

Array

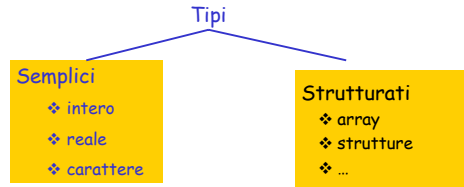
Antonella Santone

Anno Accademico 2008/2009

1

Tipi di dati

- ❖ VALORI: un insieme dei valori del tipo
- ❖ OPERAZIONI: per operare su tali valori



2

Array

Trattare un insieme omogeneo di dati

Invece di usare tante variabili dello stesso tipo



ARRAY

Variabile strutturata dove è possibile memorizzare più valori dello stesso tipo

3

Tipi di array

Monodimensionali
(vettore)

Multidimensionali



4

Vettore

Programmi fanno spesso uso di dati omogenei

Esempio: elaborare voti

```
int voto0, voto1, voto2, voto3;
```

Se il numero di voti è grande, l'uso di identificatori unici diventa pesante

→ **vettore**

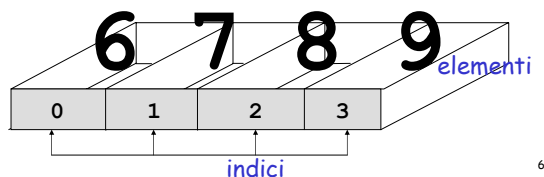
5

Rappresentazione intuitiva di un vettore

Contenitore suddiviso in tanti scomparti quanti sono i dati da memorizzare

Ogni scomparto, detto **elemento**, contiene un unico dato ed è individuato da un numero progressivo detto **indice**, che specifica la posizione dell'elemento all'interno del vettore

vettore di numeri



6

Dichiarazione di un array

```
tipo NomeVariabile[dimensione_costante];
```

tipo

è il tipo degli elementi del vettore

Lunghezza del vettore

dimensione_costante

è una costante che indica quanti elementi deve contenere l'array (intero positivo)

7

Dichiarazione di un array (cont.)

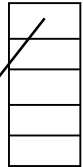
La dichiarazione serve al compilatore per riservare in memoria spazio sufficiente al vettore. Lo spazio occupato dal vettore sarà:

```
numero_byte = sizeof(tipo)*dimensione_costante ;
```

```
int a[5];  
...  
printf("dim. di a %d", sizeof(a));
```

20

int occupano 4 byte



8

Rappresentazione interna

Il C alloca gli elementi di un array in posizioni adiacenti, associando l'indirizzo più basso al primo elemento, e il più alto all'ultimo.

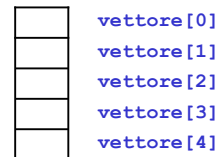
E' molto importante conoscere come sono stati memorizzati gli elementi poiché questo consente di capire come sia possibile puntare ad un preciso elemento.

9

Esempi

```
int vettore[5]; // un vettore di 5 interi
```

Il C indicizza gli elementi a partire da 0 fino a 5-1



10

Consiglio

E' buona pratica di programmazione utilizzare una costante simbolica per definire la lunghezza di un array

```
#define N 5  
int vettore[N]; // un vettore di 5 interi
```

11

Inizializzazione di un array

All'atto della dichiarazione è possibile inizializzare un array:

```
int vet[5] = {0,1,2,3,4};
```

E' possibile non specificare la dimensione dell'array:

```
int vet[] = {0,1,2,3,4}
```

12

Costanti

Dichiarare costanti

```
const int vet[5] = {0,1,2,3,4}
```

13

Accedere ad un elemento

La funzione di accesso ad un elemento è del tipo

nomearray[espressione]

```
#include<stdio.h>
main()
{
  int vet[] = {10,11,12,13,14};
  printf ( "%d\n",vet[1] );
  printf ( "%d\n",vet[2+1] );
}
```

14

Accedere ad un elemento (cont.)

```
int vet[3];
```

```
vet[0] = 3;
```

```
vet[1] = vet[0]*2;
```

3	vet[0]
6	vet[1]
	vet[2]

15

Elaborazione di un array

L'usuale costrutto per elaborare gli elementi di un vettore è il ciclo `for`

Esempio

```
#define N 5
...
int v[N];
for (i=0; i<N; i++)
  scanf("%d", &v[i]);
```

Utente inizializza il vettore

16

Operazioni sugli array

Agli elementi sono applicabili tutte le operazioni previste per il tipo componente

17

NON posso fare

```
int v1[3]={10,11,12};
int v2[3]={0,1,2};
int v3[3];
```

Copiare v1 in v2, cioè: v2

10	11	12
----	----	----

~~v2=v1;~~

Mettere in v3 la somma degli elementi di v1 e v2,

cioè: v3

10	12	14
----	----	----

~~v3=v1+v2;~~

18

Per copiare

Per copiare gli elementi da un array all'altro bisogna copiare singolarmente ogni elemento:

```
for (i=0; i<3; i++)  
    v2[i]=v1[i];
```

~~v2 = v1~~



19

Per sommare

```
for (i=0; i<3; i++)  
    v3[i]=v1[i]+v2[i];
```

~~v3 = v1+v2~~



20

```
int i, voti[6], max, min, somma;
```

Dichiara le variabili intere
i, max, min, somma

e la variabile array di tipo intero
voti

21

Problema dello sconfinamento

Il linguaggio C non effettua controlli per verificare che non si acceda a posizioni fuori dal vettore.

E' possibile (ma è un errore gravissimo anche se comune) accedere ad una locazione di memoria fuori dal vettore, con vari effetti possibili

22

Problema dello sconfinamento: accesso in lettura

Può accadere, a seconda del sistema operativo:

- ❖ nessun problema apparente, solo errore logico
- ❖ segmentation fault

Esempio

```
int a[3]={0,1,2};  
printf("%d", a[3]);
```

illegale



23

Problema dello sconfinamento: accesso in scrittura

Danni maggiori. Oltre ai due problemi già citati nel lucido precedente, si sovrascriverà una locazione di memoria che potrebbe contenere dati essenziali al programma o peggio ancora al sistema operativo.

Per esempio, si potrebbe modificare il valore delle variabili dichiarate immediatamente sopra o sotto al vettore.

24

Esempio di errore di sconfinamento

```
#include<stdio.h>
main()
{int i;
 int vet[10];
 int pippo = 0;
 for ( i=-1; i<10; i++) vet[i] = 99;
 printf("pippo = %d\n", pippo);
}
```

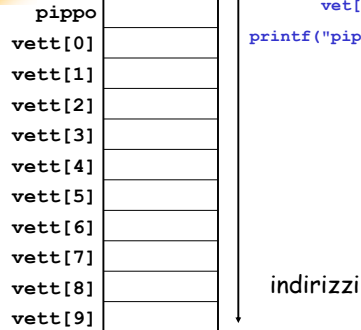
Quanto vale pippo?

99

25

Motivazione

```
int vet[10];
int pippo = 0;
for ( i=-1; i<10; i++)
    vet[i] = 99;
printf("pippo = %d\n", pippo);
```



26

Riassumendo

```
int v[5];
```

Legale

```
x = v[4];
v[1] = 2;
```

Illegale

```
x = v[5];
x = v[6];
v[7] = 9;
```

Attenzione:
indice da 0 a 4

27

Esempi di uso di

array

28

Riempire un array dall'esterno

```
#include<stdio.h>
#define N 3
main()
{
    int a[N],i;
    for (i=0; i<N; i++)
    {
        printf("dammi l'el. %d-esimo di a\n", i);
        scanf("%d", &a[i]);
    }
}
```

29

Cardinalità e riempimento

- ❖ In C è necessario dichiarare la dimensione dell'array (affinché il compilatore possa allocare la memoria necessaria)
- ❖ Per generalizzare i programmi ad array di dimensione variabile bisogna definire una dimensione massima **MAX_ELEM** e di volta in volta utilizzare un numero di elementi **n** compreso tra 1 e **MAX_ELEM**
- ❖ Valore **MAX_ELEM** detto **cardinalità dell'array**
- ❖ Valore **n** detto **riempimento dell'array**

30

Esempio: riempio un array di dimensione massima 100 con n elementi

```
#include<stdio.h>
#define MAX_ELEM 100
main()
{
    int a[MAX_ELEM],n,i;
    printf("Numero elementi che vuoi inserire\n");
    scanf("%d", &n);
    for (i=0; i<n; i++)
    {
        printf("dammi l'el. %d-esimo di a\n", i);
        scanf("%d", &a[i]);
    }
}
```

Controllare che: $n \leq 100$

Stampare tutti gli elementi di un vettore

```
for (i=0; i<n; i++)
    printf("%d\n", a[i]);
```

32

```
#include<stdio.h>
#define MAX_ELEM 100
main()
```

Programma completo

```
{
    int a[MAX_ELEM],n,i;
    printf("Numero elementi che vuoi inserire\n");
    scanf("%d", &n);
    for (i=0; i<n; i++)
    {
        printf("dammi l'el. %d-esimo di a\n", i);
        scanf("%d", &a[i]);
    }
    for (i=0; i<n; i++)
        printf("%d\n", a[i]);
}
```

leggo

stampo

Visita degli elementi di un array

Visita totale:
vengono analizzati tutti gli elementi

Usare un ciclo

Uscita?

Dopo che tutti gli elementi sono stati visitati

Visita finalizzata:

la visita termina quando un elemento dell'array verifica una certa condizione

Usare un ciclo

Uscita?

2 condizioni:
- una sull'indice;
- l'altra che dipende dal problema specifico ... vedremo poi un esempio

34

Ricerca del massimo elemento

Visita totale

3	vet[0]
6	vet[1]
27	vet[2]
4	vet[3]
7	vet[4]

35

Ricerca del massimo elemento (cont.)

```
int a[N];
...
max=a[0];
for (i=1; i<N; i++)
    if (a[i] > max) max=a[i];
```

36

Programma completo

```
#include<stdio.h>
#define N 3
main()
{
    int a[N],i, max;
    // inizializzo l'array
    for (i=0; i<N; i++)
        scanf("%d", &a[i]);
    max=a[0];
    for (i=1; i<N; i++)
        if (a[i] > max) max=a[i];
    printf("Massimo = %d", max);
}
```

37

Esempio

Stabilire se un array contiene 100

Visita finalizzata

3
44
55
100
7

SI

3
45
27
4
7

NO

38

Condizione di uscita

Visito l'array ed esco quando:

- ho trovato 100

oppure

- ho visto tutti gli elementi dell'array e 100 non c'era

39

Cosa mi serve?

Variabile boelana

trovato

Inizialmente posta a zero

40

Attenzione alla condizione di uscita

```
trovato = 0;
```

```
i=0;
```

```
while ( i<N && !trovato)
```

```
{
    trovato = (a[i]==100);
    i++;
}
```

Finché $i < N$ e non ho trovato 100: vado avanti

Esco quando:

$i \geq N \rightarrow$ non c'era 100

oppure

$trovato=1 \rightarrow$ c'era 100

41

Programma completo

```
#include<stdio.h>
#define N 3
main()
{ int a[N],i, trovato;
  // inizializzo l'array
  for (i=0; i<N; i++)
      scanf("%d", &a[i]);
  trovato=0;
  i=0;
  while (i<N && !trovato)
  {
      trovato = (a[i]==100);
      i++;
  }
  if (trovato) printf("SI");
  else printf("NO");
}
```

Se avessi scritto ...

```
trovato=0;
i=0;
while (!trovato)
{
    trovato = (a[i]==100);
    i++;
}
```

Sbagliato!!!

3
45
27
4
7

43

E se avessi scritto ...

```
trovato=0;
i=0;
while (i<N)
{
    trovato = (a[i]==100);
    i++;
}
```

Visito inutilmente gli altri elementi

100
45
27
4
7

Sbagliato perché

trovato = (a[N-1]==100);

44

E se avessi scritto ...

```
trovato=0;
i=0;
while (i<N)
{
    if (a[i]==100) trovato = 1;
    i++;
}
```

100
45
27
4
7

Corretto, ma visito inutilmente gli altri elementi

45

Esercizio di analisi

46

```
#include<stdio.h>
main()
{ int i;
  int a[5], b[5];
  for (i=0; i<5; i++)
  {
    printf("dammi l'el. %d-esimo di a\n", i);
    scanf("%d", &a[i]);
    b[4-i]=a[i];
  }
  printf("stampo a:\n");
  for (i=0; i<5; i++) printf("%d ", a[i]);
  printf("\n");
  printf("stampo b:\n");
  for (i=0; i<5; i++) printf("%d ", b[i]);
}
```

Cosa stampa?

```
for (i=0; i<5; i++)
{
    printf("dammi l'el. %d-esimo di a\n", i);
    scanf("%d", &a[i]);
    b[4-i]=a[i];
}
```

a

1
2
3
4
5

b

5
4
3
2
1

48

```

Santone@PC ~
$ ./a.exe
dammi l'el. 0-esimo di a
1
dammi l'el. 1-esimo di a
2
dammi l'el. 2-esimo di a
3
dammi l'el. 3-esimo di a
4
dammi l'el. 4-esimo di a
5
stampa a:
1 2 3 4 5
stampa b:
5 4 3 2 1
Santone@PC ~
$

```

Esercizi

... sugli array

Esercizi

link nella home page

[esercizi array](#)

Esercizio

Scrivere un programma che, inizializzato un vettore a di lunghezza N con valori interi, stampa tutti gli elementi in posizione pari.

Esempio

```

a = [3, 22, 44, 55, 7, 0]
3
44
7

```

Uso ciclo for

```

for (i=0; i<N; i=i+2)
    printf("l'el. %d-esimo di a: %d\n", i, a[i]);

```

Programma completo

```

#include<stdio.h>
#define N 5
main()
{ int i;
  int a[N];
  // inializzo il vettore leggendo gli
  // elementi dall'esterno
  for (i=0; i<N; i++)
  {
    printf("dammi l'el. %d-esimo di a\n", i);
    scanf("%d", &a[i]);
  }
  // stampa tutti gli elementi in posizione pari
  for (i=0; i<N; i=i+2)
    printf("l'el. %d-esimo di a: %d\n", i, a[i]);
}

```

Esercizio

Scrivere un programma che visualizza **SI** se, dato un vettore **a**, esiste un elemento in posizione **i** che ha valore **i**, cioè **a[i]==i**

Esempio

a = [3,1,22] SI perché a[1]==1
a = [3,22,44] NO

55

Visita finalizzata

```
trovato=0;
i=0;
while (i<N && !trovato)
{trovato = (a[i]==i);
  i++;
}
```

56

Programma completo

```
#include<stdio.h>
#define N 5
main()
{int i, trovato;
 int a[N];
 // inializzo il vettore leggendo gli elementi
 // dall'esterno
 for (i=0; i<N; i++)
 {
  printf("dammi l'el. %d-esimo di a\n", i);
  scanf("%d", &a[i]);
 }
 // cerco un i tale che a[i]=i
 trovato=0;
 i=0;
 while (i<N && !trovato)
 {trovato = (a[i]==i);
  i++;
 }
 if (trovato) printf("SI");
 else printf("NO");
}
```

58

Esercizio

Dato un vettore **voti** contenente il punteggio raggiunto da **N** studenti, determinare il maggiore, il minor e la media

Esempio

Se **N = 6** e
voti = [23, 18, 24, 27, 18, 29]
→
maggiore → 29
minore → 18
media → 23.16

Programma

```
...
int i, max, min;
float media;
int voti[N];
// calcolo max, min, media;
max = voti[0];
min = voti[0];
media = voti[0];
for (i=1; i<N; i++)
{ if (voti[i] > max) max=voti[i];
  if (voti[i] < min) min=voti[i];
  media = media + voti[i];
}
media = media / N;
printf("Maggiore = %d, Minore = %d, Media = %f",
max, min, media);
...
```

Esercizio

Un vettore di interi non negativi rappresenta un passaggio minato: gli elementi di valore 0 rappresentano zone minate, gli elementi di valore positivo **n** indicano salti in avanti di **n** passi. Ad esempio, il vettore:

vett = [2,0,4,1,0]

ha come zone minate gli elementi **vet[1]** e **vet[4]**; l'elemento **vet[0]** indica un salto di 2 passi, **vet[2]** rappresenta un salto di 4 passi, etc..

Partendo dall'elemento di indice 0 (**vet[0]**), con un salto di due passi, si arriva in **vet[2]**, da qui, con uno salto di 4 passi si esce dal passaggio (si esce dall'array). Scrivere un programma che, dato in input un vettore di interi non negativi **vett** e un indice di partenza **ind**, stampa:

- **NOK**: se a partire da **ind** si arriva in una zona minata;
- **OK**: se a partire da **ind** si esce dal passaggio, senza aver incontrato una zona minata.

Esempio:

```
vett = [2,0,4,1,0]      e ind = 0 →    OK
vett = [2,0,4,1,0]      e ind = 3 →    NOK
```

```
#include<stdio.h>
#define N 5
main()
{
  int vet[N],i,ind, uscito;
  // riempimento del vettore
  for (i=0; i<N; i++)
  {
    printf("dammi l'el. %d-esimo di vet\n", i);
    scanf("%d", &vet[i]);
  }
  printf("dammi l'indirizzo iniziale\n");
  scanf("%d", &ind);
  i = ind, uscito = 0;
  while (!uscito && vet[i]!=0) {
    i += vet[i];
    if (i >= N) uscito = 1;
  }
  if (uscito) printf("OK\n");
  else printf("NOK\n");
}
```

Soluzione

Esercizio

Scrivere un programma che ricevuto in input un array di interi, restituisce in output l'array in cui i numeri dispari precedono i numeri pari. Non usare nessun array d'appoggio.

```
#include<stdio.h>
#define N 3
main()
{
  int vett[N], i, j, temp;
  for (i=0; i<N; i++){
    printf("dammi elemento %d-esimo: ", i);
    scanf("%d", &vett[i]);
  }
  i=0; j=N-1;
  while (i<j){
    while (i<N && vett[i]%2==1) i++;
    while (j>=0 && vett[j]%2==0) j--;
    if (i<j){
      temp=vett[i];
      vett[i]=vett[j];
      vett[j]=temp;
      i++; j--;
    }
  }
  for (i=0; i<N; i++) printf("%d\n", vett[i]);
}
```

Soluzione

Esercizio

Scrivere un programma che, ricevuto in input un array v , elimina da v tutti i pari. I valori rimanenti sono spostati nelle prime posizioni e le ultime posizioni vengono riempite con degli zeri. Non è possibile usare un vettore d'appoggio.

Esempio

$v = [35, 18, 2, 17, 4, 1, 3]$
 \rightarrow
 $v = [35, 17, 1, 3, 0, 0, 0]$

E' corretto?

```
...
for(i = 0; i < N; i++)
  if (vett[i] % 2 == 0){
    for (j = i; j < N-1; j++)
      vett[j]=vett[j+1];
    vett[N-1] = 0;
  }
}
```

NO! Se vett è: 3 2 4 \rightarrow 3 4 0

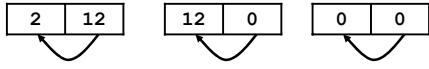
Programma

```
...
i=0;
lung=N;
while (i < lung)
{
  if (v[i]%2 == 0)
  {for (j=i; j<lung-1; j++) v[j]= v[j+1];
   v[lung-1] = 0;
   lung = lung-1;
  }
  else i++;
}
```

i non va incrementato qui !!!

**..se avessi
scritto**

```
...  
i=0;  
while (i < N)  
{  
  if (v[i]%2 == 0)  
    {for (j=i; j<N-1; j++) v[j]= v[j+1];  
    v[N-1] = 0;  
    }  
  else i++;  
}
```



loop