TIPI DI DATO STRUTTURATI o STRUTTURE DATI

Un **tipo di dato** è una entità caratterizzata dai seguenti elementi:

- **un insieme X di valori** che rappresenta il "*dominio*" del tipo di dato;
- **un insieme opzionale di costanti** che caratterizzano l'insieme X;
- **un insieme di operazioni** che si possono effettuare sull'insieme X.

I tipi di dato possono essere classificati in :

a) **tipi elementari o tipi semplici**: i cui dati non sono costituiti da altri dati ed inoltre nel corso del processo risolutivo, in un certo istante di tempo, possono assumere un unico valore;
(Es. **INT**, **REAL**, **CHAR**, **BOOL**)

b) **tipi strutturati:** i cui dati sono aggregazioni di tipi elementari e/o strutturati che è possibile estrarre tramite opportune operazioni ed inoltre nel corso del processo risolutivo, in un certo istante di tempo, possono assumere più valori
(Es. **ARRAY o VETTORE** ad UNA dimensione, o a DUE dimensioni (**MATRICE**), **RECORD, ARRAY DI RECORD**)

ARRAY o VETTORE MONODIMENSIONALE

DEFINIZIONE: Un **array o vettore monodimensionale** è una struttura dati di tipo ***sequenziale*** costituita da un insieme di elementi ***omogenei*** tra loro, individuabili per mezzo di un indice (ossia ad ***accesso diretto***)

SEQUENZIALE: (Significato: GLI ELEMENTI DEL VETTORE OCCUPANO CELLE CONSECUTIVE DI MEMORIA) **contrario NON SEQUENZIALI**

OMOGENEA: (Significato: GLI ELEMENTI DEL VETTORE DEVONO AVERE TUTTI LO STESSO TIPO) **contrario ETEROGENEO**

ACCESSO DIRETTO: (Significato: POSSO RAGGIUNGERE/ACCEDERE DIRETTAMENTE AD UN CERTO ELEMENTO DEL VETTORE SENZA DOVER PASSARE PER TUTTI QUELLI CHE LO PRECEDONO) **contrario ACCESSO SEQUENZIALE**

N.B. Nel nostro, caso al momento, tratteremo **vettori o array a carattere statico** memorizzati in opportune **strutture di memoria** presenti nella RAM.

A CARATTERE STATICO: (Significato: LO SPAZIO DI MEMORIA – OSSIA IL NUMERO DI BYTE TOTALI - ASSEGNATO AL VETTORE NON CAMBIA NEL CORSO DEL PROCESSO RISOLUTIVO)
Contrario: A CARATTERE DINAMICO

STRUTTURA DI MEMORIA: (Significato: i valori della struttura dati sono memorizzati nella RAM (ossia nella memoria di lavoro volatile e vengono quindi perduti ogni qualvolta si spegne il calcolatore)
Contrario: FILE O ARCHIVI

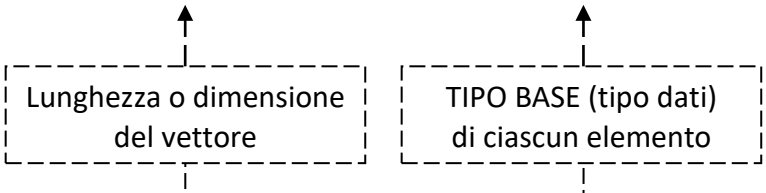
MODELLO ASTRATTO (PSEUDOCODIFICA) DELL'ARRAY MONODIMENSIONALE)

ARRAY o VETTORE v

4	12	34	-8	45	-23	17	3
1	2	3	4	5	6	7	8

PSEUDOCODIFICA: DICHIARAZIONE

<nome_vettore> **: ARRAY [**<num_elementi>**] DI** <tipo_elemento>
<indice> : **INT**



Quindi nel nostro caso

v : ARRAY[**8**] DI **INT**
i : **INT**

v[6] cosa rappresenterà? il 6° elemento del vettore quindi vale -23

i ← 3
v[i] cosa rappresenterà? il 3° elemento del vettore quindi vale 34

v[2] ← v[1] + v[3] assegna al 2° elemento dell'array 4 + 34 = 38

v[9] cosa rappresenterà? IMPOSSIBILE sfondamento array (a destra)
v[0] cosa rappresenterà? IMPOSSIBILE sfondamento array (a sinistra)

NOTA BENE

ARRAY o VETTORE v

0	1	2	3	4	5	6	7
4	12	34	-8	45	-23	17	3
1	2	3	4	5	6	7	8

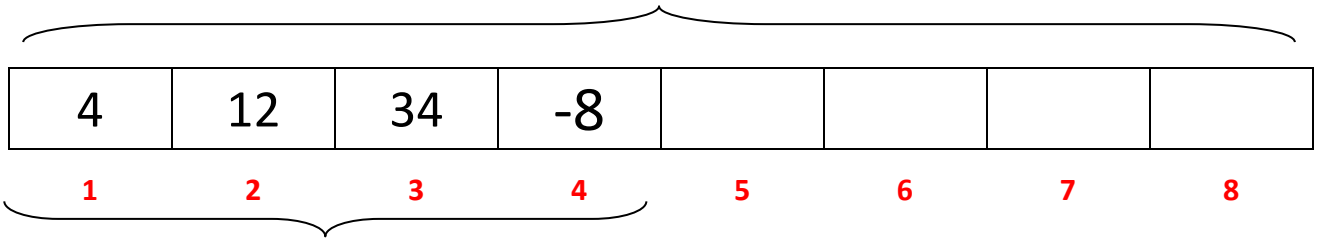
LINGUAGGIO C: La dimensione del vettore è pari ad 8 però l'indice degli elementi va da 0 a 7

PSEUDOCODIFICA: La dimensione del vettore è pari ad 8 però l'indice degli elementi va da 1 a 8

FALSA DINAMICITA'

ARRAY o VETTORE v

(formato al MASSIMO da **MAXDIM** elementi)



Vettore è come se avesse ADESSO n = 4 elementi

PROBLEMA: progettare un algoritmo che si occupi di caricare per poi visualizzare i valori contenuti in un vettore o array monodimensionale di n interi qualunque

TABELLE DEI DATI

DATI DI INPUT DEL PROBLEMA PRINCIPALE (PROCEDURA MAIN)				
Nome variabile	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**
v	**ARRAY[MAXDIM] DI INT**	**STATICA**	**TUTTI**	Vettore monodimensionale con valori interi qualsiasi da acquisire da tastiera
n	**INT**	**STATICA**	**1 ≤ n ≤ MAXDIM**	Dimensione (o lunghezza) effettiva del vettore da trattare

DATI DI OUTPUT DEL PROBLEMA PRINCIPALE (PROCEDURA MAIN)				
Nome variabile	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**
v	**ARRAY[MAXDIM] DI INT**	**STATICA**	**TUTTI**	Vettore monodimensionale con valori interi qualsiasi da visualizzare a video

DATI DI ELABORAZIONE o DI LAVORO DEL PROBLEMA PRINCIPALE (PROCEDURA MAIN)				
Nome variabile oppure **nome costante**	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**
MAXDIM	**INT**	**STATICA**	**8**	Dimensione (o lunghezza) massima di elementi gestibili
i	**INT**	**STATICA**	**1 ≤ i ≤ n + 1**	Dimensione (o lunghezza) massima di elementi gestibili

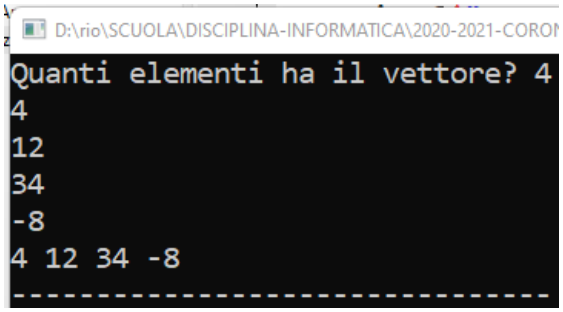
PSEUDOCODIFICA

```
ALGORITMO CaricaVisualizzaVettore

// DIMENSIONE massima di elementi del vettore
MAXDIM     8

PROCEDURA main ( )

/* variabile di INPUT ed OUPUT – vettore da caricare e visualizzare */
v: ARRAY [MAXDIM] DI INT
/* variabile di INPUT – numero di elementi del vettore che si intende utilizzare (DIMENSIONE) */
n : INT
/* variabile di LAVORO o ELABORAZIONE – indice per scorrere il vettore v */
i : INT

INIZIO

/* leggo la dimensione del vettore da caricare non superando il valore MAXDIM */
RIPETI
   Scrivi("Quanti elementi ha il vettore? ")
   Leggi (n)
FINCHE' (n ≥ 1) AND (n ≤ MAXDIM)          /* Nel nostro caso ipotizziamo n = 4 */

/* carico (leggo) gli elementi nel vettore */
PER i ← 1 A n ESEGUI                      /* Quindi n vale ADESSO 4 */
   Leggi (v[i])
   i ← i + 1
FINE PER

/* visualizzo (stampo) gli elementi precedentemente immessi nel vettore */
PER i ← 1 A n ESEGUI
   Scrivi (v[i])
   i ← i + 1
FINE PER

FINE
```

CODIFICA C

Versione 1 "**poco**" USER-FRIENDLY

```c
#include <stdio.h>
#include <stdlib.h>

#define MAXDIM 8

int main(int argc, char *argv[])
{
/* variabile di INPUT ed OUPUT - vettore da caricare e visualizzare */
int v[MAXDIM];
/* variabile di INPUT - numero di elementi del vettore che si intende
utilizzare (DIMENSIONE) */
int n;
/* variabile di LAVORO o ELABORAZIONE - indice per scorrere il vettore v */
int i;

/* leggo la dimensione del vettore da caricare non superando il valore MAXDIM */
do
   {
   Printf ("Quanti elementi ha il vettore? ");
   scanf ("%d", &n);
   }
while ((n < 1) || (n > MAXDIM));

/* carico (leggo) gli elementi nel vettore */
for (i = 0; i < n; i++)
   {
   scanf ("%d", &v[i]);
   }

/* visualizzo (stampo) gli elementi precedentemente immessi nel vettore */
for (i = 0; i < n; i++)
   {
   printf ("%d ", v[i]);
   }

return 0;
}
```



CODIFICA C

Versione 2 "**molto**" USER-FRIENDLY

```c
#include <stdio.h>
#include <stdlib.h>

#define MAXDIM 8

int main(int argc, char *argv[])
{
/* variabile di INPUT ed OUPUT - vettore da caricare e visualizzare */
int v[MAXDIM];
/* variabile di INPUT - numero di elementi del vettore che si intende utilizzare (DIMENSIONE) */
int n;
/* variabile di LAVORO o ELABORAZIONE - indice per scorrere il vettore v */
int i;

/* leggo la dimensione del vettore da caricare non superando il valore MAXDIM */
do
   {
   system("CLS");
   printf("Quanti elementi ha il vettore? ");
   scanf ("%d", &n);

   /* gestione valore errato per la dimensione */
   if ((n < 1) || (n > MAXDIM))
      {
      printf ("La dimensione deve essere un numero compreso tra 1 e %d\n", MAXDIM);
      system("PAUSE");   /* in alternativa è possibile usare l'istruzione getch(); */
      getch();
      }
   }
while ((n < 1) || (n > MAXDIM));

/* carico (leggo) gli elementi nel vettore */
for (i = 0; i < n; i++)
   {
   printf ("Inserisci il %d^ elemento: ", i + 1);
   scanf ("%d", &v[i]);
   }
```

```
D:\rio\SCUOLA\DISCIPLINA-INFORMATICA\2020-2021-CORONA
Quanti elementi ha il vettore? 4
Inserisci il 1^ elemento: 4
Inserisci il 2^ elemento: 12
Inserisci il 3^ elemento: 34
Inserisci il 4^ elemento: -8

v = [4 12 34 -8 ]
--------------------------------
```

ATTENZIONE: In questo caso la variabile i è di OUTPUT

```c
/* visualizzo (stampo) gli elementi precedentemente immessi nel vettore */
printf("\nv = [");
for (i = 0; i < n; i++)
   {
   printf ("%d ", v[i]);
   }
printf("]");

return 0;
}
```

E se volessi visualizzare il vettore in questo modo:

v = [4, 12, 34, -8]

come dovrei "aggiustare" il codice?

IL TIPO DI DATI STRUTTURATO MATRICE o ARRAY BIDIMENSIONALE

Per risolvere particolari tipi di problemi si ricorre a strutture dati più complesse del vettore monodimensionale che permette soltanto la memorizzazione di insiemi lineari di informazioni.

*Esempio: Se volessimo memorizzare tutti i voti ottenuti da un gruppo di alunni nelle varie materie seguite invece che una serie di **vettori paralleli** da gestire attentamente da programma (uno per memorizzare gli alunni ed uno per ciascuna materia di cui memorizzare i voti), saremmo interessati ad utilizzare una tabella che avesse tante righe quanti sono gli alunni e tante colonne quante sono le materie, dove l'incrocio tra riga e colonna conterrebbe la votazione di un determinato alunno in una determinata materia.*

Questa struttura dati prende il nome di **matrice o array bidimensionale.**

DEFINIZIONE: Si definisce **MATRICE o ARRAY BIDIMENSIONALE ad n righe ed m colonne** o più brevemente **MATRICE o ARRAY n * m** una tabella formata da **n * m elementi omogenei**, disposti su **n** linee orizzontali (**righe**) ed **m** linee verticali (**colonne**).

L'elemento generico di una matrice si indica con il simbolo $a_{i\,j}$ dove **i** è l'indice di riga e **j** è l'indice di colonna.

Rappresentazione estesa di una matrice:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & \dots & a_{1m} \\ a_{21} & a_{22} & a_{23} & \dots & \dots & a_{2m} \\ a_{31} & a_{32} & a_{33} & \dots & \dots & a_{3m} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & a_{ij} & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & \dots & a_{nm} \end{pmatrix}$$

i-esima riga

j-esima colonna

Rappresentazione compatta di una matrice:

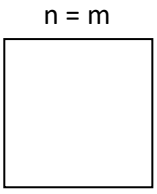
$$A = a_{ij} \text{ con } i = 1 \text{ a } n \text{ e con } j = 1 \text{ a } m$$

Nella PSEUDOCODIFICA per dichiarare all'interno di un algoritmo una variabile di questo tipo occorre usare la seguente pseudoistruzione:

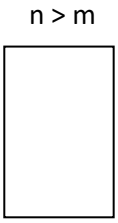
<nome_matrice> **: ARRAY [**<num_righe>**] [**<num_colonne>**] DI** <tipo_elemento>

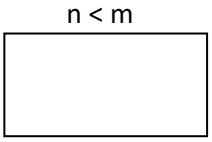
Le matrici si dividono in:

- **matrici quadrate**: nel caso in cui il numero di righe sia uguale al numero di colonne (n = m)
In questo caso la forma teorica che essa assumerà può essere sintetizzata da un **quadrato**



- **matrici rettangolari**: nel caso in cui il numero di righe sia diverso al numero di colonne (n ≠ m)
In questo caso la forma teorica che essa assumerà può essere sintetizzata da un **rettangolo**





Le operazioni di base sulle matrici sono le stesse di quelle già esaminate per i vettori (quindi caricamento, visualizzazione, ordinamento, ricerca).
Qui ci limitiamo ad illustrare quelle relative al **CARICAMENTO** ed alla **VISUALIZZAZIONE** di una matrice rettangolare n x m e di una matrice quadrata n x n

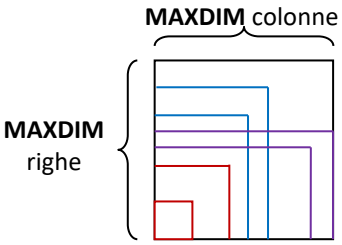
ALGORITMO CaricaVisualizzaMatriceRettangolare

/ N.B. Si usa un unica costante MAXDIM per gestire la FALSA DINAMICITA'anche nel caso degli array bidimensionali (ossia delle matrici). La decisione di allocare uno spazio di memoria "quadrato" non impedisce di utilizzare al suo interno una parte "rettangolare" */*

MAXDIM 10

PROCEDURA main ( )

matr: **ARRAY** [MAXDIM] [MAXDIM] **DI INT**
n, m: INT
i, j: INT



INIZIO

/ leggo il numero di righe e di colonne della matrice rettangolare che si desidera caricare rispettando in entrambi i casi i vincoli imposti da* MAXDIM **/*
RIPETI
  Leggi (n)
FINCHE' (n > 0) AND (n ≤ MAXDIM)

RIPETI
  Leggi (m)
FINCHE' (m > 0) AND (m ≤ MAXDIM) **AND (n ≠ m)** /* Così si escludono le matrici quadrate */

/ carico per riga e per colonna crescenti gli elementi nella matrice rettangolare */*
PER i ← 1 **A** n **ESEGUI**
  PER j ← 1 **A** m **ESEGUI**
    Leggi (matr[i][j])
    j ← j + 1
  FINE PER
  i ← i + 1
FINE PER

/ visualizzo per riga e per colonna crescenti gli elementi della matrice rettangolare */*
PER i ← 1 **A** n **ESEGUI**
  PER j ← 1 **A** m **ESEGUI**
    Scrivi (matr[i][j])
    j ← j + 1
  FINE PER
  i ← i + 1
FINE PER

FINE

ALGORITMO CaricaVisualizzaMatriceQuadrata

MAXDIM 10

PROCEDURA main ( )

matr: **ARRAY** [**MAXDIM**] [**MAXDIM**] **DI INT**
n: INT
i, j: INT

INIZIO

/ leggo il numero di righe e di colonne della matrice quadrata che si desidera caricare rispettando i vincoli imposti da* MAXDIM. N.B. *Non ha senso usare un'altra variabile per le colonne visto che tratiamo una matrice quadrata */*
RIPETI
  Leggi (n)
FINCHE' (n > 0) AND (n ≤ MAXDIM)

/ carico per riga e per colonna crescenti gli elementi nella matrice quadrata */*
PER i ← 1 **A** n **ESEGUI**
  PER j ← 1 **A** n **ESEGUI**
    Leggi (matr[i][j])
    j ← j + 1
  FINE PER
  i ← i + 1
FINE PER

/ visualizzo per riga e per colonna crescenti gli elementi della matrice quadrata */*
PER i ← 1 **A** n **ESEGUI**
  PER j ← 1 **A** n **ESEGUI**
    Scrivi (matr[i][j])
    j ← j + 1
  FINE PER
  i ← i + 1
FINE PER

FINE

MODELLO ASTRATTO (PSEUDOCODIFICA DI UN RECORD)

POSSIBILI RAPPRESENTAZIONI DI UN TRACCIATO RECORD

A) Metodo di rappresentazione tabellare

Numero	Nome Campo	Tipo Campo	Lunghezza[1]	Descrizione	FlagSubRec[2]
1	CodProdotto	ARRAY DI CHAR	6	Codice del Prodotto	
2	NomeProdotto	ARRAY DI CHAR	25	Nome del prodotto	
3	Giacenza	INT	4	Giacenza del prodotto	
4	Prezzo	REAL	3,2	Prezzo unitario del prodotto	

B) Metodo dichiarazione sintattica (PSEUDOCODIFICA)

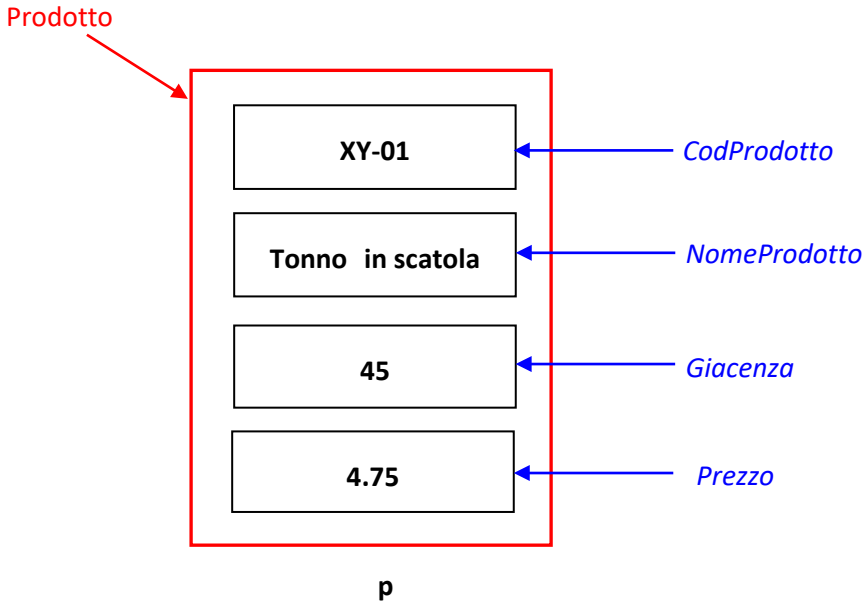
```
.....
TIPO Prodotto = RECORD
                    CodProdotto: ARRAY[6] DI CHAR
                    NomeProdotto: ARRAY[25] DI CHAR
                    Giacenza: INT
                    Prezzo: REAL
                FINE RECORD
.....
p : Prodotto
.....
```

Cosa accade fisicamente.....

Prodotto

CodProdotto	*NomeProdotto*	*Giacenza*	*Prezzo*
XY-01	Tonno in scatola	45	4.75

Cosa accade logicamente (a livello astratto)....



ALGORITMO CaricaVisualizzaAssegnaRecord_Prodotto_NOCONTROL_NOSUBREC

TIPO Prodotto = **RECORD**
CodProdotto : **ARRAY**[6] **DI CHAR**
NomeProdotto : **ARRAY**[25] **DI CHAR**
Giacenza : **INT**
Prezzo :**REAL**
FINE RECORD

} **Ambiente globale**

PROCEDURA main ( )

p1, p2 : **Prodotto**

} **Ambiente locale**

INIZIO

/ leggo il record **p1** campo x campo (operazione NON ATOMICA) SENZA EFFETTUARE tutti i controlli */*
Leggi (p1.CodProdotto)
Leggi (p1.NomeProdotto)
Leggi (p1.Giacenza)
Leggi (p1.Prezzo)

/ visualizzo il record **p1** campo x campo (operazione NON ATOMICA)*/*
Scrivi (p1.CodProdotto)
Scrivi (p1.NomeProdotto)
Scrivi (p1.Giacenza)
Scrivi (p1.Prezzo)

/ Assegnazione tra 2 record (operazione ATOMICA) – in questo caso equivale a 4 assegnazioni */*

p2 ← p1 (*)

/ visualizzo il record **p2** campo x campo (operazione NON ATOMICA)*/*
Scrivi (p2.CodProdotto)
Scrivi (p2.NomeProdotto)
Scrivi (p2.Giacenza)
Scrivi (p2.Prezzo)

FINE

(*)

p2.CodProdotto ← p1.CodProdotto
p2.NomeProdotto ← p1.NomeProdotto
p2.Giacenza ← p1.Giacenza
p2.Prezzo ← p1.Prezzo

Cosa accade se voglio considerare insieme, ad esempio, 5 variabili TUTTE DELLO STESSO TIPO Prodotto ?

Qual è la struttura dati, se esiste, in grado di fare cio'?

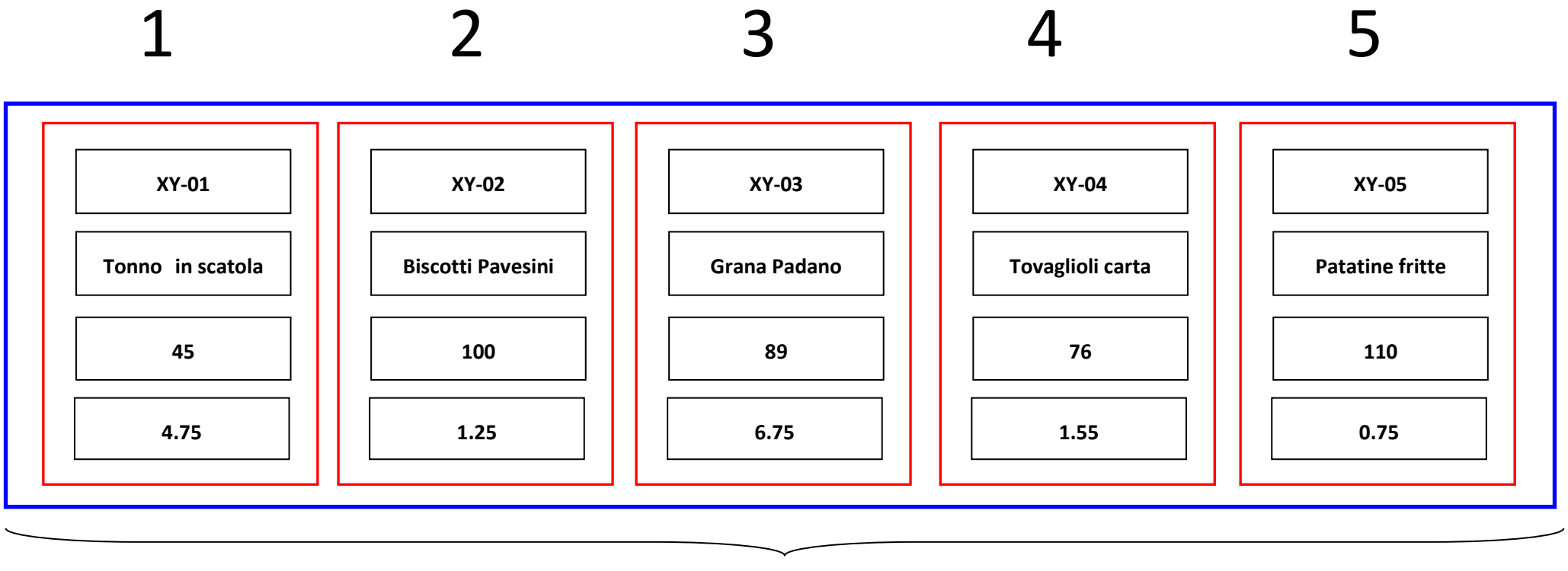

```
.....
TIPO Prodotto = RECORD
                    CodProdotto: ARRAY[6] DI CHAR
                    NomeProdotto: ARRAY[25] DI CHAR
                    Giacenza: INT
                    Prezzo: REAL
                FINE RECORD

.....
mag1, mag2, mag3, mag4, mag5 : Prodotto
.....
```

Tentativo 1

Ho messo così insieme 5 variabili DELLO STESSO TIPO Prodotto

... apparentemente sembra che io abbia dato una valida risposta.

... peccato che queste 5 variabili, seppur dello stesso tipo, siano INDIPENDENTI TRA LORO e NON FACCIANO PARTE DI UNA UNICA STRUTTURA DATI....




```
.....
TIPO Prodotto = RECORD
                    CodProdotto: ARRAY[6] DI CHAR
                    NomeProdotto: ARRAY[25] DI CHAR
                    Giacenza: INT
                    Prezzo: REAL
                FINE RECORD
.....
v : ARRAY DI [5] DI Prodotto
.....
```

Tentativo 2 (quello giusto)

Costruisco un ARRAY MONODIMENSIONALE costituito da 5 RECORD di tipo Prodotto

ARRAY MONODIMESIONALE DI RECORD

ARRAY MONODIMESIONALE DI RECORD (NO sottorecord)

```
.....
TIPO Prodotto = RECORD
                    CodProdotto: ARRAY[6] DI CHAR
                    NomeProdotto: ARRAY[25] DI CHAR
                    Giacenza: INT
                    Prezzo: REAL
                FINE RECORD
.....
v : ARRAY DI [10] DI Prodotto
.....
```

v[3].CodProdotto ← "XYZ"

v.Giacenza ← 10

v[0].Prezzo ← "1200.75"

Leggi(v[4].NomeProdotto)

Scrivi(v[10])

v[1].Prezzo ← v[1].Prezzo + 100

Leggi(v[8].CodProdotto[6])

v[7].NomeProdotto[26] ← '$'

v[3] ← v[5]

Scrivi(v[7].Giacenza)

v[1].Prodotto ← "PIPPO"

v[3].CodProdotto ← v[8].NomeProdotto

v[3].CodProdotto ← v[6].CodProdotto[4]

Assegna all'ottavo carattere del campo "NomeProdotto" del quarto elemento dell'array **v** il valore '@'

Diminuisci di 20 unita il campo "Giacenza" del sesto elemento dell'array **v**

Diminuisci del 10% il prezzo del quinto elemento dell'array **v**

```
ALGORITMO ArrayDiRecord_Prodotto_NOCONTROL_NOSUBREC
MAXDIM 10
TIPO Prodotto = RECORD
                    CodProdotto: ARRAY[6] DI CHAR
                    NomeProdotto: ARRAY[25] DI CHAR
                    Giacenza: INT
                    Prezzo: REAL
                FINE RECORD
PROCEDURA main( )
magazzino: ARRAY[MAXDIM] DI Prodotto
i,n: INT
INIZIO
/* leggo la dimensione del vettore da caricare */
RIPETI
   Leggi(n)
FINCHE' (n ≥ 1) AND (n ≤ MAXDIM)
/* Carico l'array di record valorizzando ciascun campo del record */
PER i ← 1 A n ESEGUI
   Leggi (magazzino[i].CodProdotto)
   Leggi (magazzino[i].NomeProdotto)
   Leggi (magazzino[i].Giacenza)
   Leggi (magazzino[i].Prezzo)
   i ← i + 1
FINE PER

/* Visualizzo l'array di record stampando ciascun campo del record */
PER i ← 1 A n ESEGUI
   Scrivi (magazzino[i].CodProdotto)
   Scrivi (magazzino[i].NomeProdotto)
   Scrivi (magazzino[i].Giacenza)
   Scrivi (magazzino[i].Prezzo)
   i ← i + 1
FINE PER
FINE
```

Ambiente globale

Ambiente locale

```
ALGORITMO ArrayDiRecord_Prodotto_SICONTROL_NOSUBREC
MAXDIM 10
TIPO Prodotto = RECORD
                    CodProdotto: ARRAY[6] DI CHAR
                    NomeProdotto: ARRAY[25] DI CHAR
                    Giacenza: INT
                    Prezzo: REAL
                FINE RECORD
```

Ambiente globale

```
PROCEDURA main( )
magazzino: ARRAY[MAXDIM] DI Prodotto
i,n: INT
```

Ambiente locale

```
INIZIO
/* leggo la dimensione del vettore da caricare (vedi esempi precedente) */
.....
/* Carico l'array di record controllando e valorizzando ciascun campo del record */
PER i ← 1 A n ESEGUI
   RIPETI
      Leggi (magazzino[i].CodProdotto)
   FINCHE' (magazzino[i]. CodProdotto ≠ "") AND (Lunghezza(magazzino [i].CodProdotto) ≤ 6)
   RIPETI
      Leggi (magazzino[i].NomeProdotto)
   FINCHE' (magazzino[i]. NomeProdotto ≠ "") AND (Lunghezza(magazzino [i].NomeProdotto) ≤ 25)
   RIPETI
      Leggi (magazzino[i].Giacenza)
   FINCHE' (magazzino[i].Giacenza ≥ 0) AND (magazzino[i].Giacenza) ≤ 9999)
   RIPETI
      Leggi (magazzino[i].Prezzo)
   FINCHE' (magazzino[i].Prezzo ≥ 0) AND (magazzino[i].Prezzo) ≤ 999.99)
   i ← i + 1
FINE PER

/* Visualizzo l'array di record stampando ciascun campo del record */
PER i ← 1 A n ESEGUI
   Scrivi (magazzino[i].CodProdotto)
   Scrivi (magazzino[i].NomeProdotto)
   Scrivi (magazzino[i].Giacenza)
   Scrivi (magazzino[i].Prezzo)
   i ← i + 1
FINE PER
FINE
```

Uso del SOTTORECORD

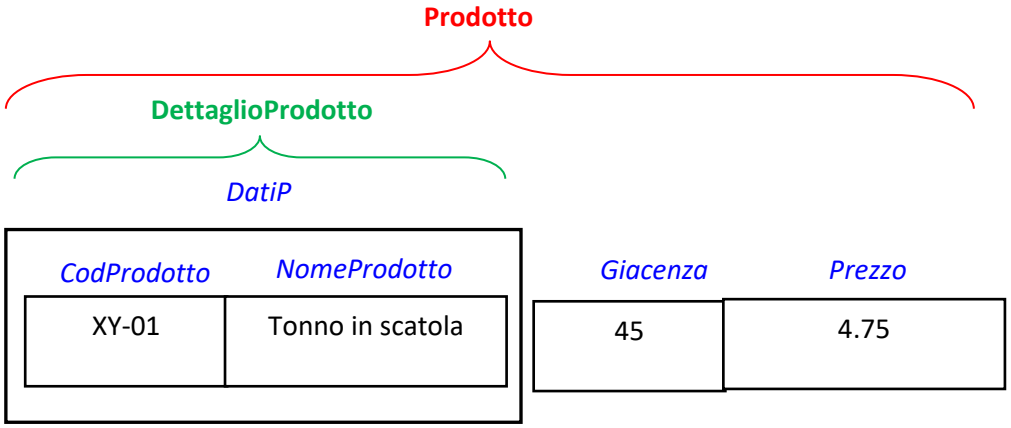
A) Metodo di rappresentazione tabellare

Numero	Nome Campo	Tipo Campo	Lunghezza[1]	Descrizione	FlagSubRec[2]
1	CodProdotto	ARRAY DI CHAR	6	Codice del Prodotto	X
2	NomeProdotto	ARRAY DI CHAR	25	Nome del prodotto	X
3	Giacenza	INT	4	Giacenza del prodotto	
4	Prezzo	REAL	3,2	Prezzo initario del prodotto	

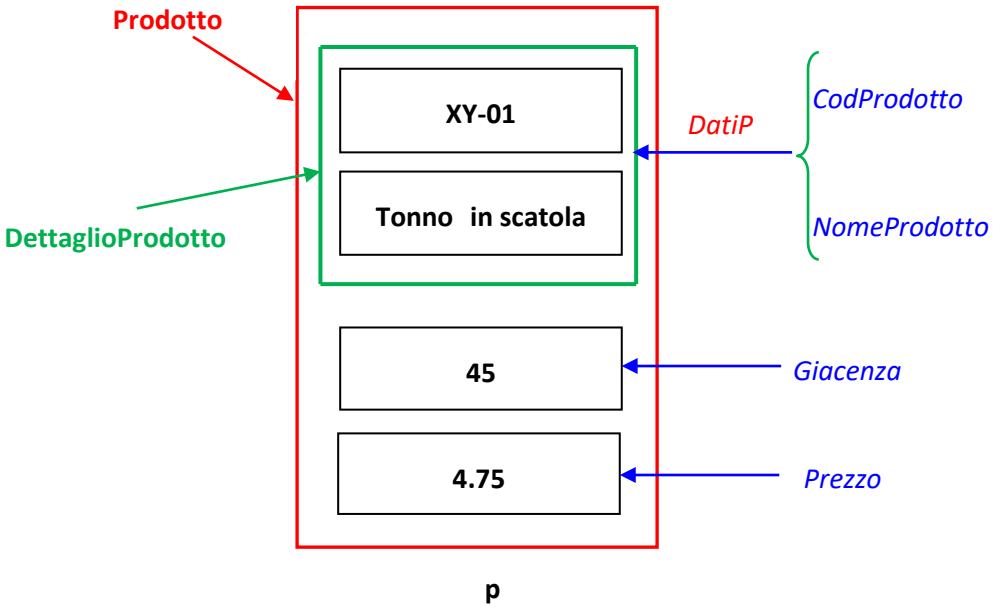
B) Metodo dichiarazione sintattica (PSEUDOCODIFICA)

```
.....
TIPO DettaglioProdotto = RECORD
                            CodProdotto: ARRAY[6] DI CHAR
                            NomeProdotto: ARRAY[25] DI CHAR
                         FINE RECORD
.....
TIPO Prodotto = RECORD
                   DatiP : DettaglioProdotto
                   Giacenza: INT
                   Prezzo: REAL
                FINE RECORD
.....
p : Prodotto
.....
```

Cosa accade fisicamente.....



Cosa accade logicamente (a livello astratto)....



USO DEL SOTTORECORD

ALGORITMO CaricaVisualizzaAssegnaRecord_Prodotto_NOCONTROL_SISUBREC

TIPO DettaglioProdotto = **RECORD**
CodProdotto: ARRAY[6] DI CHAR
NomeProdotto: ARRAY[25] DI CHAR
FINE RECORD

.....

TIPO Prodotto = **RECORD**
DatiP : DettaglioProdotto
Giacenza: **INT**
Prezzo: **REAL**
FINE RECORD

Ambiente globale

PROCEDURA main ( )

p1, p2 : **Prodotto**

Ambiente locale

INIZIO

/ leggo il record **p1** campo x campo (operazione NON ATOMICA) SENZA EFFETTUARE tutti i controlli */*

Leggi (p1.DatiP.CodProdotto)
Leggi (p1.DatiP.NomeProdotto)
Leggi (p1.Giacenza)
Leggi (p1.Prezzo)

/ visualizzo il record **p1** campo x campo (operazione NON ATOMICA)*/*

Scrivi (p1.DatiP.CodProdotto)
Scrivi (p1.DatiP.NomeProdotto)
Scrivi (p1.Giacenza)
Scrivi (p1.Prezzo)

/ Assegnazione tra 2 record (operazione ATOMICA) – in questo caso equivale a 4 assegnazioni */*

p2 ← p1 (*)

/ visualizzo il record **p2** campo x campo (operazione NON ATOMICA)*/*

Scrivi (p2.DatiP.CodProdotto)
Scrivi (p2.DatiP.NomeProdotto)
Scrivi (p2.Giacenza)
Scrivi (p2.Prezzo)

FINE

(*)

p2.DatiP.CodProdotto ← p1.DatiP.CodProdotto
p2.DatiP.NomeProdotto ← p1.DatiP.NomeProdotto
p2.Giacenza ← p1.Giacenza
p2.Prezzo ← p1.Prezzo

ALGORITMO CaricaVisualizzaAssegnaRecord_Prodotto_SICONTROL_SISUBREC

TIPO DettaglioProdotto = **RECORD**
CodProdotto: ARRAY[6] DI CHAR
NomeProdotto: ARRAY[25] DI CHAR
FINE RECORD

.....

TIPO Prodotto = **RECORD**
DatiP : DettaglioProdotto
Giacenza: **INT**
Prezzo: **REAL**
FINE RECORD

Ambiente globale

PROCEDURA main ( )

p1, p2 : **Prodotto**

Ambiente locale

INIZIO

/ leggo il record **p1** campo x campo (operazione NON ATOMICA) SENZA EFFETTUARE tutti i controlli */*

RIPETI
Leggi (p1.DatiP.CodProdotto)
FINCHE' (p1.DatiP.CodProdotto ≠ "") AND (Lunghezza(p1.DatiP.CodProdotto) ≤ 6)

RIPETI
Leggi (p1.DatiP.NomeProdotto)
FINCHE' (p1.DatiP.NomeProdotto ≠ "") AND (Lunghezza(p1.DatiP.NomeProdotto) ≤ 25)

RIPETI
Leggi (p1.Giacenza)
FINCHE' (p1.Giacenza ≥ 0) AND (p1.Giacenza ≤ 9999)

RIPETI
Leggi (p1.Prezzo)
FINCHE' (p1.Prezzo ≥ 0) AND (p1.Prezzo ≤ 999.99)

/ visualizzo il record **p1** campo x campo (operazione NON ATOMICA) */*

Scrivi (p1.DatiP.CodProdotto)
Scrivi (p1.DatiP.NomeProdotto)
Scrivi (p1.Giacenza)
Scrivi (p1.Prezzo)

/ Assegnazione tra 2 record (operazione ATOMICA) – in questo caso equivale a 4 assegnazioni */*

p2 ← p1 (*)

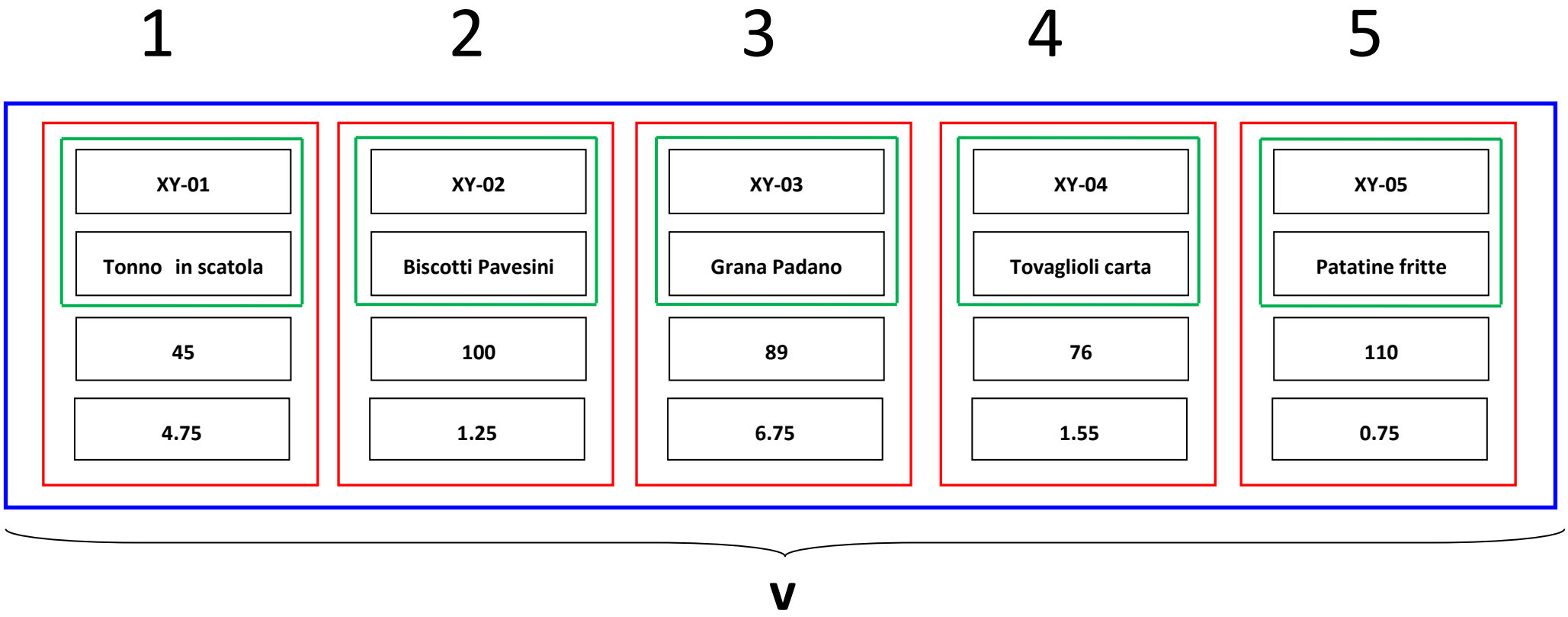
/ visualizzo il record **p2** campo x campo (operazione NON ATOMICA) */*

Scrivi (p2.DatiP.CodProdotto)
Scrivi (p2.DatiP.NomeProdotto)
Scrivi (p2.Giacenza)
Scrivi (p2.Prezzo)

FINE

(*)

p2.DatiP.CodProdotto ← p1.DatiP.CodProdotto
p2.DatiP.NomeProdotto ← p1.DatiP.NomeProdotto
p2.Giacenza ← p1.Giacenza
p2.Prezzo ← p1.Prezzo




```
TIPO DettaglioProdotto = RECORD
                            CodProdotto: ARRAY[6] DI CHAR
                            NomeProdotto: ARRAY[25] DI CHAR
                         FINE RECORD
.....
TIPO Prodotto = RECORD
                    DatiP : DettaglioProdotto
                    Giacenza: INT
                    Prezzo: REAL
                FINE RECORD

v : ARRAY DI [5] DI Prodotto
```

SOLUZIONE

Costruisco un ARRAY MONODIMENSIONALE costituito da 5 RECORD di tipo Prodotto contenenti un campo di tipo DettaglioProdotto

ARRAY MONODIMESIONALE DI RECORD

ARRAY MONODIMESIONALE DI RECORD (SI sottorecord)

```
.....
TIPO DettaglioProdotto = RECORD
                             CodProdotto: ARRAY[6] DI CHAR
                             NomeProdotto: ARRAY[25] DI CHAR
                         FINE RECORD
.....
TIPO Prodotto = RECORD
                    DatiP : DettaglioProdotto
                    Giacenza: INT
                    Prezzo: REAL
                FINE RECORD
.....
v : ARRAY DI [10] DI Prodotto
.....
```

v[3].CodProdotto ← "XYZ"

v[4].DettaglioProdotto ← 10

v[1].Prezzo ← 1200.75

Leggi(v[4].DatiP.NomeProdotto)

Scrivi(v[10].DatiP)

v[1].DatiP.Prezzo ← v[1].DatiP.Prezzo + 100

Leggi(v[8].DatiP.CodProdotto[6])

v[7].NomeProdotto[26] ← '@'

v[8] ← v[4]

Scrivi(v[3].Giacenza)

v[1].DatiP.NomeProdotto ← PIPPO

v[3].DatiP.CodProdotto ← v[0].DatiP.CodProdotto

v[3].CodProdotto[4] ← '$'

Assegna al quarto carattere del campo "CodProdotto" del sesto elemento dell'array **v** il valore '#'

Aumenta di 15 unita il campo "Giacenza" dell'ottavo elemento dell'array **v**

Aumenta del 15% il prezzo del secondo elemento dell'array **v**

```
ALGORITMO ArrayDiRecord_Prodotto_NOCONTROL_SISUBREC
MAXDIM 10
TIPO DettaglioProdotto = RECORD
                             CodProdotto: ARRAY[6] DI CHAR
                             NomeProdotto: ARRAY[25] DI CHAR
                         FINE RECORD
TIPO Prodotto = RECORD
                    DatiP : DettaglioProdotto
                    Giacenza: INT
                    Prezzo: REAL
                FINE RECORD
PROCEDURA main( )
magazzino: ARRAY[MAXDIM] DI Prodotto
i,n: INT
INIZIO
/* leggo la dimensione del vettore da caricare */
RIPETI
   Leggi(n)
FINCHE' (n ≥ 1) AND (n ≤ MAXDIM)
/* Carico l'array di record valorizzando ciascun campo del record */
PER i ← 1 A n ESEGUI
   Leggi (magazzino[i].DatiP.CodProdotto)
   Leggi (magazzino[i].DatiP.NomeProdotto)
   Leggi (magazzino[i].Giacenza)
   Leggi (magazzino[i].Prezzo)
   i ← i + 1
FINE PER

/* Visualizzo l'array di record stampando ciascun campo del record */
PER i ← 1 A n ESEGUI
   Scrivi (magazzino[i].DatiP.CodProdotto)
   Scrivi (magazzino[i].DatiP.NomeProdotto)
   Scrivi (magazzino[i].Giacenza)
   Scrivi (magazzino[i].Prezzo)
   i ← i + 1
FINE PER
FINE
```

Ambiente globale

Ambiente locale

```
ALGORITMO ArrayDiRecord_Prodotto_SICONTROL_SISUBREC
MAXDIM 10
TIPO DettaglioProdotto = RECORD
                            CodProdotto: ARRAY[6] DI CHAR
                            NomeProdotto: ARRAY[25] DI CHAR
                         FINE RECORD
TIPO Prodotto = RECORD
                   DatiP : DettaglioProdotto
                   Giacenza: INT
                   Prezzo: REAL
                FINE RECORD
```

Ambiente globale

```
PROCEDURA main( )
magazzino: ARRAY[MAXDIM] DI Prodotto
i,n: INT
```

Ambiente locale

```
INIZIO
/* leggo la dimensione del vettore da caricare */
RIPETI
   Leggi(n)
FINCHE' (n ≥ 1) AND (n ≤ MAXDIM)
/* Carico l'array di record controllando e valorizzando ciascun campo del record */
PER i ← 1 A n ESEGUI
   RIPETI
      Leggi (magazzino[i].DatiP.CodProdotto)
   FINCHE' (magazzino[i].DatiP.CodProdotto ≠ "") AND (Lunghezza(magazzino[i].DatiP.CodProdotto) ≤ 6)
   RIPETI
      Leggi (magazzino[i].DatiP.NomeProdotto)
   FINCHE' (magazzino[i].DatiP.NomeProdotto ≠ "") AND (Lunghezza(magazzino[i].DatiP.NomeProdotto) ≤ 25)
   RIPETI
      Leggi (magazzino [i].Giacenza)
   FINCHE' (magazzino[i].Giacenza ≥ 0) AND (magazzino[i].Giacenza) ≤ 9999)
   RIPETI
      Leggi (magazzino[i].Prezzo)
   FINCHE' (magazzino[i].Prezzo ≥ 0) AND (magazzino[i].Prezzo) ≤ 999.99)
   i ← i + 1
FINE PER
/* Visualizzo l'array di record stampando ciascun campo del record */
PER i ← 1 A n ESEGUI
   Scrivi (magazzino[i].DatiP.CodProdotto)
   Scrivi (magazzino[i].DatiP.NomeProdotto)
   Scrivi (magazzino[i].Giacenza)
   Scrivi (magazzino[i].Prezzo)
   i ← i + 1
FINE PER
FINE
```