

Tipi strutturati

.... l'informazione può essere decomposta in tipi più semplici

.... più informazioni aggregate fra loro in base ad una relazione per costituire una informazione più complessa

Es:

data (giorno, mese, anno)

numero complesso (parte reale, coefficiente immaginario)

generalità anagrafiche (cognome, nome, data nascita, indirizzo)

Un tipo strutturato è caratterizzato da:

- tipi componenti
- costruttore
- funzione d'accesso agli elementi

COSTRUTTORI DI TIPO

un *tipo strutturato* si ottiene dai tipi componenti attraverso apposite *operazioni*

due costruttori:

- *prodotto cartesiano*
- *sequenza*

Prodotto cartesiano

tipo T = cartesiano (t1, t2, ..., tn)

t1, t2, ..., tn: tipi componenti, non necessariamente tutti uguali

forma *n-ple ordinate* di elementi dei tipi t1, t2, ..., tn

COSTRUTTORI DI TIPO

Prodotto cartesiano

... due modi per definire il tipo strutturato:

- indicando solo i tipi componenti

tipo T = cartesiano (t1, t2, ..., tn)

tipo complesso: cartesiano(reale, reale)

tipo data: cartesiano(intero, intero, intero)

- indicando attributi e tipi componenti

tipo T = cartesiano (a1:t1, a2:t2, ..., an:tn)

tipo complesso: cartesiano(ParteReale:reale, CoeffImm: reale)

tipo data: cartesiano(giorno: intero, mese:intero,anno:intero)

COSTRUTTORI DI TIPO

Sequenza

tipo T = sequenza (t1, t2,, tn)

dove $t1 = t2 = \dots = tn$ tipi componenti tutti uguali

Es.;

parola = sequenza (carattere)

numero decimale = sequenza (cifra decimale)

capitolo = sequenza (paragrafi)

coda = sequenza (persone)

Variabili di Tipo strutturato

Una variabile (informazione) di tipo strutturato va dichiarata, come qualsiasi altra variabile, indicandone il tipo ed il nome

<nome_tipo_strutturato> <nome_variabile>

Es.:

tipo complesso: cartesiano(ParteReale:reale, CoeffImm: reale)
complesso A;

Funzione di Accesso

... per potere accedere ad uno degli elementi costituenti la struttura:

- tramite l'indicazione di un attributo:
tipo T = cartesiano (n1:t1, n2:t2, ..., nk:tk)

indicando con A una informazione del tipo T, con

A.ni

si indica l'accesso alla i-esima componente di T

Es.:

tipo complesso: cartesiano(ParteReale:reale, CoeffImm: reale)
complesso A;
A.ParteReale

Funzione di Accesso

tramite l'indicazione della posizione:

.... Uso di un'informazione *indice* ...

... *indice* informazione di un *tipo ordinato* ...

Detta A una variabile del tipo strutturato:

A[*indice*]

indica la componente che occupa la posizione indicata da indice

Es.:

tipo `GraduatoriaConcorso:cartesiano(persona, persona, ..., persona)`

`GraduatoriaConcorso A;`

`A[3]` indica il componente in terza posizione

Array monodimensionali

Tipo strutturato praticamente presente in tutti i linguaggi come tipo strutturato primitivo

- Costruttore: Cartesiano
- Tipi componenti: tutti dello stesso tipo
- Funzione d'accesso : per posizione

tipo *array monodimensionale* = `cartesiano(T1, T2, ..., Tn)`

con $T_1 = T_2 = \dots = T_n = T$

bisogna:

definire T, il *tipo* dei componenti

definire la *dimensione* (o *cardinalità*) dell'array

Array monodimensionali

... un modello interpretativo ...

1	
2	
3	A[3]
4	
5	
6	

A
Nome array

Accesso diretto ai componenti

Elementi di Informatica
Prof. G. A. Di Lucca - Univ. del Sannio

9

Array monodimensionali

... concettualmente ...

- un nome 'collettivo' associato ad un insieme ordinato di elementi tutti dello stesso tipo
- ciascun elemento è direttamente accessibile attraverso una informazione (l'indice) che ne individua la posizione nella struttura

Es.:

- classifica campionato: array di squadre
- graduatoria concorso: array di persone
- prenotati esame universitario: array di studenti

Elementi di Informatica
Prof. G. A. Di Lucca - Univ. del Sannio

10

Array monodimensionali

Parametri caratterizzanti un array:

- **Strutturali:**

- Nome dello array
- Tipo componente
- Cardinalità: numero di elementi costituenti lo array

Array monodimensionali

Parametri caratterizzanti un array:

- **di utilizzo:**

- **Indice:** appartenete ad un tipo ordinato, indica la posizione di un elemento nello array.

Può essere espresso tramite:

- una costante (es. $A[4]$, quarto elemento dello array)
- una variabile (es. $A[I]$, i -esimo elemento dello array)
- una espressione (es. $A[c+d*e]$,
(elemento dello array nella posizione corrispondente al valore di $c+d*e$)

- **Riempimento:** numero di elementi effettivamente utilizzati
N.B. non confondere Riempimento con Cardinalità

Deve sempre essere :

$\text{Riempimento} \leq \text{Cardinalità}$

$\text{indice} \leq \text{Riempimento} \leq \text{Cardinalità}$

Array monodimensionali

La dichiarazione di una variabile di tipo array è fatta specificando:

- tipo componente
- nome dello array (nome della variabile)
- cardinalità dello array (racchiusa tra parentesi quadre)

`<tipo> <nome_array> [<cardinalità>]`

Es.:

```
int A[10];
```

```
float SpesaSettimanale[7];
```

Array monodimensionali

ad un elemento di un array è possibile assegnare un valore tramite operazioni di:

- lettura

Es.:

```
read("%d", A[3]);
```

legge il valore da assegnare al terzo elemento dello array A

- calcolo e assegnazione

Es.:

```
A[4] = C;
```

```
A[I] = 5;
```

```
A[I] = f * 9 + d;
```

Array monodimensionali

un elemento di un array può essere utilizzato in operazioni di:

- stampa

Es.:

```
write("%d", A[3]);
```

stampa il valore contenuto nel terzo elemento dello array A

- calcolo e assegnazione

Es.:

```
C = A[4];
```

```
B = f * 9 + d - A[I];
```

- condizione

Es.:

```
if (A[I] > 7) .....
```

1	18
2	22
3	5
4	8
5	21
6	6

A

$B = A[2] + A[4]; \Rightarrow B = 22 + 8 = 30$

`printf("%d --- %d\n", A[1], A[5]);`

18 --- 21

$A[1] = 7; \Rightarrow$

1	7
2	22
3	5
4	8
5	21
6	6

A

Array Bidimensionali

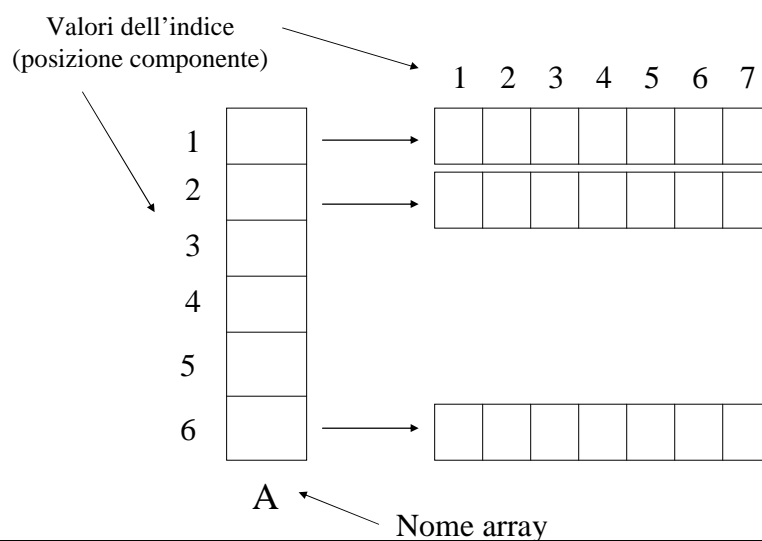
tipo *array Bidimensionale* = cartesiano(T_1, T_2, \dots, T_n)
con $T_1 = T_2 = \dots = T_n = T$

Tipo T = cartesiano (q_1, q_2, \dots, q_m)
 $q_1 = q_2 = \dots = q_m$

.... Ovvero una array bidimensionale è un array in cui
ciascun elemento è a sua volta un array
... array tutti della stessa dimensione e tipo

Array Bidimensionali

... un modello interpretativo ...



Array Bidimensionali

... la posizione di ciascun elemento è individuata da 2 indici ...

	1	2	3	4	5
1	[1,1]	[1,2]	[1,3]	[1,4]	[1,5]
2	[2,1]	[2,2]	[2,3]	[2,4]	[2,5]
3					
4					
5					
6	[6,1]	[6,2]	[6,3]	[6,4]	[6,5]

A[4,2]

... il primo indice, indica la *riga* ...
... il secondo indice, indica la *colonna* ...

Elementi di Informatica
Prof. G. A. Di Lucca - Univ. del Sannio

19

Array Bidimensionali

... la posizione di ciascun elemento è individuata da 2 indici ...

... ciascuna posizione conterrà un solo valore ...

	1	2	3	4	5
1	23	47	63	96	8
2	27	66	23	44	567
3					
4		57			
5					
6	42	28	34	125	47

A[4,2] = 57

A[6,3] = 34

... il primo indice, indica la *riga* ...
... il secondo indice, indica la *colonna* ...

Elementi di Informatica
Prof. G. A. Di Lucca - Univ. del Sannio

20

Array Bidimensionali

Parametri caratterizzanti un array bidimensionale:

- **Strutturali:**
 - Nome dello array
 - Tipo componente
 - Cardinalità di *riga*: numero di righe costituenti lo array
 - Cardinalità di *colonna*: numero di colonne costituenti lo array

Array Bidimensionali

Parametri caratterizzanti un array bidimensionale:

- **di utilizzo:**
 - Indice di *riga*: appartenete ad un tipo ordinato
 - Indice di *colonna*: appartenete ad un tipo ordinato
la coppia ordinata [indice di *riga*, indice di *colonna*] indica la posizione di un elemento nello array bidimensionale.
- Gli indici possono essere espressi tramite:
- una costante (es. $A[4,2]$, elemento in quarta riga e seconda colonna)
 - una variabile (es. $A[I, J]$, elemento in riga i -esima e colonna j -esima)
 - una espressione (es. $A[c+d*e, k]$,
(elemento dello array nella riga corrispondente al valore di $c+d*e$, e k -esima colonna)

Array Bidimensionali

Parametri caratterizzanti un array bidimensionale:

- di utilizzo:
 - Riempimento: numero di elementi effettivamente utilizzati
 - Riempimento di *riga*
 - Riempimento di *colonna*
- N.B. non confondere Riempimento con Cardinalità

Deve sempre essere :

indice *riga* \leq Cardinalità *riga*

indice *colonna* \leq Cardinalità *colonna*

Riempimento *riga* \leq Cardinalità *riga*

Riempimento *colonna* \leq Cardinalità *colonna*

Array Bidimensionali

La dichiarazione di una variabile di tipo array bidimensionale è fatta specificando:

- tipo componente
- nome dello array (nome della variabile)
- cardinalità di *riga* e di *colonna* dello array (racchiusa tra parentesi quadre)

<tipo> <nome_array_bidim> [<cardinalità_riga> [cardinalità_colonna>]

Es.:

```
int A[10] [15];
```

```
float Mat[7] [22];
```

Array Multidimensionali

... generalizzando opportunamente
... un array di array di array di array Quante sono le dimensioni
... tutti array dello stesso tipo ...
... per ciascuna dimensione, tutti array con stessa cardinalità ...
... posizione individuata mediante tanti indici quanti sono le dimensioni
... un indice per ciascuna dimensione ...

<tipo><nome_array_multidim>[<card_dim1> [card_dim2>] ...[card_dimK]

Es.:

```
int A[10] [15] [20];
```

```
float Multi [7] [22] [30] [45];
```

Il tipo stringa di caratteri

- Una stringa di caratteri è una sequenza di caratteri ASCII, atti a rappresentare una parola, una frase, un testo
- E' un tipo strutturato costruito con l'operatore sequenza
- Una delle operazioni tipiche sulle stringhe di caratteri è la concatenazione di due o più stringhe, tipicamente indicata con il simbolo //.

Es.

```
A="calcolatore "    B="elettronico"
```

```
C=A//B="calcolatore elettronico"
```

Il tipo record

- Quando si vogliono rappresentare informazioni che sono collezioni di dati diversi,
 - ad esempio:
 - un libro è una entità composta da un titolo, un autore, un editore, il testo
 - Le generalità anagrafiche di una persona: cognome, nome, data di nascita, luogo di nascita, indirizzo attuale
- Il tipo *record* è usato per rappresentare e gestire queste collezioni di dati,
- Il tipo *record* permette di raggruppare le diverse informazioni, elementari o strutturate, che caratterizzano la collezione di dati.

Il tipo record

- Il *record* è un tipo strutturato formato da un predeterminato numero di *campi*, ciascuno dei quali può essere un tipo semplice o strutturato
- E' un tipo strutturato costruito con l'operatore prodotto cartesiano

$\text{tipo record} = \text{cartesiano}(a1:T1, a2:T2, \dots, an:Tn)$

dove $a1, \dots, an$ sono i nomi dei campi del record
e $T1, \dots, Tn$ i rispettivi tipi

Il tipo record

- dichiarazione in PDL di un tipo record:

```
record <nome record>=  
  begin  
    <nome campo1> :<tipo campo1>;  
    .....  
    <nome campo_n> : <tipo campo_n>;  
  end;
```

Il tipo record

Es.

```
record anagrafico =  
  begin  
    cognome:string;  
    nome:string;  
    data_nascita: record  
      begin  
        giorno:integer;  
        mese: integer;  
        anno:integer;  
      end;  
    indirizzo:string;  
  end;
```

Per indicare un campo del record si usa la notazione:

<nome record>.<nome campo>

Es. anagrafico.nome

Tipo strutturato File

- Tipo strutturato formato da un insieme di elementi tutti dello stesso tipo
- Gli elementi sono registrati su un **supporto di memoria di massa**
- Individuato da un **nome**
- Caratterizzato da un **metodo di accesso** ai suoi elementi (detti genericamente records)
 - File ad accesso **Sequenziale**
 - File ad accesso **Diretto**

Accesso sequenziale

Per accedere all' N -mo record bisogna accedere agli $N-1$ record precedenti (*nastro*)

Accesso diretto

Individuazione ed accesso diretti ad un record tramite:

- o il **riferimento** alla sua posizione
- o i **valori** di uno o piu' campi (*chiavi*)

FILE SEQUENZIALE

- è una *sequenza* di oggetti tutti dello stesso tipo T
- ..ed una ulteriore informazione di tipo T che diremo *buffer* ed indicheremo con $f \uparrow$
- la definizione è completata con le operazioni caratteristiche del tipo

- Operazioni possibili su un file sequenziale:
 - Inserimento di un nuovo elemento (PUT)
 - Lettura degli elementi (GET)
 - Riposizionamento all'inizio del file (RESET)
 - Riscrittura del file (REWRITE)
 - Verifica di fine file (EOF)

operazione **PUT** (f)

- è una operazione di inserzione che modifica lo stato del file
- inserzione del valore del buffer in f; il buffer avrà valore indefinito dopo l'operazione

... prima di PUT(f) ...

f = a b c d

$f \uparrow$ x

... dopo PUT(f) ...

f = a b c d x

$f \uparrow$?

...una osservazione

- in ogni istante la sequenza di oggetti di un file f può essere vista come concatenazione di altre due sequenze

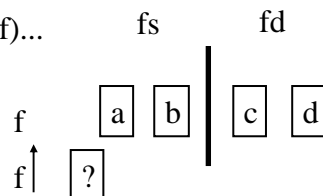
....una parte sinistra f_s ed una parte destra f_d

$$f = \overleftarrow{f_s} / \overrightarrow{f_d}$$

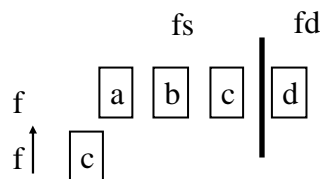
operazione **GET (f)**

non modifica il contenuto del file f , ma quello del buffer con un valore di f ben definito, inoltre..... modifica ...sinistra..e destra..

... prima di GET (f)...

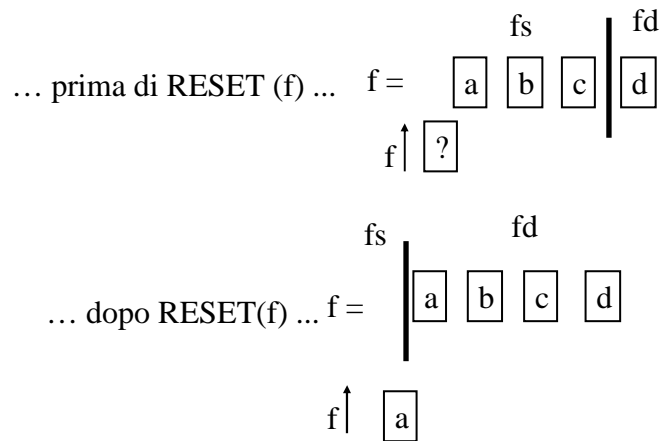


... dopo GET(f) ... Nel buffer sarà presente il primo elemento di f_d



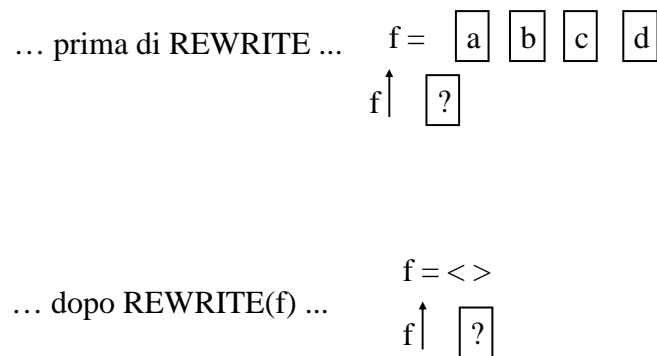
operazione **RESET (f)**

- posiziona il buffer sul primo elemento del file
la parte sinistra risulterà vuota



operazione **REWRITE (f)**

- cancella gli elementi del file svuotandolo



il predicato EOF (f)

E' usato per verificare se si è raggiunta la fine del file

EOF = End Of File

- EOF(f)
 - TRUE se la parte destra della sequenza è vuota
 - FALSE se non lo è

Alcune regole

Dopo un RESET sono possibili solo operazioni di GET

Dopo un REWRITE sono possibili solo operazioni di PUT

Non è possibile mischiare operazioni di PUT e GET

Generazione di un File:

un REWRITE seguito da una o più operazioni di PUT

Ispezione di un FILE:

un RESET seguito da una o più operazioni di GET,
fino alla fine del file