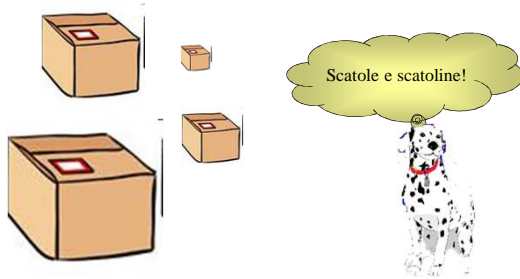


Vettori

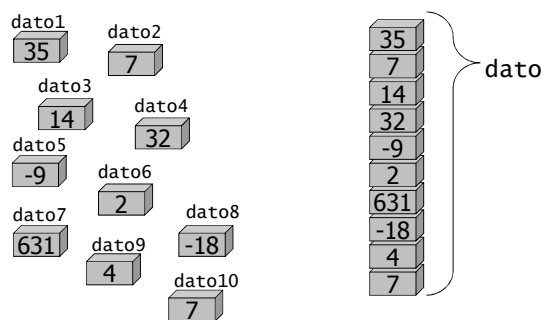


Vettori



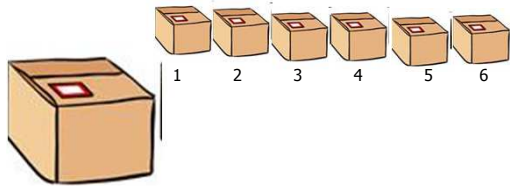
2

Variabili e vettori



3

Vettori

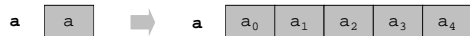


Nome della scatola (esempio: vettore)
Nome della scatola (vettore[3])

4

Vettori

- Insiemi di variabili *dello stesso tipo* aggregate in un'unica entità
 - Identificate globalmente da un nome
 - Singole variabili (*elementi*) individuate da un *indice*, corrispondente alla loro posizione rispetto al primo elemento
 - L'indice degli elementi parte da 0
 - Gli elementi di un vettore sono memorizzati in celle di memoria contigue!



5

Dichiarazione di un vettore

- Sintassi:
`<tipo> <nome vettore> [<dimensione>];`
- Accesso ad un elemento:
`<nome vettore> [<posizione>]`
- Esempio:

```
int v[10];
```

 - Definisce un insieme di 10 variabili intere
`v[0], v[1], v[2], v[3], v[4], v[5], v[6], v[7], v[8], v[9]`

6

Da evitare...

```
int main(void)
{
    int dato1, dato2, dato3, dato4, dato5 ;
    int dato6, dato7, dato8, dato9, dato10 ;
    scanf("%d", &dato1) ;
    scanf("%d", &dato2) ;
    scanf("%d", &dato3) ;
    scanf("%d", &dato10) ;

    printf("%d\n", dato10) ;
    printf("%d\n", dato9) ;
    printf("%d\n", dato8) ;
    printf("%d\n", dato1) ;
}
```

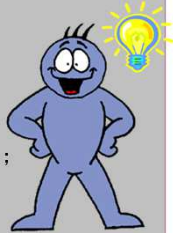


7

...così è meglio!

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

    for( i=9; i>=0; i--)
        printf("%d\n", dato[i]) ;
}
```



8

Definizione di vettori in C

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

Tipo di dato
base

Nome del
vettore

Numero di
elementi

9

indici

- Il generico elemento di un vettore si indica come:

`dato[i]`

- Con `i` indice che inizia da 0 (`dato[0]` è il primo elemento)
- `i` è una variabile intera
- Esempi:
`i=3;`
`A=dato[i]` (è uguale a `A=dato[3]`)

10

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**: `[]`

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

`int dato[10];`

11

11

Sintassi

`nomevettore[valoreindice]`

Come nella
dichiarazione

Costante, variabile o
espressione aritmetica
con valore intero

Valore intero compreso
tra 0 e (dimensione del
vettore - 1)

12

12

Vettori e indici

- L'indice che definisce la posizione di un elemento di un vettore DEVE essere intero!
 - Non necessariamente costante
 - Può essere un'espressione complessa (purché intera)

- Esempi:

```
double a[100]; /* a vettore di double */
double x;
int i, j, k;
... ..
x = a[2*i+j-k]; /* è corretto! */
```

13

Vincoli

- Il valore dell'indice deve essere compreso tra 0 e N-1. La responsabilità è del programmatore
- Se l'indice non è un numero intero, viene automaticamente troncato
- Il nome di un vettore può essere utilizzato solamente con l'operatore di indicizzazione

14

14

Uso di un elemento di un vettore

- L'elemento di un vettore è utilizzabile come una qualsiasi variabile:
 - utilizzabile all'interno di un'espressione
 - tot = tot + dato[i] ;
 - utilizzabile in istruzioni di assegnazione
 - dato[0] = 0 ;
 - utilizzabile per stampare il valore
 - printf("%d\n", dato[k]) ;
 - utilizzabile per leggere un valore
 - scanf("%d\n", &dato[k]) ;

15

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

16

16

Vettori e cicli

- I cicli sono particolarmente utili per "scandire" un vettore
- Utilizzo tipico: Applicazione iterativa di un'operazione sugli elementi di un vettore
- Schema:

```
...  
int data[10];  
for (i=0; i<10; i++)  
{  
    // operazione su data[i]  
}  
...
```

17

Problemi

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

Devono essere
tutte uguali.
Chi lo
garantisce?

18

18

Problemi

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

Se volessi lavorare con 20 dati dovrei modificare in tutti questi punti.

19

19

Costanti

- La dimensione di un vettore deve essere specificata utilizzando una costante intera positiva
 - **Costante** = valore numerico già noto al momento della compilazione del programma

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

20

20

Soluzione

- Per risolvere i problemi visti si può ricorrere alle **costanti simboliche**
 - Associamo un nome simbolico ad una costante
 - Nel programma usiamo sempre il nome simbolico
 - Il compilatore si occuperà di sostituire, ad ogni occorrenza del nome, il valore numerico della costante

21

21

Costanti simboliche

- Costrutto **#define**
 - Metodo originario, in tutte le versioni del C
 - Usa una sintassi particolare, diversa da quella del C
 - Definisce costanti valide su tutto il file
 - Non specifica il tipo della costante
- Modificatore **const**
 - Metodo più moderno, nelle versioni recenti del C
 - Usa la stessa sintassi di definizione delle variabili
 - Specifica il tipo della costante

22

22

Costrutto #define

```
#define N 10
```

```
int main(void)
```

```
{
```

```
    int dato[N] ;
```

```
    . . .
```

```
}
```

Definizione
della
costante

Uso della
costante

23

23

Particolarità (1/2)

- La definizione non è terminata dal segno ;
- Tra il nome della costante ed il suo valore vi è solo uno spazio (non il segno =)
- Le istruzioni **#define** devono essere una per riga
- Solitamente le **#define** vengono poste subito dopo le **#include**

```
#define N 10
```

24

24

Particolarità (2/2)

- Non è possibile avere una `#define` ed una variabile con lo stesso nome
- Per convenzione, le costanti sono indicate da nomi TUTTI_MAIUSCOLI

```
#define N 10
```

25

25

Esempio

```
#define MAX 10

int main(void)
{
    int i ;
    int dato[MAX] ;

    . . .

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

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

26

26

Modificatore const

```
int main(void)
{
    const int N = 10 ;
    int dato[N] ;
    . . .
}
```

Definizione
della
costante

Uso della
costante

27

27

Sintassi

- Stessa sintassi per dichiarare una variabile
- Parola chiave `const`
- Valore della costante specificato dal segno `=`
- Definizione terminata da segno `;`
- Necessario specificare il tipo (es. `int`)
- Il valore di `N` non si può più cambiare

```
const int N = 10 ;
```

28

28

Esempio

```
int main(void)
{
    const int MAX = 10 ;
    int i ;
    int dato[MAX] ;

    . . .

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

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

29

29



Suggerimenti

- Utilizzare **sempre** il costrutto `const`
- Permette maggiori controlli da parte del compilatore
- Gli eventuali messaggi d'errore sono più chiari
- Aggiungere **sempre** un commento per indicare lo scopo della variabile
- Utilizzare la convenzione di assegnare nomi `TUTTI_MAIUSCOLI` alle costanti

30


30



Errore frequente

- Dichiarare un vettore usando una variabile anziché una costante

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



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

31


31



Errore frequente

- Dichiarare un vettore usando una variabile non ancora inizializzata

```
int N ;  
int dato[N] ;  
...  
scanf("%d",&N) ;
```



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

32


32



Errore frequente

- 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] ;
```

33

33

Inizializzazione di un vettore

- E' possibile assegnare un valore iniziale ad un vettore (solo) *al momento della sua dichiarazione*
- Equivalente ad assegnare OGNI elemento del vettore
- Sintassi (vettore di N elementi):
`{<valore 0>, <valore 1>, ... ,<valore N-1>};`
- Esempio:

```
int lista[4] = {2, 0, -1, 5};
```
- NOTA: Se vengono specificati meno di N elementi, l'inizializzazione assegna a partire dal primo valore. I successivi vengono posti a zero.
 - Esempio:

```
int s[4] = {2, 0, -1};  
/* s[0]=2, s[1]=0, s[2]=-1, s[3]=0 */
```

34

Lettura vettore di interi

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

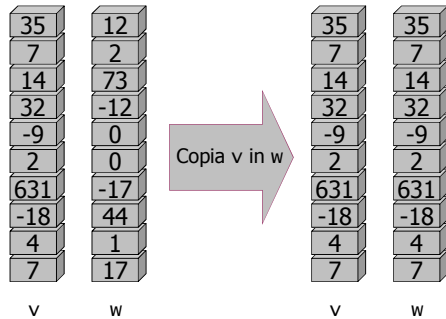
35

Stampa vettore di interi

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

36

Copia di un vettore



37

Copia di un vettore

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

38

Esercizio 1

- Leggere 10 valori interi da tastiera, memorizzarli in un vettore e calcolarne il minimo ed il massimo
- Analisi:
 - Il calcolo del minimo e del massimo richiedono la scansione dell'intero vettore
 - Il generico elemento viene confrontato con il minimo corrente ed il massimo corrente
 - Se minore del minimo, aggiornare il minimo
 - Se maggiore del massimo, aggiornare il massimo
 - Importante l'inizializzazione del minimo/massimo corrente!

39

Esercizio 1: Soluzione

```
#include <stdio.h>
main()
{
    int v[10];
    int i, max, min;

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

    /* uso il primo elemento per inizializzare min e max */
    max = v[0];
    min = v[0];

    for (i=1; i<10; i++) {
        if (v[i] > max)
            max = v[i];
        if (v[i] < min)
            min = v[i];
    }
    printf("Il massimo e': %3d\n", max);
    printf("Il minimo e' : %3d\n", min);
}
```

40

Esercizio 2

- Scrivere un programma che legga un valore decimale minore di 1000 e lo converta nella corrispondente codifica binaria

- Analisi:

- Usiamo l'algoritmo di conversione binaria visto a lezione
 - Divisioni successive per 2
 - Si memorizzano i resti nella posizione del vettore di peso corrispondente
 - La cifra meno significativa è l'ultima posizione del vettore!
- Essenziale determinare la dimensione massima del vettore
 - Per codificare un numero < 1000 servono 10 bit ($2^{10}=1024$)

41

Esercizio 2: Soluzione

```
#include <stdio.h>
main()
{
    int v[10] = {0};
    int i=9; /* ultima posizione del vettore */
    int N;

    printf("Inserire un numero positivo (<1000): ");
    scanf("%d", &N);

    if (N > 1000)
        printf("Errore: il numero deve essere < 1000\n");
    else {
        while(N != 0) {
            v[i] = (N % 2); /* resto della divisione per 2 */
            N = N/2; /* divido N per 2 */
            i--;
        }
        for (i=0; i<10; i++)
            printf("%d", v[(10-1)-i]); /* stampa in ordine inverso */
        printf("\n");
    }
}
```

42

Fine Capitolo