ADA

L'evoluzione del linguaggio

di Fabio Marzocca

Seconda parte

Sottoprogrammi

Il concetto di sottoprogramma è ormai ben noto e radicato in ognuno di noi; difficilmente si riuscirebbe ad immaginare di scrivere un programma in un qualsiasi linguaggio senza avere a disposizione questa specifica caratteristica.

Lo scopo principale di un sottoprogramma è di consentire la parametrizzazione di una particolare algoritmo o processo, senza doverne riscrivere il codice ogni volta che serve.

In Ada esistono due forme di sottoprogrammi: le procedure e le funzioni. La differenza sostanziale sta nel fatto che una procedura definisce una sequenza di azioni e comunica con il programma chiamante tramite una serie di parametri: la funzione consente la definizione di un calcolo restituendo un solo valore.

Tutte le unità di programma di Ada (sottoprogrammi, package e task) sono caratterizzate da due parti:

— un'interfaccia, la quale riassume le sue proprietà all'utente;

— un'implementazione, la quale descrive i suoi compiti interni.

In particolare, ogni sottoprogramma può essere diviso in due parti separate: le dichiarative (l'interfaccia) ed il «body» (l'implementazione). Questa suddivisione consente di poter effettuare la scrittura e la compilazione delle due parti in tempi diversi. In definitiva, l'utente può avere libero accesso alle parti dichiarative di una procedura, senza dover necessariamente conoscere il modo di funzionamento di un body.

In un sottoprogramma, le dichiarazioni definiscono il nome della procedura o funzione, i suoi parametri ed il tipo delle variabili di ritorno. Il body del sottoprogramma invece fornisce le dichiarazioni locali e gli statement di esecuzione del processo (vedi fig. 1).

Ad esempio, consideriamo la sequenza di dichiarazioni:

```
PROCEDURE aggiungi
(I: IN articolo; Q: IN OUT lista)
PROCEDURE togli
(I: OUT articolo; Q: IN OUT lista)
```

Esse definiscono due procedure per aggiungere e togliere un articolo da una lista, e possono essere inserite in un qualunque punto del programma. I body possono essere definiti dopo tutte le dichiarative, ed assumono la seguente forma:

```
PROCEDURE aggiungi
(I: IN articolo; Q: IN OUT lista) IS
.....
END;
PROCEDURE togli
(I: OUT articolo; Q: IN OUT lista) IS
.....
END;
```

In ogni body dei sottoprogrammi viene ripetuta la definizione dei parametri. Questa caratteristica di poter separare una dichiarazione dal suo «corpo», anche in fase di compilazione, è una peculiarità di Ada specificatamente richiesta in sede di progetto.

Ogni parametro di una procedura è caratterizzato da un *modo*, definito come segue:

— IN, agisce come input logico. Il valore non viene modificato;

— OUT, agisce come output logico;

— IN OUT, è un parametro il cui valore può essere usato e modificato durante l'esecuzione della procedura.

Vediamo ora un esempio di body di un sottoprogramma:

```
PROCEDURE proiettile
(velx, vely, distanza: IN FLOAT; salita: OUT
FLOAT) IS
tempo: FLOAT;
BEGIN
tempo := distanza/velx;
Salita := vely*tempo - (g/2.0)*(tempo**2);
END;
```

La procedura calcola la traiettoria di un proiettile (la variabile g è definita dal programma chiamante come costante = 9.81). I parametri sono rappresentati dalla velocità orizzontale (velx), verticale (vely), la distanza e la salita, mentre il tempo è una variabile locale.

Un sottoprogramma di tipo funzione, è definito formalmente come la procedura; la differenza sta nel fatto che, come abbiamo detto, questo è usato per calcolare un singolo valore

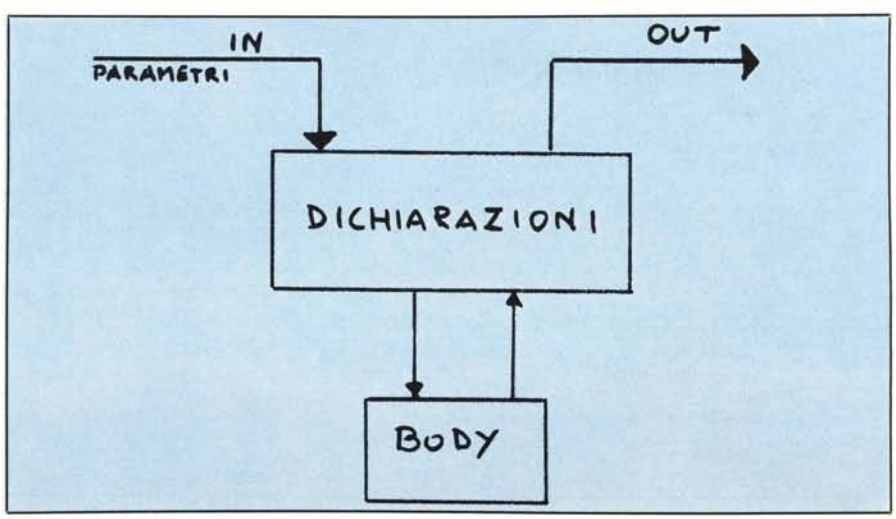

Figura 1 - Struttura di un'unità di programma Ada.

da usare in un'espressione. Una funzione può avere solo parametri di modo «IN».

Supponendo di voler realizzare con una funzione lo stesso sottoprogramma della procedura vista precedentemente, si avrà:

```
FUNCTION proiettile
(velx, vely, distanza: FLOAT)
RETURN FLOAT IS
tempo salita: FLOAT;
BEGIN
tempo:=distanza/velx;
Salita:=vely*tempo - (g/2.0)*(tempo**2)
RETURN salita;
END;
```

Per quanto riguarda le varie possibilità di chiamata di un sottoprogramma, la più intuitiva è rappresentata dal nome del sottoprogramma, seguito dai parametri richiesti. Ad esempio:

```
PROIETTILE (xv, yv, dist, salita)
```

Gli argomenti di chiamata possono essere anche passati al sottoprogramma senza rispettare lo stesso ordine dei parametri formali; in questo caso, però bisognerà indicare con il simbolo → l'associazione tra i corrispondenti argomenti di chiamata e parametri di sottoprogramma. Ad esempio:

```
PROIETTILE (distanza → 150;
            vely     → yv;
            velx     → xv);
```

Spesso accade però che sottoprogrammi di grande utilità, siano caratterizzati da lunghe liste di parametri dotati di valori di default; questo porta ad una nuova peculiarità di Ada. In particolare, tutti i parametri di modo IN, possono essere dichiarati nel sottoprogramma con dei valori di default, cosicché se nella chiamata non dovessero essere esplicitati gli argomenti, il sottoprogramma impiegherà i valori di default. Ricollegandoci sempre alla procedura PROIETTILE, si ha, ad esempio:

```
PROCEDURE proiettile
(velx : IN FLOAT := minx;
vely : IN FLOAT := miny;
distanza : IN FLOAT := 150);
```

In questo modo, volendo impiegare i valori di default nel programma chiamante, sarà sufficiente richiamare PROIETTILE senza specificare gli argomenti.

Esistono dei casi in cui c'è la necessità di definire le stesse operazioni concettuali su argomenti di tipo diverso. Il classico esempio è rappresentato da un processo relativo a numeri interi, da estendere anche ai numeri reali. In questo caso occorrerà definire due procedure con due diversi body, due differenti tipi di parametri, ma con lo stesso nome. Questa caratteristica che possono assumere i sottoprogrammi Ada, viene detta *sovraccarico.*

In figura 2 è riportato il listato di un programma di esempio con sovraccarico sulla funzione SQRT, impiegata per calcolare con metodi numerici la radice quadrata di interi o reali. Il body del sottoprogramma corrispondente verrà selezionato automaticamente in base all'appropriato tipo dell'argomento di ingresso. Il concetto fondamentale è un po' quello che conduce, in questo ed in altri linguaggi, ad impiegare ad esempio l'operatore «+» per addizionare sia interi che reali.

Bibliografia

Ada: An Introduction
Henry Ledgard, Springer Verlag

Il linguaggio Ada
D.J. David Jackson

Beginning Programming with Ada
Saxon & Fritz, Prentice-Hall

Ada: manuale di riferimento
Intel


```
PROCEDURE test_radice_quadrata IS
-----dimostrazione di sovraccarico della funzione SQRT

FUNCTION SQRT(A: integer) RETURN integer IS
c,s : integer;
begin
    s := 1;
    c := 3;
    WHILE s <= A LOOP
            s := s + c;
            c := c + 2;
    END LOOP;
    RETURN C/2-1 ;
END SQRT;


FUNCTION SQRT(A: FLOAT) RETURN FLOAT IS
trova, delta : FLOAT;
x1, x2, x3 : FLOAT;
BEGIN
    IF A = 0.0 then
           RETURN 0.0  ;
    ELSIF A = 1.0 then
           RETURN 1.0 ;
    ELSE
           trova := A/2.0;
           delta := 1.0;
           WHILE trova**2 > A LOOP
                 trova := trova/2.0;
           END LOOP;
           WHILE ABS(trova**2 - A) > 0.0001 LOOP
                 WHILE (trova + delta)**2 > A LOOP
                       IF delta < 0.000001 THEN
                                  RETURN trova;
                       END IF;
                       delta := delta/2.0;
                 END LOOP;
                 trova := trova + delta ;
            END LOOP;
            RETURN trova;
     END IF;
END SQRT;

----- questo e' il body della procedura test_radice_quadrata

BEGIN
     FOR I IN 0..20 LOOP
              PUT ("La radice quadrata di ");
              PUT (I);
              PUT ("e' ");
              PUT (SQRT(I));
              PUT (", ");
              PUT (SQRT(real(I)));
              PUT ("in floating point.")
              newline;
       END LOOP;
       NEWLINE;
       PUT ("Fine del programma demo.");
       PUT (character(7));
END;
```

Figura 2 - Listato del programma test Radice quadrata.

Dai un taglio al passato.

OPEN ACCESS,
l'unico sistema a memoria virtuale, per chi aveva bisogno di più programmi.

Con Open Access si valutano cifre, si disegnano grafici a colori a tre dimensioni, si producono dattiloscritti, si trasmettono dati in tutto il mondo, si gestiscono gli appuntamenti. Basterà inserire i dati una sola volta, qualsiasi numero di applicazioni si vorrà usare. Il segreto delle possibilità eccezionali di Open Access è la gestione delle informazioni con un sistema relazionale di data base. Open Access garantisce un vero «accesso aperto» ai dati con modalità a piacere. Si potrà per esempio, avere accesso fino a cinque file contemporaneamente e in seguito trasferire le informazioni di data base in fogli elettronici, inserirli in rapporti e trasmetterli ai vari partner in affari con l'accesso ad altri computer. Naturalmente si avrà sempre accesso ad altre informazioni e funzioni che aiuteranno a risolvere i problemi quotidiani di lavoro. Open Access offre una straordinaria funzionalità, documentazione e supporti dettagliati in italiano, display a finestre, memorizzazione virtuale e soprattutto integrazione.

OPEN ACCESS,
nato dall'esperienza SPI

NUOVA RELEASE
VERSIONE ITALIANA

Dati tecnici:

data base:	32.000 records; relaziona fino a 5 file
spreadsheet:	3.000 x 216; linka 4 fogli in contemporanea
agenda:	multi-utente
comm.:	9.600 baud in duplex o semi-duplex, accede direttamente ai file di altri computer.

SVPT SRL
Sviluppo Vendite Prodotti Tecnologici
Via Val Cristallina, 3 - 00141 Roma (Italia)
Tel. (06) 8278951 Ricerca automatica - Telex 622147 SVPT I

Azeta