



Programmazione I

Il Linguaggio C

Strutture Dati

Daniel Riccio

Università di Napoli, Federico II

20 ottobre 2021



Sommario



- Argomenti
 - Operazioni elementari sui vettori
 - Definizioni
 - Copia di un vettore
 - Ricerca di un elemento
 - Vettori multidimensionali

I vettori

Caratteristiche statiche

Nome

dato

Tipo di dato base

int

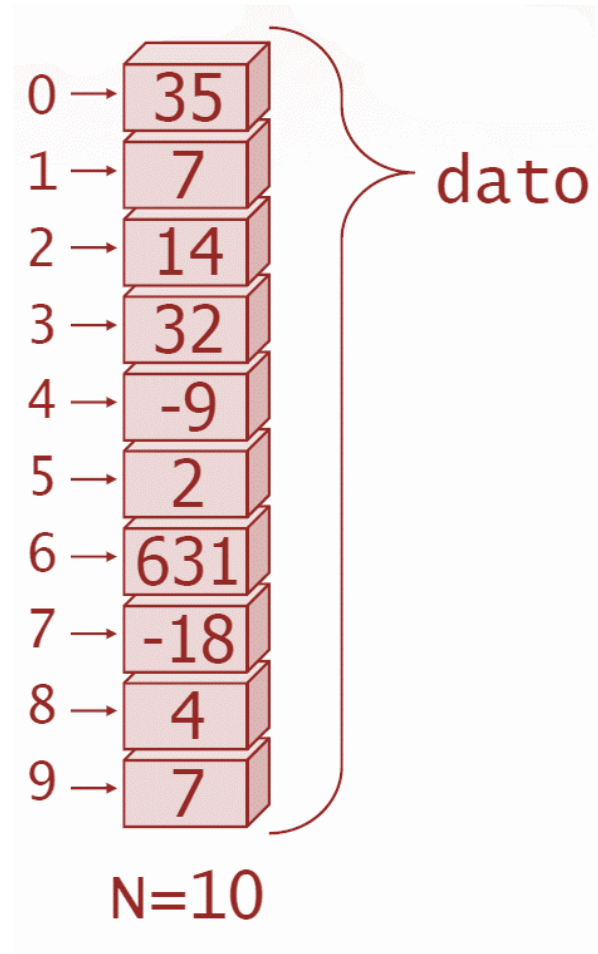
Dimensione totale

10

Caratteristiche dinamiche

Valori assunti dalle singole celle

35, 7, 14, 32, ...



Ricapitolando



Tutte le celle di un vettore avranno lo stesso

nome

Tutte le celle di un vettore devono avere lo

stesso tipo base

La dimensione del vettore è fissa e deve essere determinata al momento della sua definizione

La dimensione è sempre un numero intero

Ogni cella ha sempre un valore

Impossibile avere celle “vuote”

Le celle non inizializzate contengono valori ignoti

Ciascuna cella è identificata dal proprio indice

Gli indici sono sempre numeri interi

In C, gli indici partono da 0

Ogni cella è a tutti gli effetti una variabile il cui tipo è pari al tipo base del vettore

Ogni cella, indipendentemente dalle altre

deve essere inizializzata

può essere letta/stampata

può essere aggiornata da istruzioni di assegnazione

può essere usata in espressioni aritmetiche

Errori frequenti

Non confondere mai l'indice con il contenuto

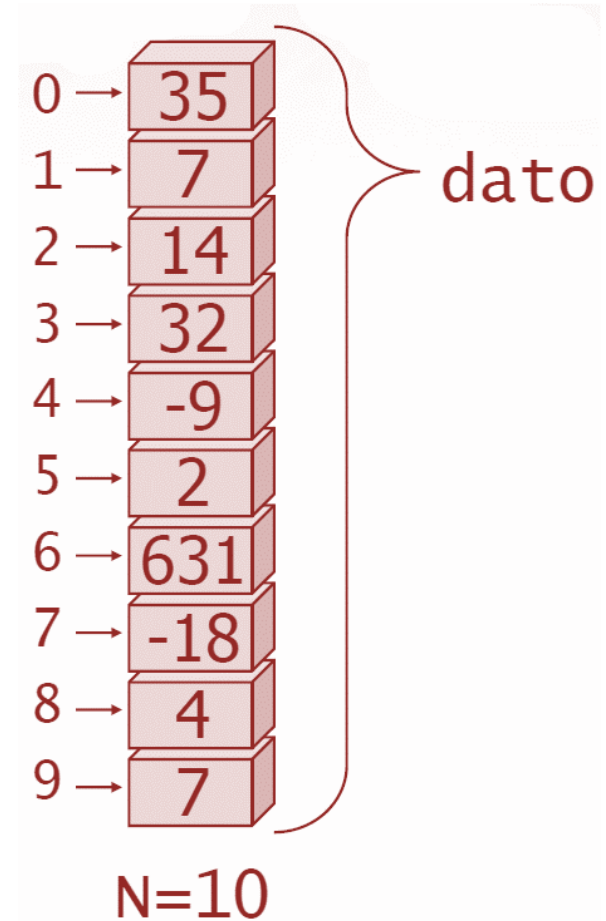
```
dato[5] = 2  
dato[9] == dato[1]  
dato[i] > dato[j]  
i > j
```

Non si può effettuare alcuna operazione sul vettore

```
dato = 0  
printf("%d", dato)
```

Occorre operare sui singoli elementi

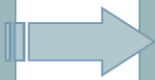
Solitamente all'interno di un ciclo for



Errori frequenti

Dichiarare un vettore usando una variabile anziché una costante


```
int N = 10 ;  
int dato[N] ;
```



```
int dato[10] ;
```

Dichiarare un vettore usando il nome dell'indice

```
int i ;  
int dato[i] ;  
for(i=0; i<10; i++)  
    scanf("%d",&dato[i]) ;
```



```
int dato[10] ;
```



Accesso ai valori di un vettore

Ciascun elemento di un vettore è equivalente ad una singola variabile avente lo stesso tipo base del vettore

È possibile accedere al contenuto di tale variabile utilizzando l'operatore di indicizzazione: []

dato[0]	→	35
dato[1]	→	7
dato[2]	→	14
dato[3]	→	32
dato[4]	→	-9
dato[5]	→	2
dato[6]	→	631
dato[7]	→	-18
dato[8]	→	4
dato[9]	→	7

int dato[10] ;

Esempi:

```
if (dato[i]==0)  
se l'elemento contiene zero
```

```
if (dato[i]==dato[i+1])  
due elementi consecutivi uguali
```

```
dato[i] = dato[i] + 1 ;  
incrementa l'elemento i-esimo
```

```
dato[i] = dato[i+1] ;  
copia un dato dalla cella successiva
```

Definizione

```
#define N 10      /* dimensioni dei vettori */
```

```
int v[N] ;        /* vettore di N interi */
```

```
float r[N] ;      /* vettore di N reali */
```

```
int i, j ;        /* indici dei cicli */
```

```
#define M 100     /* dimensioni dei vettori */
```

```
int w[N] ;        /* vettore di N interi */
```

```
int h[M] ;        /* vettore di M interi */
```

```
int dato ;        /* elemento da ricercare */
```


Stampa di un vettore



Occorre stampare un elemento per volta, all'interno di un ciclo for

Ricordare che

gli indici del vettore variano tra 0 e N-1

gli utenti solitamente contano tra 1 e N

$v[i]$ è l'elemento $(i+1)$ -esimo

Esempio:

```
printf("Vettore di %d interi\n", N);  
  
for( i=0; i<N; i++ ){  
    printf("Elemento %d: ", i+1);  
    printf("%d\n", v[i]);  
}
```

```
g++ Prompt dei comandi  
  
Stampa di un vettore di 10 interi  
Elemento 1: 3  
Elemento 2: 4  
Elemento 3: 7  
Elemento 4: 5  
Elemento 5: 3  
Elemento 6: -1  
Elemento 7: -3  
Elemento 8: 2  
Elemento 9: 7  
Elemento 10: 3
```

Stampa in linea

Occorre stampare un elemento per volta, all'interno di un ciclo for

Ricordare che

- gli indici del vettore variano tra 0 e N-1

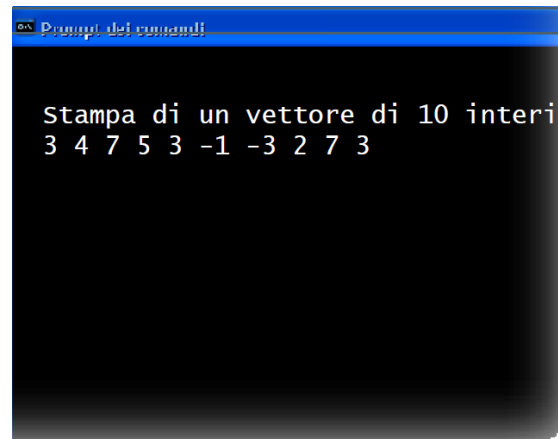
- gli utenti solitamente contano tra 1 e N

- $v[i]$ è l'elemento $(i+1)$ -esimo

Esempio:

```
printf("Vettore di %d interi\n", N);
```

```
for( i=0; i<N; i++ ){  
    printf("%d ", v[i]);  
}  
printf("\n");
```



```
Principi dei calcoli  
Stampa di un vettore di 10 interi  
3 4 7 5 3 -1 -3 2 7 3
```

Copia di un vettore

Si tratta di copiare il contenuto di un vettore in un altro vettore

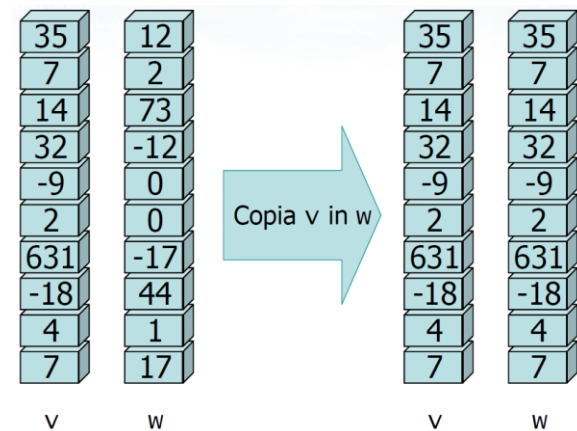
Occorre copiare un elemento per volta dal vettore “sorgente” al vettore “destinazione”, all’interno di un ciclo for

I due vettori devono avere lo stesso numero di elementi, ed essere dello stesso tipo base

Esempio:

```
/* copia il contenuto di v[] in w[] */  
for( i=0; i<N; i++ ) {  
    w[i] = v[i] ;  
}
```

Nonostante siano coinvolti **due** vettori, occorre **un solo ciclo** for, e **un solo indice** per accedere agli elementi di entrambi i vettori



```
w = v ;  
w[] = v[] ;  
w[N] = v[N] ;  
w[1,N] = v[1,N] ;
```

Ricerca di un elemento



Dato un valore numerico, verificare

se almeno uno degli elementi del vettore è uguale al
valore numerico

in caso affermativo, dire dove si trova

in caso negativo, dire che non esiste

Si tratta di una classica istanza del problema di “ricerca di
esistenza”

Se l'array non è ordinato ricerca lineare

Se l'array è ordinato **ricerca binaria** o **dicotomica**

Ricerca di un elemento



Programma

```
main.c
1  #include <stdio.h>
2  #define N 10
3
4  int main ()
5  {
6      int dato;          /* dato da ricercare */
7      int trovato;       /* flag per ricerca */
8      int pos;           /* posizione elemento */
9
10     int v[N] = {-18, -9, 2, 4, 7, 8, 14, 32, 35, 631};
11
12     printf ("Elemento da ricercare? ");
13     scanf ("%d", &dato);
14
15     trovato = 0;
16     pos = -1;
17     for (int i = 0; i < N; i++){
18         if (v[i] == dato){
19             trovato = 1;
20             pos = i;
21         }
22     }
23
24     if (trovato == 1){
25         printf ("Elemento trovato alla posizione %d\n", pos + 1);
26     } else {
27         printf ("Elemento non trovato\n");
28     }
29
30     return 0;
31 }
```



Esecuzione



```
Elemento da ricercare? 8
Elemento trovato alla posizione 6
```

Ricerca binaria



Sapendo che il vettore è **ordinato** (esiste una relazione d'ordine totale sul dominio degli elementi), la ricerca può essere ottimizzata

– Vettore ordinato in senso **non decrescente**:

se $i < j$ si ha $V[i] \leq V[j]$

2	3	5	5	7	8	10	11
---	---	---	---	---	---	----	----

– Vettore ordinato in senso **crescente**:

se $i < j$ si ha $V[i] < V[j]$

2	3	5	6	7	8	10	11
---	---	---	---	---	---	----	----

In modo analogo si definiscono l'ordinamento in senso **non crescente** e **decrescente**

Ricerca binaria

Definizione:

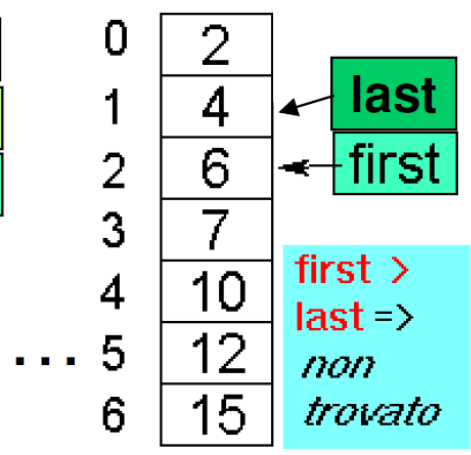
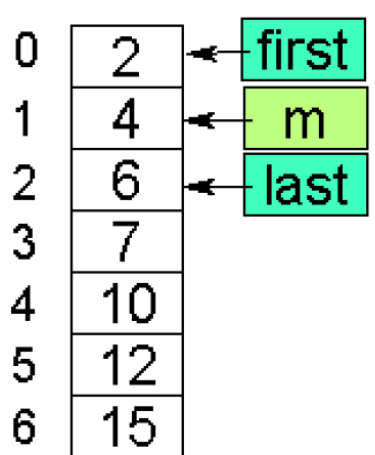
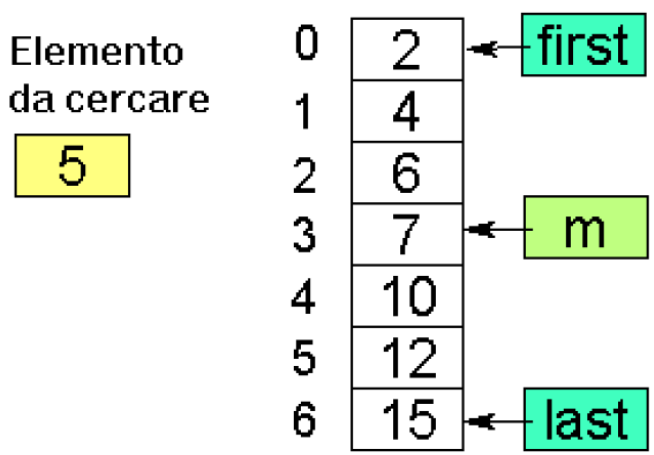
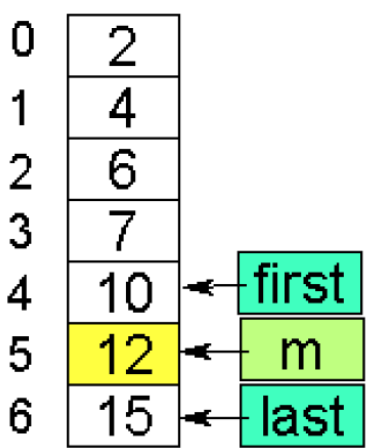
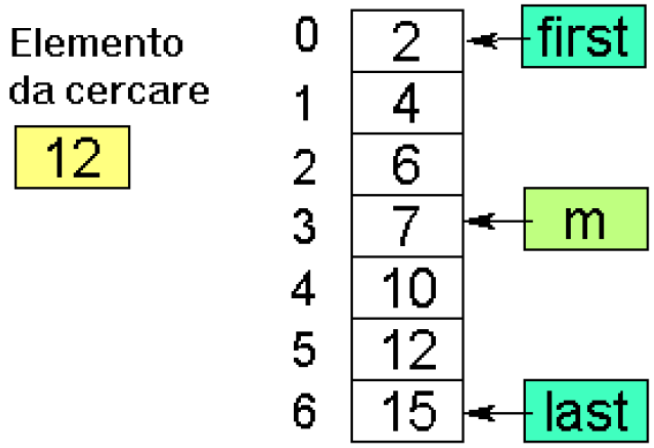
Sia **dim** la dimensione dell'array

- Se l'elemento mediano (posizione **med**) dell'array è l'elemento da cercare **elemento trovato**
- Se l'elemento mediano dell'array è maggiore dell'elemento da cercare **cercare nella prima metà dell'array** (dalla posizione "0" alla posizione **med-1**)
- Se l'elemento mediano dell'array è minore dell'elemento da cercare **cercare nella seconda metà dell'array** (dalla posizione **med+1** alla posizione "finale")



Ricerca binaria

Esempio:



Ricerca binaria



Ricerca binaria di un elemento in un vettore ordinato in senso non decrescente in cui il primo elemento è **Primo** e l'ultimo **Ultimo**

La tecnica di **ricerca binaria**, rispetto alla ricerca esaustiva, consente di **eliminare ad ogni passo metà degli elementi del vettore**

Si confronta l'elemento cercato `val` con quello mediano del vettore, `V[Medio]`

- Se `val==V[Medio]`, fine della ricerca (**Trovato=true**)
- Altrimenti, se il vettore ha almeno una componente (**Primo<=Ultimo**):
 - se `val<V[Medio]`, ripeti la ricerca nella prima metà del vettore (indici da **Primo** a **Medio-1**)
 - se `val>V[Medio]`, ripeti la ricerca nella seconda metà del vettore (indici da **Medio+1** a **Ultimo**)

Ricerca binaria



Il vantaggio di usare un algoritmo di ricerca binaria è che il numero di elementi tra cui cercare viene diviso per 2 ogni volta che si ripete il ciclo **while**. Così,

- la prima volta che si esegue il ciclo si deve cercare fra n elementi;
- la seconda volta $n/2$ elementi sono stati eliminati e ne rimangono solo $n/2$;
- la terza volta si elimina un'altra metà degli elementi, e ne rimangono $n/4$, così via.

In generale, dopo p ripetizioni del ciclo, il numero di elementi che rimangono da confrontare è $n / (2^p)$. Nel caso peggiore, la ricerca continua finché gli elementi che rimangono da confrontare siano ≤ 1 .

Matematicamente ciò si esprime dicendo che

$$n / (2^p) \leq 1$$

o, in modo alternativo, che p è l'intero più piccolo tale che $2^p \geq n$.

$$p \leq \log_2(n)$$

Ricerca di un elemento



Programma

```
main.c
1  #include <stdio.h>
2  #define N 10
3
4  int main ()
5  {
6      int v[N] = {-18, -9, 2, 4, 7, 8, 14, 32, 35, 631};
7
8      int val;
9      int Trovato=0;
10     int Primo=0;
11     int Ultimo=N-1;
12     int Medio;
13
14     printf ("Valore da cercare: ");
15     scanf ("%d",&val);
16
17     while ((Primo<=Ultimo) && (Trovato==0)){
18
19         Medio = (Primo+Ultimo)/2;
20
21         if (val==v[Medio])
22             Trovato = 1;
23         else if(val<v[Medio])
24             Ultimo = Medio - 1;
25         else
26             Primo = Medio + 1;
27     }
28
29     if(Trovato)
30         printf("Elemento trovato in posizione %d\n", Medio+1);
31     else
32         printf("Elemento non trovato\n");
33
34     return 0;
35 }
```



Esecuzione



Valore da cercare: 8

Elemento trovato in posizione 6

Vettori multidimensionali



Il concetto di vettore come collezione di elementi consecutivi, può essere esteso immaginando che gli elementi siano a loro volta dei vettori: si ottiene così un vettore multidimensionale o **matrice**

La definizione di **matrice** ricalca pienamente quella del vettore:

tipo_comp **nome** [**dim1**] [**dim2**].....;

tipo_comp può essere un qualunque tipo semplice,

dim1, **dim2**, ecc.; racchiusi tra parentesi quadre, definiscono il numero di elementi di ogni dimensione.

Vettori multidimensionali



Esempio:

matrice bidimensionale di numeri interi formata da **3** righe e **5** colonne:

```
int a[3][5];
```

a

a[0][0]	a[1][0]	a[2][0]	a[3][0]	a[4][0]	a[0]
a[0][1]	a[1][1]	a[2][1]	a[3][1]	a[4][1]	a[1]
a[0][2]	a[1][2]	a[2][2]	a[3][2]	a[4][2]	a[2]

Vettori multidimensionali

Accesso ad un elemento:

<nome vettore> [**<posizione1>**] [**<posizione2>**].....

Per esempio

matrix [10][20][15]

individua l'elemento di coordinate rispettivamente **10**, **20** e **15** nella matrice a **3** dimensioni **matrix**

L'inizializzazione di un vettore multidimensionale, deve essere effettuata per righe:

```
int vett[3][2] = { {8,1},      /* vett[0]  */
                  {1,9},      /* vett[1]  */
                  {0,3}        /* vett[2]  */
                };
```

Vettori multidimensionali

Per un vettore a più dimensioni, la scansione va applicata a tutte le dimensioni: in questo caso si devono utilizzare **cicli annidati**

Esempio:

elaborazione degli elementi di un vettore bidimensionale

```
int vett [3][5];  
...  
for (i = 0; i < 3; i++) {           /* per ogni riga */  
    for (j = 0; j < 5; j++) {       /* per ogni colonna */  
        ... elaborazione su vett[i][j]  
    }  
}
```



1. Sottosequenza comune

Due colleghi intendono fissare una riunione, pertanto devono identificare dei giorni nei quali sono entrambi liberi da impegni. A tale scopo, essi realizzano un programma C che permetta a ciascuno di immettere le proprie disponibilità, e che identifichi i giorni nei quali entrambi sono liberi

