 secondi si userà:
10 **tp=timer**
20 **while timer<tp+20 : wend**

In VISUAL BASIC

espansioni del linguaggio

Numerose e rilevanti sono le novità che WINDOWS, e dunque VB, aggiunge e che fanno del calcolatore un potente strumento per il controllo e la misurazione del tempo, con date ammesse comprese tra il 1 gennaio 100 e il 31 dicembre 9999.
Oltre all'oggetto ***orologio*** che, come gli altri oggetti collocabili nelle applicazioni, impareremo ad utilizzare l'anno prossimo, VB dispone delle seguenti istruzioni:

Date > restituisce la data di sistema in formato *gg-mm-aaaa oppure gg-mm-aa*
ad esempio sulla finestra immediata con Print Date il calcolatore scrive 24/07/02
Date è una variabile riassegnabile
ad esempio scrivendo sulla f. i. Date = "24/07/02" il calcolatore aggiorna la data
Time > restituisce l'ora di sistema in formato *hh:mm:ss*
Now > restituisce una variabile che rappresenta la data e l'ora di sistema in formato *data.ora*
DateValue > restituisce una valore relativo ad una data definita, in forma alfanumerica, dal giorno, dal mese e dall'anno, permettendo così di calcolare i giorni trascorsi da un'altra data.
ad esempio se Date = 24/07/02 battendo sulla f.i.
Print Date - DateValue("10/02/00")
il calcolatore scriverà 895 e cioè il numero di giorni trascorsi tra il *24/07/02 e il 10/02/00*
Day (*data***) >** restituisce un numero compreso fra 1 e 31 corrispondente al giorno della data inserita come parametro
ad esempio se Date = 24/07/02 battendo sulla f.i.
Print Day(Date) oppure Print Day(Now)
il calcolatore scriverà 24
Month (*data***) >** restituisce un numero compreso fra 1 e 12 corrispondente al mese della data inserita come parametro
Year (*data***) >** restituisce un numero compreso fra 100 e 9999 corrispondente all'anno della data inserita come parametro
Weekday (*data***) >** restituisce un numero compreso fra 1 (per domenica) e 7 (per sabato) corrispondente al giorno della settimana della data inserita come parametro
ad esempio se Date = 24/07/02 battendo sulla f.i.
Print Weekday(Date) oppure Print Weekday(Now)
il calcolatore scriverà 4 perché il 24 luglio 02 è stato un sabato mentre con

Print Weekday(DateValue("01,08,02"))
il calcolatore scriverà 5 perché il 1 agosto 02 è stato un giovedì

Hour (*ora di sistema***) >** restituisce un numero compreso fra 0 e 23 corrispondente all'ora inserita come parametro
ad esempio se Time = 19.48.44 battendo sulla f.i.
Print Hour(Time) oppure Print Hour(Now)
il calcolatore scriverà 19

Minute (*ora di sistema***) >** restituisce un numero compreso fra 0 e 59 corrispondente ai minuti dell'ora inserita come parametro

Second (*ora di sistema***) >** restituisce un numero compreso fra0 e 59 corrispondente ai secondi dell'ora inserita come parametro

… diversi utilizzi del ciclo con WHILE … WEND

per estrarre a sorte (RND e RANDOMIZE)

Con:

➢ RND

il calcolatore estrae un numero compreso tra 0 e 1 da una sequenza di numeri generati dal calcolatore. Per generare interi casuali in un dato intervallo, utilizzare la seguente formula:

INT((*limitesup - limiteinf + 1*) * **RND** + *limiteinf)*

dove: *limiteinf* è il numero più basso da estrarre
limitesup è il numero più alto da estrarre

*Esempio: con A=INT((20-10+1)*RND+10) il calcolatore colloca nella variabile A un numero compreso tra 10 e 20*

Come abbiamo detto, con RND il numero viene estratto da una sequenza fissa. Di conseguenza il calcolatore, dopo l'ordine di esecuzione (RUN), darà gli stessi numeri.

Per rendere realmente casuali le estrazioni dovrà essere prima inserito l'ordine:

➢ RANDOMIZE *numero*

dove: *numero* è il numero (compreso tra –32768 a 32767) da cui partire per l'estrazione.
Comunque l'estrazione è così veloce da rendere indifferente il numero dato

Esercizio: fare un programma con il quale il calcolatore chiede all'operatore di battere un numero compreso tra 0 e 10. Successivamente il calcolatore estrae un numero compreso tra 0 e 10 e lo confronta con il numero dato dall'operatore. Se sono uguali il calcolatore scriverà "Hai vinto!" altrimenti scriverà "Hai perso!".
Provarlo poi tra i componenti del gruppo vediamo chi vince per primo.

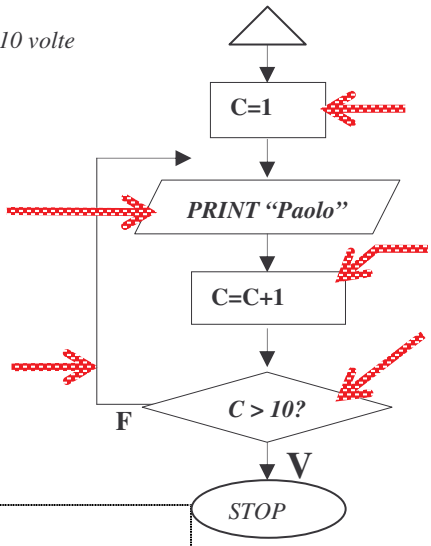
10	gwbasic

Quali modifiche vanno introdotte se il numero da battere e da estrarre dovrà essere compreso tra 1 e 10 invece che tra 0 e 10?

Ripetere contando

Quando la ripetizione consiste nella ripetizione di un'azione o di una serie di azioni per un numero predeterminato di volte è necessario utilizzare una variabile numerica per tenere il conto del numero di volte in cui l'azione sarà eseguita.
Proviamo a risolvere un problema di questo tipo utilizzando ancora l'istruzione IF ... THEN ...

Esempio: scrivi il nome "Paolo" 10 volte



assegnazione iniziale
C è la variabile numerica usata come "contatore"
1 è il numero da assegnare inizialmente al "contatore"

azione che viene ripetute più volte

incremento
viene aggiunta un'unità alla variabile "contatore"

rinvio istruzione che riporta l'esecuzione all'azione seguente l'assegnazione iniziale

controllo
viene controllato il contenuto della variabile "contatore". Se esso ha superato il numero 10 il calcolatore termina l'esecuzione, altrimenti un rinvio (che nel programma sarà gestito dall'istruzione di salto) riporterà l'esecuzione all'azione da ripetere

10	gwbasic C=1
20	**PRINT** "Paolo"
30	C=C+1
40	**IF** C>10 **THEN STOP ELSE GOTO** 20

Con la soluzione adottata, nel momento in cui viene svolta l'azione da ripetere (PRINT "Paolo") il numero inserito nella variabile "contatore" corrisponderà al numero di parole scritte. L'incremento situato prima del controllo fa si che la variabile arrivi al controllo con un numero maggiore di un'unità rispetto al numero di operazioni svolte. Per questo motivo nel controllo si verifica se il numero limite è stato superato (C>10) e non se esso è stato raggiunto (C=10).

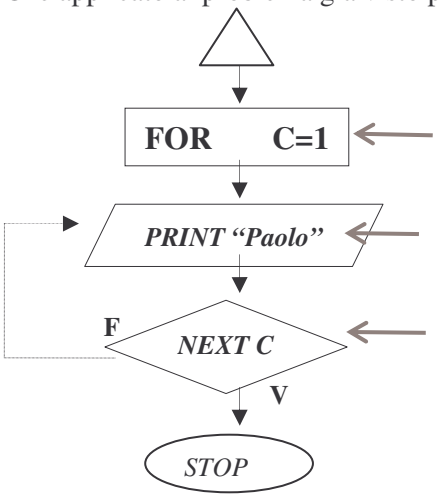
Esiste però un apposito ordine per gestire cicli di questo tipo:

FOR *variabile* **=** *numero iniziale* **TO** *numero finale*

_____ (azione o azioni che vengono ripetute più volte)

NEXT *variabile*

Che applicato al problema già visto precedentemente funzionerà così:



assegnazione iniziale
oltre a svolgere la funzione di puntatore di inizio ciclo, questa istruzione organizza una particolare variabile "contatore" nella quale viene collocato 1 come numero di partenza e viene anche memorizzato 10 come numero massimo da non superare

azione che viene ripetute più volte

incremento / controllo / rinvio
oltre a svolgere la funzione di puntatore di fine ciclo, questa istruzione svolge automaticamente alcune operazioni:

1- incrementa automaticamente il numero contenuto nel contatore di un unità
2- controlla se il "contatore" ha superato il numero massimo:
   - se ciò si è verificato il calcolatore continuerà l'esecuzione del programma alla linea successiva
   - se ciò non si è verificato il calcolatore rinvierà automaticamente l'esecuzione all'istruzione che segue l'assegnazione iniziale

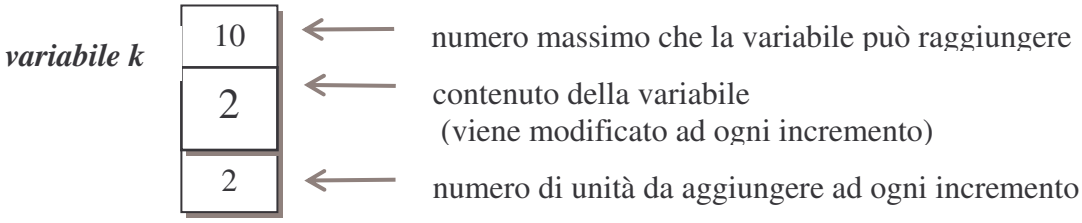
E' importante notare come l'incremento, gestito automaticamente dal NEXT, è sempre di un unità. E'

possibile però ordinare al calcolatore un aumento della variabile "contatore" diverso da uno aggiungendo l'ordine:

STEP *incremento che si vuole ottenere dal calcolatore*

Ad esempio con la proposizione: FOR K=2 TO 20 STEP 2

il calcolatore organizza una variabile strutturata nel seguente modo:



il calcolatore organizza una variabile che ha come numero di partenza 2, come numero di arrivo 20 e come "passo" 2. Questo significa che ad ogni esecuzione dell'ordine NEXT la variabile k sarà aumentata di due unità Il ciclo sarà dunque eseguito 10 volte.
La variabile assegnata con FOR è dunque una variabile particolare che memorizza, in appositi spazi, oltre al suo valore, anche il numero massimo che la variabile può raggiungere e l'eventuale "passo", se diverso da +1.

Se necessario il "passo" potrà avere un valore negativo.
Ad esempio con la proposizione FOR C=20 TO 2 STEP -2 il calcolatore inserirà inizialmente nella variabile C il numero 20 a cui sottrarrà due unità ad ogni esecuzione di NEXT sino a raggiungere il numero 2 che è il minimo prefissato.

In VISUAL BASIC

espansioni del linguaggio
La gestione dei cicli con FOR NEXT (attendere contando) non subisce sostanziali variazioni in VB. Vi sono alcune novità collegate alla mancanza dei numeri di linea.

… diversi utilizzi del ciclo con FOR … NEXT

Oltre che per ripetere un determinato numero di volte una o più istruzioni, il ciclo gestito da FOR può essere utilizzato per altre necessità:
a) L'uso del ciclo di ritardo è molto simile a ciò che accade quando giochiamo a nascondino. Chi "sta sotto" deve contare fino a 50 o a 100. Ovviamente questo non serve come ripasso di matematica; il contare serve a far perdere tempo a chi "sta sotto" affinché, i giocatori abbiano il tempo di nascondersi.

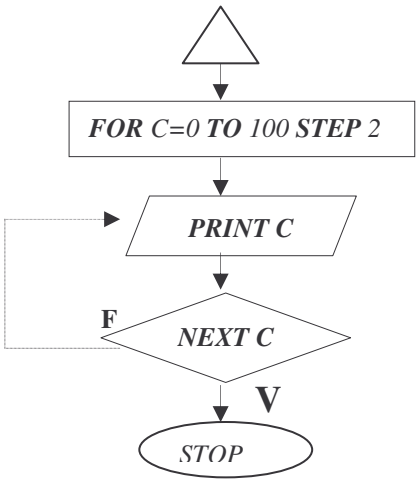
Se inseriamo in un programma la proposizione: ***FOR K=1 TO 2000: NEXT K***

il calcolatore si fermerà in questo punto a contare. Si viene così a creare una pausa nell'esecuzione del programma.

Nota Bene: quando utilizziamo cicli di ritardo dobbiamo tener conto che il numero di operazioni che un calcolatore è in grado di svolgere in un secondo (viene misurata in herz) è molto elevata è può essere molto diversa a seconda del modello posseduto. Per questo motivo se si vuole essere precisi nella misurazione del tempo di ritardo conviene controllare la variabile TIMER.

b) Frequentemente i cicli vengono usati in situazioni nelle quali ciò che interessa è il numero contenuto nella variabile "contatore".
Esempio: scrivere tutti i multipli di 2 da 0 a 100

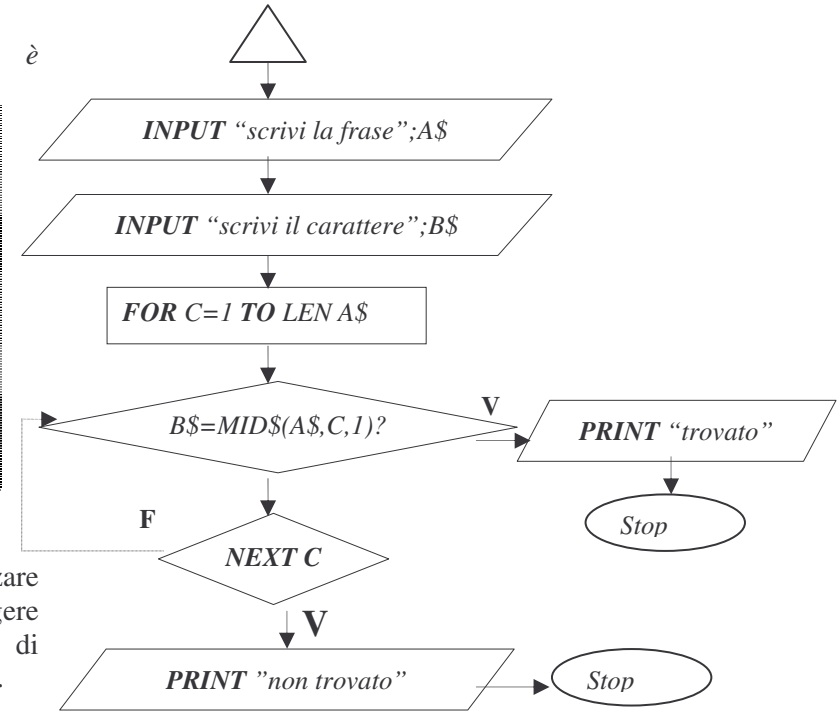
	gwbasic
10	**FOR** C=0 **TO** 100 **STEP** 2
20	**PRINT** C
30	**NEXT** C
40	**STOP**



c) Come abbiamo visto nella precedente U.D. Algoritmi, una variabile alfanumerica è un insieme di byte ognuno dei quali memorizza un carattere ed è contrassegnato da un numero crescente che lo rende raggiungibile separatamente dagli altri.
Ad esempio: se nella variabile A$ vi è la parola "scolaro" con MID$(A$,3,1) potrò ottenere il carattere "o".
Una variabile "contatore" può essere utilizzata per scorrere i caratteri di una variabile alfanumerica evitando di ripetere molte volte la stessa istruzione.

Esempio: trovare se un carattere è presente in una frase.

	gwbasic
10	**INPUT** "scrivi la frase";A$
20	**INPUT** "scrivi il carattere";B$
30	**FOR** C=1 **TO** LEN (A$)
40	**IF** B$=MID$(A$,C,1) **THEN** **PRINT "**trovato": **STOP**
50	**NEXT** C
60	**PRINT** "non tovato"
70	**STOP**



Come abbiamo visto è possibile utilizzare una variabile "contatore" per raggiungere un dato depositato in uno spazio di memoria contrassegnato con un numero.

Ed ora al lavoro!

Esegui i seguenti esercizi utilizzando la scheda di programmazione B:

a) data una frase il calcolatore deve contare il numero di parole da cui è formata (ricordare che le parole sono separate tra di loro da spazi).

b) data una frase e una parola da cercare nella frase, il calcolatore segnala se la parola è presente.

c) data una frase, una parola da cercare nella frase ed una con cui sostituire la precedente (con lo stesso numero di caratteri di quello da cercare), il calcolatore effettuerà la sostituzione scrivendo la nuova frase sullo schermo

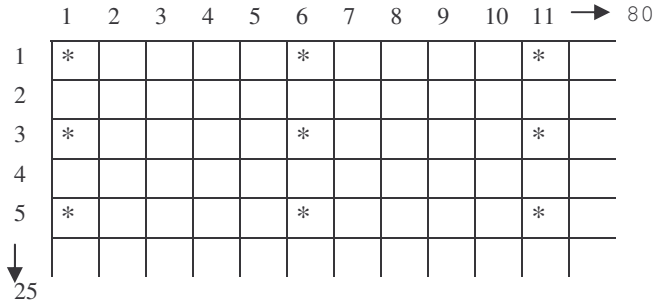
d) data una frase, una parola da cercare nella frase ed una con cui sostituire la precedente, il calcolatore effettuerà la sostituzione scrivendo a nuova frase sullo schermo.**…**

La nidificazione dei cicli

Abbiamo già incontrato un esempio di **ciclo nidificato** a pag. 4 quando abbiamo realizzato il diagramma di flusso della ricetta **RISOTTO CON IL SEDANO DI VERONA.** Nei procedimenti di informatica capita frequentemente di avere **cicli nidificati**.
Esempio:
vogliamo collocare una serie di asterischi sullo schermo del calcolatore sulle righe dispari dalla uno alla ventuno e collocandone uno ogni cinque colonne a partire dalla colonna 1 sino alla 76. Realizza, utilizzando l'apposita scheda, il diagramma di flusso e il programma

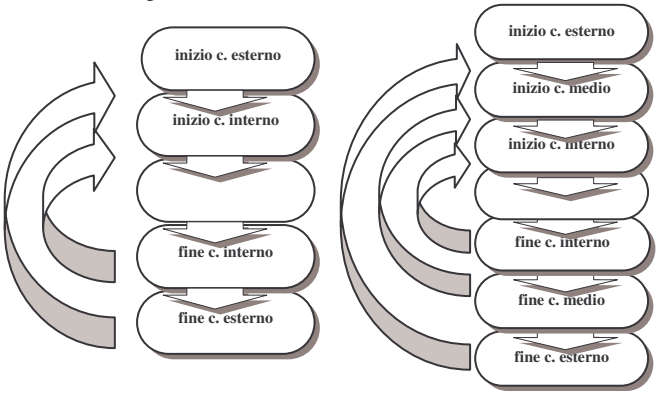
Si dovrà seguire il seguente schema:

Nota Bene: lo schermo GW_BASIC dispone di 25 righe (numerate da 1 a 25) e di 80 colonne (numerate da 1 a 80). La venticinquesima riga sarà però utilizzabile solo se ripulita dai codici dei tasti funzione mediante l'ordine:
- **KEY OFF**



Per collocare il calcolatore in posizione di scrittura utilizzare LOCATE con le coordinate di riga e colonna, che dovranno essere gestite da due appositi cicli. Seguirà PRINT""; che assicurerà la scrittura dell'asterisco (il punto e virgola serve ad evitare, dopo la scrittura, il rinvio automatico alla riga successiva*

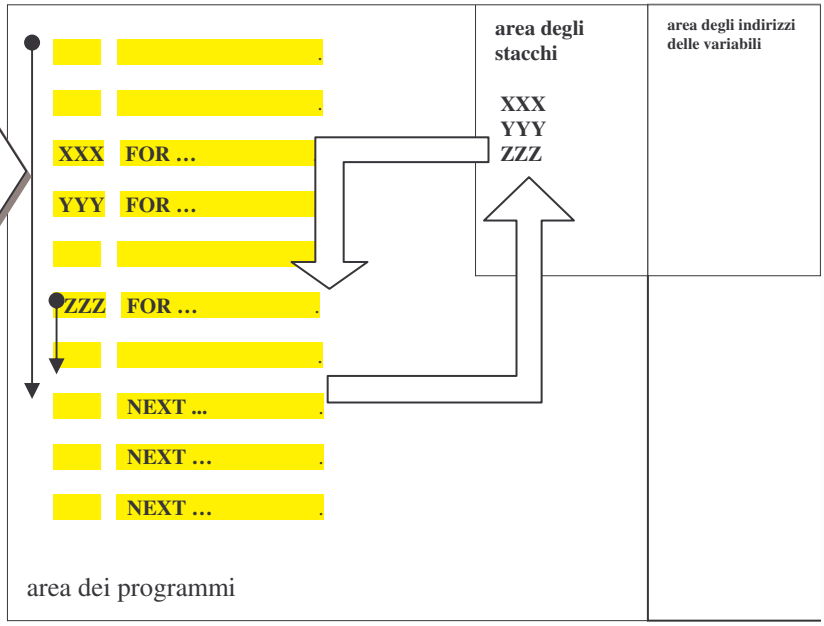
Anche nell'esempio appena visto si possono cogliere le caratteristiche fondamentali dei cicli nidificati:
- se i cicli sono due, uno dei due è completamente interno (ciclo interno) all'altro (ciclo esterno); questa situazione è detta ***nidificazione***
- se vi sono più di due cicli, essi devono comunque essere nidificati l'uno dentro l'altro



l'area degli stacchi

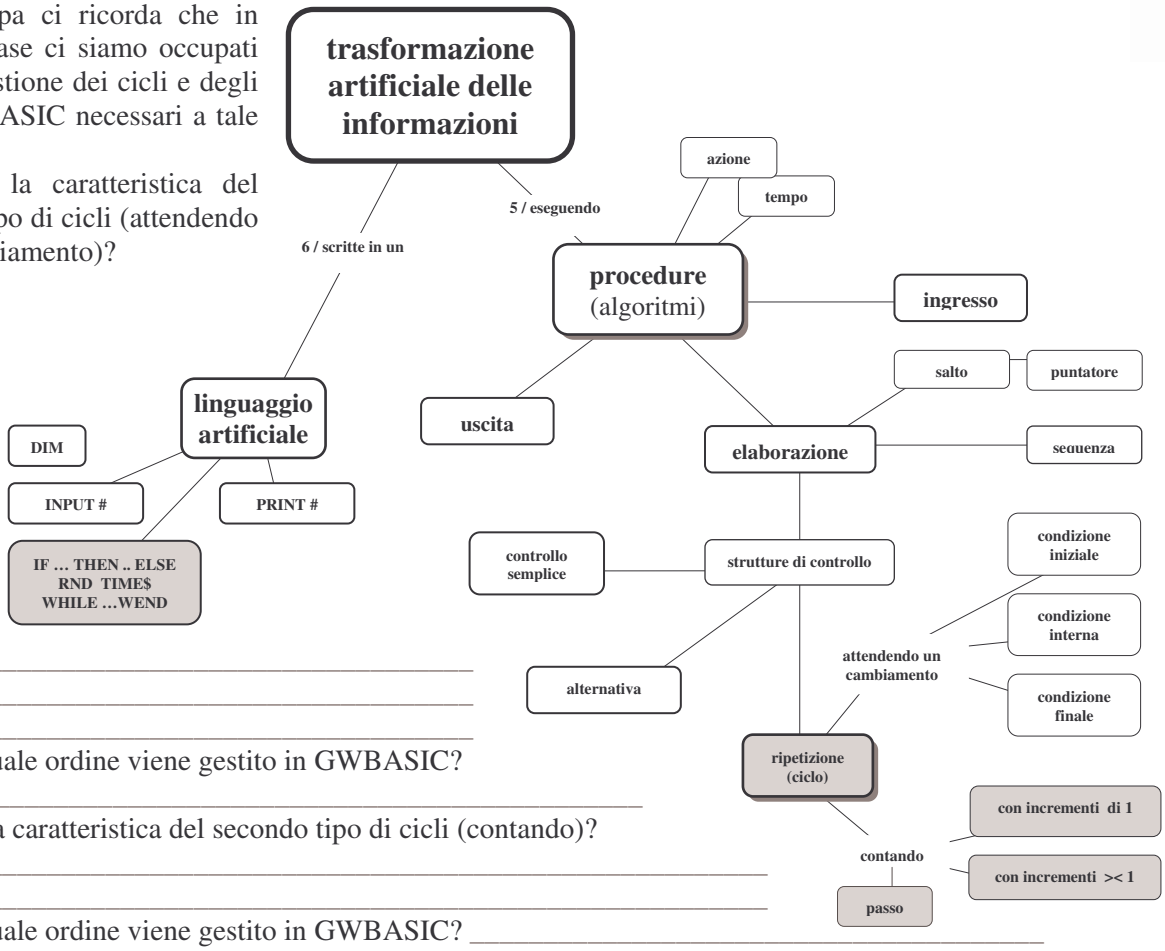
Ma come fa il calcolatore a sapere a quale linea deve tornare quando incontra un NEXT o un WEND. E' importante sapere che quando il calcolatore organizza la propria RAM, oltre alle aree che già conosciamo, ne organizza un'altra strettamente legata all'area dei programmi: è **l'area degli stacchi**. Ogni volta che, durante l'esecuzione di un programma, il calcolatore incontra l'ordine FOR oppure WHILE ne annota l'**indirizzo di memoria** in questa apposita area. Attenzione! Il calcolatore annota l'indirizzo di memoria e non il numero di linea che, come sappiamo, è un puntatore a nostra disposizione per collocare le istruzioni sequenzialmente ma non c'entra con il "numero della casellina" dove il calcolatore ha collocato il dato.
Appena incontra il NEXT o il WEND il calcolatore prosegue l'esecuzione dall'istruzione che segue l'ultimo indirizzo registrato nell'area degli stacchi avendo anche cura di cancellarlo subito dopo l'utilizzo se il controllo segnala la fine del ciclo.
L'area degli stacchi è una pila, cioè una struttura di dati dove l'ultimo dato arrivato viene prelevato per primo (funziona proprio come una pila di piatti da lavare).
Queste modalità di funzionamento spiegano anche il meccanismo delle nidificazioni. Mettiamo di avere un programma con tre cicli nidificati con il FOR .. NEXT. Durante l'esecuzione il calcolatore incontrerà prima i tre FOR e ne annoterà, uno dopo l'altro, i tre indirizzi di memoria. Il controllo del primo NEXT porterà il calcolatore a proseguire dall'ultimo indirizzo segnato nell'area degli stacchi che però verrà cancellato in caso di uscita dal ciclo. Questo spiega perché, incontrando il secondo NEXT, il calcolatore troverà nell'area degli stacchi, come ultimo indirizzo segnato, l'indirizzo del secondo NEXT.

schema che illustra il funzionamento del calcolatore durante l'esecuzione di un ciclo interno. L'uscita dal ciclo comporterà la cancellazione dall'area degli stacchi dell'indirizzo dell'ultimo FOR



La mappa ci ricorda che in questa fase ci siamo occupati della gestione dei cicli e degli ordini BASIC necessari a tale scopo.

Qual è la caratteristica del primo tipo di cicli (attendendo un cambiamento)?



_____

_____

_____

E con quale ordine viene gestito in GWBASIC?

_____

Qual è la caratteristica del secondo tipo di cicli (contando)?

_____

_____

E con quale ordine viene gestito in GWBASIC? _____

Che cosa è il passo e quando viene gestito automaticamente dal calcolatore?

_____