PASCAL: LE PROCEDURE 1

TRATTO DA CAMAGNI-NIKOLASSY, CORSO DI INFORMATICA, VOL. 2, HOEPLI

Informatica

Le procedure senza parametri

Problemi e sottoproblemi: top-down

- Quando un problema da risolvere è molto complesso conviene utilizzare il metodo top-down (cioè per "affinamenti successivi" dall'alto verso il basso)
- Ogni problema viene scomposto in problemi più semplici di più facile risoluzione
- Il codice risolutivo di problemi semplici può essere visto come un programma "autonomo" da utilizzare all'interno di altri programmi (sottoprogramma)

TOP-DOWN

L'espressione inglese *top-down* letteralmente significa "dall'alto al basso", e sta a indicare una modalità operativa che affronta un problema nella sua interezza scomponendolo in sottoproblemi sempre più elementari, che consentono di giungere agevolmente alla sua soluzione.

I sottoprogrammi

- I sottoprogrammi sono da intendersi come componenti software da utilizzarsi come "mattoni" per la costruzione di programmi più complessi
- Utilizzando i sottoprogrammi:
  - È inutile scrivere due volte lo stesso codice
  - È inutile scrivere un codice che qualcun altro ha già scritto
  - È inutile scrivere un codice che qualcun altro ha già scritto e che funziona

Esempio: sqrt()

- Un esempio di sottoprogramma è la funzione sqrt()
- È stata utilizzata più volte ma il suo codice non è mai stato scritto nei programmi che la utilizzavano!
- Questo perché "qualcuno" l'aveva già scritta ed era stata salvata "da qualche parte"

Meglio usare del codice già scritto e garantito che riscriverlo *ex novo*

I sottoprogrammi: vantaggi

- I vantaggi dell'uso dei sottoprogrammi sono:
  - Riusabilità, cioè poter utilizzare lo stesso codice per la soluzione di problemi diversi
  - Astrazione, cioè la possibilità di esprimere in modo sintetico operazioni complesse
  - Risparmio, cioè scrivere una sola volta un codice e usarlo più volte

I sottoprogrammi

- I sottoprogrammi possono essere di due tipi:
  - Procedure
    - Procedure senza parametri
    - Procedure con parametri
      - Passati per valore
      - Passati per riferimento
  - Funzioni

Procedure senza parametri

- Le procedure sono segmenti di codice che vengono isolati e scritti a parte nella sezione dichiarativa del programma
  - Hanno un nome
  - Vengono successivamente richiamate all'interno del programma proprio tramite il nome
- Il programma esterno che le contiene e le manda in esecuzione è detto programma principale o main program

MAIN PROGRAM

Con il termine main program si indica il programma principale di un'applicazione informatica, che viene mandato in esecuzione per "avviare" l'elaborazione. Generalmente è costituito da più sottoparti, che possono essere procedure o funzioni.

Definizione di una procedura

- Una procedura si definisce nella **sezione dichiarativa** del programma principale in questo modo:

```
procedure nome_procedura;
    begin
      .
      .
      .
    end;
```

La struttura della procedura è identica a quella di un programma; l'unica differenza consiste nel fatto che termina con un `end` seguito dal punto e virgola al posto del punto.

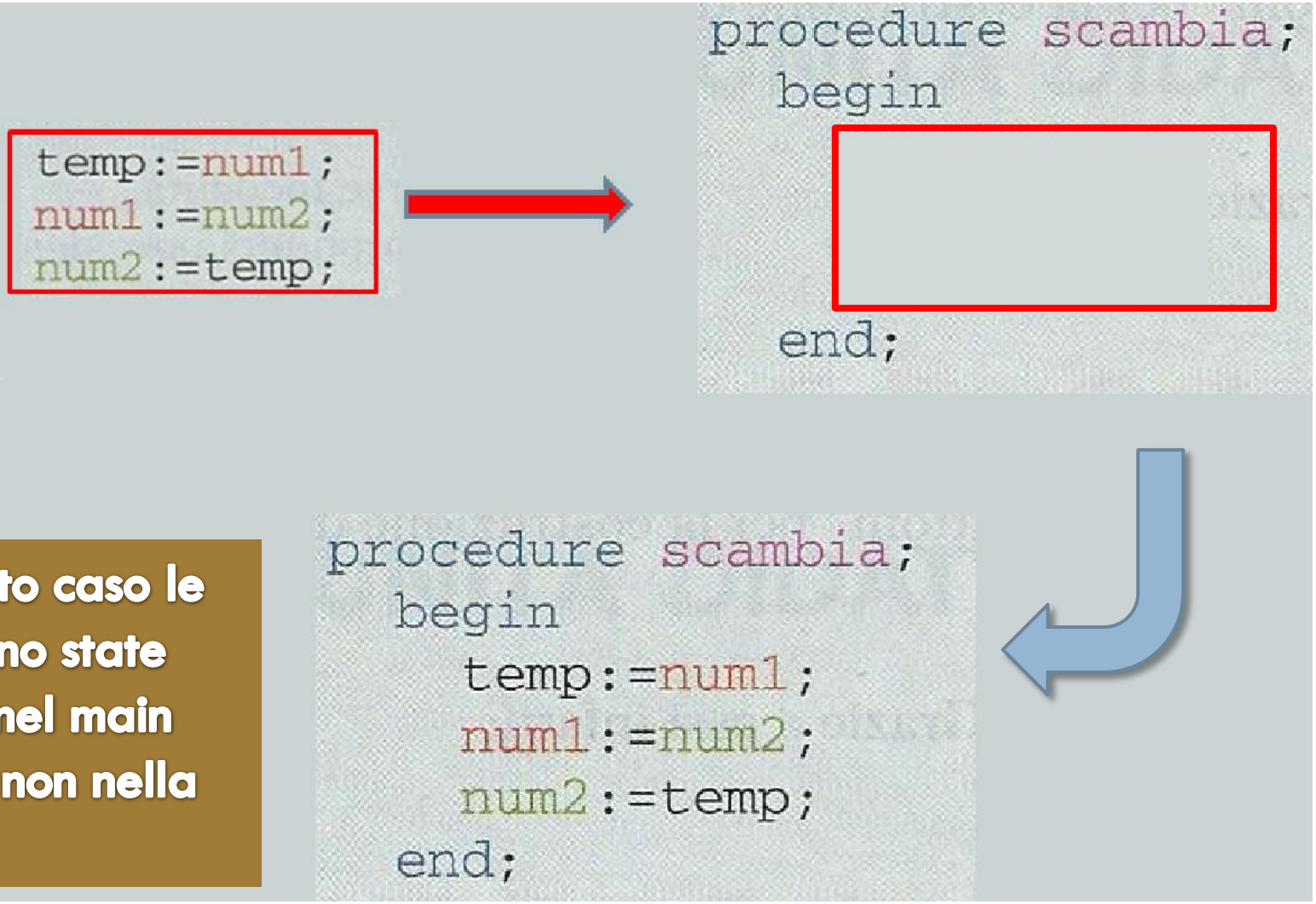
Esempio: swap

- Il seguente programma effettua lo swap di due variabili
- Vogliamo rendere lo swap una procedura

```
program scambiaconprocedura;
var
(* dichiarazione delle variabili *)
  num1,num2,temp:integer;
(* inizio programma principale *)
begin
  write('inserisci il valore di num1 ');
  readln(num1);
  write('inserisci il valore di num2 ');
  readln(num2);
  temp:=num1;
  num1:=num2;
  num2:=temp;
  writeln('la variabile num1 contiene ora ', num1);
  writeln('la variabile num2 contiene ora ', num2);
  readln;
end.
```

Swap di variabili

procedure scambia;



NB: In questo caso le variabili sono state dichiarate nel main program e non nella procedura!

Codice completo

```
0  program scambiaconprocedura;
1  var
2  (* dichiarazione delle variabili *)
3    num1,num2,temp:integer;
4  (* dichiarazione delle procedure *)
5    procedure scambia;
6    begin
7      temp:=num1;
8      num1:=num2;
9      num2:=temp;
10   end;
11 (* inizio programma principale *)
12 begin
13   write('inserisci il valore di num1 ');
14   readln(num1);
15   write('inserisci il valore di num2 ');
16   readln(num2);
17   scambia;                  // chiamata della procedura
18   writeln('la variabile num1 contiene ora ', num1);
19   writeln('la variabile num2 contiene ora ', num2);
20   readln;
21 end.
```

Uso di tre procedure

- Possiamo ulteriormente scomporre il programma scrivendo altri due sottoprogrammi (procedure):
  - Uno per effettuare le operazioni di **input**
  - L'altro per eseguire **l'output**
- In questo modo il programma principale (main program) si riduce a tre istruzioni di chiamata delle procedure

```
. . .
begin
   input;   // chiamata della procedura che esegue l'input
   scambia;// chiamata della procedura che effettua i calcoli
   output; // chiamata della procedura che esegue l'output
end.
```

Programma completo

```
0    program scambiaconprocedure;
1  var
2  (* dichiarazione delle variabili *)
3    num1,num2,temp:integer;
4  (* dichiarazione delle procedure *)
5    procedure scambia;
6    begin
7      temp:=num1;
8      num1:=num2;
9      num2:=temp;
10   end;
11   procedure input;
12   begin
13     write('inserisci il valore di num1 ');
14     readln(num1);
15     write('inserisci il valore di num2 ');
16     readln(num2);
17   end;
```

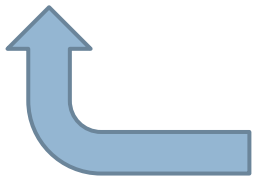
Procedure SWAP

Procedure INPUT

Programma completo (continuo)

Procedure OUTPUT

```
18    procedure output;
19    begin
20      writeln('la variabile num1 contiene ora ', num1);
21      writeln('la variabile num2 contiene ora ', num2);
22      readln;
23    end;
24  (* inizio programma principale *)
25  begin
26    input;      // chiamata della procedura che esegue l'input
27    scambia;    // chiamata della procedura che effettua i calcoli
28    output;     // chiamata della procedura che esegue l'output
29  end.
```



MAIN PROGRAM

Esercizio

Prova adesso!

- Definizione di procedure
- Utilizzo di procedure

APRI IL FILE esempio01

1. Modifica il programma eseguendo un ulteriore scambio tra i numeri in modo che "tutto ritorni come all'inizio".
2. Confronta il codice con quello presente nel file esercizio01. Quante istruzioni hai aggiunto?
3. Quante istruzioni avresti dovuto aggiungere se non avessi utilizzato le procedure?

Esercizi: Sottoprobemi 1-6, pag. 7

Sottoproblemi

Scomponi i seguenti problemi in sottoproblemi risolvibili mediante procedure.

1. Leggi tre numeri da tastiera, individua il minore, il maggiore e visualizzali sullo schermo.
2. Leggi il valore dell'area di un quadrato e calcola il perimetro della circonferenza a esso circoscritta.
3. Leggi il valore del lato di un quadrato e calcola il perimetro e l'area della circonferenza a esso inscritta e circoscritta.
4. Leggi tre valori corrispondenti ai lati di un triangolo; in base al loro valore richiama, per ogni tipologia di triangolo, una procedura che esegue il calcolo del perimetro e dell'area.
5. Leggi i due cateti di un triangolo rettangolo e calcola il valore dell'area e del perimetro di un quadrato costruito sulla sua ipotenusa.
6. Leggi 40 numeri e memorizzali in un array. Quindi separa in due vettori i numeri pari da quelli dispari e visualizzali entrambi sullo schermo ordinati in senso crescente.

Esercizi: Probemi 1-6, pag. 7

Problemi

Codifica i seguenti problemi utilizzando le procedure.

1. Scrivi un programma che esegue il calcolo dell'area di un quadrato mediante una procedura dopo aver letto il valore del lato per mezzo di una seconda procedura.
2. Scrivi un programma che esegue il calcolo dell'area di un cerchio mediante una procedura dopo aver letto il valore del lato impiegando una seconda procedura.
3. Scrivi un programma che leggendo il valore del perimetro di un quadrato ne calcola l'area. Definisci tre funzioni: una per effettuare l'input, una per ricavare il lato e la terza per effettuare il calcolo dell'area.
4. Scrivi un programma che leggendo il valore dell'area di un quadrato calcola la lunghezza di una circonferenza a esso circoscritta. Definisci tre funzioni: una per effettuare l'input, una per ricavare il lato e la terza per eseguire il calcolo della circonferenza.
5. Scrivi un programma che leggendo il valore dell'area di un triangolo rettangolo calcola il lato di un quadrato a esso equivalente. Definisci tre funzioni: una per effettuare l'input, una per calcolare l'area del triangolo e la terza per calcolare l'area del quadrato.
6. Scrivi un programma per effettuare le conversioni di base dei sistemi di numerazione. Dopo aver letto una sequenza di numeri terminante con 0 (inserita dall'utente), il programma deve offrire all'utente la possibilità di convertire tale numero nei seguenti formati:
   - binario;
   - ottale;
   - esadecimale;
   - vigesimale.

Il modello ad ambienti: *Ambiente locale e globale*

Ambiente locale e globale

- Nel programma principale abbiamo la sezione var e successivamente la sezione delle procedure, all'interno delle quali possiamo avere un'ulteriore sezione var

```
program main;              // main: I livello
var
  qui:integer;
  procedure ali;           // procedura  al II livello
  var
    quo:integer;
  begin                    // inizio procedura ali
    . . .
  end;                     // fine procedura ali
  procedure baba;          // procedura  al II livello
  var
    qua:integer;
  begin                    // inizio procedura baba
    . . .
  end;                     // fine procedura baba
begin                      // inizio programma principale
  . . .
end.                       // fine programma principale
```

Livelli di annidamento: ambiente locale

- In analogia a quanto accade con le istruzioni annidate anche per le procedure si parla di livelli di annidamento:
  - le procedure interne sono a un *livello superiore* rispetto alla procedura che le definisce

Il programma principale è al primo livello e dichiara:
- la variabile `qui`;
- la procedura `ali`;
- la procedura `baba`.

Al secondo livello si trovano le due procedure dichiarate del `main`, cioè:
- la procedura `ali`, che dichiara la variabile `quo`;
- la procedura `baba`, che dichiara la variabile `qua`.

Ambiente e variabili globali e locali

- Le variabili dichiarate nella procedura sono dette variabili locali
- Le variabili dichiarate nel main program sono dette variabili globali

AMBIENTE LOCALE

Per ciascuna procedura viene definito con l'espressione ambiente locale l'insieme delle variabili che essa definisce: queste variabili prendono il nome di variabili locali.

AMBIENTE GLOBALE

L'unione delle variabili locali e delle variabili globali di una procedura prende il nome di ambiente globale: costituisce l'insieme di tutte le variabili che la procedura può utilizzare nelle sue elaborazioni.

Esempio

Per esempio, la procedura `ali` ha:

`qui`: variabile globale
`quo`: variabile locale

} ambiente globale

La procedura `baba` ha:

`qui`: variabile globale
`qua`: variabile locale

} ambiente globale

Regole di visibilità

Le due regole che indicano come possono essere utilizzate le variabili si chiamano ◀ **regole di visibilità** ▶ (o scoping rules), e sono le seguenti:

- ogni procedura vede le variabili che dichiara (ambiente locale);
- ogni procedura può vedere le variabili del programma esterno che la contiene, cioè le variabili locali del programma che la dichiara.

L'insieme di queste variabili è l'ambiente globale.

- Quindi il programma esterno non vede le variabili dei programmi più interni
- Il *main program* ha come ambiente globale solo le variabili solo le variabili che dichiara lui, cioè ambiente globale e locale coincidono.

Esempio

```
0  program main;                    // main: I livello
1  var
2    qui:integer;
3    procedure ali;                 // procedura al II livello
4    var
5      quo:integer;
6    begin                          // inizio procedura ali
7      write('introduci il valore di quo ');
8      readln(quo);
9      qui:=qui+quo;
10     writeln('ora il valore di qui vale ',qui);
11   end;                           // fine procedura ali
12   procedure baba;                // procedura  al II livello
13   var
14     qua:integer;
15   begin                          // inizio procedura baba
16     write('introduci il valore di qua ');
17     readln(qua);
18     qui:=qui+qua;
19     writeln('ora il valore di qui vale ',qui);
20   end;                           // fine procedura baba
21 begin                           // inizio programma principale
22   write('introduci il valore di qui ');
23   readln(qui);
24   ali;                           // chiamata procedura ali
25   baba;                          // chiamata procedura baba
26   writeln('alla fine qui vale',qui);
27 end.                            // fine programma principale
```

Esempio

1 Nel `main` viene letto il valore della variabile `qui` con le istruzioni **22–23**, per esempio inseriamo 1. Quindi viene chiamata la procedura `ali` con l'istruzione **24**.

2 Il controllo passa alla procedura `ali` a partire dall'istruzione **7**.
In `ali` viene letto il valore della variabile locale `quo` con le istruzioni **7–8**, per esempio inseriamo 2.
In `ali` viene modificata e visualizzata la variabile globale `qui` che assume il valore 3.
In `ali` l'istruzione **11** termina la procedura e il controllo ritorna al `main`.

3 Quindi viene chiamata la procedura `baba` con l'istruzione **25**.

4 Il controllo passa alla procedura `baba` a partire dall'istruzione **16**.
In `baba` viene letto il valore della variabile locale `qua` con le istruzioni **16–17**, per esempio inseriamo 3.
In `baba` viene modificata e visualizzata la variabile globale `qui` che assume il valore 6.
In `baba` l'istruzione **20** termina la procedura e il controllo ritorna al `main`.

5 Quindi viene eseguita l'istruzione **26**, che visualizza `qui`, cioè ancora il valore 6.

Possiamo osservare che le due procedure hanno modificato in modo permanente il valore della variabile `qui`.

Esercizio

Prova adesso!

- Ambiente locale
- Ambiente globale
- Visibilità delle variabili

APRI IL FILE esempio04

1 Aggiungi nel `main` e in ogni procedura le seguenti istruzioni:

```
qui:=qui+quo+qua;
writeln('le variabili valgono ',qui,quo,qua);
```

Che cosa ti aspetti che venga visualizzato sullo schermo? Perché?

Omonimie

- È possibile che nel main program e nella procedura vengano dichiarate due variabili con lo stesso nome. Vediamo cosa accade.

```
0  program main2;                    // main: I livello
1  var
2    qui:integer;
3    quo:integer;
4    procedure ali;                  // procedura al II livello
5    var
6      qui:integer;
7      qua:integer;
8    begin                           // inizio procedura ali
9      write('introduci il valore di qui e qua ');
10     readln(qui,qua);
11     quo:=quo+qua;
12     writeln('dentro ali qui e quo valgono ',qui,' ',quo);
13   end;                            // fine procedura ali
14 begin                             // inizio programma principale
15   write('introduci il valore di qui e quo ');
16   readln(qui,quo);
17   writeln(prima di ali le variabili valgono ',qui,' ',quo);
18   ali;                            // chiamata procedura ali
19   writeln('dopo ali le variabili valgono ',qui,' ',quo);
20   readln;
21 end.                              // fine programma principale
```

Omonimie

Possiamo osservare che sono presenti due celle con lo stesso nome, `qui`, che però sono contenute in ambienti diversi:

- `qui`: appartiene all'ambiente locale del `main`;
- `qui`: appartiene all'ambiente locale della procedura `ali`.

Omonimie

Mandiamo in esecuzione il programma e inseriamo 1 e 2 alla prima richiesta (istruzione **15**): i dati vengono scritti nelle variabili locali del `main` `qui` e `quo`.

La chiamata di `ali` con l'istruzione **18** passa il controllo alla procedura che legge due nuovi valori memorizzandoli nelle variabili locali `qui` e `qua`.

La procedura utilizza la cella `qui` definita dalla stessa, quindi prevale la dichiarazione più interna, senza modificare la cella `qui` esterna, dichiarata dal `main`.

Esercizi 1-2, pag. 13

Indicare l'output dei seguenti programmi:

1
```
program visibilita1;
var a,b,c:integer;
 procedure p2;
 var c,d:integer;
 begin
  d:=2;
  c:=d;
  b:=c+d;
  writeln(a,b,c);
 end;
begin
 a:=1;b:=1;c:=1;
 p2;
 writeln(a,b,c);
end.
```

2
```
program visibilita2;
var a,b,c:integer;
 procedure p2;
 var c,d:integer;
 begin
  d:=2;c:=3;
  b:=c+d;
  writeln(a,b,c);
 end;
begin
 a:=1;b:=1;c:=1;
 p2;
 writeln(a,b,c);
end.
```

Esercizi 3-4, pag. 13

3
```
program visibilita3;
var a,b,c:integer;
 procedure p2;
 var a,d:integer;
 begin
  a:=1;d:=2;
  b:=c+d;
  c:=b-1;
  writeln(a,b,c);
 end;
begin
 a:=1;b:=1;c:=1;
 p2;
 writeln(a,b,c);
end.
```

4
```
program visibilita4;
 var a,b,c:integer;
  procedure p2;
  var a,d:integer;
  begin
   a:=2;d:=2;
   c:=b+a;
   b:=c+d;
   writeln(a,b,c);
  end;
  procedure p3;
  var x:integer;
  begin
   x:=3+a;
   a:=b+x-c;
   writeln(a,b,c);
  end;
begin
 a:=1;b:=1;c:=1;
 p2;
 p3;
 writeln(a,b,c);
end.
```

Problemi1-2, pag. 13

1 Scrivi un programma che legge quattro numeri e ne trova il maggiore confrontandoli a coppie e scambiandoli tra loro mediante la procedura scambia.

2 Scrivi un programma che legge quattro numeri e li ordina in senso crescente confrontandoli a coppie e scambiandoli tra loro mediante la procedura scambia.