LABORATORIO DI PROGRAMMAZIONE
Corso di laurea in matematica

11 – LE PROCEDURE

Marco Lapegna
Dipartimento di Matematica e Applicazioni
Universita' degli Studi di Napoli Federico II

wpage.unina.it/lapegna

Dalle lezioni precedenti

Sviluppo di algoritmi in pascal-like

- Variabili e costanti
- Variabili strutturate
  - array
- Strutture di controllo
  - if-then-else-endif
  - for-endfor
  - repeat-until
  - while-endwhile

concetti universali presenti in tutti i linguaggi di programmazione

Qualunque algoritmo puo' essere ricondotto ad un algoritmo equivalente basato su tali costrutti

Problema

Siano dati due array di reali

A e B rispettivamente di lunghezza N e M

Si vuole scrivere un algoritmo che calcoli il

massimo MAX dei due valori medi MediaA e MediaB dei due array

Esempio: N=8 M=10

A = (3 ; 6 ; 2 ; 9 ; 10 ; 1 ; 2 ; 5) ➔ MediaA = 4,75

B = (8 ; 3 ; 5 ; 2 ; 4 ; 9 ; 10 ; 0 ; 5 ; 7) ➔ MadiaB = 5,3

MAX = 5,3

possibile soluzione

L'algoritmo puo' essere basato sui seguenti passi

1) **Leggi** N e l'array A
2) **Leggi** M e l'array B
3) **Calcola** la media MediaA dei valori di A
4) **Calcola** la media MediaB dei valori di B
5) **Calcola** il massimo MAX tra MediaA e mediaB
6) **Stampa** il valore di MAX

Osservazione

le istruzioni relative **all'array B (blu)** sono le stesse di quelle relative **all'array A (rosse)**

(cambiano i **dati** su cui le istruzioni agiscono)

```
begin maxmedia
var A(100), B(100): array of float
var MediaA, MediaB, MAX: float
var i, j, N, M: integer
read N
for i=1 to N
   read A(i)
endfor
read M
for i=1 to M
   read B(i)
endfor
MediaA = 0
for i=1 to N
   MediaA = MediaA + A(i)
endfor
MediaA = MediaA/N
MediaB = 0
for j=1 to M
   MediaB = MediaB + B(i)
endfor
MediaB = MediaB/M
if (MediaA > MediaB) then
    MAX = MediaA
else
    MAX = MEDIAB
endif
print MAX
end maxmedia
```

algoritmo in pascal like per il calcolo del massimo dei due valori medi di due array

problema ulteriore

Nel caso in cui il numero di array dipende dal contenuto di una variabile NUM
non e' possibile scrivere l'algoritmo
(quante volte scrivere il gruppo di istruzioni nell'algoritmo?)

OBIETTIVO

individuare una metodologia per **scrivere una sola volta le istruzioni e riutilizzarle tutte le volte che servono**

SOLUZIONE

scrivere un "**sottoalgoritmo**" che puo' essere richiamato piu' volte con dati diversi

uso di un sottoalgoritmo

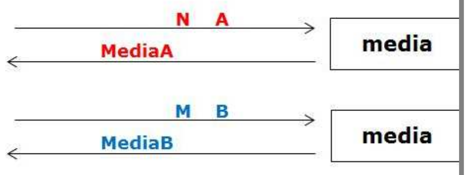
si supponga di avere a disposizione un **sottoalgoritmo "media"** che ha come

- **dati di input**: un array di una certa lunghezza
- **dati di output**: la media degli elementi dell'array

si supponga inoltre che tale sottoalgoritmo "**media**" sia in grado di **ricevere e restituire dati ad un altro algoritmo**

Possibile **affidare il calcolo della media** a tale sottoalgoritmo ogni volta che occorre

```
begin maxmedia
var A(100), B(100): array of float
var MediaA, MediaB, MAX: float
var i, j, N, M
read N
for i=1 to N
   read A(i)
endfor
read M
for i=1 to M
   read B(i)
endfor
```




```
if (MediaA > MediaB) then
    MAX = MediaA
else
    MAX = MediaB
endif
print MAX
end maxmedia
```

ruolo del sottoalgoritmo "media"

le procedure

l'esempio precedente mostra che sarebbe utile affidare il calcolo della media degli elementi di un array ad un opportuno sottoalgoritmo
(analogamente per la lettura degli array)

sottoalgoritmi richiamabili da altri algoritmi
=
PROCEDURE

PROBLEMA

stabilire il protocollo di comunicazione tra i due algoritmi

sviluppo di una procedura

una **procedura** e' un sottoalgoritmo
richiamabile da un altro algoritmo
in cui l'input e l'output **non avvengono mediante le istruzioni read e print**,
ma scambiando dati
con l'algoritmo chiamante

operativamente:

- rimozione delle istruzioni read e print
- sostituzione dell'inizio dell'algoritmo con la **testata della procedura** contenente i dati di input e output

```
begin media
var  X(100): array of float
var  k, P: integer
read   P
for k=1 to P
   read X(k)
endfor
Media = 0
for k=1 to P
   Media = Media + X(k)
endfor
Media = Media/P
print   Media
end media
```

algoritmo per la media degli elementi di un vettore

```
procedure media(in: P, X; out: Media)
var  X(100): array of float
var  i, P: integer
Media = 0
for i=1 to P
   Media = Media + X(i)
endfor
Media = Media/P
end media
```

procedura per la media degli elementi di un vettore

Testata della procedura

In generale

procedura *nomeprocedura* (in: *dati di input*; out: *dati di output*)

La testata della procedura contiene:

- Il **nome** della procedura
- I **dati di input e di output** da scambiare con l'algoritmo chiamante

La testata della procedura rappresenta l'unico canale di comunicazione con l'algoritmo chiamante

Utilizzo della procedura

L'algoritmo chiamante utilizza la procedura **attraverso il nome**, scambiando con essa i dati di input e di output

Piu' precisamente:

1) vengono scambiati i dati di input tra algoritmo chiamante e procedura

2) l'algoritmo chiamante viene sospeso fino al completamento della procedura

3) Al termine della procedura vengono scambiati i dati di output tra procedura e algoritmo chiamante

4) Viene ripresa l'esecuzione dell'algoritmo chiamante

La procedura puo' essere
utilizzata piu' volte con dati diversi

```
begin maxmedia
var  A(100), B(100): array of float
var  MediaA, MediaB, MAX: float
var  i, j, N, M: integer
read   N
for i=1 to N
   read A(i)
endfor
read   M
for i=1 to M
   read B(i)
endfor

media(in: N, A; out: MediaA);

media(in: M, B; out: MediaB);

if (MediaA > MediaB) then
    MAX = MediaA
else
    MAX = MEDIAB
endif
print   MAX
end maxmedia
```

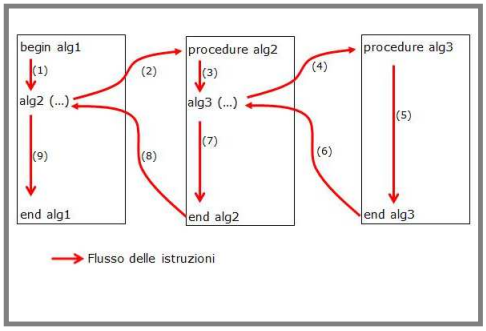
Utilizzo della procedura da parte dell'algoritmo chiamante.

Inoltre

Una procedura puo' richiamare a sua volta un'altra procedura

Ovviamente

1) L'ultima procedura ad iniziare sara' la prima a finire
2) L'algoritmo chiamante sara' l'ultimo a terminare



Chiamate di procedure innestate

Osservazioni importanti (1)

I dati nella testata della procedura sono chiamati **argomenti formali**

I dati nella chiamata della procedura all'interno dell'algoritmo chiamante sono chiamati **argomenti attuali (o effettivi)**

LA CORRISPONDENZA E' PER POSIZIONE

(e' possibile utilizzare cioe' nomi diversi per gli argomenti formali ed attuali)

E' necessario che gli argomenti che occupano la **stessa posizione** abbiano lo **stesso tipo**

Osservazioni importanti (2)

l'unico canale di comunicazione tra procedura ed algoritmo chiamante e' costituito dai dati presenti nella testata

Conseguenza:

variabili che hanno lo stesso nome nei due algoritmi **non hanno niente in comune !**
(si noti la variabile "i")

Problema

cosa viene scambiato tra algoritmo chiamante e procedura al momento della chiamata della procedura?

```
begin maxmedia
var A(100), B(100): array of float
var MediaA, MediaB, MAX: float
var i, j, N, M: integer
read  N
for i=1 to N
   read A(i)
endfor
read  M
for i=1 to M
   read B(i)
endfor

media(in: N, A; out: MediaA);

media(in: M, B; out: MediaB);

if (MediaA > MediaB) then
   MAX = MediaA
else
   MAX = MEDIAB
endif
print  MAX
end maxmedia
```

Algoritmo chiamante

```
procedure media(in: P, X; out: Media)
var  X(100): array of float
var  i, P: integer
Media = 0
for i=1 to P
   Media = Media + X(i)
endfor
Media = Media/P
end media
```

procedura.

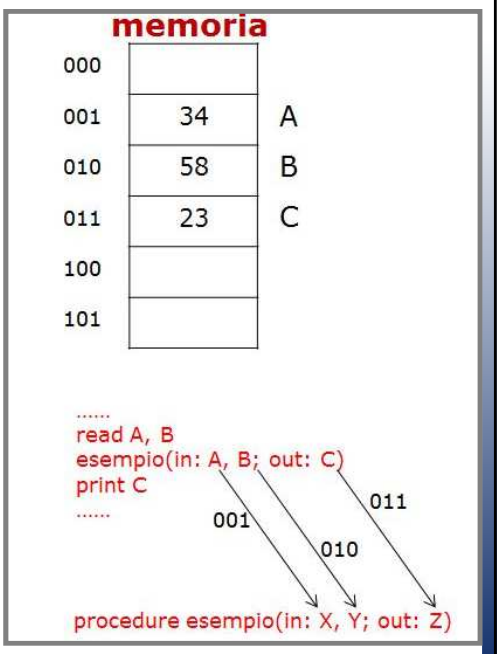
Passaggio per indirizzo

Viene comunicata la **posizione (indirizzo) in memoria** delle variabili che conengono i dati di input e output della procedura

La procedura opera direttamente sulle stesse variabili dell'algoritmo chiamante

Nell'esempio, le variabili A, B e C dell'algoritmo chiamante prendono temporaneamente i nomi X, Y e Z nella procedura

Utilizzato nel linguaggio Fortran



Passaggio per indirizzo

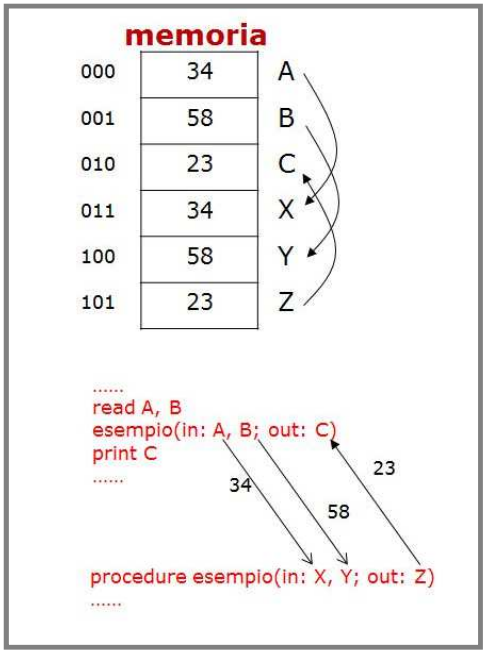
Passaggio per valore

Viene comunicato il **valore** delle variabili che conengono i dati di input e output della procedura

La procedura utilizza delle variabili proprie, in cui copia i valori provenienti dall'algoritmo chiamante

Nell'esempio, le variabili A e B dell'algoritmo chiamante vengono copiate nelle variabili X e Y nella procedura, mentre il valore di output Z viene copiato nella variabile C dell'algoritmo chiamante

Utilizzato nel linguaggio C (e derivati)



Passaggio per valore

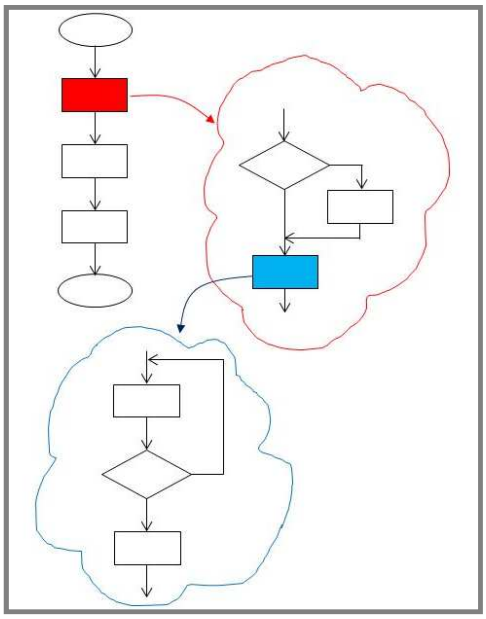
Sviluppo modulare degli algoritmi

Algoritmi complessi possono essere decomposti in procedure piu' semplici mediante "raffinamenti successivi"

Vantaggi:

- Minore probabilita' di errori
- E' possibile testare separatamente le procedure per verificarne il funzionamento
- Possibilita' di dividere il lavoro tra piu' persone

Tecnica di sviluppo TOP-DOWN



Sviluppo top-down di un algoritmo

Esempio

Sviluppare un algoritmo "somma" per la somma di due vettori

C = A + B

con la tecnica dei raffinamenti successivi

Primo raffinamento:

Individuare i passi fondamentali e esprimerli sotto forma di procedure, individuandone gli argomenti di input e output

```
begin somma
var A(100), B(100): array of float
var C(100):  array of float
var N: integer

leggivettore(out: N, A)

leggivettore(out: N, B)

sommavettori(in: A, B, N; out: C)

stampavettore(in: N, C)

end somma
```

Somma di vettori: primo raffinamento

Esempio (cont.)

le procedure "leggivettore" e "stampavettore" rispettivamente leggono e stampano un vettore

leggivettore

- Dati di input: nessuno
- Dati di output: un array e la sua lunghezza

```
procedure leggivettore(out: P, X)
var X(100): array of float
var i, P: integer

read  P

for i=1 to P
     read  X(i)
endfor

end leggivettore
```

Procedura leggivettore

stampavettore

- Dati di input: un array e la sua lunghezza
- Dati di output: nessuno

```
procedure stampavettore(in: P, X)
var X(100): array of float
var i, P: integer

for i=1 to P
     print  X(i)
endfor

end stampavettore
```

Procedura stampavettore

Esempio (cont.)

La procedura **sommavettori** esegue la somma di due vettori

Dati di input:
- A e B (due vettori)
- N (lunghezza dei vettori)

Dati di output
- C (Il vettore somma)

Secondo raffinamento:

E' possibile esprimere la somma di una singola coppia di componenti mediante la procedura "somma2elementi" che esegue la somma di due dati float R e S

```
procedure sommavettori(in: X, Y, P; out: Z)
var  X(100), Y(100), Z(100): array of float
var  i, P: integer

for i=1 to P
     somma2elementi(in: X(i), Y(i); out: Z(i))
endfor

end sommavettori
```

Procedura sommavettori

```
procedure somma2elementi(in: R, S; out: T)
var  R, S, T :  float

T = R + S

end somma2elementi
```

Procedura somma2elementi

In conclusione

Le procedure

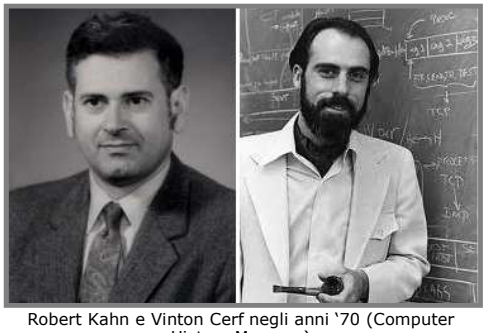
- Possono essere richiamate piu' volte con dati diversi
- Possibile innestare piu' livelli di procedure
- Rappresentano "scatole nere" per sviluppare algoritmi piu' complessi

Sono concetti universali presenti in tutti i linguaggi di programmazione

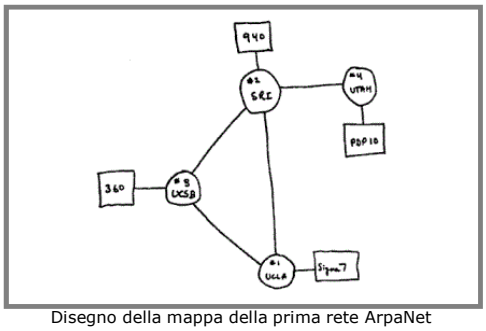
Un po' di storia (11)

Robert Kahn (1938) - Vinton Cerf (1943)

- Ingegnere il primo (New York) e matematico il secondo (Stanford), svilupparono nel 1972 i protocolli di trasmissione dati TCP e IP per la rete Arpanet, finanziata dal governo americano per collegare universita' e centri di ricerca.
- ArpaNet era composta da 4 nodi che si interfacciavano tra loro tramite computer piu' piccoli (router) che scomponevano i messaggi in pacchetti di dimensione fissa (commutazione di pacchetto). Il bandwidth era di 50 Kbit/sec.
- I primi 4 nodi di ArpaNet furono: l'UCLA, UC Santa Barbara, Stanford Research Institute e l'Università dello Utah.
- Vinsero entrambi il Turing Award nel 2004, ed insieme a L. Kleinrock e L. Roberts sono noti come i padri di Internet



Robert Kahn e Vinton Cerf negli anni '70 (Computer History Museum)



Disegno della mappa della prima rete ArpaNet