UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Fondamenti di Informatica

Ing. Alba Amato, PhD

alba.amato@unina2.it

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

LA STRUTTURA DEI PROGRAMMI

Lucidi tratti da:

Alla scoperta dei fondamenti dell'informatica. Un viaggio nel mondo dei bit
di Angelo Chianese, Vincenzo Moscato, Antonio Picariello

capitolo 4 -par. 4.1, 4.3

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Le frasi di un linguaggio di programmazione

- Tutti i linguaggi di alto livello prevedono quattro tipologie di frasi diverse:
  - le **frasi di commento**, che permettono l'introduzione di frasi in linguaggio naturale utili a rendere più comprensibili i programmi ad un lettore umano; le frasi di commento non vengono tradotte in linguaggio macchina
  - le **dichiarazioni**, con le quali il programmatore da ordini al traduttore del linguaggio di programmazione; anche le dichiarazioni non vengono tradotte in linguaggio macchina poiché servono solo a guidare il processo di traduzione.
  - le **istruzioni** di calcolo ed assegnazione, che tradotte in linguaggio macchina indicano al processore le operazioni da svolgere;
  - le **strutture di controllo**, che definiscono l'ordine di esecuzione delle istruzioni;

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Le frasi di commento

- Le frasi di commento sono frasi in linguaggio naturale prive di ogni valore esecutivo o dichiarativo che consentono di migliorare la leggibilità e la chiarezza del programma.

  Si distinguono in asserzioni e motivazioni.

  –*Le asserzioni sono commenti destinati a fissare in un punto del programma lo stato di una o più variabili. Per tale motivo permettono di chiarire le condizioni nelle quali vengono adoperare le istruzioni successive.*

  –*Le motivazioni sono invece commenti destinati a chiarire le ragioni e/o gli obiettivi per i quali il blocco di programma, successivo al commento, viene scritto.*

  Es: int var; //var è il nome di una variabile

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Le dichiarazioni

- Tra le tante dichiarazioni le più importanti sono quelle con cui si definiscono le **variabili** sulle quali si svolgono le elaborazioni dell'algoritmo.
  - **Le variabili vanno sempre dichiarate prima di poter essere usate!!**

  Ad una variabile corrisponde una porzione di memoria la cui dimensione e le cui regole di uso dipendono dal suo tipo.

  per *tipo di dato* si intende *un insieme di valori associati a delle operazioni definite su di essi*

  - Es: int var;
  - In questo modo si riserva uno spazio in memoria che può contenere un intero egli si assegna l'etichetta **var**;
  - Lo spazio di memoria potrebbe contenere già qualche valore spurio, per cui è necessario inizializzare la variabile

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Le istruzioni di calcolo ed assegnazione

- Con le istruzioni elementari di calcolo ed assegnazione si prescrive il calcolo di una qualche espressione con memorizzazione del risultato.
- Tali istruzioni prevedono che l'esecutore debba prima risolvere un'espressione e, solo quando ne ha prodotto il risultato, assegnare quest' ultimo ad una variabile
  - Es: v ← *espressione;*

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

I costrutti di controllo

- I ***costrutti di controllo*** indicano all'esecutore l'ordine in cui le operazioni devono essere eseguite. Essi devono essere scelti in modo da rispecchiare quanto più possibile i meccanismi che naturalmente vengono usati quando si descrive (ad esempio in italiano) una qualsiasi successione di operazioni.

  Si dividono in costrutti di sequenza, selezione ed iterazione.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

I costrutti di controllo: Sequenza

- Il costrutto di controllo più semplice è quello che specifica che due o più operazioni (elementari o no) devono essere eseguite una dopo l'altra.
- Tali costrutti sono detti *costrutti di sequenza* e vengono indicati nel seguente modo:

```
PRIMA     stacca il contatore
POI       sostituisci la presa
QUINDI    riattacca il contatore
```

- o anche con una notazione più compatta che usa il carattere punto e virgola come indicatore dell'istruzione successiva:
- Il punto e virgola consente anche di scrivere più azioni della sequenza su di uno stesso rigo:

```
stacca il contatore;
sostituisci la presa;
riattacca il contatore
```

```
stacca il contatore; sostituisci la presa;
riattacca il contatore
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

I costrutti di controllo: Costrutti Condizionali

- Alcune volte i costrutti di sequenza possono prescrivere la selezione di un'operazione
- I **costrutti condizionali** consentono di subordinare l'esecuzione di una certa operazione al verificarsi o meno di una specificata condizione. La notazione tipicamente utilizzata è IF-THEN-ELSE, ma si può anche avere solo IF-THEN

SE (la carta scoperta è quadri)
ALLORA vinci
ALTRIMENTI perdi

IF (la carta scoperta è quadri)
then vinci
else perdi

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

I costrutti di controllo: Costrutti di Selezione

- In alcuni casi i costrutti di sequenza possono prescrivere la selezione di una sola operazione:

SE (hai fame) **ALLORA** mangia	**IF** (hai fame) **then** mangia

- Altre volte, invece, si può voler selezionare un'operazione tra più di due alternative:

NEL CASO (in cui il colore del semaforo) è ROSSO: fermati è VERDE: passa l'incrocio è GIALLO: passa con prudenza o fermati	**CASE** (colore del semaforo) **OF** ROSSO: fermati VERDE: passa l'incrocio GIALLO: passa con prudenza o fermati **END**

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

I costrutti di controllo: Costrutti Iterativi

- I *costrutti iterativi* prescrivono di ripetere l'esecuzione di una o più operazioni; tale ripetizione viene sospesa al verificarsi di un evento. Sono costrutti iterativi:

RIPETI i compiti **FINCHE'** (suona la sveglia)	**REPEAT** i compiti **UNTIL** (suona sveglia)
MENTRE (piove) **DEVI** usare l'ombrello	**WHILE** (piove) **DO** usa l'ombrello
PER (10 giorni) **DEVI** non venire all'università	**FOR** giorni:=1 **TO** 10 **DO** non venire all'università

  - Il primo costrutto ha termine quando diventa vera la condizione (la sveglia suona),
  - il secondo invece quando la condizione diventa falsa (finisce di piovere).
  - Infine nel terzo esempio il numero di ripetizioni è noto a priori: chiameremo queste iterazioni cicliche o iterazioni enumerative per distinguerle dalle altre due.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

I costrutti Equivalenti e Funzionalmente Equivalenti

- Per poter confrontare costrutti con caratteristiche simili si introducono i concetti di *equivalenza* e di *equivalenza funzionale*.
  - Si dicono ***equivalenti*** due programmi che evocano le stesse sequenze di esecuzione.
  - Sono invece ***funzionalmente equivalenti*** due programmi che, sollecitati nello stesso modo, producono lo stesso risultato.
  - Si noti che due programmi equivalenti sono anche funzionalmente equivalenti, mentre non è vero, in generale, che due programmi funzionalmente equivalenti sono anche equivalenti.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

I costrutti Equivalenti e Funzionalmente Equivalenti

- I due programmi seguenti producono lo stesso risultato (il prodotto di x per y) se le due variabili sono positive,
- e risultati diversi negli altri casi (la somma e la divisione se sono entrambe negative, la somma o il prodotto e il prodotto e il rapporto se il loro segno è discorde).

assegna valore ad x e y; **if** (x≤0) **then** stampa il valore di x+y **else** stampa il valore di x*y	assegna valore ad x e y; **if** (y>0) **then** stampa il valore di x*y **else** stampa il valore di x/y

- L'equivalenza funzionale si riferisce così ad un ben preciso insieme di valori di ingresso: nel caso delle variabili x e y entrambe positive i due programmi sono funzionalmente equivalenti, mentre non lo sono negli altri casi.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

I costrutti Equivalenti e Funzionalmente Equivalenti

- è sempre possibile ricondurre un costrutto **if then-else** ad una sequenza di soli **if-then**

Sono equivalenti solo se azione1 non cambia il valore di condizione!!

```
if    (condizione)
then  azione 1
else  azione 2
```

```
if    (condizione)
then  azione 1
if    (not condizione)
then  azione 2
```

- La struttura case può essere ricondotta ad un insieme di if disposti l'uno dentro l'altro.

```
case (espressione) of
      a: azione 1
      b: azione 2
      c: azione 3
end;
```

```
if (espressione=a)
then azione 1
else
     if (espressione=b)
     then azione 2
     else
          if (espressione=c)
          then azione 3
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

I costrutti Equivalenti e Funzionalmente Equivalenti

- Per quanto riguarda le due strutture iterative **while** e **repeat** si noti che è sempre possibile ricondurre l'una all'altra.
- Il while prescrive prima la valutazione della condizione e dopo l'esecuzione delle azioni qualora il valore ottenuto sia vero, mentre con il repeat la condizione viene valutata dopo e quindi le azioni vengono fatte almeno una volta

repeat azione **until** (condizione)	Azione **while** (**not** condizione) **do** azione

while (condizione) **do** azione	**repeat** **if** (condizione) **then** azione **until** (**not** condizione)

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

I costrutti Equivalenti e Funzionalmente Equivalenti

- Anche la struttura ciclica **for** è riconducibile ad una iterativa **while** o **repeat**
- Un ciclo iterativo prescrive la ripetizione di azioni un numero di volte fissato a priori e determinato dal fatto che una variabile detta *contatore* di ciclo, a partire da un valore iniziale raggiunge un valore finale o per valori crescenti (indicato dal to) o decrescenti (indicato dal down to).

for i:=vp **to** vg **do** azione	i:= vp; **while** i≤vg **do begin** azione; i:= i +1 **end**

for i:=vg **downto** vp **do** azione	i:= vg **while** i≥vp **do begin** azione; i:=i-1 **End**

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Modularità

- Per facilitare la scrittura e la comprensione di un programma è utile individuare al suo interno dei **moduli funzionali**, ciascuno dei quali realizza un singolo e ben preciso compito e ha un solo punto di ingresso e di uscita.
  - Ogni sotto-problema viene risolto da un pezzo di programma detto **sottoprogramma**
  - Un sottoprogramma ha un **nome**, con cui può essere richiamato all'interno di un programma
  - Se il sottoprogramma fornisce un risultato, allora si chiama **funzione**, altrimenti si chiama **procedura**

programma torta;

{

I_1: impasta farina, burro, uova, sale e zucchero fino ad ottenere una pasta soffice;

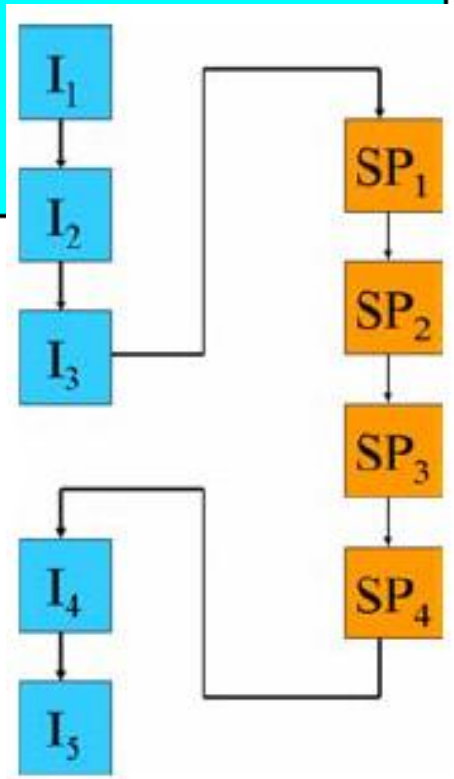
I_2: inforna la pasta per una decina di minuti;

I_3: ***prepara_la_crema;***

I_4: distribuisci la crema sulla sfoglia;

I_5: ricopri con frutta di stagione tagliata a dadini e gelatina

}



sottoprogramma prepara_la crema;

{

SP_1: sbatti le uova con lo zucchero;

SP_2: stempera la farina nel latte;

SP_3: versa il latte nelle uova e nello zucchero;

SP_4: mette sul fuoco e porta ad ebollizione

}

Indicazione di un sottoprogramma

- L'*indicazione* o *definizione* di un sottoprogramma si compone di due parti: il **titolo** e il **corpo**.
  - Il titolo comprende: il nome del sottoprogramma; i parametri di ingresso con il loro tipo (tipicamente racchiusi fra parentesi tonde); il risultato con il suo tipo (una funzione ritorna un solo risultato)
  - Il corpo comprende una sequenza di dichiarazioni e di istruzioni tipicamente racchiusa fra parentesi graffe {}
- Una volta definito, è possibile richiamare il sottoprogramma utilizzando il suo nome e fornendo i parametri di ingresso, se presenti.
  - –L'invocazione del sottoprogramma è un'istruzione come le altre
  - –Evita di riscrivere più volte lo stesso insieme di istruzioni laddove queste vadano ripetute
  - –Consente di concentrarsi su un problema di piccole dimensioni rispetto al programma principale

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Parametrizzazione

- L'utilizzo dei sottoprogrammi è particolarmente vantaggioso quando le operazioni da essi effettuate possono essere «adattate» su dati diversi

```
{
    stampa('Dammi riempimento:');
    leggi(riemp)
}
```

```
{
    stampa('Numero fogli :');
    leggi(num_fogli)
}
```

NOTA: stampa() e leggi() sono dei sottoprogrammi; riemp e num_fogli sono variabili che devono essere dichiarate

- I due sottoprogrammi differiscono solo per le costanti da stampare e per le variabili di cui si devono leggere i valori

```
{
    stampa(messaggio);
    leggi(x)
}
```

- *Messaggio* è una variabile che contiene una stringa che deve essere passata al sottoprogramma(input);
- *X* è una variabile dove verrà salvato un numero fornito tramite tastiera (input/output)

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Parametrizzazione

- Si definiscono **parametri formali** i parametri utilizzati nel corpo del sottoprogramma (*messaggio* e *x* dell'esempio); essi vanno indicati nel titolo del sottoprogramma al momento della definizione del sottoprogramma.
- Quando il sottoprogramma viene richiamato all'interno di un altro programma, ai parametri formali verranno sostituiti i **parametri effettivi**
- **I parametri effettivi e quelli formali devono essere dello stesso tipo e devono essere specificati nello stesso ordine!**

DEFINIZIONE:

```
stampa_valore (messaggio, x){
  stampa(messaggio);
  leggi (x);
}
```

INVOCAZIONE:

```
stampa_valore('Dammi il riempimento:',riemp); oppure
stampa_valore('Numero di fogli:',num_fogli);
```

Sostituzione dei Parametri Formali

- I parametri effettivi possono essere sostituiti a quelli formali secondo 3 diverse modalità: *per valore, per riferimento* o *per nome*.
- **Passaggio per valore**: al parametro formale viene assegnato il valore del parametro effettivo
  - –Il parametro effettivo può essere un'espressione e, come casi particolari di espressione, una costante o una variabile. Se è un‘ espressione, se ne calcola il valore e lo si assegna al corrispondente parametro formale.
  - –Nel caso di variabili, la sostituzione per valore garantisce che la variabile passata come parametro effettivo non venga alterata (il sottoprogramma lavora sulla sua copia)
  - –All’atto della chiamata viene fatta la copia del parametro effettivo con conseguente consumo di tempo e spazio

Sostituzione dei Parametri Formali

- **Passaggio per riferimento**: al parametro formale viene assegnato l'indirizzo del parametro effettivo
  - –Il parametro effettivo può essere solo una variabile
  - –Il sottoprogramma lavora direttamente sulla variabile in memoria
  - –Il parametro effettivo occupa solo lo spazio necessario per contenere un indirizzo
- **Passaggio per nome**: il parametro formale viene testualmente sostituito dai parametri effettivi senza alcuna valutazione
  - –Durante l'esecuzione del programma ogni volta che compare il parametro formale esso viene sostituito al quello effettivo (è come se si riscrivesse il programma)
  - –Questo metodo è ormai presente solo in pochissimi linguaggi

Sostituzione dei Parametri Formali

- Per quanto concerne la scelta del metodo di sostituzione dei parametri si possono seguire le seguenti regole base:
  - quando un parametro rappresenta un argomento di una procedura, si sceglie la sostituzione per valore;
  - quando un parametro rappresenta invece un risultato, occorre usare la sostituzione per riferimento.
- Osservando tale problema da una visuale differente, possiamo classificare i parametri formali in:
  - parametri di ingresso al sottoprogramma;
  - parametri di uscita dal sottoprogramma;
  - parametri modificabili dal sottoprogramma.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Programma Principale e Funzioni

- Si chiamerà **programma principale** quella unità di programma che si interfaccia direttamente con il sistema operativo.
- In molti linguaggi il programma principale si chiama **main**
  - La funzione main è il punto da cui comincia l'esecuzione, indipendentemente dalla posizione nel listato
  - La funzione main può prendere dei parametri in ingresso (che servono per interagire col sistema operativo) e restituisce solitamente un valore intero, che indica l'esito dell'esecuzione (codice di errore)
  - **Tutti i programmi devono contenere la funzione main**

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Programma Principale e Funzioni

- Quando una procedura fornisce un unico risultato, può essere organizzata in funzione, in modo che il suo nome corrisponda proprio a tale risultato.
- Per esempio, si organizzano in funzione le funzioni matematiche che devono essere utilizzate all'interno del programma, o le espressioni o porzioni di esse che devono essere usate più volte all'interno di uno stesso programma. In tali casi si ha che:
  - il nome della funzione è l'unico risultato della procedura;
  - poiché il nome della procedura è una variabile a tutti gli effetti, deve essere dotata di un tipo che deve essere indicato nell'intestazione;
  - si può assegnare valore a tale variabile soltanto nel corpo della funzione stessa;
  - si può usare il valore di tale variabile inserendone il nome in espressioni che si trovano all'esterno del corpo della funzione.

```
funzione somma(a,b:interi):intera
{
      somma:=a+b
}
```

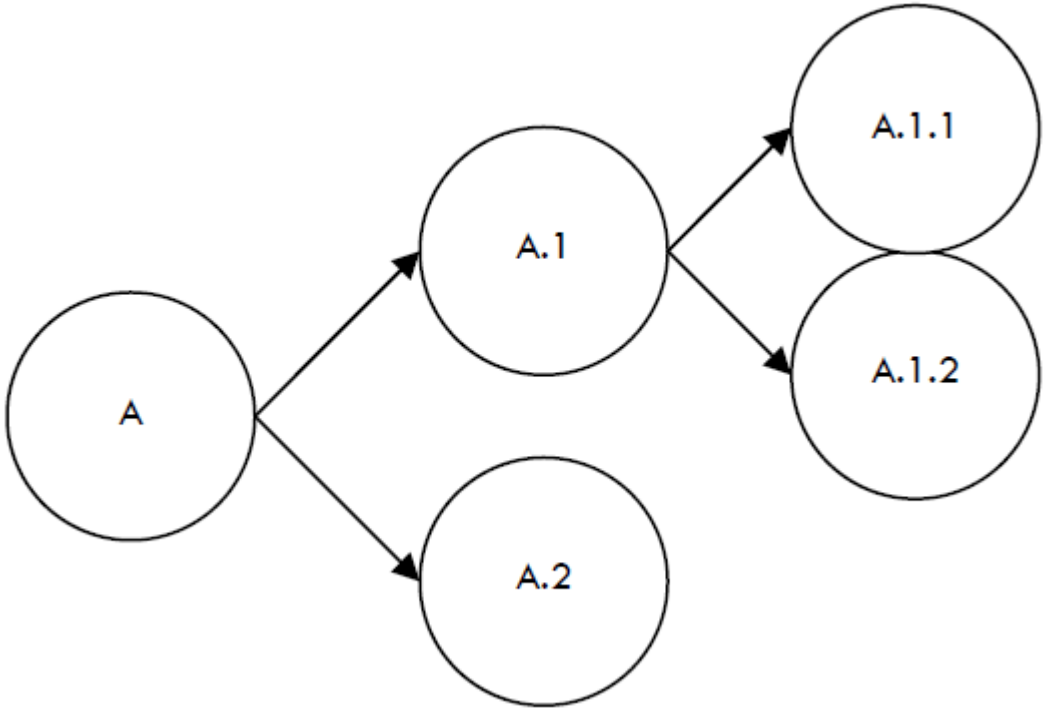
Può essere utilizzata così:

```
x:=a+somma(3,y)/3;
stampa(somma(alfa,beta));
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Visibilità

- L'uso dei sottoprogrammi permette di costruire programmi non costituiti da una unica sequenza di istruzioni ma da una gerarchia di unità di programma ciascuna delle quali realizza una particolare istruzione in modo autonomo, con proprie definizioni di tipi, dichiarazioni di variabili e istruzioni.
- Ogni unità forma la base per il livello superiore (le unità che la chiamano) e si appoggia eventualmente su un livello di macchina inferiore (le procedure che sono chiamate al suo interno).



UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Visibilità

- Con il termine regole di ***scope*** (***visibilità***) si fa riferimento a quelle regole che consentono di determinare i campi di influenza di una variabile in tutte le varie parti costituenti un programma. In altre parole si dice scope di un oggetto la porzione di programma che è in grado di usarla.
- Se un oggetto - una costante, un tipo, una variabile, una procedura o una funzione - è definito ed usato solo all'interno di una determinata procedura, allora viene detto *locale* a quella procedura.
- Se, viceversa, è definito nell'unità di programma chiamante, ma risulta visibile alla procedura chiamata attraverso qualche meccanismo, allora viene detto *non locale* alla procedura.
- Infine viene detto *globale* se risulta definito nel programma principale, perché tale unità è l'unica che può rendere visibili i propri oggetti a tutte le altre in quanto rappresenta la radice della gerarchia di unità di programma (in altre parole è l'unica unità che chiama tutte le altre e che non è chiamata da nessun'altra).