

Algoritmi e Strutture Dati A-L. Informatica

Esercitazioni 1

1 marzo 2007

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 ad altri linguaggi, nel caso del Pascal rappresenta un *separatore* viceversa in Java (ad esempio) ha il ruolo semantico di *terminatore*.

2 Introduzione al linguaggio Pascal

In generale si consiglia di far riferimento al testo online “Introduzione al linguaggio Pascal” (<http://amalfi.dis.unina.it/infappl/pascal.html>) e Pascal programming language su Wikipedia (http://en.wikipedia.org/wiki/Pascal_programming_language).

2.1 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 altri linguaggi, 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.
```

2.2 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.

2.2.1 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.
```

2.2.2 Tipi record

```
type Persona = record
    nome: string[30];
    cognome: string[30];
    eta : integer
end;
var p : persona;
begin
    p.nome := 'Niklaus';
    p.cognome := 'Wirth';
    p.eta := 70;
    ....
```

3 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 maggiormente diffusi.

Ad esempio:

- Free Pascal Compiler [FPC] (<http://www.freepascal.org>)
Disponibile per molte architetture e sistemi operativi (Linux, Win32 ecc.), pacchetto fp-compiler in Debian unstable.
Gli esempi proposti a lezione solitamente sono testati con questo compilatore.

- GNU Pascal Compiler [GPC] (<http://www.gnu-pascal.de/>)

4 I puntatori

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 plnt : ^integer;
begin
    new(plnt);
    plnt^ := 2001;
    writeln(plnt^);
    dispose(plnt)
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 plnt = ^integer;
var punta1 : plnt;
    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
    pippo := 101;

    new(pluto);
    pluto^ := 201;
    writeln('pippo ', pippo, ' pluto ', pluto^);
    (* che output otteniamo? *)
    dispose(pluto);
    pluto^ := pippo;
    (* operazione corretta? *)
    (* NO la memoria è deallocata! *)
    writeln('pippo ', pippo, ' pluto ', pluto^);
    (* cosa succede? *)
    pluto := pippo;
    writeln('pippo ', pippo, ' pluto ', pluto^);
    paperino := pippo;
    writeln('pippo ', pippo, ' paperino ', paperino^)
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(inizio^.successivo);
scandisci := inizio^.successivo;
new(scandisci^.successivo);
scandisci^.successivo^.successivo := nil;

```

4.1 Errori tipici

1. utilizzo di tipi incompatibili (vedi esempi precedenti)
2. assenza dell'operatore di deferenziamento: a differenza del C non esiste in Pascal un operatore del tipo "indirizzo di" con valore di tipo puntatore.
3. aritmetica 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)`]