Nel capitolo introduttivo è stato visto l'uso di variabili identificabili semplicemente con il loro nome. La programmazione elementare richiede anche l'utilizzo di strutture di dati più complesse; le stesse stringhe sono degli array di caratteri e come tali vanno trattate.

Array

Gli array in Pascal sono una sequenza ordinata, in una quantità prestabilita, di elementi dello stesso tipo. Gli elementi possono essere composti da qualunque tipo di dati, nativo o derivato.

Una caratteristica importante del linguaggio Pascal sta nel fatto che nel momento della dichiarazione di un array, viene definito anche il valore iniziale dell'indice da utilizzare per la scansione dei vari elementi.

Dichiarazione e accesso

```
var nome : array[inizio..fine] of tipo
```

La sintassi indicata, dove le parentesi quadre fanno parte dell'istruzione, mostra in breve in che modo si possa dichiarare un array, a una sola dimensione, di elementi di un certo tipo di dati.

È importante osservare che vengono stabiliti in modo esplicito sia l'indice iniziale del primo elemento, sia quello finale dell'ultimo, stabilendo implicitamente la quantità di questi elementi.

L'esempio seguente mostra la dichiarazione di tre array simili, composti tutti da sette interi, dove, rispettivamente, il primo elemento si raggiunge con l'indice iniziale 1, 0 e 2.

```
var elenco  : array[1..7] of integer;
    elenco2 : array[0..6] of integer;
    elenco3 : array[2..8] of integer;
```

Per accedere agli elementi di un array si usa la sintassi seguente e anche qui le parentesi quadre fanno parte dell'istruzione.

```
nome[indice]
```

Quello che conta è che l'indice indicato sia valido, in funzione della dichiarazione fatta in origine. L'esempio seguente assegna al primo elemento il valore 10.

```
elenco[1] := 10;
```

Gli array multidimensionali non sono altro che array di array. Il modo più semplice per dichiarare un array multidimensionale è quello di indicare due o più intervalli di valori per gli indici, secondo la sintassi seguente:

```
var nome : array[inizio..fine,inizio..fine...] of tipo
```

Per esempio, l'istruzione seguente dichiara un array a due dimensioni di tre elementi per otto, di tipo intero. Si osservi in particolare il secondo intervallo di indici, dove il primo elemento viene raggiunto con l'indice zero.

```
var elenco : array[1..3,0..7] of integer;
```

In modo analogo, si raggiunge un elemento di un array multidimensionale utilizzando due o più indici, secondo la sintassi seguente:

nome [***indice*** , ***indice***…]

L'esempio seguente assegna un valore all'elemento «1,0».

```
elenco[1,0] := 10;
```

Scansione di un array

La scansione di un array avviene generalmente con un ciclo enumerativo, '**`for`**', come nell'esempio seguente:

```
…
var indice : integer;
var elenco : array[1..7] of integer;
…
begin
    …
    for indice := 1 to 7 do begin
        …
        elenco[indice] := …
        …
    end;
    …
end.
```

La scansione di array multidimensionali avviene generalmente attraverso una serie di cicli enumerativi, uno per ogni dimensione, annidati opportunamente. L'esempio seguente mostra la scansione di un array a tre dimensioni.

```
…
var i,j,k : integer;
var elenco : array[1..7,0..8,2..10] of integer;
…
begin
    …
    for i := 1 to 7 do begin
        …
        for j := 0 to 8 do begin
            …
            for k := 2 to 10 do begin
                …
                elenco[i,j,k] := …
                …
            end;
            …
        end;
        …
    end;
    …
end.
```

Stringhe

Nel linguaggio Pascal, così come in molti altri, le stringhe sono semplicemente degli array di caratteri, con qualche piccola differenza per facilitarne l'utilizzo.

La dichiarazione di una variabile stringa è quindi la dichiarazione di un array composto da una quantità predefinita di caratteri. Nell'esempio seguente, viene creato una variabile stringa di 20 caratteri.

```
var cognome : array[1..20] of char;
```

La variabile dichiarata in questo modo può essere usata come un array, cioè accedendo alle informazioni carattere per carattere, oppure nel suo insieme. Nell'esempio seguente si assegna un nome alla variabile stringa mostrata sopra.

```
cognome := 'Rossi';
```

Se si utilizza un assegnamento di questo tipo, vengono ricoperti anche gli elementi successivi alla lunghezza della stringa letterale assegnata. Quindi, seguendo l'esempio, l'array riceve il nome «Rossi» nei suoi primi cinque elementi, mentre negli altri viene comunque inserito uno spazio.

Tipi

Il linguaggio Pascal permette di definire dei tipi di dati derivati, a partire da quelli elementari, o a partire da altri tipi composti dichiarati precedentemente.

`type` ***tipo_nuovo*** `=` ***definizione_del_tipo***

La definizione di un nuovo tipo va posta nella zona dichiarativa del programma, della procedura o della funzione. L'esempio seguente serve a dichiarare il tipo «Numero» come equivalente al tipo intero standard.

```
type Numero = integer;
```

Naturalmente, la definizione di un nuovo tipo è sensata quando serve a individuare qualcosa di più complesso dei dati elementari, come nel caso di un array. L'esempio seguente dichiara il tipo «Stringa» come un array di 80 caratteri, quindi dichiara il tipo «Nominativo» come array composto da due elementi «Stringa» (probabilmente uno per il nome e l'altro per il cognome).

```
type Stringa = array[1..80] of char;
type Nominativo = array[1..2] of Stringa;
```

A questo punto, per seguire l'esempio, se si generasse una variabile di tipo «Nominativo», si otterrebbe un array di due elementi, che in realtà sono array di 80 caratteri.

```
…
var Nome : Nominativo;
…
begin
    …
    Nome[1] := 'Pinco';
    Nome[2] := 'Pallino';
    …
end.
```

L'esempio mostra in che modo si potrebbe usare una variabile del genere. Tuttavia, si potrebbe accedere anche al singolo elemento carattere, utilizzando due indici.

```
    …
    Nome[1,1] := 'P';
    Nome[1,2] := 'i';
    Nome[1,3] := 'n';
    Nome[1,4] := 'c';
    Nome[1,5] := 'o';
    …
end.
```

Convenzionalmente, quando si dichiara un nuovo tipo di dati, si usa l'iniziale maiuscola, per distinguerlo facilmente dagli altri tipi nativi.

Costanti

Il linguaggio Pascal offre qualcosa di simile alle costanti macro di altri linguaggi come il C. Non si tratta di un linguaggio di precompilazione, ma proprio del Pascal, anche se si tratta comunque di costanti letterali, senza la definizione di un tipo a priori.

`const` ***nome_della_costante*** `=` ***valore_letterale***

La dichiarazione di queste costanti va fatta, come prevedibile, nella zona dichiarativa del programma, della procedura o della funzione. L'esempio seguente dichiara la costante «DIMENSIONE», che poi viene usata per definire la dimensione di una serie di array.

```
…
const DIMENSIONE = 11;
…
var elenco  : array[1..DIMENSIONE] of integer;
    elenco2 : array[1..DIMENSIONE] of integer;
    elenco3 : array[1..DIMENSIONE] of integer;
```

Il vantaggio di utilizzare le costanti sta nel facilitare la lettura del sorgente, nel riconoscere il significato di determinate costanti e nel facilitare la modifica di tali valori, senza dover rileggere tutto il sorgente alla loro ricerca.

Tipo enumerativo, sottointervallo e insieme

Il linguaggio Pascal offre dei tipi di dati particolari, che non sono ancora stati descritti, il cui scopo è solo quello di facilitare il compito del programmatore.

Tipo enumerativo

Il tipo enumerativo, o scalare, secondo la terminologia del Pascal, è una forma di rappresentazione di un intero attraverso costanti mnemoniche. In pratica, si definisce una variabile che può assumere un elenco di valori simbolici possibili, valori che in realtà sono solo delle costanti simboliche, senza un legame con il termine letterale usato per distinguerle, salva la convenienza del programmatore.

(*costante* , *costante* [, …])

La sintassi indicata mostra il modo in cui si definisce un tipo del genere: all'interno di parentesi tonde si elencano i nomi delle costanti che possono essere assegnate a una variabile di questo tipo.

L'esempio seguente mostra la dichiarazione di una variabile scalare che può assumere i valori «VERDE», «BLU» e «ROSSO».

```
var colore : (VERDE, BLU, ROSSO);
```

L'esempio stesso dovrebbe chiarire l'utilità di questo tipo di dati: si lascia al compilatore il compito di stabilire i valori più appropriati per i simboli che possono essere associati a una variabile. Tuttavia, è importante chiarire che non è possibile visualizzare il contenuto di una variabile del genere, in quanto questo non è prevedibile.

```
if colore = VERDE then
    begin
        …
        Writeln( "Il colore è verde" );
    end;
else
    …
;
```

Naturalmente, questo tipo di dati si presta particolarmente per la definizione di tipi derivati, come nell'esempio seguente, dove prima si dichiara un tipo e più avanti si utilizza nella dichiarazione di una nuova variabile.

```
type Sapore = (INSIPIDO, DOLCE, SALATO, ACIDO, PICCANTE, AMARO);
…
var pietanza : Sapore;
…
```

Sottointervallo

Il sottointervallo è la definizione di un tipo derivato che può utilizzare solo un intervallo stabilito di valori. Questo intervallo si definisce solo con l'indicazione di due costanti dello stesso tipo, separate da due punti in sequenza.

Per esempio, per indicare la serie di numeri interi che va da uno a sette, si può utilizzare la notazione '**`1..7`**', mentre per indicare la serie delle lettere alfabetiche minuscole, si può utilizzare la notazione '**`'a'..'z'`**'.

Naturalmente, si possono indicare anche degli intervalli di un tipo enumerativo dichiarato in precedenza. Seguono alcuni esempi.

```
type Settimana = (LUNEDÌ, MARTEDÌ, MERCOLEDÌ,
                  GIOVEDÌ, VENERDÌ, SABATO, DOMENICA);

type Feriale   = LUNEDÌ..VENERDÌ;
…
var lavoro    : Feriale;
    minuscola : 'a'..'z';
…
```

Le variabili dichiarate in questo modo, ottengono dal compilatore il tipo più adatto a contenere l'informazione indicata, senza la necessità di doverlo indicare in modo esplicito.

Insieme

Una variabile può contenere un'informazione riferita a un insieme di elementi enumerativi. In pratica, si tratta di un tipo simile a quello enumerativo, dove ogni elemento può essere presente o meno. Si dichiara questo tipo di dati con le parole chiave '**`set of`**'. Si osservi l'esempio seguente:

```
type Settimana = (LUNEDÌ, MARTEDÌ, MERCOLEDÌ,
                  GIOVEDÌ, VENERDÌ, SABATO, DOMENICA);
…
type Lavoro    = set of Settimana;
…
var tutti     : Lavoro;
    presenze  : Lavoro;
    assenze   : Lavoro;
    altri     : Lavoro;
…
```

Le variabili '**`tutti`**', '**`presenze`**' e '**`assenze`**', definite del tipo '**`Lavoro`**', il quale a sua volta è definito come insieme di tutti i simboli del tipo '**`Settimana`**', possono contenere un sottoinsieme di tali simboli.

```
…
begin
    …
    presenze := (LUNEDÌ, MERCOLEDÌ, VENERDÌ,
                 DOMENICA);
    …
    tutti := (LUNEDÌ..DOMENICA);
    …
    assenze := tutti - presenze;
    …
    altri := assenze;
    …
    tutti := assenze + presenze;
    …
end.
```

L'esempio mostra alcuni modi in cui possono essere utilizzate le variabili contenenti insiemi e quali espressioni si possono realizzare. In pratica:

- due variabili dello stesso tipo di insieme possono essere assegnate l'una nell'altra;
- due variabili dello stesso tipo di insieme possono essere sommate, generando un insieme risultato dell'unione dei due;
- tra due variabili dello stesso tipo di insieme può essere indicata una sottrazione, con la quale si genera un insieme risultato dall'eliminazione degli elementi presenti nella seconda variabile.

A parte gli assegnamenti che possono essere fatti alle variabili contenenti un insieme, è poi necessario poter verificare il contenuto di tali variabili, con istruzioni apposite. Per questo si usa la parola chiave '**`in`**'. L'esempio seguente dovrebbe essere autoesplicativo.

```
    if LUNEDÌ in presenze then begin
        …
        …
    end;
    if MARTEDÌ in presenze then begin
        …
        …
    end;
```

Un insieme può essere definito anche come gruppo di valori di un intervallo, come nell'esempio seguente in cui si definisce un tipo nuovo che rappresenta l'insieme delle lettere minuscole.

```
type Lettere    = set of 'a'..'z';
```

Nello stesso modo, si può utilizzare la parola chiave '**`in`**' per verificare che un valore appartenga a un insieme definito in forma di intervallo.

```
if iniziale in 'a'..'z' then begin
    …
end;
```

Record

Il record è un tipo di dati composto dall'insieme di altri tipi, ognuno con una sua denominazione. L'esempio seguente mostra in che modo possano essere creati tipi nuovi definiti come record.

```
type Datario =
    record
        anno    : integer;
        mese    : integer;
        giorno  : integer;
    end;

type Anagrafico =
    record
        cognome : array[1..40] of char;
        nome    : array[1..40] of char;
        luogo   : array[1..40] of char;
        data    : Datario;
    end;
```

L'esempio vuole mostrare la creazione di un record anagrafico con tutti i dati (riferiti alla nascita) che permettono di identificare una persona. Si può osservare che la data (di nascita) è stata definita come tipo '**`Datario`**', che a sua volta è un altro record.

Quando si dichiara una variabile come tipo record, si pone il problema di accedere ai vari elementi di questo. Per farlo si usa l'operatore punto ('**`.`**'). Si osservi l'esempio seguente, in cui si dichiara un array di dati anagrafici e quindi si assegnano i valori per il primo elemento di questo array.

```
…
var anagrafe : array[1..10] of Anagrafico;
…
begin
    …
    anagrafe[1].cognome         := 'Pallino';
    anagrafe[1].nome            := 'Pinco';
    anagrafe[1].luogo           := 'Sferopoli';
    anagrafe[1].data.anno       := 1990;
    anagrafe[1].data.mese       := 1;
    anagrafe[1].data.giorno     := 31;
    …
end;
```

Come si può osservare, per inserire le informazioni sulla data di nascita, è stato necessario usare due volte il punto per accedere agli elementi del sottorecord '**`data`**'.

Una variabile definita come record può ricevere l'assegnamento in blocco di un'altra variabile record, purché dello stesso tipo.

With

Quando si utilizzano frequentemente i record, potrebbe essere conveniente specificare che in una porzione di codice sorgente si vuole fare riferimento a elementi di una variabile determinata. Si osservi l'esempio seguente, che è una variante di quanto già visto in precedenza.

```
…
var anagrafe : array[1..10] of Anagrafico;
…
begin
    …
    with anagrafe[1] do begin
        cognome         := 'Pallino';
        nome            := 'Pinco';
        luogo           := 'Sferopoli';
        data.anno       := 1990;
        data.mese       := 1;
        data.giorno     := 31;
    end;
    …
end;
```

Il significato dovrebbe essere evidente: nell'intervallo delimitato dalle parole chiave '**`begin`**' '**`end`**', tutti i nomi si riferiscono a elementi di '**`anagrafe[1]`**'.

Riferimenti

- Gordon Dodrill, *Pascal Language Tutorial*
  http://www.geocities.com/hotdogcom/ptutor/paslist.html
  http://packetstormsecurity.nl/programming-tutorials/Pascal/pascal-tutorial/paslist.htm