Esercitazione 1 (09/10/02)

18 febbraio 2003

1 Programmazione Pascal, esempi

Come da tradizione incominciamo con un esempio banale di programma Pascal:

```
(* esempio minimale di programma Pascal *)
program Salve;
begin
 writeln('Salve mondo!')
end.
```

Vediamo ora un programma leggermente più complesso:

```
program AreaDelCerchio;
const pi = 3.1416;
var raggio: real;
area: real;
begin
 (* chiede all'utente di specificare il raggio *)
 write('Inserisci il raggio: ');
 (* legge il valore del raggio *)
 readln(raggio);
 (* esegue il calcolo dell'area del raggio *)
 area := pi * raggio * raggio;
 (* visualizza il risultato *)
 writeln('L"area del cerchio è: ', area)
end.
```

Da notare la stretta divisione in

- intestazione
- sezione_dichiarativa
- sezione_esecutiva

Il simbolo ";" assume un diverso significato rispetto al Java, nel caso del Pascal rappresenta un *separatore* viceversa in Java ha il ruolo semantico di *terminatore*.

2 Una semplice tabella comparativa tra Pascal e Java

Fare riferimento al documento all'indirizzo:
http://www.cs.cornell.edu/courses/cs410/1999fa/Handouts/java-compare.htm

3 Altri costrutti "particolari" del linguaggio Pascal

In generale si consiglia di far riferimento al testo
http://amalfi.dis.unina.it/infappl/pascal.html

3.1 Funzioni e procedure

```
(* Questa funzione calcola il quadrato di un numero *)
function Quadrato(x:real):real;
begin
 Quadrato := x*x
end;

(* Questa procedura stampa a video una tabellina *)
procedure StampaTabellina(n:integer);
var i:integer;
begin
 for i:=1 to 10 do
  writeln(n, ' per ' , i , ' = ', n*i)
end;
```

A differenza di quanto avviene in Java, funzioni e procedure possono essere "innestate" e di conseguenza avere diversa "visibilità".

Esempio di programma corretto:

```
program AreaDelCerchio;
const   pi = 3.1416;
var     raggio: real;
        area: real;
(* funziona Calcola Area del cerchio *)
function CalArea(x:real):real;
        (* funzione Eleva al Quadrato *)
        function Eleva2(y:real):real;
        begin
                Eleva2 := y*y;
        end;
begin
        CalArea := pi * Eleva2(x);
end;
begin
```

```
        (* chiede all'utente di specificare il raggio *)
                write('Inserisci il raggio: ');
        (* legge il valore del raggio *)
        readln(raggio);
        (* esegue il calcolo dell'area del raggio *)
        area := CalArea(raggio);
        (* visualizza il risultato *)
        writeln('L"area del cerchio è: ', area)
end.
```

3.2 Dichiarazioni di tipo

In aggiunta ai "tipi" predefiniti dal linguaggio il programmatore può definire dei nuovi tipi, come vedremo in seguito questo risulta molto comodo lavorando con i puntatori e record.

3.2.1 Tipi enumerativi

```
type    Mese = (Gennaio, Febbraio, Marzo, Aprile, Maggio,
        Giugno, Luglio, Agosto, Settembre, Ottobre, Novembre,
        Dicembre);

        GiornoSettimana = (lun, mar, mer, gio, ven, sab, dom);
(* Questo frammento stampa per ciascun giorno
tra lunedì e venerdì la posizione del giorno
nella sequenza dei giorni della  settimana *)
var giorno: GiornoSettimana;
begin
        giorno := lun;
        while giorno <= ven do
        begin
                writeln(ord(giorno)); (* posizione del giorno *)
                giorno := succ(giorno)
        end;
end.
```

3.2.2 Tipi record

```
type Persona = record
                nome: string[30];
                cognome: string[30];
                eta : integer;
var p : persona;
begin
        p.nome := 'Richard M.';
        p.cognome := 'Fujimoto';
```

```
p.eta = 45;
....
```

4 Strumenti di sviluppo

Durante lo studio di un nuovo linguaggio è estremamente importante provare i frammenti di codice che vengono scritti. Fortunatamente esistono dei compilatori rilasciati sotto licenza GPL e disponibili per tutti i sistemi operativi maggiormenti diffusi.

Ad esempio:

- FPC `http://www.freepascal.org`
- GPC `http://www.gnu-pascal.de/`

5 I puntatori

Le variabili di tipo puntatore sono assenti in Java, o meglio Java per una serie di motivi progettuali non è stato dotato di aritmetica dei puntatori e puntatori espliciti.

Per una trattazione introduttiva del problema si consiglia di studiare il testo di riferimento del corso ed in particolare il paragrafo 1.3 (pag. 31-33).

Un testo che invece analizza il problema con maggiore dettaglio, facendo anche riferimento agli errori comuni d'utilizzo è reperibile all'indirizzo `http://www.disi.unige.it/person/MagilloP/ALGO/lezioni1`.

Iniziamo a parlare di puntatori attraverso un esempio:

```
(* esempio banale di utilizzo dei puntatori *)
program Puntatori;
var pInt : ^integer;
begin
        new(pInt);
        pInt^ := 2001;
        writeln(pInt^);
        dispose(pInt)
end.
```

Ovviamente le dichiarazioni di tipo possono rendere il codice più leggibile:

```
(* esempio di utilizzo dei puntatori con uso
dei tipi definiti dall'utente*)
program Puntatori;
type pInt = ^integer;
var     punta1 : pInt;
        punta2 : ^integer;
begin

        new(punta1);
```

```
        punta1^ := 2001;
        writeln(punta1^);
        dispose(punta1);

        new(punta2);
        punta2^ := 2003;
        writeln(punta2^);
        dispose(punta2)
end.
```

NOTA: in questo caso risulta che punta1 e punta2 hanno tipi incompatibili fra di loro!

Vediamo qualche altro esempio tipico di uso dei puntatori:

```
program Puntatori;
type puntaintero = ^integer;
var     pippo : integer;
        pluto : puntaintero;
        paperino : ^integer;
        (*      pippo è un intero,
                pluto è un puntatore ad intero *)
begin
        (* ipotizziamo che pluto = 2001, cosa significa? *)
        pippo := 101;

        new(pluto);
        pluto^ := 201;
        writeln('pippo ', pippo, ' pluto ', pluto^);
        dispose(pluto);
        (* che output otteniamo? *)
        pluto^ := pippo;
        writeln('pippo ', pippo, ' pluto ', pluto^);
        (* cosa succede? *)
        pluto := pippo;
        writeln('pippo ', pippo, ' pluto ', pluto^);

        (* è corretta? no, memoria deallocata! *)
        paperino := pippo;
        writeln('pippo ', pippo, ' paperino ', paperino^)
        (* è corretta? no, tipi incompatibili! *)
end.
```

Vediamo un esempio di utilizzo:

```
(* frammento di codice non compilabile *)
type catena = ^studente;
        studente = record
                matricola : integer;
```

```
                successivo : catena
        end;
var inizio, scandisci : catena;
(* allocazione *)
new(inizio)
new(inzio^.successivo);
scandisci := inizio^.successivo;
new(scandisci^.successivo);
scandisci^.successivo^.successivo := nil;
```

5.1 Errori tipici

1. utilizzo di tipi incompatibili (vedi esempi precedenti)
2. assenza dell'operatore di deferenziazione: a differenza del C non esiste in Pascal un operatore del tipo "indirizzo di" con valore di tipo puntatore.
3. artimetica dei puntatori: gli unici operatori di confronto validi su puntatori sono = e <>, *non* si applicano le operazioni aritmetiche tipiche degli interi.
4. errori di gestioni della memoria, assegnazioni senza allocazioni, deallocazioni multiple ecc.
5. dati due puntatori p e q, p = q produce un fenomeno detto "aliasing" ovvero entrambi i puntatori hanno come oggetto la stessa cella di memoria. Modificare la cella attraverso uno dei puntatori produce ovviamente effetto anche su quanto ottenuto dall'altro puntatore.
6. garbage: memoria allocata che rimane inutilizzabile in quanto abbiamo "perso" il puntatore per accederla [ad esempio new(p); p=q SBAGLIATO, manca il dispose(p)]