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

Unità di Apprendimento PROGRAMMI (10/2012)

area delle Informazioni / 2 media

1. Il controllo

che cosa è un controllo?

1.a – la struttura di controllo

- il controllo semplice
- l'alternativa
. il salto
. il ciclo

1.a – la rappresentazione del percorso con i diagrammi di flusso

la classificazione delle azioni con parallelogrammi, rettangoli, rombi e frecce

1.b – il programma

l'inserimento di un controllo utilizzando **IF** ... **THEN** .. **(ELSE..)**

- sul problema 01
  - l'uso della scheda di programmazione
  - verifica su un nuovo problema
- sul problema 02
  - verifica su un nuovo problema

e ora al lavoro

- l'**uso dei connettivi nei controlli**
  unire due o più controlli con **AND**, **OR** e **NOT**
  la ricerca in **INTERNET**
- **altre possibilità per gestire i controlli**
  l'uso di **END IF** per gestire percorsi complessi
  l'uso di **SELECT CASE** per gestire esiti multipli ad un controllo

e ora al lavoro

2. La ripetizione del controllo (iterazione, **ciclo**):

ripetere contando / ripetere attendendo un cambiamento

con *condizione finale* / con *condizione iniziale*

un tempo: il numero di linea come **puntatore** di inizio e di fine ciclo
il controllo abbinato all'ordine **GOTO**

2.a – ripetere attendendo un cambiamento

- il puntatore di inizio ciclo **DO** e di fine ciclo **LOOP** / l'uso di **EXIT DO**
- il controllo con **UNTIL** (fino a quando) e con **WHILE** (mentre)
- la lettura del **tempo** con **TIME$( )** e **DATE$( )**
- l'**estrazione** a sorte con **RND( )**
- la **nidificazione** dei cicli
- il controllo delle **variazioni di un sensore**

e ora al lavoro

2.b - ripetere contando

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

- diversi utilizzi del ciclo con FOR ... NEXT
  - utilizzare il numero della variabile che conta
  - contare i caratteri di una variabile alfanumerica

e ora al lavoro

3. Risolvere i problemi con i fogli elettronici

L'impostazione del foglio di lavoro.
La risoluzione del problema
Verifica su un nuovo del problema

- *e ora al lavoro*

4. Programmi complessi

la gestione di un ambiente di lavoro

4.a – programmare utilizzando la programmazione strutturata

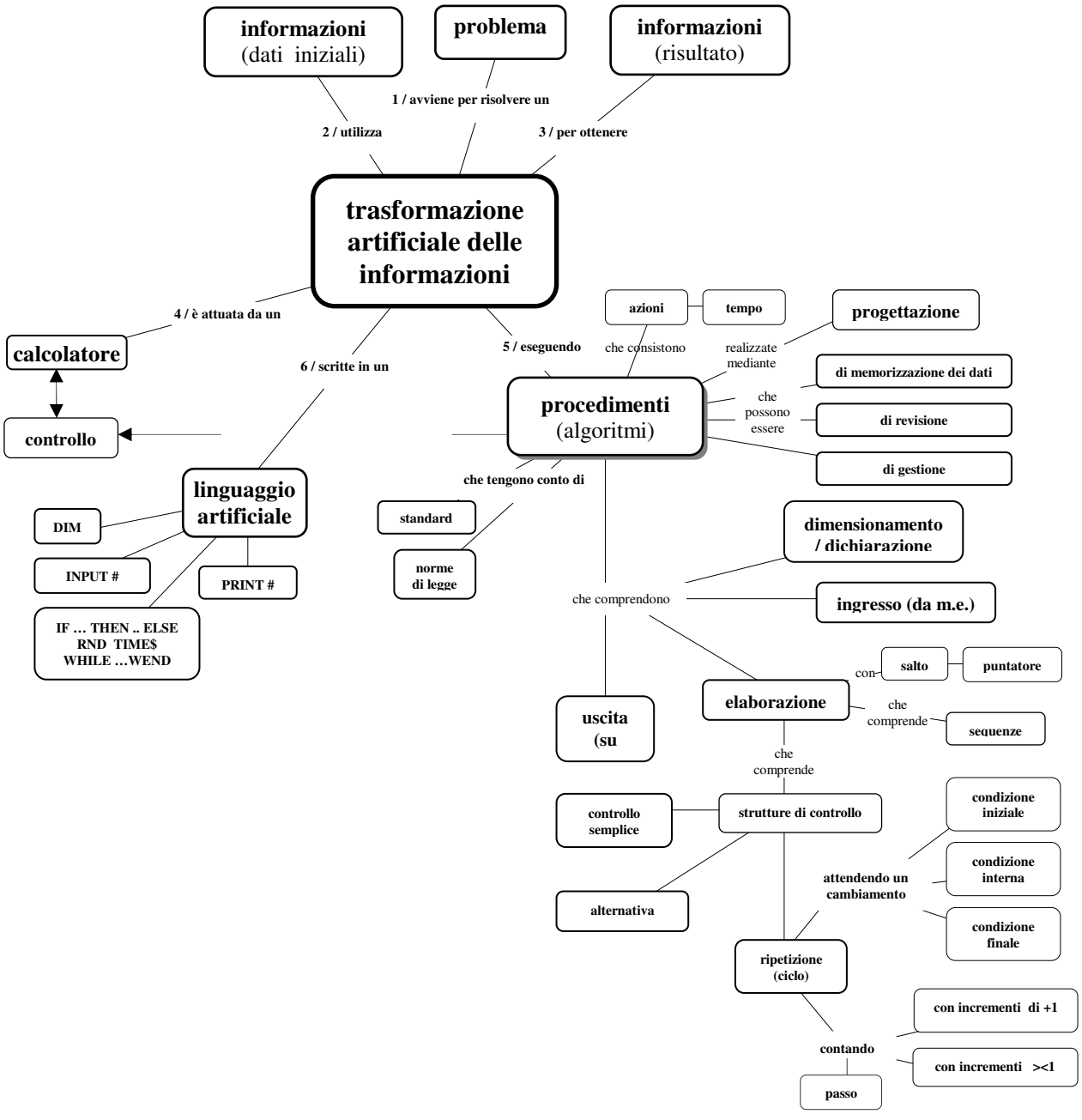
- la procedura principale o menù, le altre procedure definite con **SUB** ... ed **END SUB**, le procedure comuni
- variabili globali e variabili locali, l'ordine **GLOBAL**
- la gestione con chiamate **CALL ...** e ritorni **RETURN** e la possibilità di trasferire valori di variabili

e ora al lavoro

- **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



IL CONTROLLO

che cosa é un controllo?

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.

Anche nella risoluzione di problemi può capitare molto frequentemente di trovare dei controlli. Vediamo ad esempio due problemi già esaminati nell'ambito dellU.di A. Algoritmi.

Problema specifico 01

Una società immobiliare ha costruito degli appartamenti spendendo 1000 € al metro quadro. Ne rivende uno di 80 metri quadri al prezzo di 1400 € al metro quadro.
Quale sarà il guadagno della società?

Problema specifico 02

Un muratore di un'impresa edile percepisce una base mensile di 1200 €. Va poi aggiunta la paga per le 40 ore di straordinario che vengono pagate 20 € l'ora. Qual è la paga definitiva del muratore.

Riprendiamo il lavoro svolto: il passaggio dal problema specifico a quello generalizzato, il grafo di scomposizione, l'analisi dei dati, la procedura in linguaggio di progetto e programma. Effettuiamo poi la verifica sul calcolatore utilizzando sia i dati del problema specifico, sia altri dati suggeriti dall'insegnante. Come sappiamo i programmi realizzati non risolvano più un solo problema, ma un'ampia classe di problemi simili.
Vi sono però alcune situazioni non coperte dall'algoritmo e dal programma studiati. Sul primo problema facciamo un esempio:

la società immobiliare decide di fare uno sconto di 1/10 (dato costante) sul prezzo di vendita dell'appartamento in caso di appartamenti grandi (a partire da 100 mq).

… e un altro sul secondo problema:

l'impresa edile decide di premiare la disponibilità del dipendente ad effettuare ore di straordinario anche nei giorni festivi con una aggiunta alla paga base.

Che cosa hanno in comune queste due situazioni e come sia possibile farle risolvere al calcolatore?

_____

_____

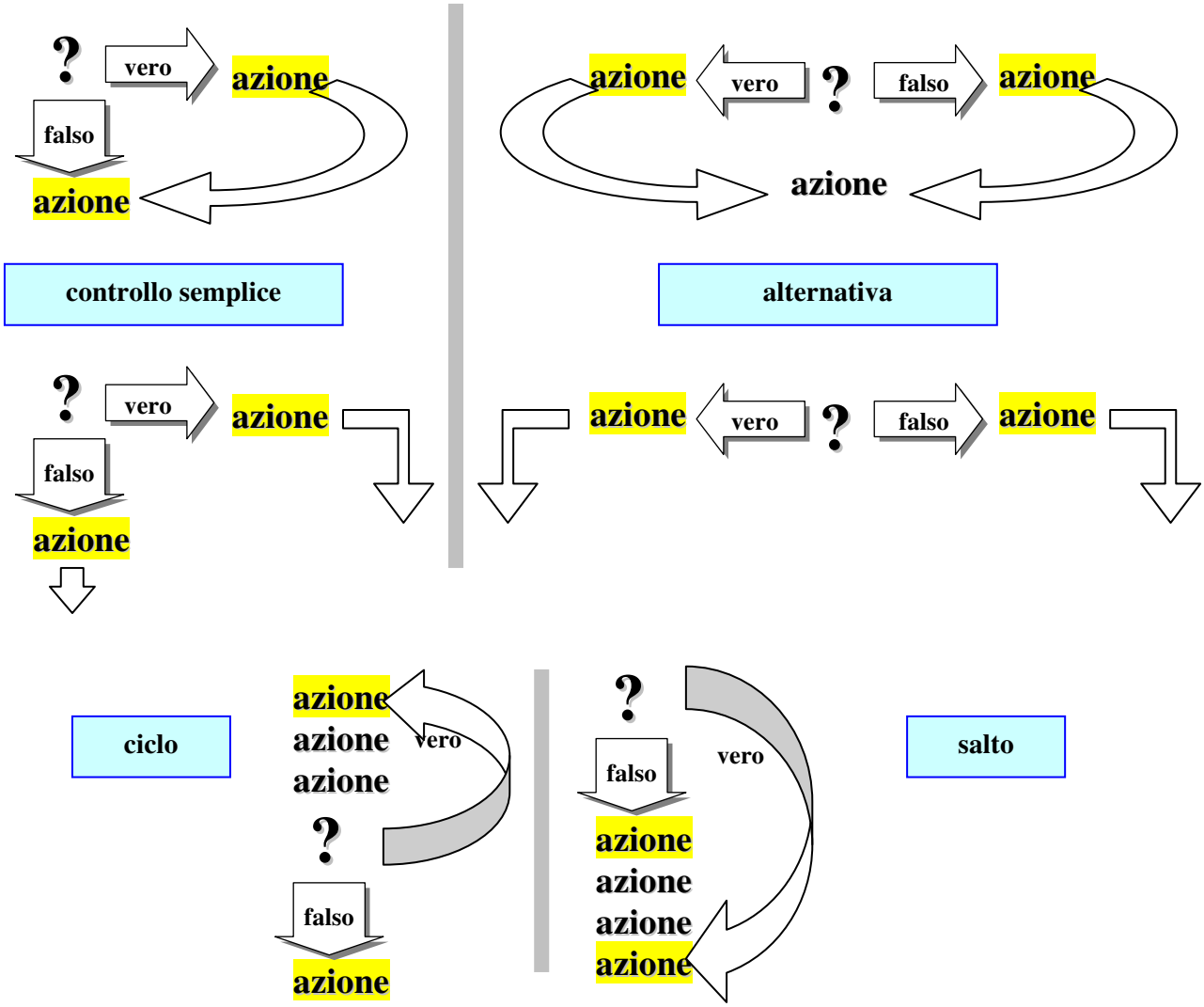
La struttura di controllo *

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 può prevedere che ***delle azioni possano essere eseguite solo in caso di risposta Vero***. Oppure che ***delle azioni vengano eseguite solo in caso di risposta Vero ed altre azioni in caso di risposta Falso***. Il controllo può anche comportare una diversificazione permanente dei due percorsi.

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.



Con l'introduzione del controllo nei procedimenti sposteremo al nostra attenzione dall'attività di analisi, che faremo mentalmente senza utilizzare più i grafi di scomposizione, all'individuazione del percorso di risoluzione che richiederà l'utilizzo di un apposito linguaggio grafico, quello dei **diagrammi di flusso**.

La rappresentazione del percorso con i diagrammi di flusso
la classificazione delle azioni con parallelogrammi, rettangoli, rombi e frecce

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.
Vengono utilizzati i seguenti codici:

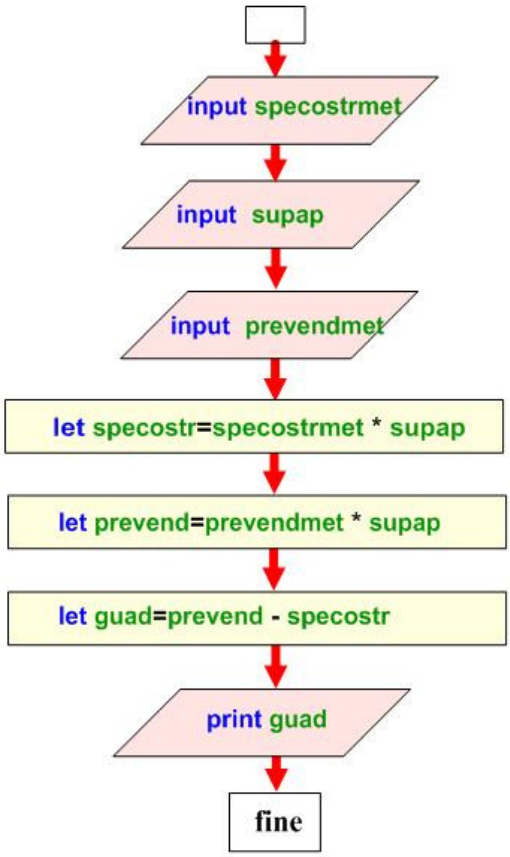
- il **parallelogramma** indica una azione in cui si ha un ingresso o un'uscita di informazioni (dalla mente di chi esegue la procedura)
- il **rettangolo** indica un'azione di elaborazione
- il **rombo** rappresenta il controllo e contiene all'interno il confronto tra i due dati. Alla domanda contenuta all'interno del rombo vi sono due risposte ***vero*** e ***falso***. Nei diagrammi di flusso il rombo è dunque l'unica figura da cui partono due frecce.
- la **freccia** indica il passaggio da un'azione a quella successiva
- un **piccolo rettangolo** indica l'inizio del programma, mentre un altro con scritto *fine* ne indica la fine

Il programma
l'inserimento di un controllo utilizzando IF ... THEN .. (ELSE..)

sul problema 01

- ***il diagramma di flusso***

A fianco vediamo rappresentato, con un diagramma di flusso, l'algoritmo del problema 01. All'interno dei grafi le istruzioni vengono scritte subito in BASIC.



- ***e il programma in BASIC***

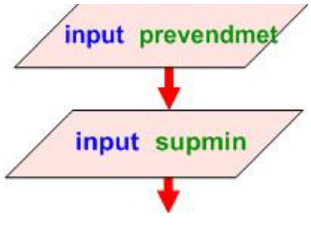
Riprendiamo ora il programma scritto per risolvere il problema 01:

```
input "scrivi la spesa di costruzione al mq. "; specostrmet
input "scrivi la superficie dell'appartamento (in mq.) "; supap
input "scrivi il prezzo di vendita al mq. "; prevendmet
let specostr=specostrmet * supap
let prevend=prevendmet * supap
let guad=prevend - specostr
print "il guadagno della società è di ";guad;" euro"
end
```

Ricordando il cambiamento da introdurre:

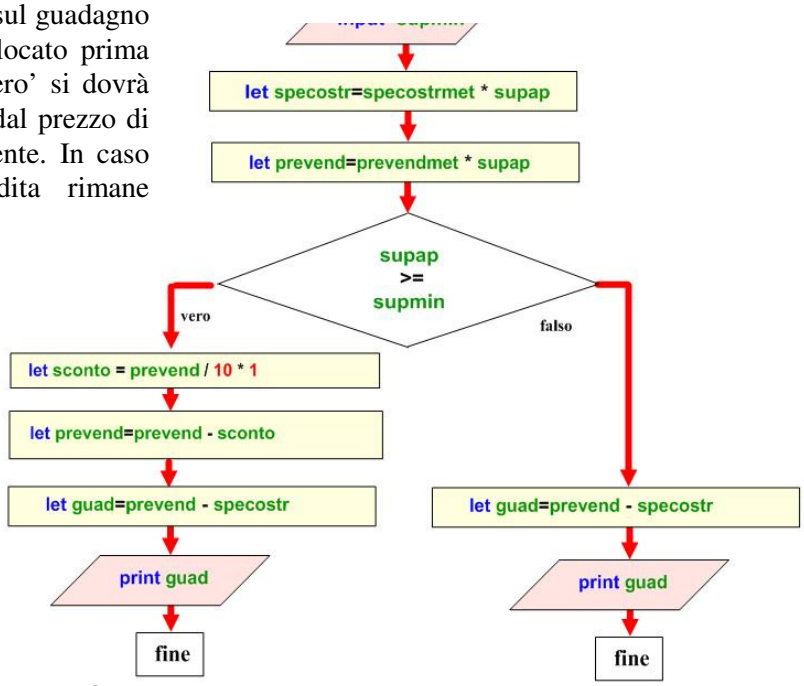
la società immobiliare decide di fare uno sconto di 1/10 (dato costante) sul prezzo di vendita dell'appartamento in caso di appartamenti grandi (a partire da 100 mq).

Il dato da controllare sarà la superficie dell'appartamento.
Per fare il controllo il calcolatore deve conoscere da quale superficie un appartamento può essere considerato grande (nel problema specifico era di 100 mq.). Di conseguenza va inserita la richiesta del nuovo dato: la superficie minima per appartamenti grandi (in mq.)



Vediamo ora come e in quale punto del grafico introdurre il controllo.

Visto che il controllo influisce sul guadagno della società, dovrà essere collocato prima di esso. In caso di risposta 'vero' si dovrà calcolare lo *sconto* e sottrarlo dal prezzo di vendita calcolato precedentemente. In caso contrario il prezzo di vendita rimane invariato.
A fianco possiamo vedere come viene scritto il controllo. All'interno del rombo viene scritta la domanda che consiste nel confronto tra due variabili o tra una variabile e un dato costante.



Nel programma invece si utilizza la proposizione:

se (controllo) allora (ciò che viene fatto in caso di risposta vero) altrimenti (ciò che viene fatto in caso di risposta falso)

che tradotta in inglese diventa:

if (controllo) then (ciò che viene fatto in caso di risposta vero) else (ciò che viene fatto in caso di risposta falso)

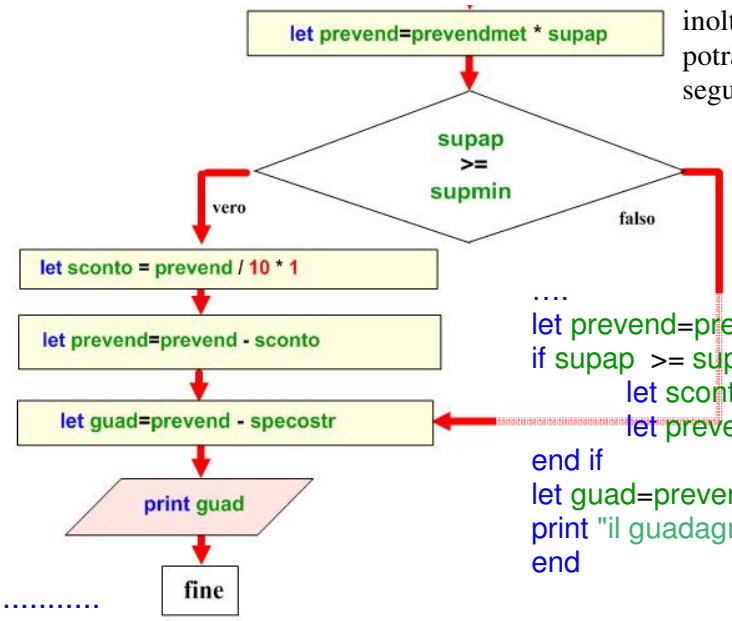
Si inserisce dunque all'interno del programma la nuova istruzione preceduta dalla richiesta della superficie minima per appartamenti grandi (in mq.): (es: problema01-a)

```
input "scrivi la spesa di costruzione al mq. "; specostrmet
input "scrivi la superficie dell'appartamento (in mq.) "; supap
input "scrivi il prezzo di vendita al mq. "; prevendmet
input "scrivi la superficie minima per appartamenti grandi (in mq.)"; supmin
let specostr=specostrmet * supap
let prevend=prevendmet * supap
if supap  >= supmin then let sconto = prevend / 10 * 1: let prevend=prevend – sconto
      : let guad=prevend – specostr : print "il guadagno della società è di ";guad;"
      euro" : end else let guad=prevend – specostr : print "il guadagno della società è
      di ";guad;" euro" : end
```

come si può notare, in caso di risposta vero la variabile prevend viene **riassegnata**. Cioè vi viene inserito un nuovo valore che cancella il precedente. Il vecchio valore del prezzo di vendita viene dunque sostituito da quello nuovo, diminuito grazie allo sconto.

Liberty Basic dà la possibilità di scrivere il programma in modo più leggibile: (es: problema01-b)

```
....
let prevend=prevendmet * supap
if supap  >= supmin then
        let sconto = prevend / 10 * 1
        let prevend=prevend – sconto
        let guad=prevend – specostr
        print "il guadagno della società è di ";guad;" euro"
        end
else
        let guad=prevend – specostr
        print "il guadagno della società è di ";guad;" euro"
        end
end if
```



inoltre il programma, nella sua parte finale potrà anche essere disegnato e scritto nel seguente, e più semplice, modo:

(es: problema01-c)

```
....
let prevend=prevendmet * supap
if supap  >= supmin then
        let sconto = prevend / 10 * 1
        let prevend=prevend – sconto
end if
let guad=prevend – specostr
print "il guadagno della società è di ";guad;" euro"
end
```

.....

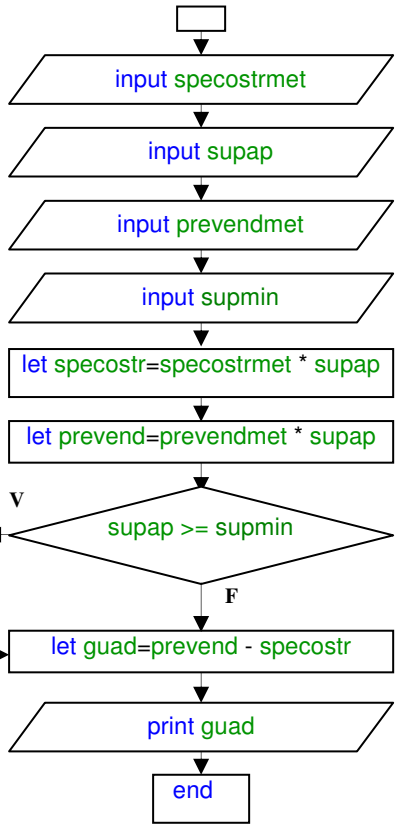
come si può notare, in caso di risposta vero la variabile prevend viene **riassegnata**. Cioè vi viene inserito un nuovo valore che cancella il precedente. Il vecchio valore del prezzo di vendita viene dunque sostituito da quello nuovo, diminuito grazie allo sconto.

l'uso della scheda di programmazione

Con questa unità di apprendimento a scheda di programmazione è stata adattata alle esigenze che pone questa Unità di Apprendimento. 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:

Una società immobiliare ha costruito degli appartamenti spendendo una certa cifra al metro quadro. Ne rivende uno di una certa superficie ad un cero prezzo al metro quadro.
Inoltre la società immobiliare decide di fare uno sconto di 1/10 (dato costante) sul prezzo di vendita dell'appartamento in caso di appartamenti grandi (a partire da 100 mq).
Quale sarà il guadagno della società?

diagramma di flusso




```
input "scrivi la spesa di costruzione al mq. "; specostrmet
input "scrivi la superficie dell'appartamento (in mq.) "; supap
input "scrivi il prezzo di vendita al mq. "; prevendmet
input "scrivi la superficie minima per appartamenti grandi (in mq.)";
supmin
let specostr=specostrmet * supap
let prevend=prevendmet * supap
if supap  >= supmin then
        let sconto = prevend / 10 * 1
        let prevend=prevend – sconto
end if
let guad=prevend – specostr
print "il guadagno della società è di ";guad;" euro"
end
```

Dalla scheda compilata si può notare che il controllo differenzia i percorsi che poi possono ricongiungersi ma possono rimanere differenziati e terminare con due differenti end.
La parte dell'algoritmo che viene eseguita in una determinata esecuzione viene denominata **processo**.

verifica su un nuovo problema

Applichiamo ad un nuovo problema ciò che abbiamo appena imparato. Trascriviamo il testo su una scheda di programmazione e mettiamoci al lavoro.

Problema generalizzato 03

Un'impresa edile calcola la paga mensile definitiva dei propri dipendenti nel seguente modo:

- dati il numero di ore di lavoro effettuate e la paga oraria il calcolatore dovrà calcolare la paga base
- alla paga base così ottenuta verrà sommata un'*aggiunta* che dipende dal numero di ore effettuate. Se è maggiore di 100 (dato costante) l'aggiunta sarà equivalente alla paga di 10 ore di lavoro (dato costante), altrimenti l'aggiunta sarà equivalente alla paga di 4 ore di lavoro.

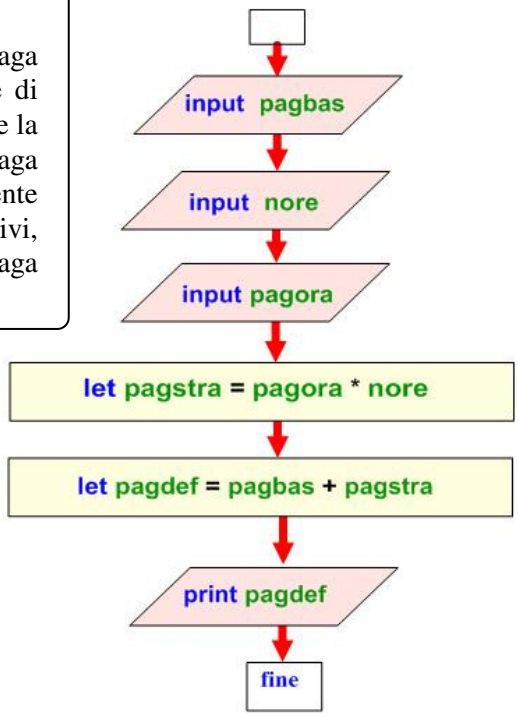
Realizzare un programma che calcoli la paga definitiva di un dipendente.

sul problema 02

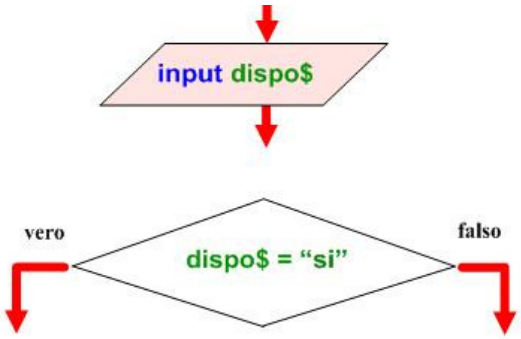
Riprendiamo ora il problema 02 e partiamo dalla soluzione che avevamo individuato all'inizi del nostro lavoro.
Sulla scheda di programmazione inseriamo già il testo con la modifica che avevamo individuato:

Problema generalizzato 02

Un muratore di un impresa edile percepisce una certa paga base mensile a cui viene aggiunta la paga per le ore di straordinario. Data la paga base, le ore di straordinario e la paga per un'ora di straordinario, calcolare la paga definitiva del muratore tenendo conto che, se il dipendente è disposto ad effettuare lo straordinario nei giorni festivi, alla paga andrà aggiunto un premio pari a metà della paga base (dato costante).



Osserviamo il diagramma di flusso del problema iniziale. Sappiamo già che il calcolatore, oltre a numeri, è in grado di memorizzare anche singoli caratteri, parole e frasi (**dati alfanumerici**).



Vediamo ora come scrivere il controllo in base al quale il calcolatore decide se dare il premio per la disponibilità ad effettuare straordinario festivo.
Vi sono varie possibilità: scegliamo di richiedere questa disponibilità con un **sì** o un **no** da memorizzare dentro una variabile.

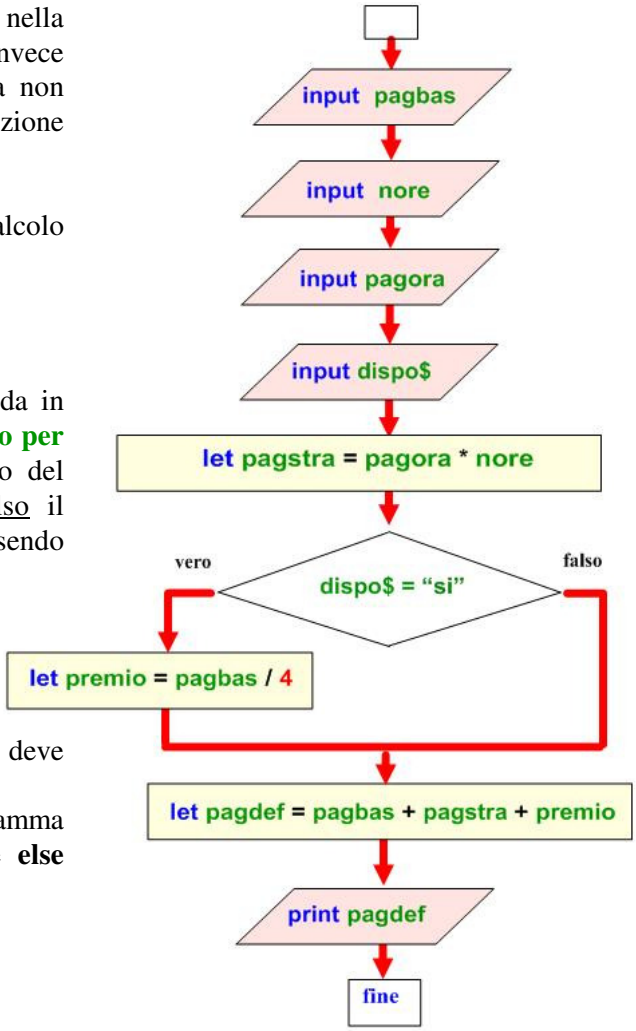
L'ingresso dei dati va messo insieme agli altri nella parte iniziale del programma, il controllo va invece collocato nel punto più funzionale, in modo da non complicare eccessivamente l'algoritmo di risoluzione del problema.

Decide dunque di collocare il controllo dopo il calcolo della paga per lo straordinario.

Ora possiamo notare che:

- l'ultima istruzione di elaborazione resta valida in ogni caso. In caso di risposta "vero" il **premio per lo straordinario festivo** conterrà il risultato del calcolo precedente, in caso di risposta falso il **premio per lo straordinario festivo**, non essendo stato assegnato, conterrà **zero**

- in caso di risposta falso il calcolatore non deve eseguire alcuna istruzione.
  Contrariamente a quanto visto nel programma precedente, in questo mancherà l'istruzione **else** che, infatti, è facoltativa.



Passiamo ora alla realizzazione del programma ricordando che ogni singolo carattere, visto che il calcolatore può lavorare solo con numeri binari, sarà tradotto in un corrispondente numero binario seguendo un apposito codice (codice ASCI - American Standard Code for Information excange).
Uno o più caratteri possono dunque essere memorizzati dentro una variabile che va contrassegnata appositamente con il segno **$,** in modo che il calcolatore possa distinguerla da quelle che memorizzano dati numerici.
Un dato alfanumerico (singolo carattere, parola, frase) può essere::

- inserito direttamente nel programma; in questo caso va messo tra virgolette.
- memorizzato dentro una variabile il cui nome deve però terminare con il segno $

Partendo poi dal programma precedentemente realizzato, vengono inseriti richiesta dei dati e controllo in BASIC.

```
input "scrivi la paga base mensile "; pagbas
input "scrivi il n° di ore di straordinario "; nore
input "scrivi la paga per un ora di straordinario "; pagora
input "il muratore è disponibile allo straordinario festivo (si/no)?"; dispo$
let pagstra = pagora * nore
if dispo$ = "si" then let premio = pagbas / 2
let pagdef = pagbas + pagstra + premio
cls : print "la paga definitiva del muratore è di ";pagdef;" euro"
end
```

Ricordiamo ancora che il "sì", come ogni dato alfanumerico che viene inserito direttamente nel programma, va scritto tra virgolette.

verifica su un nuovo problema

Ecco un nuovo problema (problema 04), con il testo generalizzato, su cui riapplicare quanto appena studiato sul problema 02:

Problema generalizzato 04

Il calcolatore, dopo aver chiesto *la paga base mensile* e *il valore delle vendite da lui effettuate*, deve calcolare e scrivere sullo schermo la **paga mensile definitiva** di un commesso sapendo che alla sua **paga base mensile** deve essere aggiunto **1/ 10** del **valore delle vendite** effettuate durante il mese.
Inoltre se il commesso abita **fuori città** verrà aggiunto un rimborso spese pari a **50** euro (dato costante).

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.

05) 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.

06) 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.

07) In una regione agricola i terreni producono una certa quantità di chili di frumento per ogni ettaro di terreno. Se il terreno è stato irrigato oppure 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'uso dei connettivi nei controlli

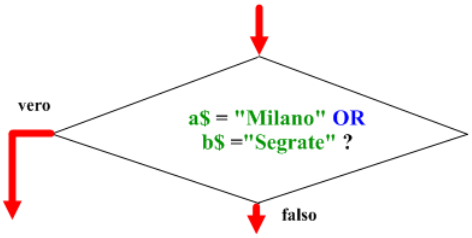
Nelle nostre conversazioni quotidiane utilizziamo spesso i connettivi "e" e "oppure" per unire due o più domande. In BASIC è possibile combinare più condizioni fra loro per mezzo di operatori logici, i più frequenti dei quali sono AND e OR.

il connettivo “e” viene tradotto con **and** mentre il connettivo “o” viene tradotto con **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:

Ad esempio:

.....
if a$ = “Milano” and b$ = “Segrate” then …. else …
…..

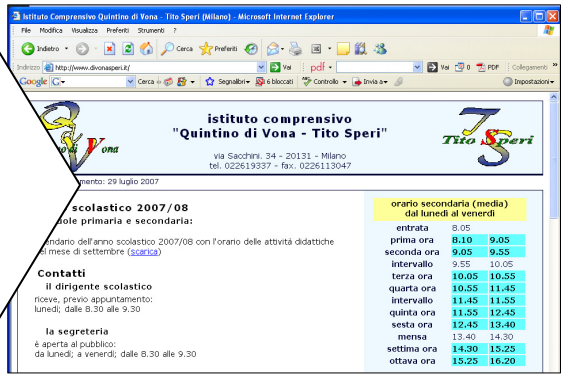


….
if a$ = “Milano” or b$ = “Segrate” then ….
else ...
….

i connettivi e la ricerca in INTERNET

Quando effettuiamo ricerche su Internet utilizziamo delle applicazioni appositamente realizzate che vengono chiamate **motori di ricerca**. Anche “dietro” di essi vi è un software realizzato con un linguaggio di programmazione.
Prendiamo ad esempio Google e immaginiamo di cercare siti di musei che sono a Milano.
Se scriviamo la parola ***Milano*** il programma cercherà la parola Milano tra le **parole chiave** dei siti presenti su Internet. Se aggiungiamo la parola ***museo*** il programma cercherà quei siti che, tra le parole chiave hanno **entrambe** le parole che ho scritto. In pratica il programma utilizza il connettivo AND per cercare entrambe le parole dando risposta **vero** se trova tutte e due.
Clikkando su **ricerca avanzata** Google apre un’apposita finestra in cui possiamo anche scrivere parole che non devono essere presenti tra le parole chiave. Ad esempio se scrivo la parola ***arte*** il programma cercherà quei siti che contengono la parola ***Milano*** e ***museo*** ma non contengono la parola ***arte***. In pratica il programma collega con il connettivo **AND** le prime due parole e con il **NOT** la terza. Darà risposta **vero** se trova le prime due ma non la terza.

La prima pagina del sito della nostra scuola su: www. divonasperi.it .
Come tutte le pagine su Internet, anch’essa è realizzata in HTML, un linguaggio particolare studiato per il WEB. Nel codice che la gestisce vi è la dichiarazione delle parole chiave a disposizione dei motori di ricerca. Eccola:
<meta name="keywords" content="Quintino di Vona, Tito Speri, divonasperi, divona, speri, scuola primaria, scuola secondaria di primo grado, istituto comprensivo, tecnologico ambientale, musicale, artistico, scientifico, linguistico" />



altre possibilità per gestire i controlli

- Quando vi è una variabile da controllare ma sono da prevedere molti possibili esiti sarà utile utilizzare l'ordine **Select Case**:

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

* *facoltativo*

ad esempio:

risolviamo ora il seguente problema:

Problema 08 (mansioni.bas)
Una ditta di trasporti paga i dipendenti in base al numero di ore di lavoro effettuate e in base alla mansione svolta (autista o impiegato).
Il calcolatore dovrà calcolare la paga definitiva del dipendente sapendo che alla ***paga base totale****, dipendente dalle* ***ore svolte*** *verrà sommata un'****aggiunta*** *che dipende dalla* ***mansione*** *che svolge il dipendente (****dirigente****, ****autista****, ****operaio*** *o* ***impiegato****). La paga verrà raddoppiata (d.c.) ai dirigenti, verrà aggiunto 1/10 (dato costante) della paga base se il dipendente svolge la mansione di impiegato mentre se il dipendente svolge la mansione di autista l'aggiunta sarà di 3/10 (dato costante) della paga base. Gli operai non avranno aumenti.*
Il programma dovrà essere protetto dagli errori di immissione di dati.

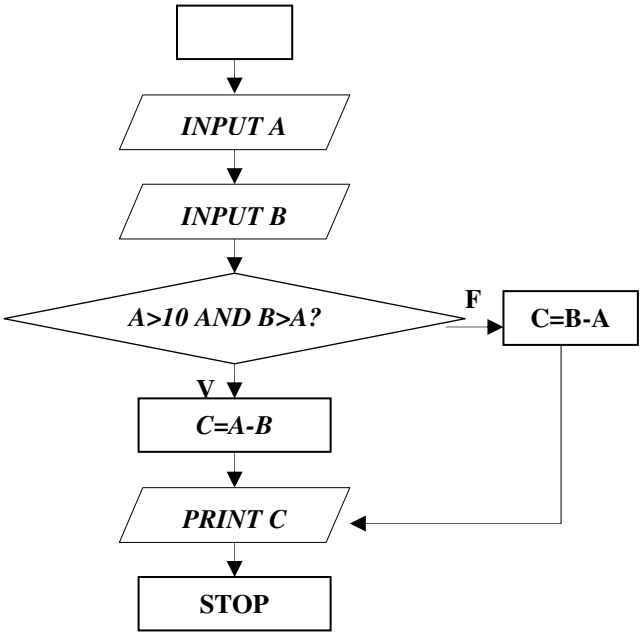
```
Input "paga base oraria "; pbo
input "ore svolte dal dipendente "; osd
input "mansione del dipendente (dirigente, autista, operaio, impiegato) "; mans$
pbt = pbo * osd
Select Case mans$
        Case "dirigente": agg = pbt
        Case "autista": agg = pbt / 10 * 3
        Case "operaio": agg = 0
        Case "impiegato": agg = pbt / 10 * 1
        Case Else: print "la mansione non è scritta in modo corretto": end
End Select
pd = pbt + agg : print "la paga definitiva del dipendente è di "; pd; " euro"
end
```

il controllo potrà essere anche scritto con la seguente sintassi che permette l'uso di simboli di confronto diversi da "=" (uguale)

```
Select Case
        Case (man$ ="dirigente") : agg = pbt
        Case (man$ ="autista"): agg = pbt / 10 * 3
        Case (man$ ="operaio"): agg = 0
        Case (man$ ="impiegato"): agg = pbt / 10 * 1
        Case Else: print "la mansione non è scritta in modo corretto": end
End Select
```

ed ora al lavoro!

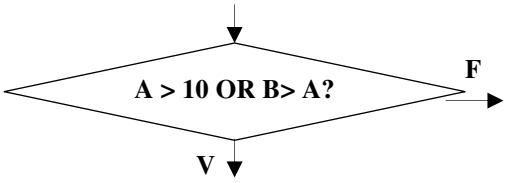
1) Analizza il seguente diagramma di flusso e rispondi alle domande sul fianco provando poi le singole soluzioni al calcolatore:



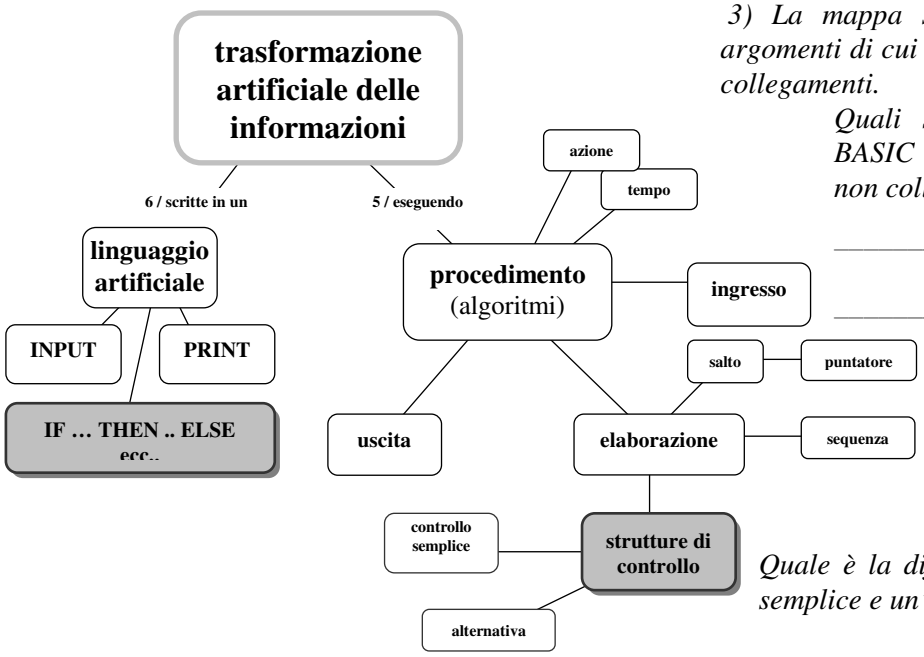
Se, alla richiesta del calcolatore, memorizzo nella variabile A il numero 6 e nella variabile B il numero 8, alla fine il calcolatore scriverà sullo schermo il numero _____.

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?

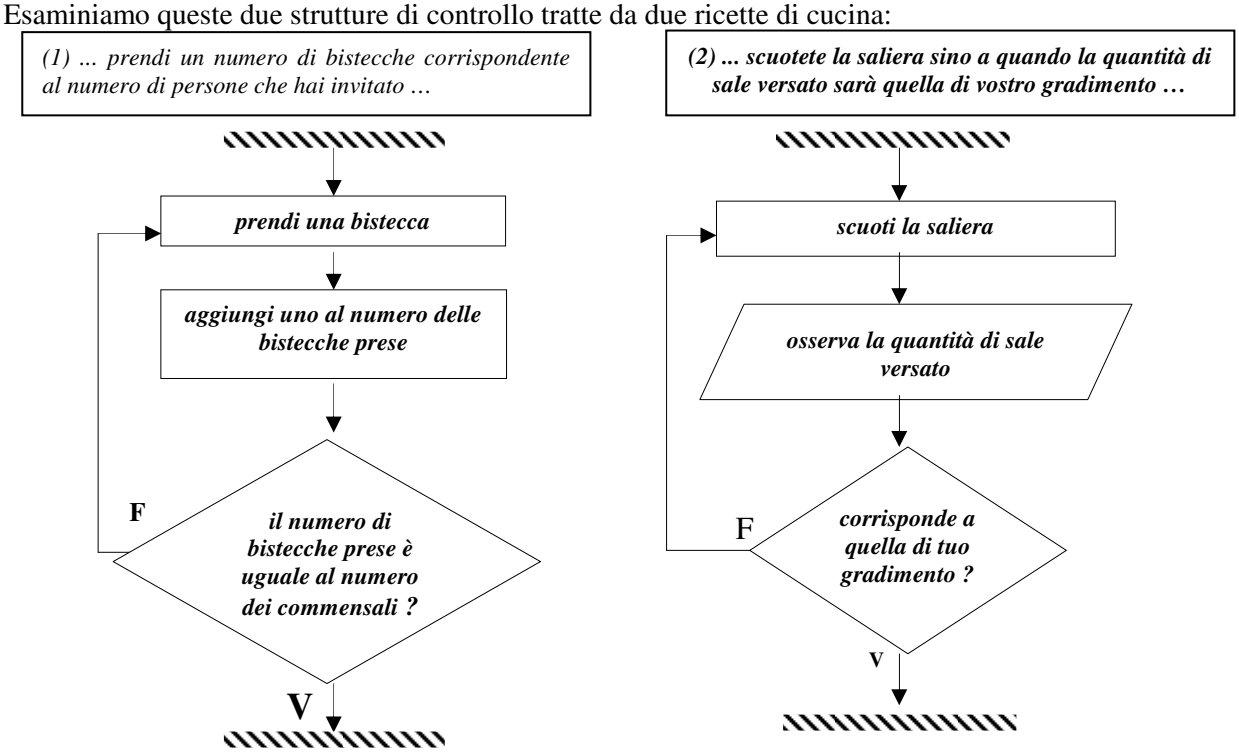
_____

_____

_____

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

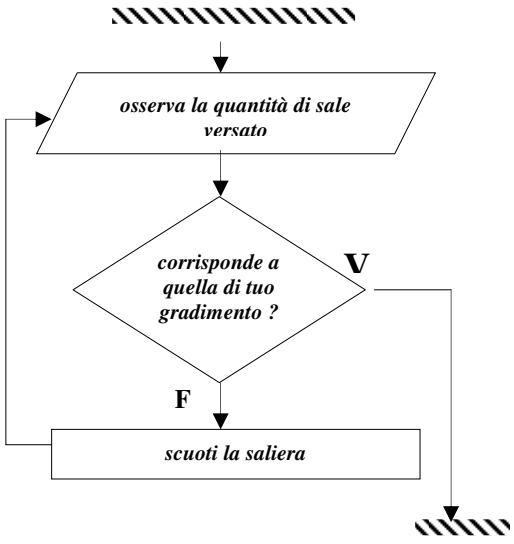
Quando una o più azioni vengono ripetute sino al cambiamento della risposta ad un controllo abbiamo una ripetizione (o ciclo).
Esaminiamo queste due strutture di controllo tratte da due ricette di cucina:



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? _____
  _____

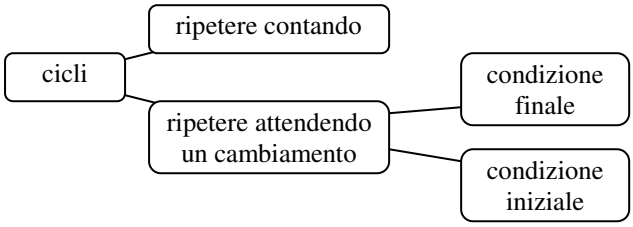
Nel primo caso l'azione che viene ripetuta ("scuoti la saliera") è posta prima del controllo e andrà comunque eseguita almeno una volta. Questa soluzione non prevede dunque che l'esecutore rifiuti il sale. Questo ciclo viene chiamato a **condizione finale**.
Nell'ultimo grafico l'azione che viene ripetuta ("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***

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

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



Gli esempi visti sono tratti dai procedimenti svolti per risolvere problemi quotidiani. Ritroveremo situazioni simili anche quando l'esecutore della procedura sarà un calcolatore.

un tempo ... 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. Nelle prime versioni di BASIC era il **numero di linea** a svolgere la funzione di inizio e di fine ciclo; per questo ogni linea veniva numerata.

un tempo ... il controllo

L'istruzione di controllo era gestita dall'ordine if ***controllo*** then … e if ***controllo*** then ... else … che, abbinato all'ordine **goto** permetteva il ritorno dell'esecuzione all'inizio del ciclo sino a quando la risposta al controllo non fosse cambiata. 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.

Le nuove versioni di BASIC permettono la rinuncia al numero di linea mettendo a disposizione diverse istruzioni che hanno il compito di svolgere la funzione di puntatore e di gestire il controllo. Queste istruzioni variano a seconda del tipo di ciclo che è necessario gestire.

Ripetere attendendo un cambiamento

➢ ***i puntatori***

Le istruzioni **do** (inizio ripetizione = "ripeti") e **loop** (fine ripetizione = "fine ripeti") vengono abbinate al controllo con le seguenti modalità:

il ciclo a condizione iniziale viene gestito dall'istruzione:

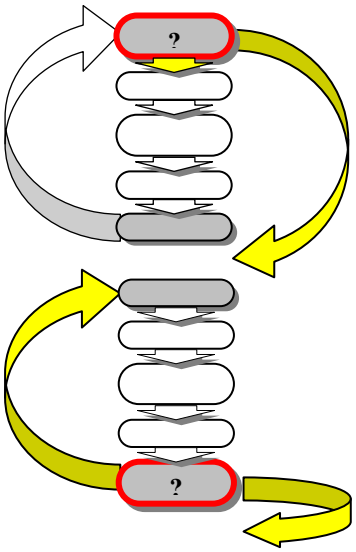
do *controllo*
istruzioni da ripetere
loop

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

il ciclo a condizione finale viene gestito dall'istruzione:

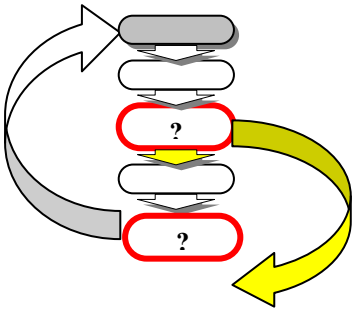
do
istruzioni da ripetere
loop *controllo*

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



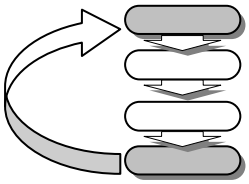
all'interno delle istruzioni da ripetere è anche possibile inserire un controllo con if .. then .. seguito dall'istruzione **exit do** prevedendo così un'altra uscita dal ciclo:

```
do
istruzioni da ripetere
controllo exit do
istruzioni da ripetere
loop controllo
```



in Visual Basic ma non in Just Basic è anche possibile non mettere il controllo dopo i puntatori creando così **un ciclo infinito**

```
do
istruzioni da ripetere
loop
```

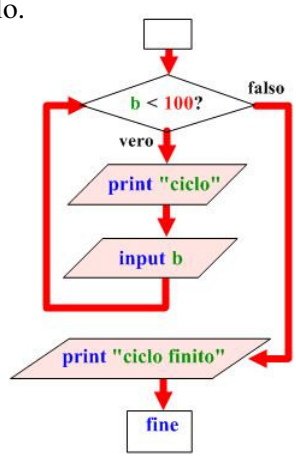


➢ ***il controllo con ... fino a quando... / con ... mentre ...***

come sappiamo, il controllo è sempre rappresentato dal confronto tra i contenuti di due variabili (es: a$=b$?) oppure dal confronto tra il contenuto di una variabile e un dato costante (es: B=17?). Le nuove versioni di BASIC ci permettono di sostituire if … then e if … then … else con gli ordini:

while che corrisponde all'italiano "mentre" e comporta un'uscita dal ciclo in caso di risposta falso. while può seguire il puntatore di inizio o di fine ciclo ed è seguito dal controllo.
Ad esempio nel programma:

```
do while b < 100
print "ciclo"
input "scrivi un numero"; b
loop
print "ciclo finito"
```

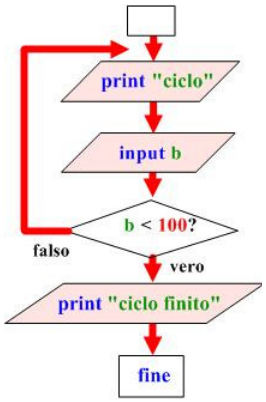


il calcolatore _____

_____

until che corrisponde all'italiano "fino a" e che comporta un'uscita dal ciclo in caso di risposta vero. Anche until può seguire il puntatore di inizio o di fine ciclo ed è seguito dal controllo.
Ad esempio nel programma:

```
do
print "ciclo"
input "scrivi un numero"; b
loop until b < 100
print "ciclo finito"
```



il calcolatore _____

_____

il controllo è abbinato al puntatore di fine ciclo perché _____

_____

➢ ***nei diagrammi di flusso***

i puntatori non vengono rappresentati. Verrà inserito, nel rombo, solo il controllo. Saranno le frecce che escono dal rombo ad indicarci l'inizio e la fine del ciclo.

la lettura del tempo con time$() e date$()

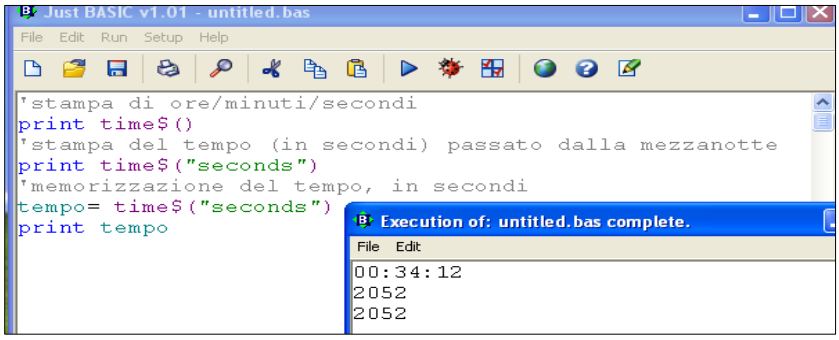
Il calcolatore possiede un orologio interno, che conta il tempo. In JB, ma anche in Visual Basic, esso viene memorizzato in una variabile alfanumerica chiamata time$() che è composta da 8 caratteri. Due caratteri sono dedicati alle ore, due ai minuti e altri due ai secondi. Come segno di separazione vengono usati i due punti ":".

se ad esempio do l'ordine: 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*

Come con tutte le variabili alfanumeriche, è possibile effettuare dello slicing
ad esempio con a$ = mid$(time$,7,2) *il calcolatore memorizzerà in* a$ *solo i secondi.*

Forniscono invece valori numerici gli ordini:

time$("seconds") fornisce i secondi passati dalla mezzanotte e time$("ms") fornisce il tempo dalla mezzanotte in millesimi di secondo

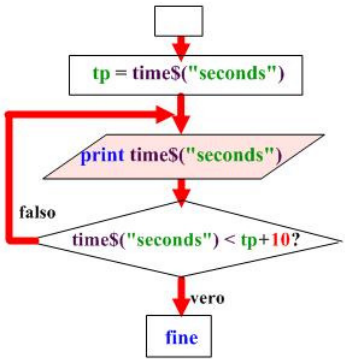


I valori di entrambi cambiano in continuazione. Se si vuole "fermare" il tempo bisognerà trasferire il valore del dato dentro un'altra variabile.

Per far trascorrere un certo tempo (*tempo di attesa*) è necessario prima memorizzare il *tempo iniziale* in una variabile numerica (ad esempio tp=time$("seconds").
Seguirà il controllo del tempo trascorso dato dall'espressione:
tempo attuale = tempo iniziale + tempo di attesa
Ad esempio se si fa il seguente controllo time$("seconds")=tp+10 ?
il calcolatore va a leggere il tempo attuale e lo va a confrontare con il tempo iniziale più 10 secondi. Con il programma :

```
tp=time$("seconds")
do
print time$("seconds")
loop until time$("seconds") > tp+10
end
```

il calcolatore resta per 10 secondi nel ciclo scrivendo il tempo.



Anche la variabile alfanumerica date$() fornisce la data corrente però con un diverso formato.
Ad esempio con: print date$() il calcolatore scriverà: Dec 19, 2006
Con:

```
print date$("mm/dd/yyyy")      12/19/2006
print date$("mm/dd/yy")        12/19/06
print date$("yyyy/mm/dd")      2006/12/19
```

Mentre con print date$("days") il calcolatore fornirà il numero (dato numerico) di giorni passati dal 1° gennaio 1901 e con date$("4/1/2002") il calcolatore fornisce il numero di giorni compresi tra il 1° gennaio 1901 e la data scritta.

con:	print date$("4/1/2002")	viene scritto	36980
mentre con	print date$(36980)	viene restituito	04/01/2002

In Visual Basic questi ultimi ordini non esistono. Il tempo che passa (in secondi) viene rilevato da **Timer** (ordine e oggetto). vedi esempio: 10 secondi in VB

l'estrazione a sorte con rnd(1)

Con **rnd(1)** il calcolatore estrae un numero compreso tra 0 e 1 da una sequenza di numeri generati dal calcolatore.
Se ad esempio eseguiamo l'ordine: **print rnd(1)** il calcolatore scriverà un numero estratto a caso compreso tra 0,00000001 e 0,99999999
Un numero così scritto ci servirà a poco ma basta introdurre poche aggiunte per renderlo davvero utile:

- l'ordine **int(**numero o variabile numerica**)** serve ad eliminare la parte decimale di un numero.
  Ad esempio: con **a=int(17.55): print a**
  il calcolatore scriverà nella finestra di esecuzione il numero ***17****.*
- usando int e moltiplicando il prodotto di rnd(1) per un certo numero si otterrà un numero che parte da 0 e può arrivare al numero inferiore a quello dato.
  Ad esempio: con **a=int(10 * rnd(1)): print a**
  il calcolatore scriverà nella finestra di esecuzione un numero compreso tra ***0*** *e* ***9****.*
- inoltre aggiungendo sempre +1 si otterrà un numero che parte da 1 e può arrivare al numero dato.

Dunque la formula:
INT(limitesup * RND(1)) + 1
dove: *limitesup* è il numero più alto da estrarre
Esempio: con a=int(20*rnd(1))+1 *il calcolatore colloca nella variabile* a *un numero compreso tra 1 e 20*

Per generare numeri interi casuali in un intervallo che non parte da 1 si può utilizzare la seguente formula un po' più complessa:
INT((*limitesup - limiteinf + 1*) * **RND(1)** + *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(1)+10) *il calcolatore colloca nella variabile* a *un numero compreso tra 10 e 20*

esempi di cicli del primo tipo con estrazione a sorte

ripetere delle estrazioni
ripetere delle estrazioni per un determinato tempo
ripetere delle estrazioni fino a quando viene estratto un certo numero e/o per un determinato tempo
estrarre dei caratteri (numeri da convertire in caratteri) **per comporre una password**
estrarre dei caratteri (numeri da convertire in caratteri) **per comporre il nome di un file**

```
'programma per continuare ad estrarre casualmente un file contenente
'i dati di un alunno della 2H sino a quando viene estratto uno
' che viene dalla scuola scelta (es: cercascuola)
input "scuola da estrarre ";sc$
do
        c=int((26-1+1)*rnd(1)+1):c$=str$(c)
        open c$+".txt" for input as#1
        input#1,a1$,a2$,a3$,a4$,a5$,a6$,a7$,a8$,a9$,a10$
        close#1
        print c$;"  ";a1$
loop while mid$(a4$,4,len(sc$))<>sc$
print
print "dati alunno estratto:"
print c$;"  ";a1$;" ";a2$;" ";a3$;" ";a4$
end
```

```
'il giocatore dà le coordinate del bersaglio che il calcolatore dovrà colpire (es: giocatore)
Input"nome giocatore ";gio1$
print "coordinata X tra 1 e 74 / coordinata Y tra 1 e 22"
print: print "inserimento dati"
Input"coordinata X dell'obiettivo ";x1
```

```
Input"coordinata Y dell'obiettivo ";y1
'il bersaglio del giocatore viene indicato con B
cls
tp=time$("seconds")
locate x1,y1:print"B"
'il calcolatore inizia ad estrarre le coordinate dei suoi colpi che vengono indicati con *
do
x=int((79-1+1)*rnd(1)+1)
      y=int((23-1+1)*rnd(1)+1)
      locate x,y:print"*";
      ' ciclo di attesa per rallentare i colpi
      do
            at=time$("ms")
        loop while time$("ms")<at+2
loop until x=x1 and y=y1
' il gioco termina quando le coordinate del proiettile sono = alle coordinate del bersaglio
tt1=time$("seconds")-tp
' comunicazione del tempo impiegato
locate 1,1:print "il bersaglio di ";gio1$;" è stato colpito in ";tt1;" secondi"
```

Nell'esempio abbiamo potuto osservare un **ciclo nidificato**, cioè un ciclo collocato all'interno di un altro ciclo. Questi cicli hanno le seguenti caratteristiche:
- se i cicli sono due, uno dei due è completamente interno (ciclo interno) all'altro (ciclo esterno); questa situazione è detta ***nidificazione***

```
'come prima ma con due giocatori (es: due giocatori)
Input"nome primo giocatore ";gio1$
Input"nome secondo giocatore ";gio2$
print "coord. X tra 1 e 74 / coord. Y tra 1 e 22"
print
print "primo giocatore - A"
Input"coordinata X dell'obiettivo ";x1
Input"coordinata Y dell'obiettivo ";y1
print
print "secondo giocatore"
Input"coordinata X dell'obiettivo ";x2
Input"coordinata Y dell'obiettivo ";y2
cls:tp=time$("ms")
do
   x=int((74-1+1)*rnd(1)+1)
   y=int((22-1+1)*rnd(1)+1)
   locate x1,y1:print"A"
   locate x2,y2:print"B"
   locate x,y:print"*";
      if x=x1 and y=y1 then tt1=time$("ms")-tp
      if x=x2 and y=y2 then tt2=time$("ms")-tp
loop until tt1<>0 and tt2<>0
' ...uscita
locate 1,1:print gio1$;" ha impiegato ";tt1;" ms"
print gio2$;" ha impiegato ";tt2;" ms"
notice "click su Ok per chiudere il gioco"
```

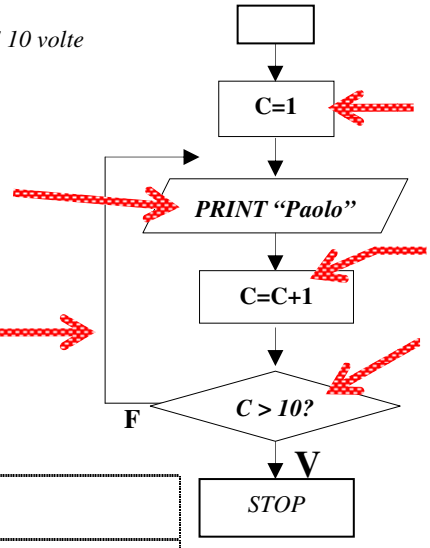
```
'un file è protetto da una password formata alcuni caratteri minuscoli (evitare più di 3 caratteri)
'il calcolatore comporrà in modo casuale una parola sino a quando troverà la password (es: password)

input "scrivi la password (si consigliano non più di tre caratteri) ";ps$
'viene stabilita la password da confrontare con quella estratta
tp=time$("seconds") 'viene memorizzato il tempo di partenza
do ' inizio del confronto tra password
        c$="" azzeramento della variabile che ospita i caratteri estratti
        do ' inizio estrazione dei caratteri
              c=int((122-97+1)*rnd(1)+97)
              'i codici dei caratteri minuscoli vanno da 97 (carattere 'a')a 122 (carattere 'z')
              c$= c$+chr$(c)
              'il numero estratto viene trasformato in carattere e aggiunto agli altri
   loop until len(c$)=len(ps$) ' fine estrazione dei caratteri
   locate 10,12:print c$ ' scrittura dei caratteri estratti
loop until c$=ps$ ' confronto tra i caratteri estratti e quelli assegnati e uscita dal ciclo
s=time$("seconds")-tp
print "password trovata dopo ";s;" secondi"
```

Dell'uso di while e until per attendere e/o contare le variazioni di un sensore ci occuperemo nell'attività di **robotica**.

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



azione che viene ripetute più volte

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

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

incremento
viene aggiunta un'unità alla variabile "contatore"

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

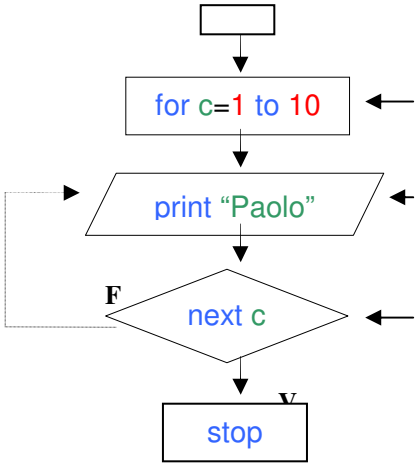
1	C=1
2	**PRINT** "Paolo"
3	C=C+1
4	**IF** C>10 **THEN STOP ELSE GOTO** 2

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)
next variabile

nel quale anche i numeri iniziale e finale possono essere sostituiti (come sempre) da variabili.
Ordine 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

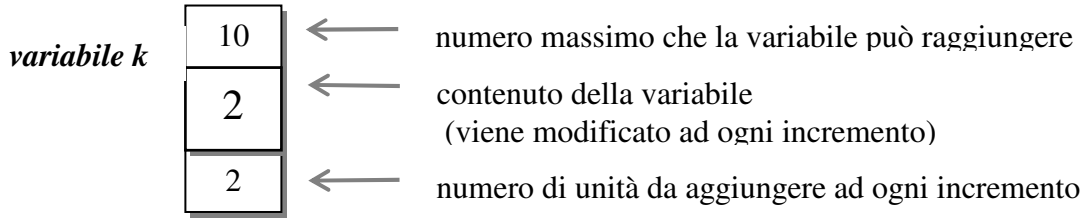
incremento / controllo / rinvio
oltre a svolgere la funzione di puntatore di fine ciclo, questa istruzione svolge automaticamente alcune operazioni:
1- incrementa automaticamente il numero contenuto nel contatore di un unità
2- controlla se il "contatore" ha superato il numero massimo:
- se ciò si è 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*

Ad esempio con: for k = 2 to 20 step 2

il calcolatore organizza una variabile strutturata nel seguente modo:

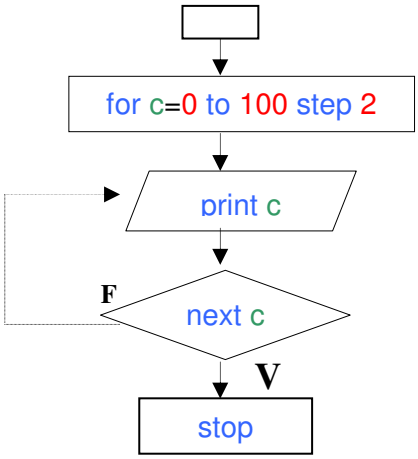


il calcolatore organizza una variabile che ha come numero di partenza 2, come numero di arrivo 20 e come "passo" 2. Questo significa che ad ogni esecuzione dell'ordine next la variabile k sarà aumentata di due unità Il ciclo sarà dunque eseguito 10 volte.
La variabile assegnata con for è dunque una variabile 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 k = 20 to 2 step -2 il calcolatore inserirà inizialmente nella variabile k il numero 20 a cui sottrarrà due unità ad ogni esecuzione di next sino a raggiungere il numero 2 che è il minimo prefissato.

… diversi utilizzi del ciclo con FOR … NEXT

a) 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

```
for c=0 to 100 step 2
print c
next c
stop
```



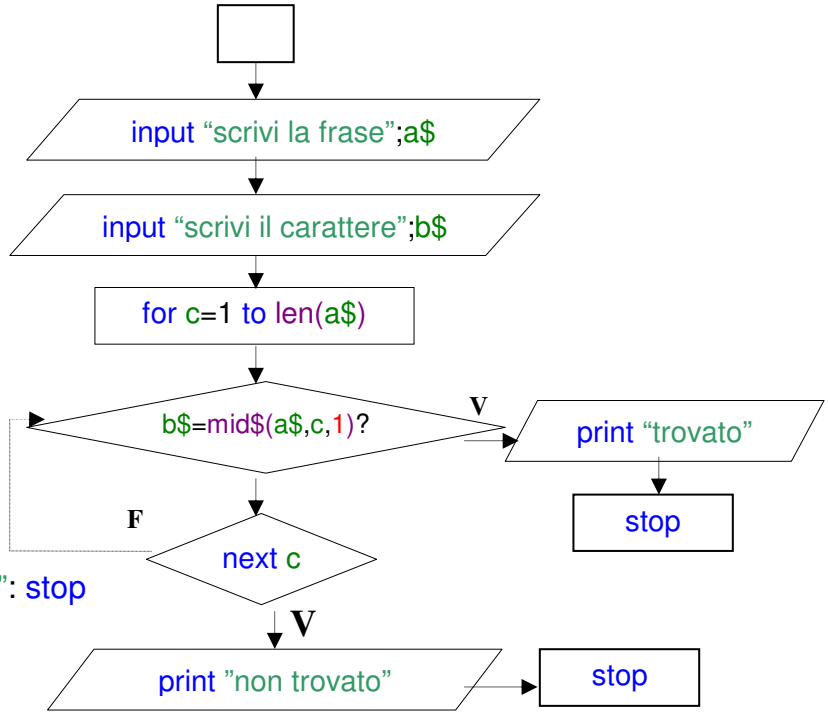
b) Come abbiamo visto nella precedente U.D. Algoritmi, una variabile alfanumerica è un insieme di byte ognuno dei quali memorizza un carattere ed è contrassegnato da un numero crescente che lo rende raggiungibile separatamente dagli altri.

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.

```
input “scrivi la frase”;a$
input “scrivi il carattere”;b$
for c=1 to len (a$)
if b$=mid$(a$,c,1) then print “trovato”: stop
next c
print “non trovato”
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: vedi esempi fase2

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).

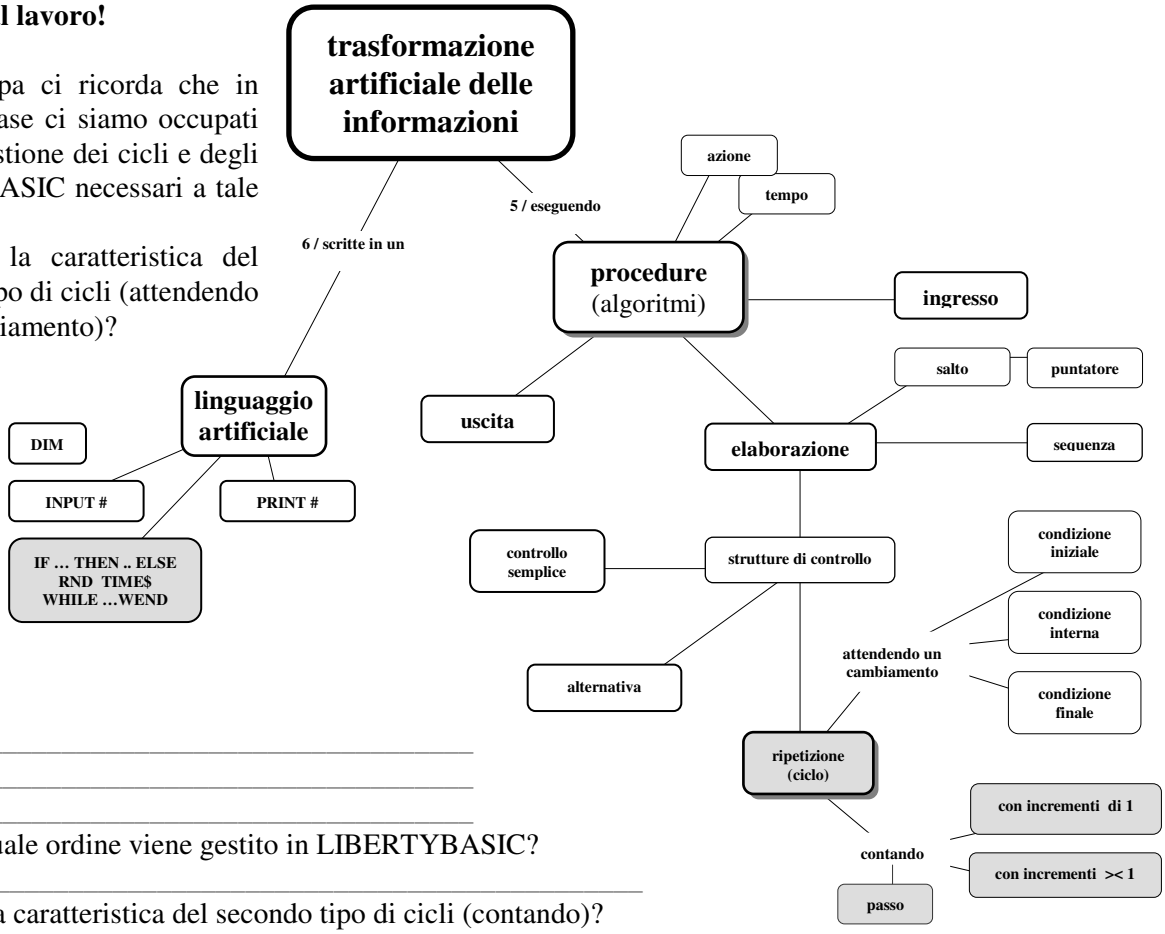
b) data una frase e una parola da cercare nella frase, il calcolatore segnala se la parola è presente.

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 la nuova frase sullo schermo.**…**

ed ora al lavoro!

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 LIBERTYBASIC?

_____

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

_____
_____

E con quale ordine viene gestito in LIBERTYBASIC? _____

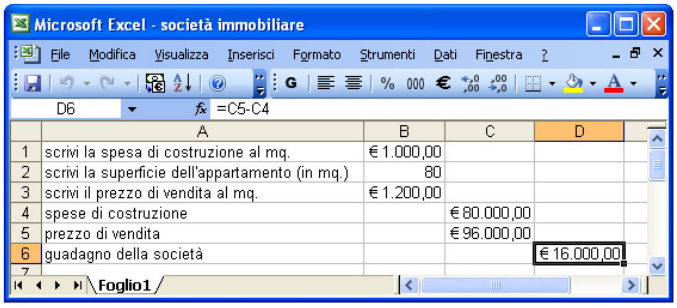
Che cosa è il passo e quando viene gestito automaticamente dal calcolatore?

_____

RISOLVERE I PROBLEMI CON I FOGLI ELETTRONICI
l'impostazione del foglio di lavoro e la risoluzione del problema 01

Sappiamo già usare i fogli elettronici per risolvere problemi sequenziali (senza controlli), vediamo ora come è possibile utilizzarli anche con i problemi che prevedono un controllo. Partiamo dai due problemi esaminati all'inizio (problemi 01 e 02) portando su foglio elettronico la loro versione priva di controllo. Preso il problema specifico 01:

> *Una società immobiliare ha costruito degli appartamenti spendendo 1000 € al metro quadro. Ne rivende uno di 80 metri quadri al prezzo di 1200 € al metro quadro.*
> *Quale sarà il guadagno della società?*



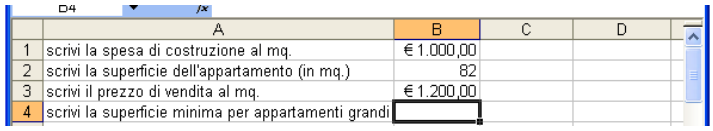
ne generalizziamo il testo e lo portiamo su un foglio di lavoro appositamente aperto sul nostro calcolatore secondo le indicazioni che già conosciamo. Infine lo proviamo utilizzando i dati del problema specifico.

Rivediamo poi il controllo da inserire:

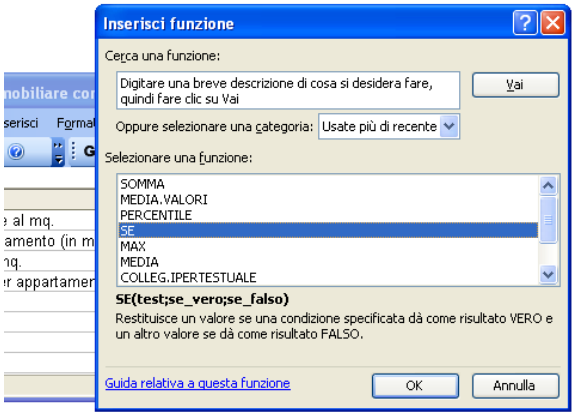
> *la società immobiliare decide di dimezzare il proprio guadagno (dato costante) in caso di appartamenti grandi.*

individuando dunque il dato che viene modificato dal controllo.
Il guadagno della società viene calcolato con la formula = C5–C4. Questa dovrà essere modificata se l'appartamento ha una superficie uguale o maggiore della superficie minima per appartamenti grandi. In questo caso la formula di calcolo sarà = (C5-C4)/2.

Come prima cosa inseriamo una nuova riga per memorizzare la superficie minima per appartamenti grandi.



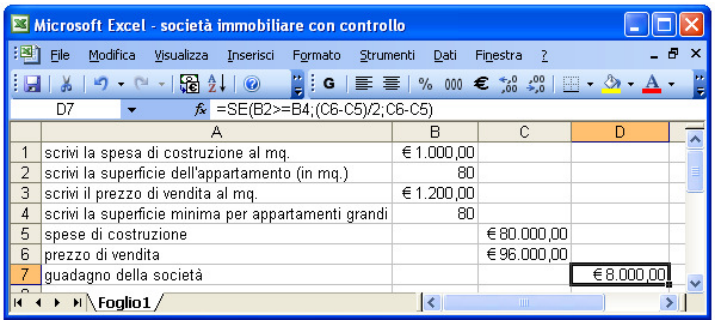
Verifichiamo poi, tramite il menù di *inserisci funzione,* se è previsto l'inserimento di un controllo. Potremo notare che è prevista la funzione *SE* e, clikkandoci sopra, la stessa finestra fornisce la sintassi da utilizzare =SE(test;se_vero;se_falso).



E' evidente che vi sono gli stessi elementi che abbiamo visto in BASIC:

- il controllo
- l'azione da fare in caso di risposta vero
- l'azione da fare in caso di risposta falso

che, preceduti da = SE e collocati tra parentesi, saranno separati tra di loro dal segno ; (punto e virgola).



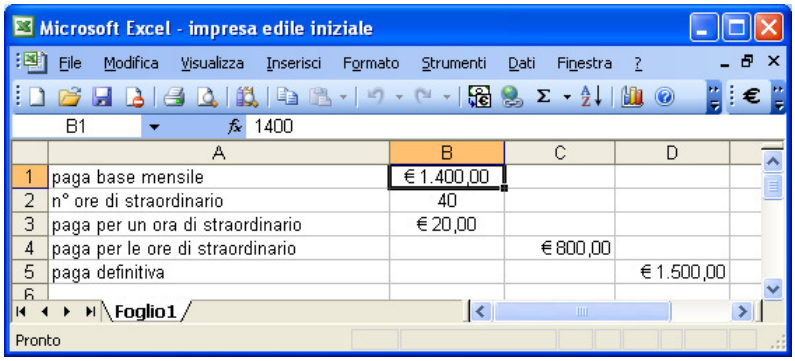
Potremo dunque individuare e inserire in **C7** la formula:
=SE(B2>=B4;(C6-C5)/2;C6-C5),
provando così la versione definitiva del foglio di lavoro.

l'impostazione del foglio di lavoro e la risoluzione del problema 02

Prendiamo ora in esame il problema specifico 02:

Un muratore di un'impresa edile percepisce una base mensile di 1400 €. Metà della paga base (dato costante) deve però essere detratta per tasse e contributi da versare allo Stato. Va invece aggiunta la paga per le 40 ore di straordinario che vengono pagate 20 € l'ora. Qual è la paga definitiva del muratore?

Anche in questo caso ne generalizziamo il testo e lo portiamo su un foglio di lavoro appositamente aperto sul nostro calcolatore secondo le indicazioni già conosciute. Infine lo proviamo utilizzando i dati del problema specifico.

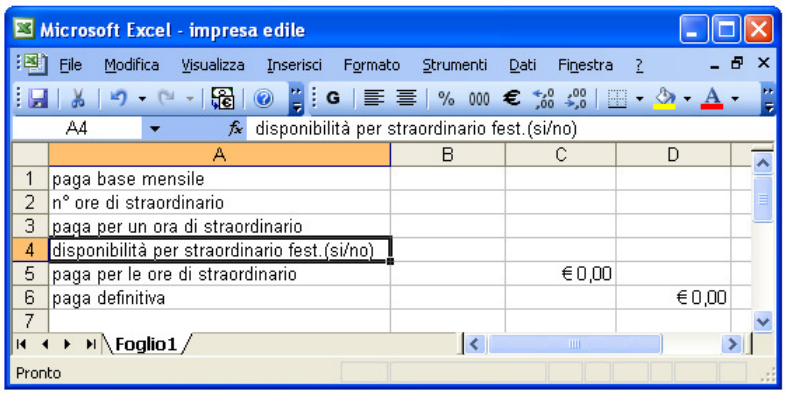


Ricordiamo poi che il controllo da inserire:

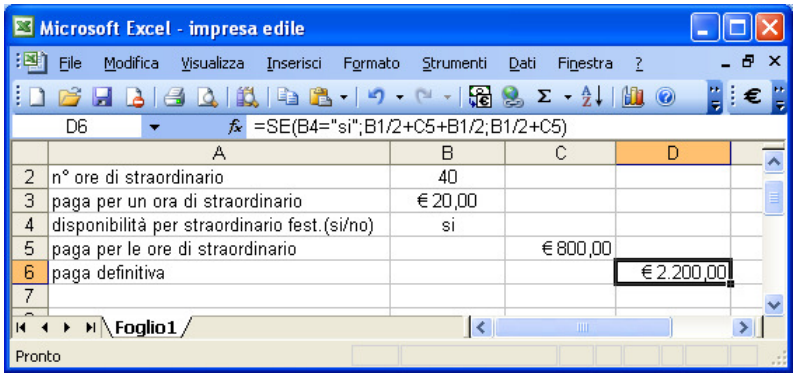
l'impresa edile decide di compensare la disponibilità del dipendente ad effettuare ore di straordinario anche nei giorni festivi con un premio (che corrisponde a metà della paga base)

aveva richiesto modifiche più profonde al programma di risoluzione.

Infatti deve essere richiesto un nuovo dato: la disponibilità ad effettuare lo straordinario festivo il cui valore alfanumerico ("si" oppure "no") determina l'aggiunta del premio.



Tenendo conto di quanto appreso riguardo ai controlli sui dati alfanumerici (attività 4/a) e di quanto visto nel problema precedente i ragazzi individuano la formula corretta:
=SE(B4="si";B1/2+C5+B1/2; B1/2+C5)



verifica su un nuovo problema

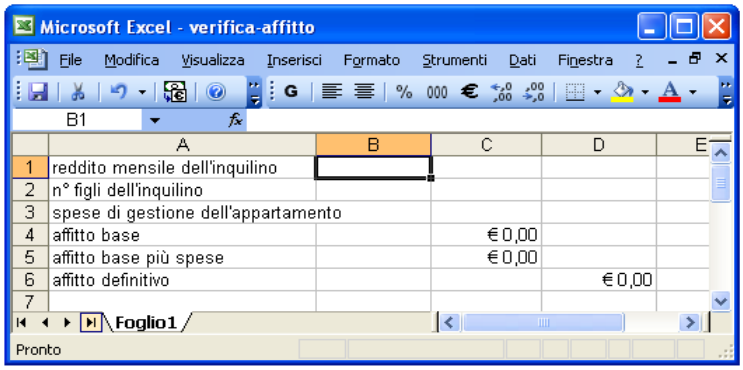
Possiamo ora individuare altri problemi, già svolti nelle fasi precedenti, e svolgere autonomamente tutto il percorso, aprendo dei fogli elettronici e compilandoli, controllando poi la correttezza dei risultati forniti da calcolatore.

ed ora al lavoro!

Per risolvere il seguente problema generalizzato:

Il Comune decide di far pagare l'affitto mensile delle case popolari in maniera proporzionale al reddito degli inquilini e cercando di agevolare le famiglie numerose.
L'affitto base sarà calcolato facendo pagare 1/10 del reddito. Ad esso saranno sommate le spese di gestione dell'appartamento.
Alla cifra così ottenuta andrà sottratto un suo 5% solo in caso di famiglie con più di 3 figli.
Il calcolatore deve calcolare l'affitto mensile definitivo di un appartamento.

è stato preparato il foglio di lavoro indicato a fianco:



1. La cella **C4** è stata formattata nella categoria: _____
2. La cella **A1** è stata formattata nella categoria: _____
3. Il contenuto della cella C4 sarà: _____
4. Il contenuto della cella C5 sarà: _____
5. La formula inserita nella cella **D6** sarà: = _____
6. Se nella cella **B1** è stato messo il valore **1000**, il valore contenuto della cella **C4** sarà: _____
7. Inoltre se nella cella **B3** vi è il valore **100**, il valore contenuto nella cella **C5** sarà: _____
8. Inoltre se nella cella **B2** è stato inserito il valore **4**, il valore contenuto nella cella **D6** sarà: _____

.

PROGRAMMI COMPLESSI

la gestione di un ambiente di lavoro

Buona parte dei programmi "professionali" che vediamo in azione sui calcolatori di banche, uffici pubblici, aziende private ecc... entrano in funzione con l'accensione dei calcolatori e terminano il loro funzionamento con il termine delle attività lavorative. Durante tutta la giornata l'operatore sceglierà, all'interno di un "menù" che compare sullo schermo, il tipo di attività che il calcolatore dovrà svolgere. Al termine di questa attività il calcolatore tornerà al menù principale pronto a svolgere nuovi lavori. Questo evita all'operatore, spesso inesperto, di dover caricare di volta in volta il programma necessario.

Per lavorare in questo modo ("user frienly") è però necessario realizzare grandi programmi che contengono tutto ciò che il calcolatore può svolgere in quel campo di attività.

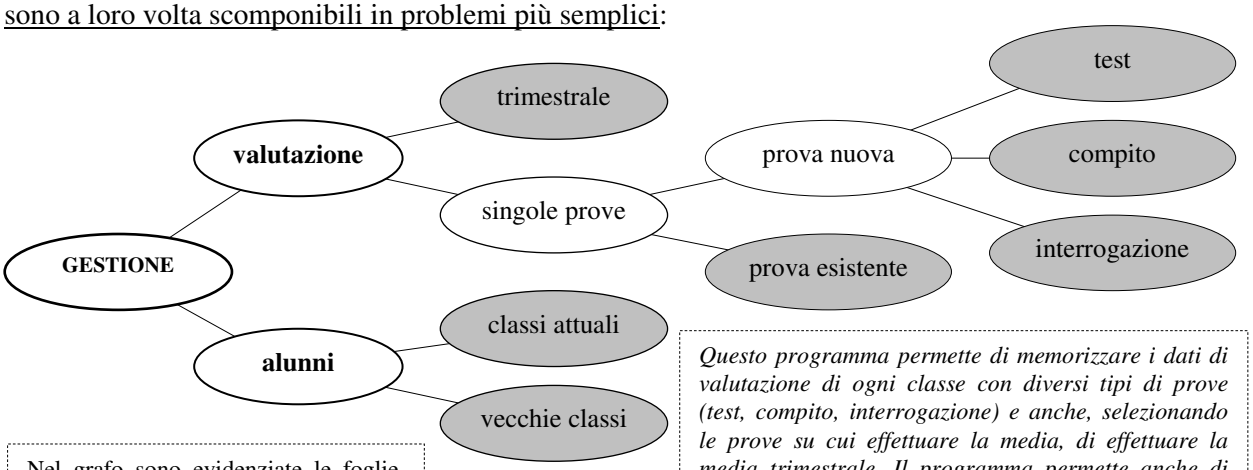
Chi progetta questi programmi (abbandonati gli oramai non più utili diagrammi di flusso) dovrà prendere in esame l'ambiente di lavoro nel quale essi verranno utilizzati ed individuare tutte le attività che fanno capo a quell'ambiente (ad esempio tutte le attività che possono essere svolte allo sportello di una banca oppure a quello di un ufficio anagrafe).

Individuare i collegamenti tra le attività connesse ad una determinata situazione di lavoro è fondamentale per poi realizzarne un buon programma di gestione. Per realizzare tale attività di studio può essere utile utilizzare dei grafi di scomposizione (vedi U.di A. Algoritmi) questa volta non per collocarvi le singoli azioni da compiere, ma per individuare i problemi semplici in cui suddividere il problema complesso.

Programmare utilizzando la programmazione strutturata

Il **grafico di progetto** è dunque un albero di scomposizione utilizzato per individuare le parti in cui dividere il campo di attività preso in esame.

Vediamo ora l'albero di scomposizione realizzato per la progettazione di un programma chiamato "gestione" con cui il nostro insegnante di E.T. gestisce la valutazione degli alunni. Attraverso vari livelli di scomposizione possiamo individuare i diversi problemi in cui può essere scomposto, alcuni dei quali sono a loro volta scomponibili in problemi più semplici:



Nel grafo sono evidenziate le foglie, cioè i diversi problemi che è possibile risolvere; a cui si accede dopo una serie di scelte

Questo programma permette di memorizzare i dati di valutazione di ogni classe con diversi tipi di prove (test, compito, interrogazione) e anche, selezionando le prove su cui effettuare la media, di effettuare la media trimestrale. Il programma permette anche di ottenere, su schermo e su stampante, gli elenchi degli alunni delle classi attuali e di quelle che oramai hanno lasciato la scuola

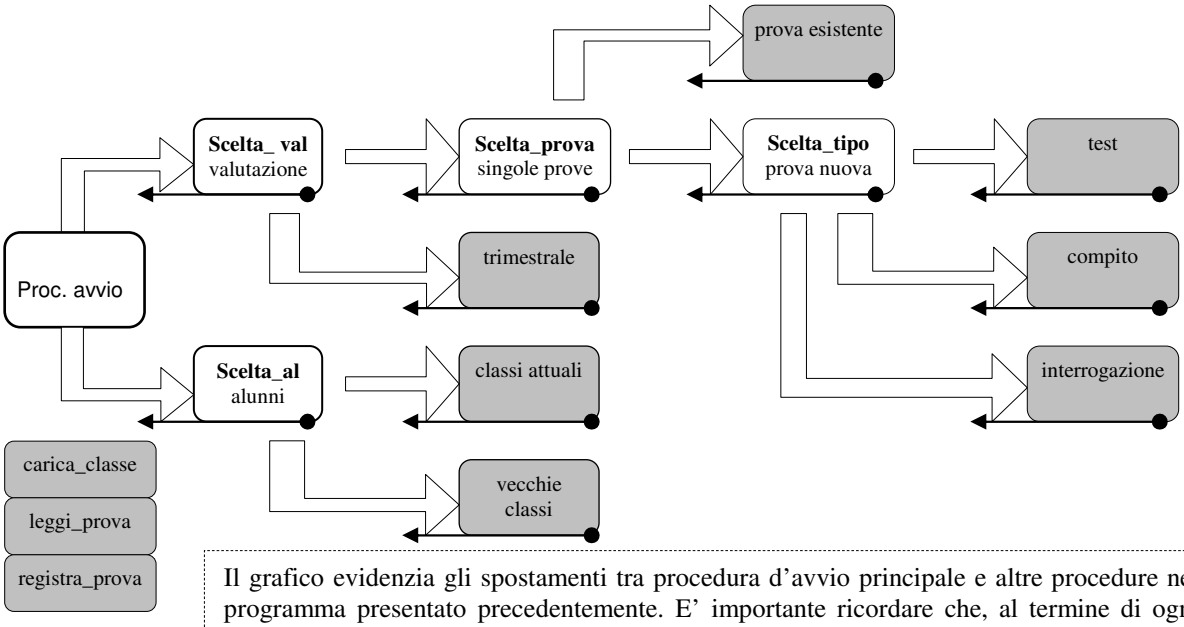
Nella nostra realtà quotidiana noi siamo abituati a suddividere un problema complesso in problemi più semplici, a ognuno dei quali assegniamo un ***nome*** che ha un ***legame di significato*** con il ***problema semplice*** che intende risolvere (*ad es. chiameremo "apparecchiare la tavola" la serie di azioni semplici che compiamo ogni volta che dobbiamo preparare la tavola per mangiare*).

Quando si deve realizzare un programma per gestire un problema complesso è utile lavorare con le stesse modalità. E' bene infatti dare un nome alle singole procedure che formano il programma. Questa possibilità permette di organizzare la propria attività di programmazione secondo un metodo che normalmente viene chiamato ***programmazione strutturata***.

procedura principale, procedure, procedure comuni

Vi sarà una **procedura principale** (o d'**avvio**) nella quale saranno collocate una serie di istruzioni di chiamata (dette **richiami di procedura**) delle procedure in cui è stato scomposto il problema complesso. La sua funzione è solo quella di indicare al calcolatore, in sequenza, i nomi delle procedure in cui è stato scomposto il problema e che saranno da eseguire quando sarà dato l'ordine di esecuzione. La procedura d'avvio è spesso chiamata **menù principale** perché la chiamata delle procedure è collegata a controlli effettuati su dati immessi dall'operatore. Anche nell'esempio appena fatto, dopo il **menù principale** vi sono altri **menù subordinati** (che nel grafo di scomposizione corrispondono ai nodi) che permettono di effettuare in successione la serie di scelte che porta alla procedura che deve essere eseguita.
E' anche compito della procedura d'avvio contenere alcuni ordini di carattere generale che servono per preparare il calcolatore al lavoro come, ad esempio, il **dimensionamento/dichiarazione** delle strutture di dati necessarie.
A questo proposito è bene ricordare che in LIBERTYBASIC, come in tutti i linguaggi di programmazione più evoluti, il valore delle variabili è **locale**. Cioè il valore inserito in una variabile è valido solo nella procedura in cui è stato inserito e non nelle altre a meno che nella procedura principale la variabile sia resa globale. Ad esempio con:

global a$ il valore che verrà inserito in a$ sarà valido in tutte le procedure del programma.
(N.B. in Visual Basic l'istruzione **global** viene sostituita dall'istruzione **public**)



Il grafico evidenzia gli spostamenti tra procedura d'avvio principale e altre procedure nel programma presentato precedentemente. E' importante ricordare che, al termine di ogni procedura, l'esecuzione ritorna alla procedura di provenienza. A parte sono collocate alcune *procedure comuni* destinate ad essere chiamate in più occasioni.

la gestione con chiamate e ritorni (call / sub … end sub)

Lavorando in LIBERTYBASIC è possibile utilizzare il metodo della programmazione strutturata per individuare la procedura d'avvio, che sarà collocata all'inizio e che terminerà con end, e le varie procedure che saranno collocate di seguito e che termineranno sempre con un end sub. Questo ordine permetterà il **ritorno** dell'esecuzione alla procedura di provenienza. E' possibile dare nomi alle diverse procedure che saranno collocate in coda alla procedura di avvio.
La **chiamata**, durante l'esecuzione, dalla procedura d'avvio o, anche, da un'altra procedura (nidificazione di procedure), avviene con l'ordine:

call nomeprocedura ad esempio con call leggi_classe il calcolatore cercherà la procedura e la eseguirà per poi tornare all'istruzione che segue l'ordine di chiamata.
L'ordine:

sub nomeprocedura segna l'inizio della procedura che terminerà con **end sub**

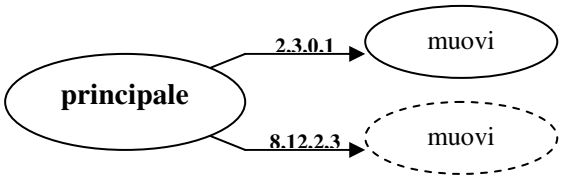
E' anche possibile, insieme alla chiamata, inviare alla procedura anche alcuni valori di dati da inserire in variabili. Ad esempio con:

```
call esempio "calcolo",12,2
end
sub esempio a$,c,d
e=c*d
print a$,e
end sub
```

il calcolatore, trovata la procedura esempio, inserisce i tre dati che seguono l'ordine di chiamata nelle tre variabili collocate all'inizio della procedura

un altro esempio

Il programma che segue è un anticipo della nostra attività di robotica. Vi è una procedura principale che, grazie ad un **puntatore** all'inizio e ad un goto alla fine, è ricorsiva.
Vi è una procedura ***muovi*** che accende un motore, ne inverte il movimento ad una variazione del primo sensore e lo ferma ad una variazione del secondo sensore. A seconda dei valori trasmessi, la procedura può controllare diversi motori (nel sottostante esempio sono i primi due).




```
' *** programma STRUTTURATA2
' *** il programma funziona con due motori (prime 2 porte) e quattro
    sensori (prime 4 porte),due per motore
' *** un sensore inverte il movimento del motore, l'altro lo arresta

call nconnectsi 'connessione con NIOM32
' procedura principale
[principale]
cls:print "scrivi quale motore vuoi attivare "
print "1 = primo motore"
print "2 = secondo motore"
print "3 = fine del programma"
input m
select case m
case 1: call muovi 2,3,0,1
case 2: call muovi 8,12,2,3
case 3: call nconnectno: end
end select
goto [principale]

' inizio procedura 'muovi'
sub muovi  mot1, mot2, numsens1, numsens2
call nwrite 9,0,mot1
a = nread(1,numsens1):b = a
   do while a = b
      b = nread(1,numsens1)
   loop
call nwrite 9,0,mot2
a = nread(1,numsens2):b = a
   do while a = b
      b = nread(1,numsens2)
   loop
call nwrite 9,0,0
end sub
```

La seguente mappa descrive ciò di cui ci stiamo occupando in questa ultima fase: