Linguaggi di Programmazione I – Lezione 2

Prof. Marcello Sette
mailto://marcello.sette@gmail.com
http://sette.dnsalias.org

6 marzo 2007

Panoramica della lezione

Procedure come astrazioni

Record di attivazione

Ereditarietà

Bibliografia

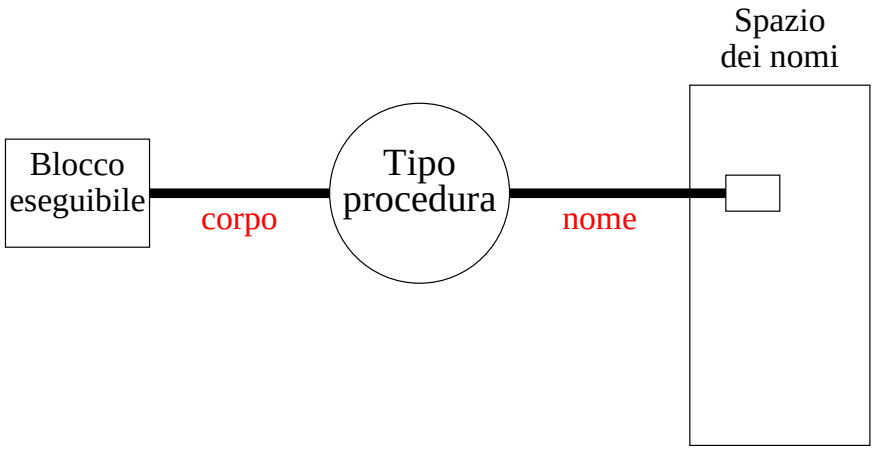
Procedure come astrazioni 3 / 18

Definizioni

1. **Procedure** sono astrazioni di parti di programma in unità di esecuzione più piccole, come enunciati o espressioni, nascondendo i dettagli irrilevanti ai fini del loro (ri-)uso.
   Vantaggi:
   - Programmi più semplici da scrivere, leggere o modificare; suddivisione dei compiti in ogni brano di programma; progettazione top-down.
   - Unità di programmi indipendenti o con dipendenze ben specificate a livello più alto.
   - Riusabilità di brani di programmi; riduzione errori.
2. Se si distinguono come unità di esecuzione, in ordine crescente di complessità, *espressioni*, *enunciati*, *blocchi*, *programmi*, allora si definisce **astrazione procedurale** la rappresentazione di una unità di esecuzione attraverso un'altra unità più semplice. In pratica è la rappresentazione di un blocco attraverso un enunciato o una espressione.

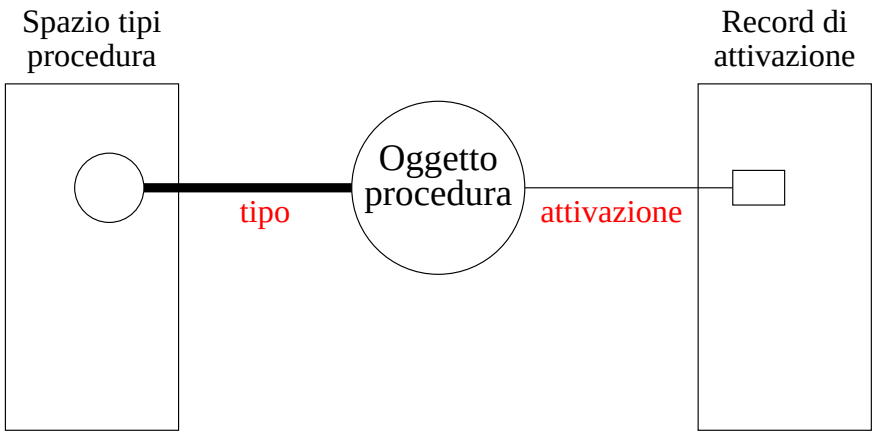
Dichiarazione di procedura

- Causa la generazione di un oggetto analogo al Type Object (anche se in questo corso non esaminato).
- Il processo avviene durante la compilazione.



Invocazione di una procedura

- Causa la generazione di un oggetto analogo al Data Object.
- Il processo avviene durante l'esecuzione, nel momento in cui c'è l'invocazione della procedura.
- Ogni invocazione diversa della stessa procedura causa la generazione di un nuovo "oggetto procedura" con lo stesso legame di tipo, ma con diverso record di attivazione.



Ambiente di esecuzione

Analogamente a quanto avveniva per un blocco (di cui la procedura è astrazione), il **record di attivazione** rappresenta l'intero ambiente di esecuzione di una procedura. Esso consiste di solito in:

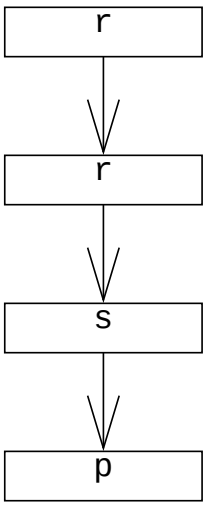
1. ambiente locale (tutti gli oggetti che sono definiti all'interno della procedura);
2. ambiente non locale (tutti gli oggetti la cui definizione è ereditata da altre procedure);
3. ambiente dei parametri (contiene informazioni sui dati che sono passati [d]alla procedura).

Ogni volta che una procedura viene invocata il suo record di attivazione viene aggiunto al cosiddetto **stack di esecuzione**.
Sul top dello stack c'è sempre il record relativo alla procedura correntemente in esecuzione.
Alla terminazione della procedura, il record di attivazione viene rimosso dallo stack.

Esempio

Stack esecuzione

r: prima di chiamare q




```
program p;
var i;

  procedure q;
  begin
    ...
  end;

  procedure r;
  begin
    i:= i-1;
    if i>0 then
      r
    else
      q
  end;

  procedure s;
  begin
    r;
    q
  end;

begin

  i:= 2;

  s

end.
```

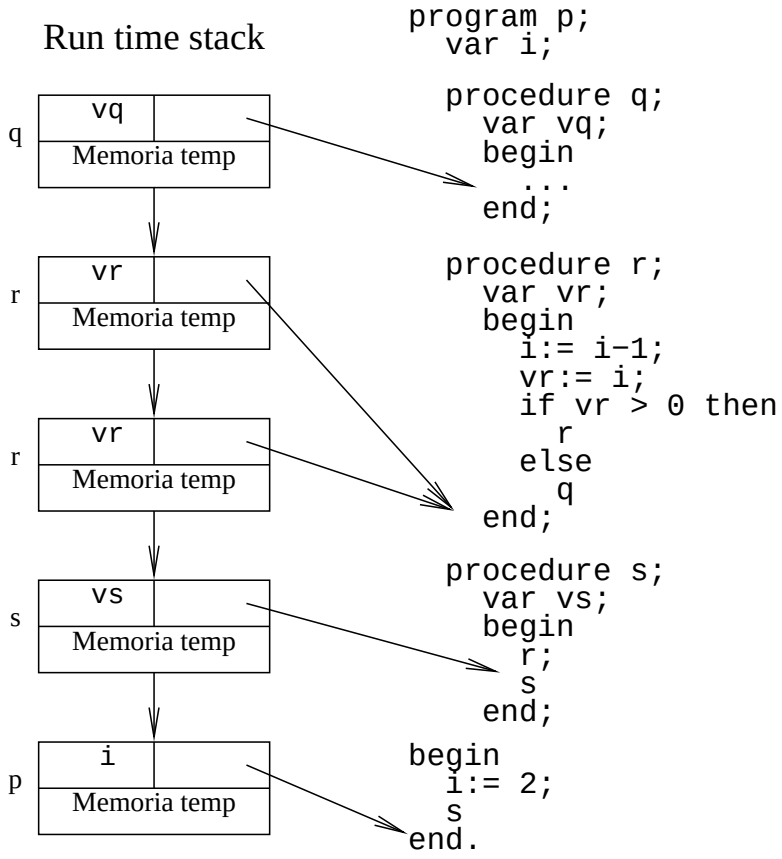

Record di attivazione

Ambiente locale

Include:

1. Tutte le variabili dichiarate localmente.
2. Puntatore alla prossima istruzione [IP] (permette di riprendere l'esecuzione quando il controllo viene restituito alla procedura chiamante).
3. Memoria temporanea necessaria alla valutazione delle espressioni contenute nella procedura (altamente dipendente dalla realizzazione).

Esempio



Ereditarietà 12 / 18

Realizzazione

- Viene realizzata aggiungendo al record di attivazione un puntatore al record di attivazione della procedura da cui devono essere ereditate le definizioni o i dati.
- Se viene richiesto l'accesso ad un dato che non è definito localmente, esso viene ricercato in modo ricorsivo nei record di attivazione precedenti.
- Tre tipologie di realizzazione dell'ereditarietà.

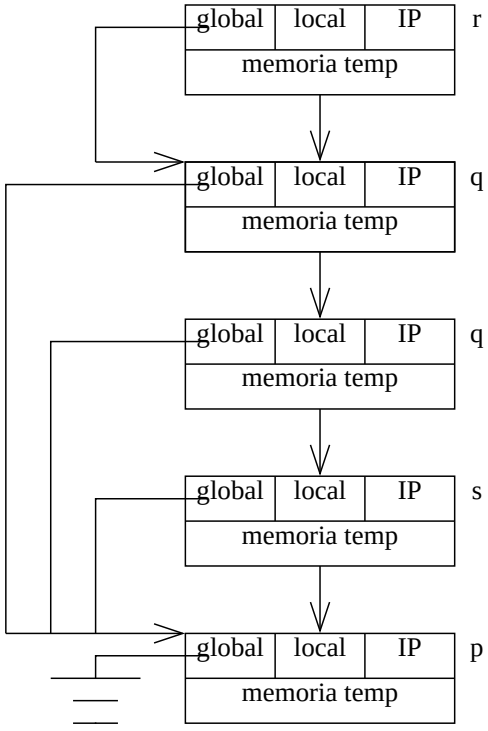
1. Ereditarietà in ambito statico. In questo caso l'ambiente non locale di una procedura è ereditato dal programma che la contiene sintatticamente: ereditarietà di posizione.
2. Ereditarietà in ambito dinamico. In questo caso l'ambiente non locale di una procedura è ereditato dal programma chiamante.
3. Nessuna ereditarietà. L'uso di ambienti non locali è scoraggiato perché produce **effetti collaterali** non facilmente prevedibili.

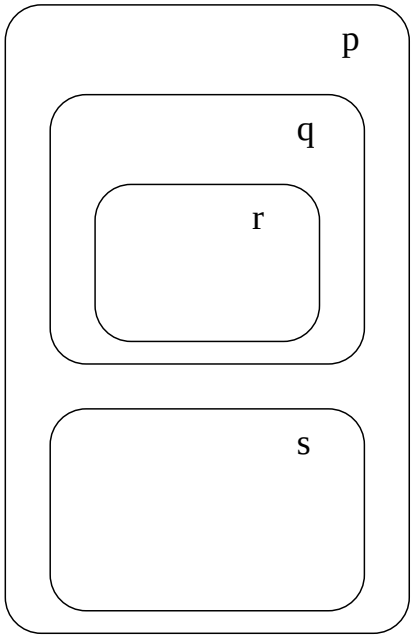
Ambito statico

```
program p;
  var a, b, c: integer;

  procedure q;
    var a, c; integer;
    procedure r;
      var a: integer;
      begin {r}          {variabili: a da r; b da p; c da q;
        ...               procedure: q da p; r da q}
      end;  {r}
    begin {q}           {variabili: a da q; b da p; c da q;
       ...               procedure: q da p; r da q}
    end;  {q}

  procedure s;
    var b: integer;
    begin {s}           {variabili: a da p; b da s; c da p;
      ...               procedure: q da p; s da p}
    end;  {s}

begin {p}               {variabili: a, b, c da p;
  ...                    procedure: q, s da p}
end.  {p}
```


Ambito statico (2)

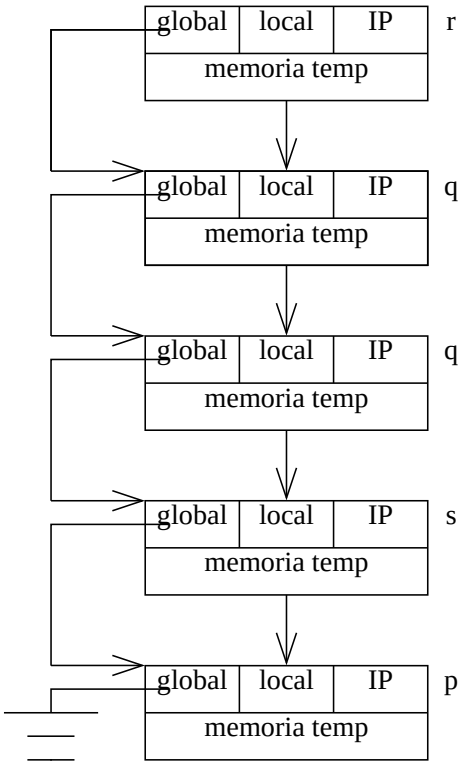
Supponendo una sequenza di attivazione (p, s, q, q, r), lo stack di esecuzione ha questa forma:

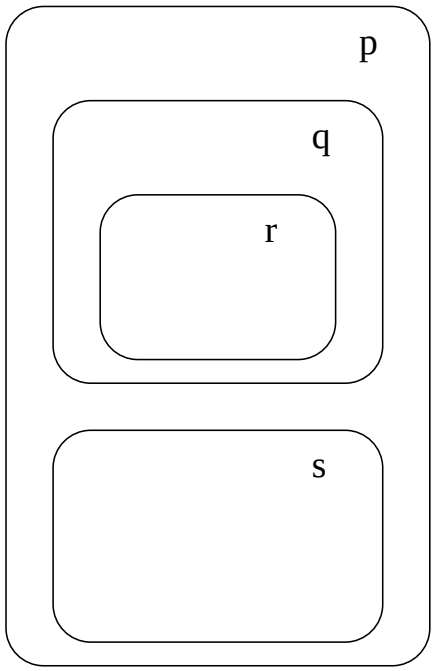




Ambito dinamico

La stessa sequenza di attivazione precedente (`p`, `s`, `q`, `q`, `r`), genera allora lo stack di esecuzione:





Osservazioni

Nell'ereditarietà in ambito dinamico:

- il puntatore all'ambiente non locale non è più necessario;
- è praticamente impossibile la determinazione dell'ambiente di esecuzione di una procedura durante la scrittura del codice sorgente.

```
program p;
  var a: integer;
  procedure q;
    begin           {vars: a da p o da r; procs: q da p}
      ...
    end;
  procedure r;
    var
      a: integer;
    begin           {vars: a da r; procs: q, r da p}
      ...
    end;
begin               {vars: a da p; procs: q, r, da p}
  ...
end.
```

Bibliografia

- H. L. Dershem. M. J. Jipping. *Programming languages: structures and models*. Second edition. Cap. 5, par. 5.1 - 5.3.