Linguaggi di Programmazione I – Lezione 3

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

11 marzo 2008

Panoramica della lezione

Parametrizzazione di procedure

Bibliografia

Parametrizzazione di procedure 3 / 15

Parametri

- Sono la terza parte necessaria a specificare l'ambiente di esecuzione di una procedura.
- Costituiscono il mezzo attraverso il quale le informazioni transitano esplicitamente tra l'unità chiamante e quella chiamata.
- Si possono distinguere:
  - paramatri IN: sono passati dalla unità chiamante alla unità chiamata al momento dell'invocazione;
  - parametri OUT: sono passati dall'unità chiamata alla unità chiamante al momento della terminazione della prima;
  - parametri IN-OUT: servono a far transitare le informazioni in entrambe le direzioni.
- Devono essere specificati in due punti:
  - nella procedura chiamante: *parametri attuali*;
  - nella procedura chiamata: *parametri formali*.

Associazione dei parametri

- **Regola**: deve essere specificato il tipo dei parametri formali. È richiesta la corrispondenza di tipo tra parametri formali e attuali.
- **Eccezioni**:
  - lasciare i parametri formali senza legame di tipo; il legame si instaura durante l'invocazione (run time) allo stesso tipo dei parametri attuali (impossibile il type checking durante la compilazione).
  - permettere solo l'eccezione degli array a dimensione non specificata come parametri formali; il legame di dimensione verrà realizzato durante l'invocazione.
- **Metodi di associazione**:
  - *per posizione*: a seconda della posizione relativa nella sequenza dei parametri;
  - *per nome*: il nome del parametro formale è aggiunto come prefisso al parametro attuale;

Esempio

Data l'intestazione della seguente procedura (ADA):

```
procedure TEST (A: in Atype; b: in out Btype; C: out Ctype)
```

allora una invocazione che usi associazione per posizione è:

```
TEST(X, Y, Z);
```

mentre una che usi associazione per nome può essere:

```
TEST(A=>X, C=>Z, b=>Y);
```

Una ulteriore tecnica è la cosiddetta *associazione di default*. Essa permette di specificare valori di default ai parametri formali che non sono stati legati a valori da parametri attuali.

Parametri IN

Possono essere realizzati in due modi:

1. con un riferimento; in questo caso la locazione del parametro attuale diventa la locazione del parametro formale;
   poiché il parametro formale è di tipo IN, allora si deve impedire la modifica all'interno della procedura;
2. con una copia; in questo caso in una nuova locazione, quella del parametro formale, viene copiato il valore del parametro attuale; parametro formale visto come variabile locale;
   modifica permessa, perché valida solo nell'ambiente di esecuzione della procedura.

Il secondo modo è meno efficiente del primo, sia rispetto allo spazio sia al tempo, ma è più flessibile e richiede meno variabili locali.

Parametri OUT

Possono essere realizzati:

1. con un riferimento;
   il compilatore impedisce in questo caso l'accesso al valore di quel data object;
2. con una copia;
   in questo caso il valore è copiato alla terminazione della procedura e non ci sono prescrizioni per il compilatore.

Parametri IN OUT

Sono la combinazione dei due precedenti. Anch'essi possono essere realizzati:

1. con un riferimento; non ci sono limitazioni all'uso all'interno della procedura;
2. con una copia; avvengono due processi di copia, uno durante l'attivazione ed uno durante la terminazione della procedura.

Aliasing

È la possibilità di riferirsi alla stessa locazione con nomi diversi.
Nel passaggio dei parametri può causare notevoli problemi di interpretazione. Per esempio:

```
program MAIN;
  var
    A: integer;
  procedure TEST (var X, Y: integer);
  begin
    X:= A + Y;
    writeln(A, X, Y)
  end;
begin
  A:= 1;
  TEST(A, A)
end.
```

Esercizio: determinare l'uscita del programma nel caso in cui i parametri VAR siano realizzati *per riferimento* e nel caso in cui siano realizzati *per copia*.

Procedure come parametri di procedura

Alcuni linguaggi permettono l'uso di procedure come argomento di altre procedure. Esempio:

```
program MAIN;
  VAR a: real;
  procedure TESTPOS (X: real; procedure ERROR (MSG: string));
    begin
      if X <= 0 then ERROR ('Negative X in TESTPOS')
    end;
  procedure E1 (M: string);
    begin
      writeln('E1 error: ', M)
    end;
  procedure E2 (M: string);
    begin
      writeln('E2 error: ', M)
    end;
begin
  readln (A);
  TESTPOS(A, E1);
  TESTPOS(A, E2)
end.
```

Macro

Generazione di un nuovo brano di codice sorgente (espansione della macro) in cui i nomi dei parametri attuali sostituiscono i nomi dei parametri formali.
Esempio: data la procedura

```
procedure swap (a, b: integer);
  var temp: integer;
  begin
    temp:= a;
    a:= b;
    b:= temp
  end;
```

allora la chiamata `swap(x, y)` esegue il seguente brano di codice:

```
temp:= x;
x:= y;
y:= temp;
```


Esercizio

Determinare i problemi che nascono dai due programmi:

```
program main;
var
  i: integer;
  m: array[1..100] of integer;
  ...
begin
  ...
  swap(m[i], i);
  ...
end.
```

```
program main;
var
  i, temp: integer;
  ...
begin
  ...
  swap(i, temp);
  ...
end.
```


Funzioni

Sono procedure che restituiscono un valore alla procedura chiamante.
Sono realizzate

- o creando una pseudovariabile nell'ambiente locale della procedura chiamata. Tale variabile può essere solo modificata; non è possibile l'accesso in lettura.
- o utilizzando una istruzione di `return` per restituire esplicitamente il controllo alla procedura chiamante inviandole allo stesso tempo il valore di una espressione.

Bibliografia 15 / 15

Bibliografia

- H. L. Dershem. M. J. Jipping. *Programming languages: structures and models*. Second edition. Cap. 5, par. 5.4.