

# Corso di Programmazione

## Sottoprogrammi

### *Procedure e funzioni*

Prof.ssa Teresa Roselli  
[roselli@di.uniba.it](mailto:roselli@di.uniba.it)

# Durata e ambito di una variabile

- In alcuni linguaggi il tempo di vita di una variabile coincide con il *tempo di esecuzione* del programma o del sottoprogramma in cui la variabile è dichiarata
- La fase in cui è definita un'area di memoria per una variabile è detta *fase di allocazione*.
  - L'allocazione può essere eseguita:
    - dal compilatore (allocazione statica) in base alla struttura lessicale del programma
    - in esecuzione (allocazione dinamica)

# Effetti Collaterali

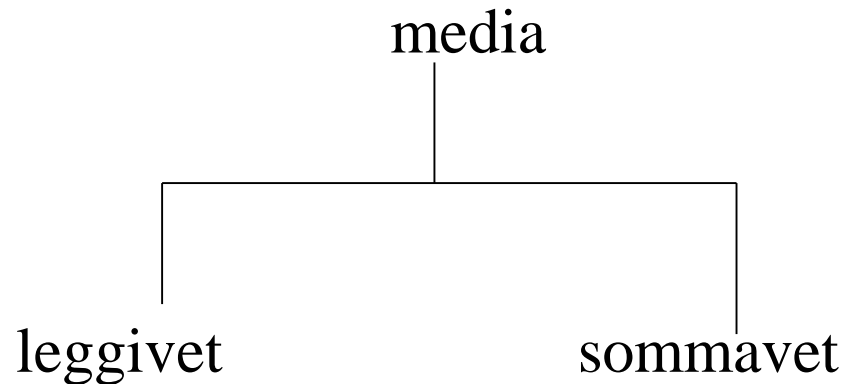
- Effetti di un sottoprogramma che altera il valore di una variabile non locale
  - La presenza di tali variabili impedisce che il sottoprogramma possa essere considerato come un'entità completa e autoconsistente
    - Non si riferisce esclusivamente alle sue costanti, variabili e parametri
- Attenzione: occorre valutare attentamente l'uso, all'interno di un sottoprogramma, di variabili non locali
  - Chiarezza
  - Sicurezza

# Sottoprogramma il countour model

```
PROGRAMMA media
  TIPO vettore=ARRAY(1,100) OF real
  vet:vettore
  n:integer
  m:real
  s:real
  SOTTOPROGRAMMA leggivet
    x:real
    i:integer
    BEGIN
      DO VARYING i FROM 1 TO n
        leggi x
        vet(i)=x
      REPEAT
    END
  SOTTOPROGRAMMA sommavet
    i:integer
    BEGIN
      s=0
      DO VARYING i FROM 1 TO n
        s=s+vet(i)
      REPEAT
    END
  BEGIN
    leggi n
    leggivet
    sommavet
    m=s/n
    stampa m
  END
```

# Sottoprogramma il countour model

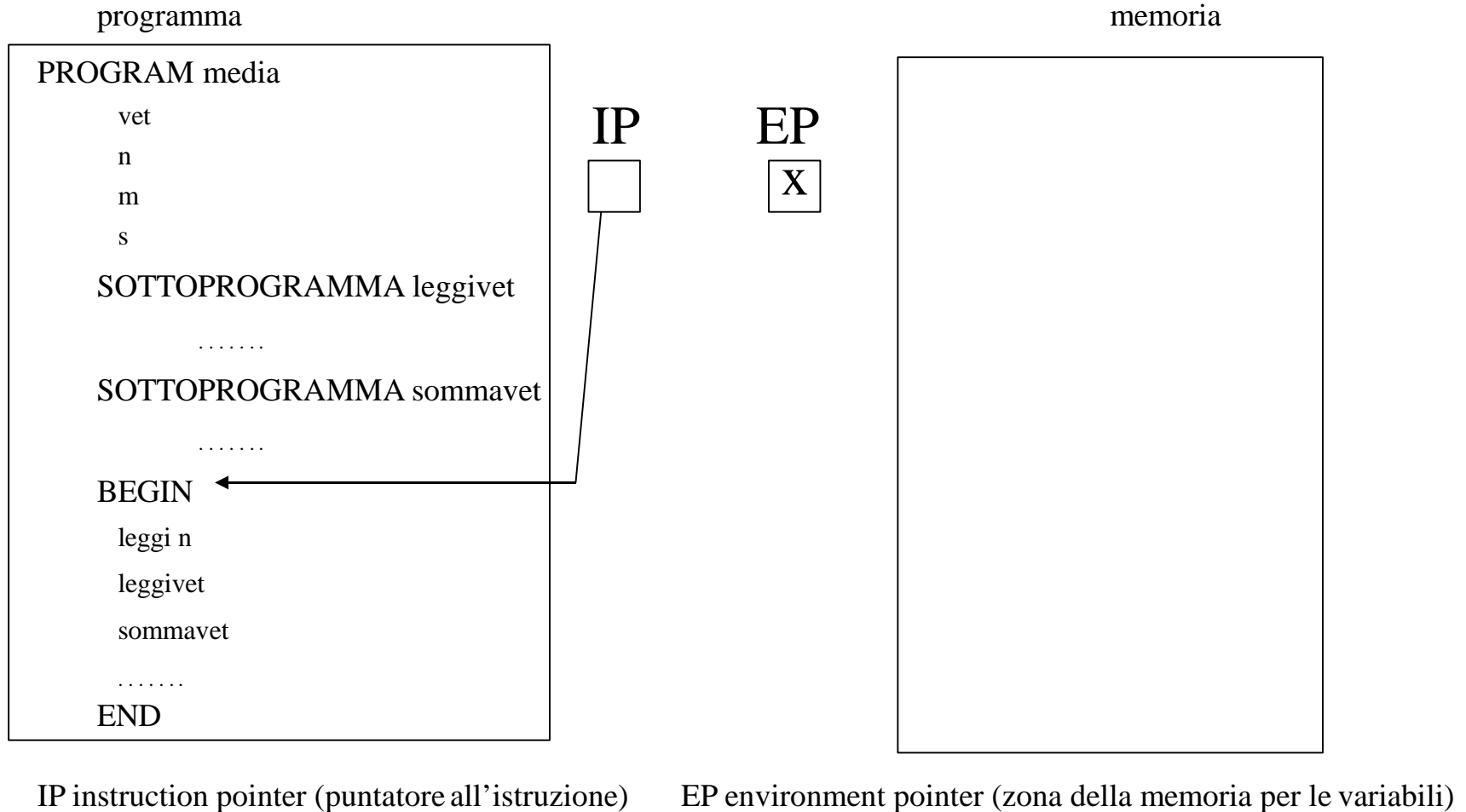
- La gerarchia di macchine è



- media cede ai sottoprogrammi il diritto di accesso a tutte le sue variabili
- ogni sottoprogramma ha le proprie variabili

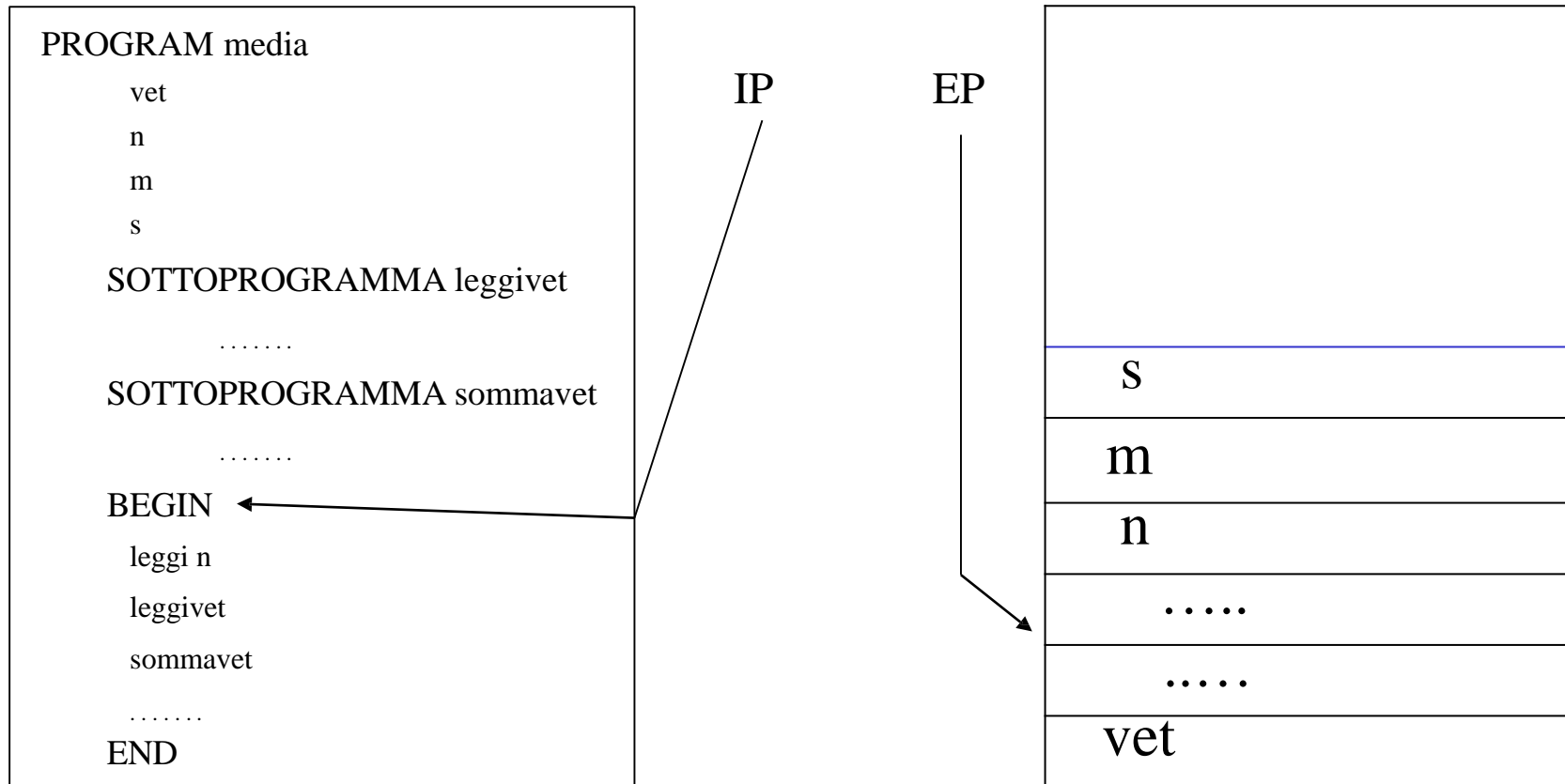
# Sottoprogramma il countour model

Il modello associato all'esecuzione del programma è detto countour model



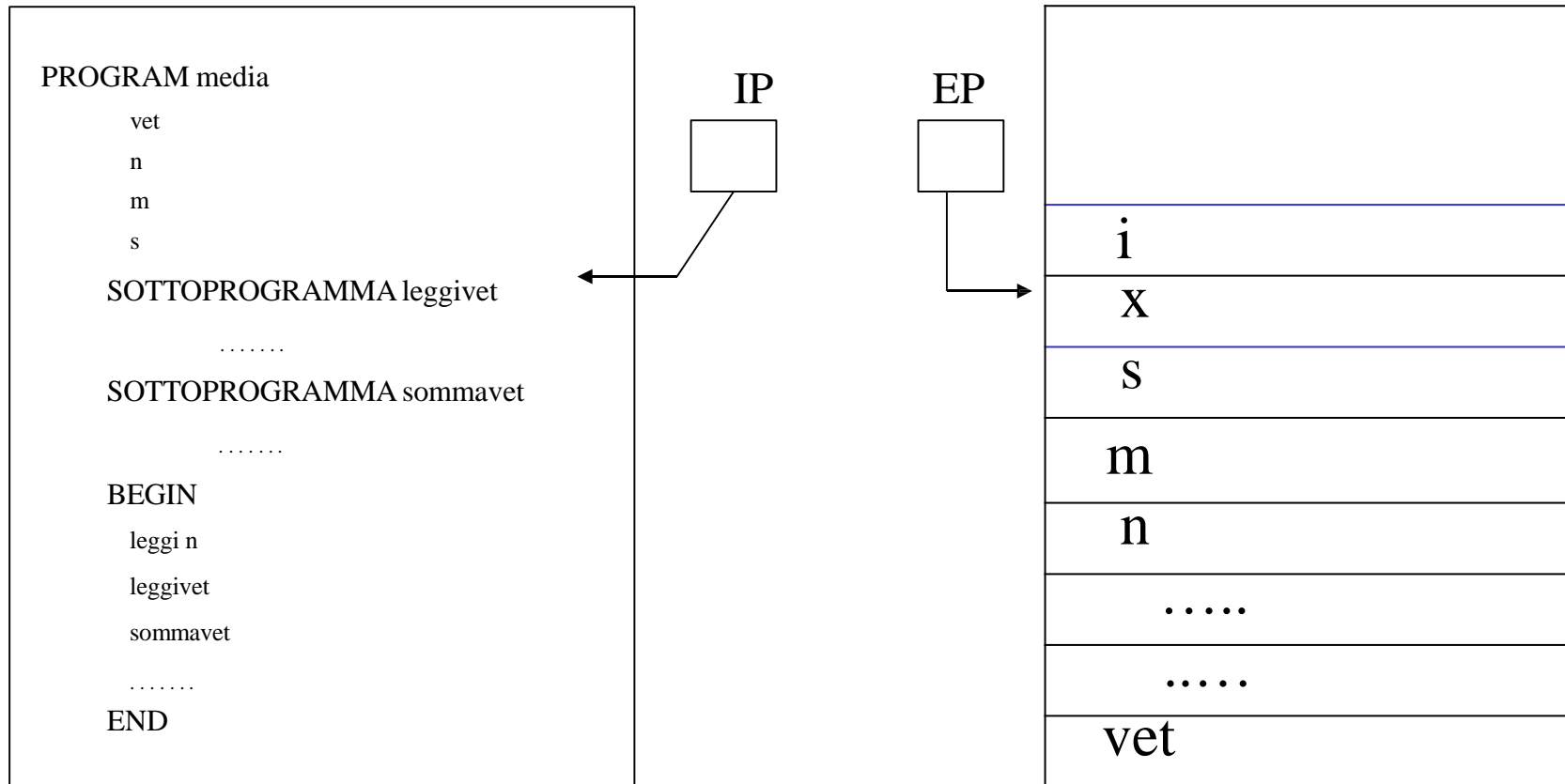
# Sottoprogramma il countour model

All'inizio dell'esecuzione vengono allocate le variabili del programma principale



# Sottoprogramma il countour model

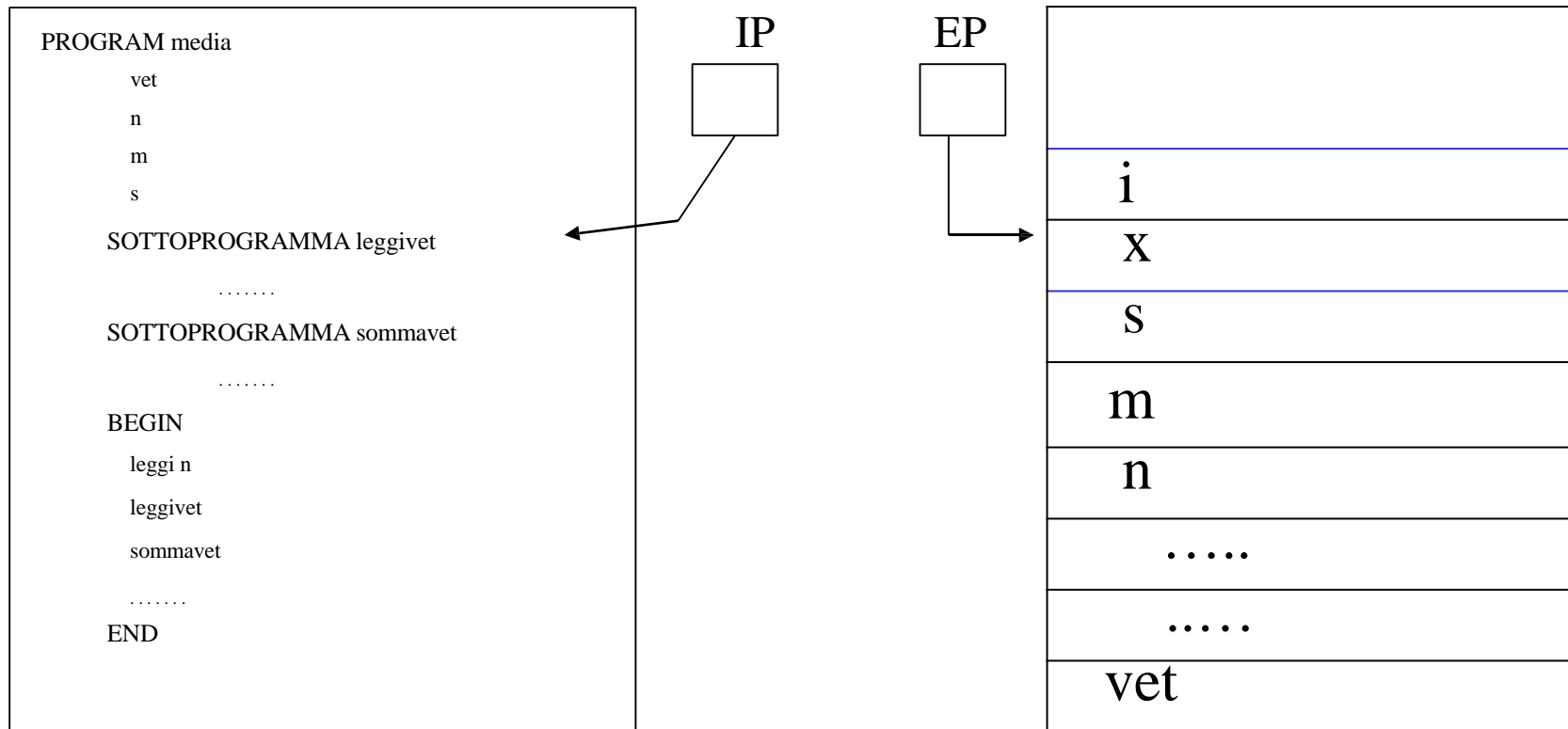
Quando viene richiamato il sottoprogramma leggivet vengono allocate nuove variabili in un nuovo ambiente a cui punterà EP





# Sottoprogramma il countour model

Leggivet userà le variabili i e x puntate da EP attuale mentre, non trovando n e vet nell'ambiente puntato da EP, risalirà negli ambienti precedenti sino a trovare la prima loro occorrenza. Questo è il meccanismo che consente ad un sottoprogramma di ereditare il diritto di accesso al programma che lo richiama



## Sottoprogramma il countour model

Terminata l'esecuzione di leggivet, l'ambiente puntato da EP non serve più quindi viene rilasciata l'area di memoria occupata e EP torna a puntare all'ambiente che aveva richiamato leggivet.

# Sottoprogramma esempio

```
sottoprogramma quadrato
  i: integer
  quad: integer
begin
  do varying i from 1 to 10
    quad ← i*i
    stampa quad
  repeat
end
```

} dichiarazione variabili

Per stampare i primi 15 quadrati?  
Per stampare i primi 35 quadrati?

# Sottoprogramma esempio

```
sottoprogramma quadrato (n: integer)
  i: integer
  quad: integer
begin
  do varying i from 1 to n
    quad      i*i
    stampa quad
  repeat
end
```

} dichiarazione variabili

```
quadrato(10)
quadrato(m)
```

} chiamate

# Sottoprogramma

Nella dichiarazione del sottoprogramma:

l'intestazione introduce il nome e gli argomenti del sottoprogramma

Il corpo descrive le regole di comportamento del sottoprogramma

Nell'intestazione del sottoprogramma appaiono i parametri formali che all'atto della chiamata vengono sostituiti dai parametri effettivi (attuali) ovvero dai dati su cui il sottoprogramma deve operare

# Parametri Formali

- Segnaposto per indicare simbolicamente
  - Gli oggetti su cui il sottoprogramma lavora
  - Il loro tipo e struttura
- I loro nomi appaiono nell'intestazione del sottoprogramma
  - Non hanno alcuna connessione con nomi usati altrove
- All'atto della chiamata vengono sostituiti dai parametri *effettivi* (o *reali*)
  - Dati su cui effettivamente il sottoprogramma deve operare

# Parametri Formali

- Specificati all'atto della definizione del sottoprogramma
    - Legati al sottoprogramma
    - Simbolici
    - Consentono di definire
      - Quale tipo di dato deve essere passato alla procedura
      - Quale argomento deve essere trasmesso alla funzione
- quando queste sono invocate

# Parametri Effettivi

- Alla chiamata di un sottoprogramma, vanno specificati i dati effettivi su cui esso dovrà operare
  - Valori, Espressioni, Variabili, ...
  - Coincidenza con i parametri formali
    - Numero, tipo e ordine
- L'esecuzione dell'istruzione di chiamata comporta la sostituzione dei parametri formali con quelli reali



# Parametri

## Tipi di Passaggio

- La sostituzione può essere:
  - Per valore
    - Si calcola il valore del parametro reale e lo si sostituisce al corrispondente parametro formale (assegnazione)
  - Per referenza
    - Il parametro effettivo è una variabile ed ha a disposizione una locazione di memoria il cui indirizzo viene “passato” al parametro formale
  - Per nome (o *valore-risultato*)
    - Il nome del parametro formale, all’occorrenza, viene sostituito col nome del parametro reale

# Parametri

## Tipi di Passaggio

- $P(pf)$  attivata con parametro reale  $pe$ 
  - Per valore:
    - Il parametro formale si comporta come una variabile locale a  $P$
  - Per riferimento:
    - Al momento dell'attivazione di  $P$  viene calcolato l'indirizzo di  $pe$  e  $pf$  viene creato con riferimento alla stessa locazione di memoria
  - Per nome:
    - Al momento dell'attivazione di  $P$  il valore di  $pe$  viene calcolato e memorizzato in una nuova locazione di indirizzo  $pf$
    - Al termine dell'esecuzione di  $P$  il contenuto di  $pf$  viene trasferito in  $pe$  e la memoria riservata per  $pf$  viene rilasciata

# Passaggio di Parametri

## Esempio

```
program legaparametri (...,...);  
  var n: integer;  
  sottoprogramma P (? x : integer)  
    begin  
    x := x + 1;  
    writeln(n);  {1}  
    writeln(x)   {2}  
  end;  
  begin  
  n := 3;  
  P(n);  
  writeln(n)    {3}  
end.
```

- Per valore
  1. 3
  2. 4
  3. 3
- Per referenza
  1. 4
  2. 4
  3. 4
- Per nome
  1. 3
  2. 4
  3. 4

# Passaggio di Parametri

## Pro (+) e Contro (-)

- Copia di valori
  - + permette la trasmissione del valore di un parametro dal chiamante
  - + permette la separazione tra programma chiamante e programma chiamato
  - aumenta l'occupazione di memoria ed il tempo
  - rende difficile la gestione di parametri di dimensione variabile
- Trasmissione per riferimento
  - + evita problemi di passaggio perché il trattamento degli indirizzi è gestito direttamente dal compilatore
  - + non occupa memoria aggiuntiva
  - causa effetti collaterali spesso imprevedibili

# Passaggio di Parametri per Valore

- Generalmente usato per parametri che
  - Rappresentano un argomento e non il risultato di un sottoprogramma
    - Inutile consentirne la modifica

# Passaggio di Parametri per Referenza

- Usato più spesso quando il parametro
  - Rappresenta un risultato
    - Necessità di conoscere la modifica
  - Ha dimensioni notevoli
    - Pesante ricopiarlo interamente
- Potenziale fonte di errori
  - Stessa variabile usata sotto diverse denominazioni
  - Errori nel sottoprogramma irrecuperabili

# Procedure

- Sottoprogrammi il cui compito è quello di produrre un effetto
  - Modifica del valore di variabili
  - Comunicazione di informazioni all'utente
- L'intestazione di procedura (e la sua chiamata) includono
  - Nome della procedura
    - Paragonabile ad una nuova istruzione del linguaggio
  - Lista di parametri