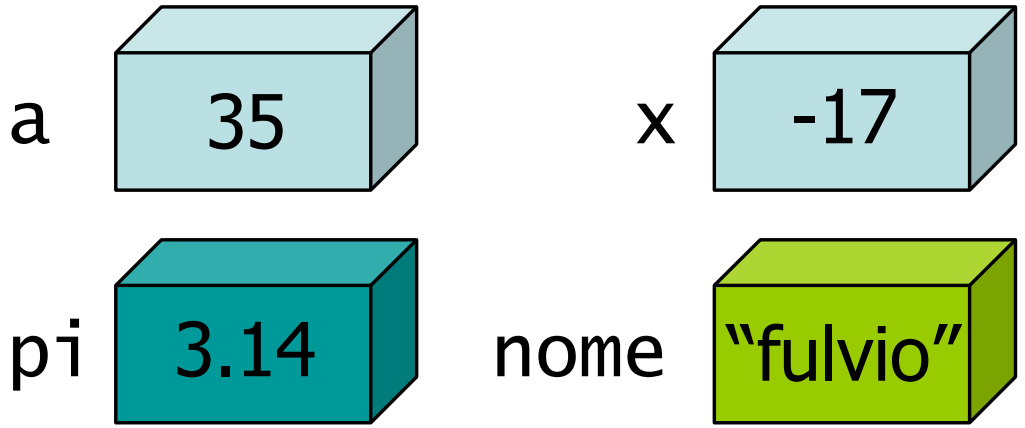
Elementi Di Informatica E Programmazione

Prof. Andrea Loreggia

UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

Tipi di dato strutturati

- Finora abbiamo utilizzato dei tipi di dato semplici
  - `int`, `float`
  - Ogni variabile può contenere un solo valore
- Il linguaggio C permette di definire tipi di dato complessi, aggregando più variabili semplici

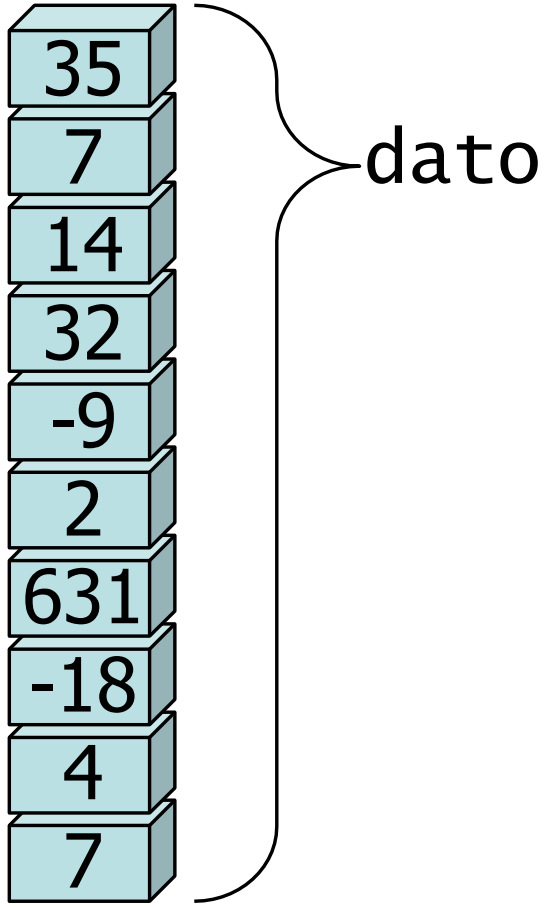


Esigenze

- Leggere da tastiera 10 numeri, e stamparli in ordine inverso
- Calcolare e stampare la tavola pitagorica
- Calcolare l'area del triangolo date le coordinate dei vertici
- Per ogni auto calcolare il tempo medio e minimo sul giro

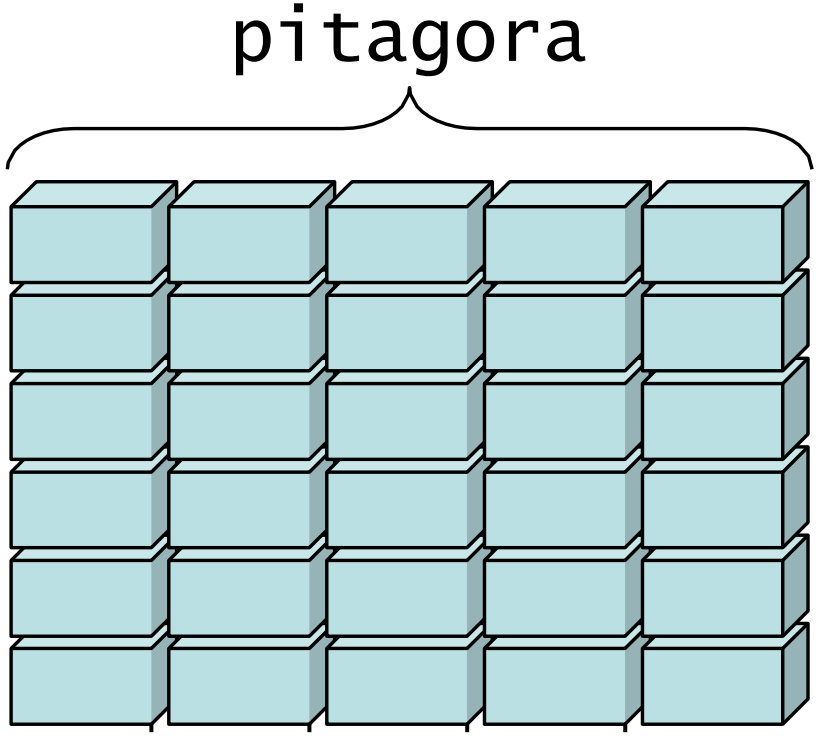
Esigenze

- Leggere da tastiera 10 numeri, e stamparli in ordine inverso
- Calcolare e stampare la tavola pitagorica
- Calcolare l'area del triangolo date le coordinate dei vertici
- Per ogni auto calcolare il tempo medio e minimo sul giro



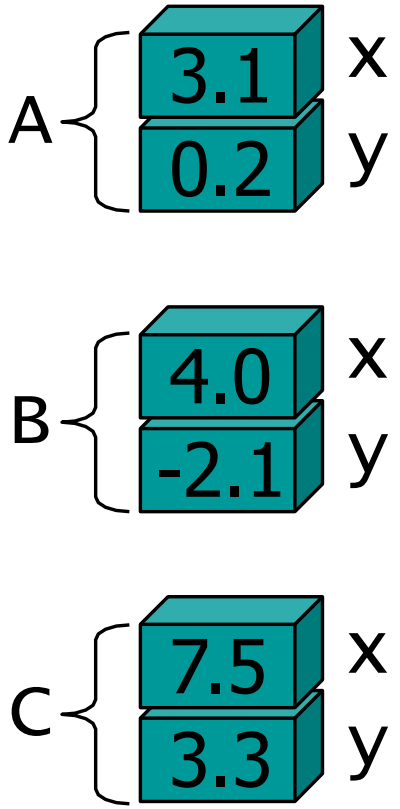
Esigenze

- Leggere da tastiera 10 numeri, e stamparli in ordine inverso
- Calcolare e stampare la tavola pitagorica
- Calcolare l'area del triangolo date le coordinate dei vertici
- Per ogni auto calcolare il tempo medio e minimo sul giro



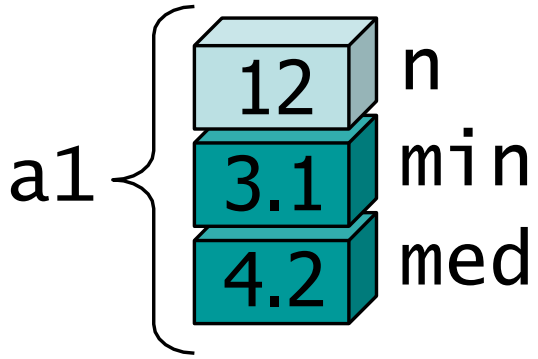
Esigenze

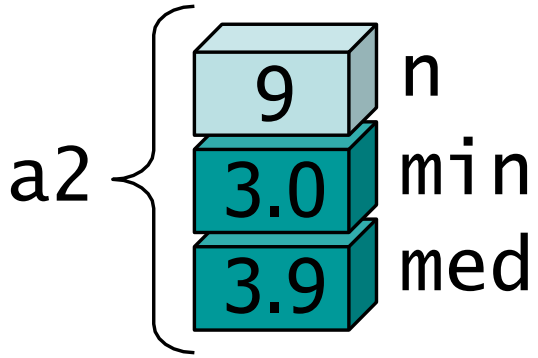
- Leggere da tastiera 10 numeri, e stamparli in ordine inverso
- Calcolare e stampare la tavola pitagorica
- Calcolare l'area del triangolo date le coordinate dei vertici
- Per ogni auto calcolare il tempo medio e minimo sul giro



Esigenze

- Leggere da tastiera 10 numeri, e stamparli in ordine inverso
- Calcolare e stampare la tavola pitagorica
- Calcolare l'area del triangolo date le coordinate dei vertici
- Per ogni auto calcolare il tempo medio e minimo sul giro





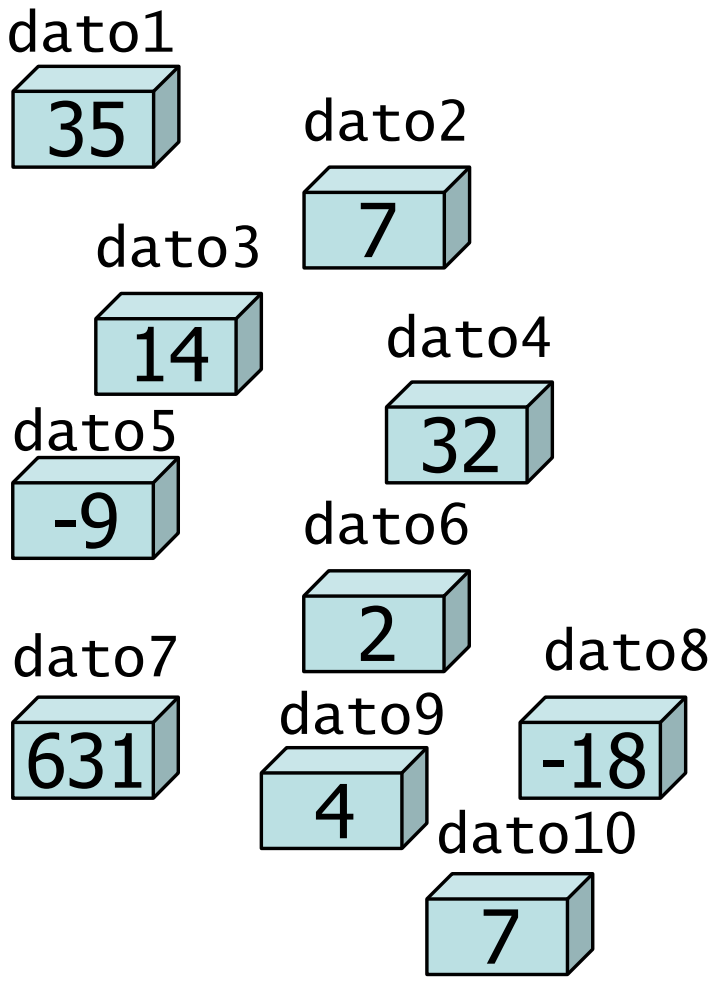
Dati strutturati (1/2)

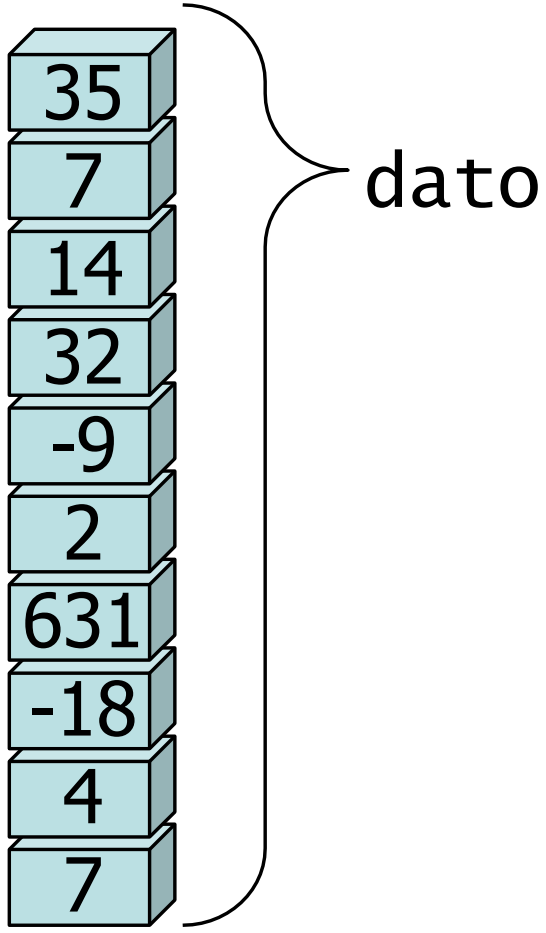
- Raggruppare più variabili semplici in un'unica struttura complessa
- Diversi meccanismi di aggregazione
  - Vettore
  - Matrice
  - Struttura

Dati strutturati (2/2)

- Una sola variabile, di tipo complesso, memorizza `tutti` i dati della struttura complessa
  - `Vettore_di_int dato`
  - `Matrice_di_int pitagora`
  - `Struttura_xy A, B, C`
  - `Struttura_auto a1, a2`

Variabili e vettori





Da evitare...

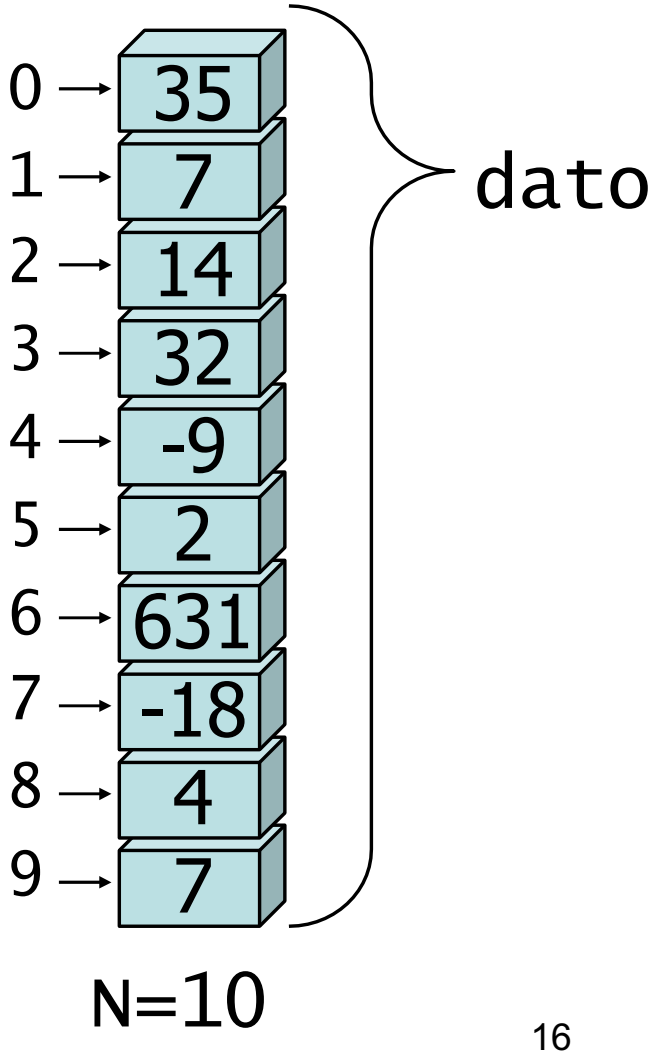

```
int main(void)
{
   int dato1, dato2, dato3, dato4, dato5 ;
   int dato6, dato7, dato8, dato9, dato10 ;
   . . . . .
   scanf("%d", &dato1) ;
   scanf("%d", &dato2) ;
   scanf("%d", &dato3) ;
   . . .
   scanf("%d", &dato10) ;

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

Vettori

- Meccanismo di strutturazione più semplice
- Utilizzato per aggregare serie di dati
- Permette di memorizzare tutti i dati acquisiti o calcolati, e potervi accedere
  - In qualsiasi momento
  - In qualsiasi ordine
- I singoli dati sono distinti dal loro numero d'ordine
  - Primo, secondo, ..., ultimo dato
  - Ciascun dato può avere un valore diverso

Vettori

- Sequenza lineare di dati elementari
- Elementi tutti dello stesso tipo
- Numero di elementi fisso (N)
- Elementi identificati dal proprio indice
  - da 0 a N-1



...così è meglio!

UNIVERSITÀ DEGLI STUDI DI BRESCIA

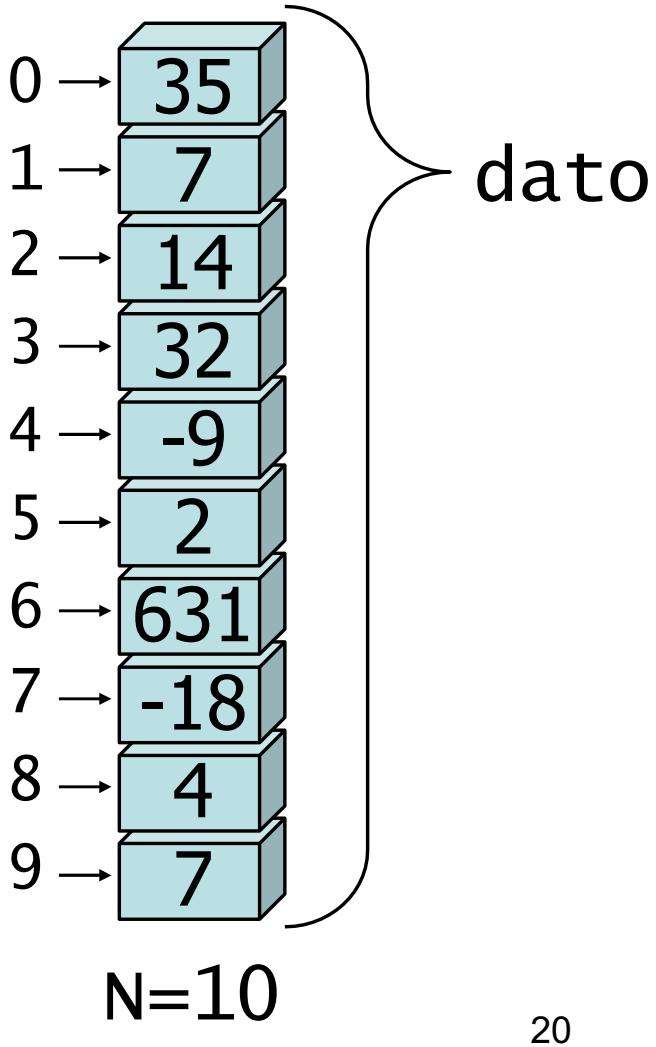
```
int main(void)
{
    int dato[10] ;
    . . . . .
    for( i=0; i<10; i++)
       scanf("%d", &dato[i]) ;

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

Insieme o separati?

- I singoli dati vengono tenuti insieme in un'unica variabile di tipo vettore
  - `int dato[10]`
- Le operazioni (lettura, scrittura, elaborazione) avvengono però sempre un dato alla volta
  - `scanf("%d", &dato[i])`
  - `printf("%d\n", dato[i])`
- Ogni singolo dato è identificato da
  - Nome del vettore
  - Posizione all'interno del vettore

Caratteristiche dei vettori

- Caratteristiche statiche
  - Nome
    - dato
  - Tipo di dato base
    - int
  - Dimensione totale
    - 10
- Caratteristiche dinamiche
  - Valori assunti dalle singole celle
    - 35, 7, 14, 32, …



Esempio

- Leggere da tastiera 10 numeri e stamparli in ordine inverso
  - Tipo base: int
  - Dimensione: 10
  - Lettura delle celle da 0 a 9
  - Stampa delle celle da 9 a 0

Vincoli

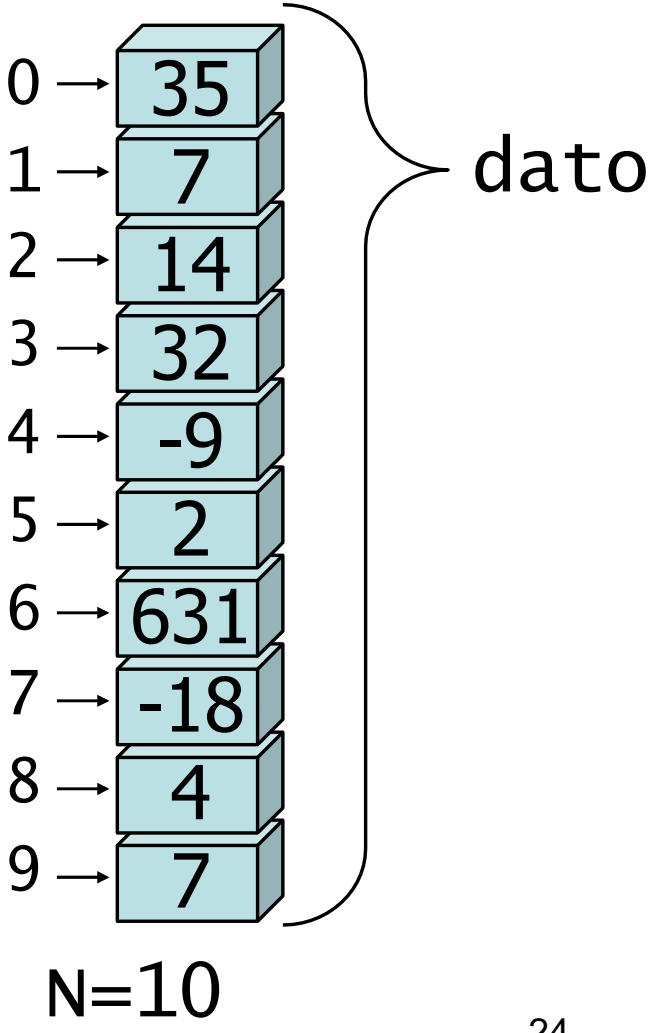
- Tutte le celle di un vettore avranno lo `stesso nome`
- Tutte le celle di un vettore devono avere lo `stesso tipo base`
- La dimensione del vettore è `fissa` e deve essere determinata al momento della sua definizione
  - La dimensione è sempre un numero intero
- Ogni cella ha `sempre` un valore
  - Impossibile avere celle "vuote"
  - Le celle non inizializzate contengono valori ignoti

Accesso alle celle

- 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

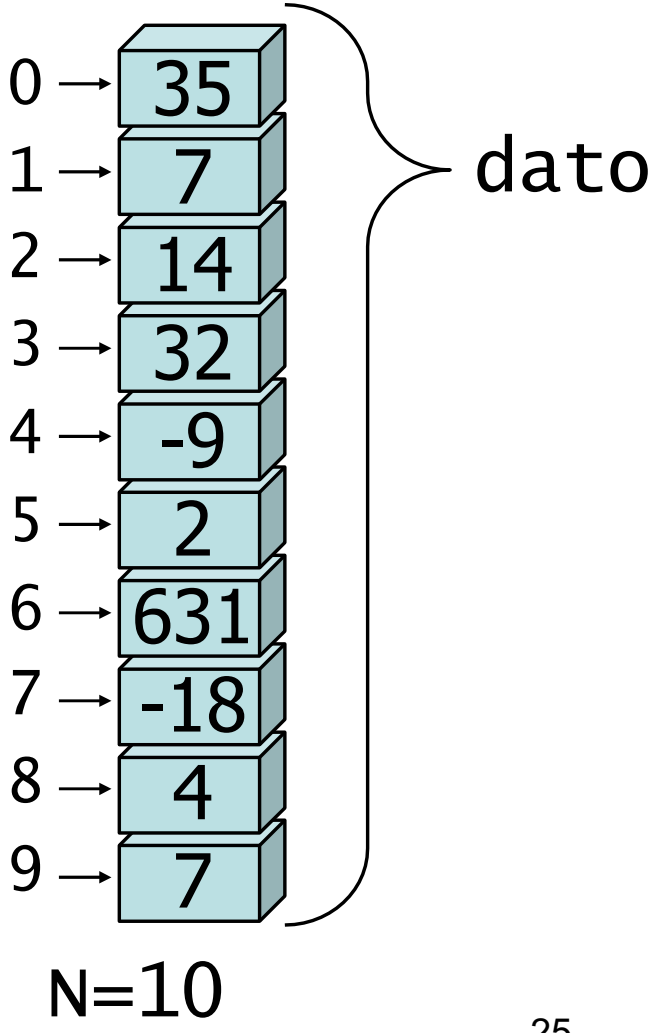
Errore frequente

- Non confondere mai l'indice con il contenuto
  - `dato[5] = 2`
  - `dato[9] == dato[1]`
  - `dato[i]>dato[j]`
  - `i>j`

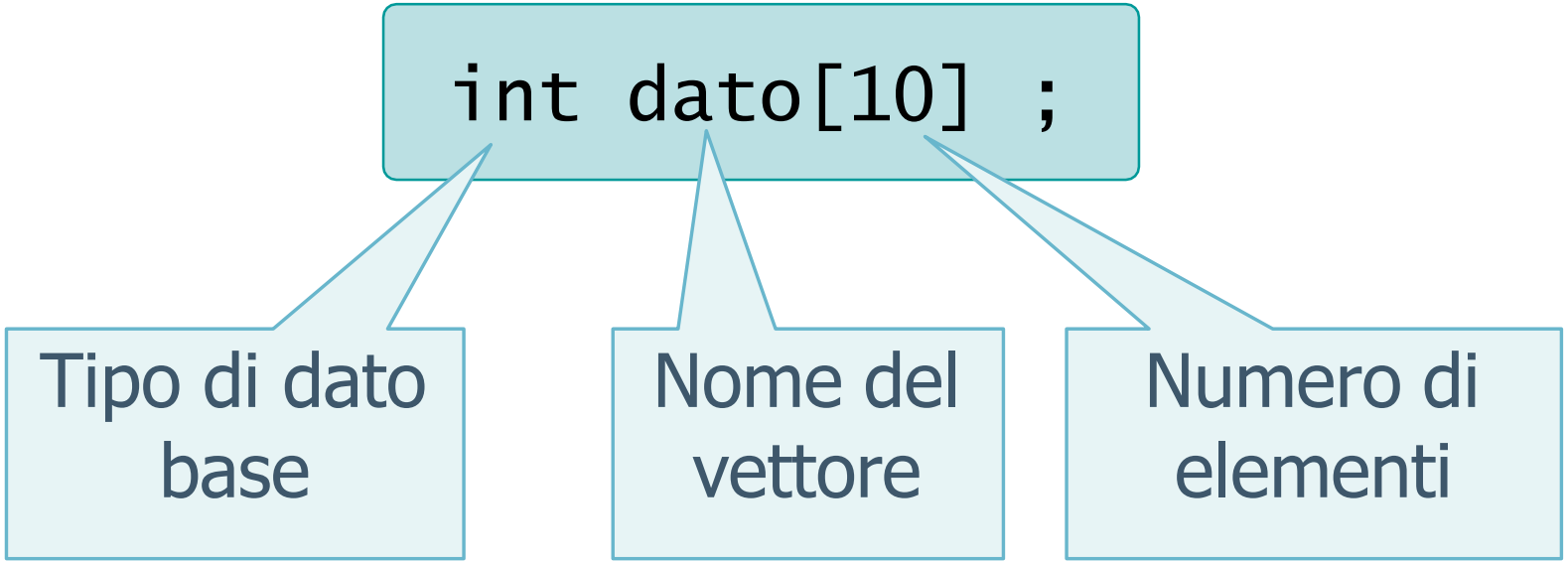


Errore frequente

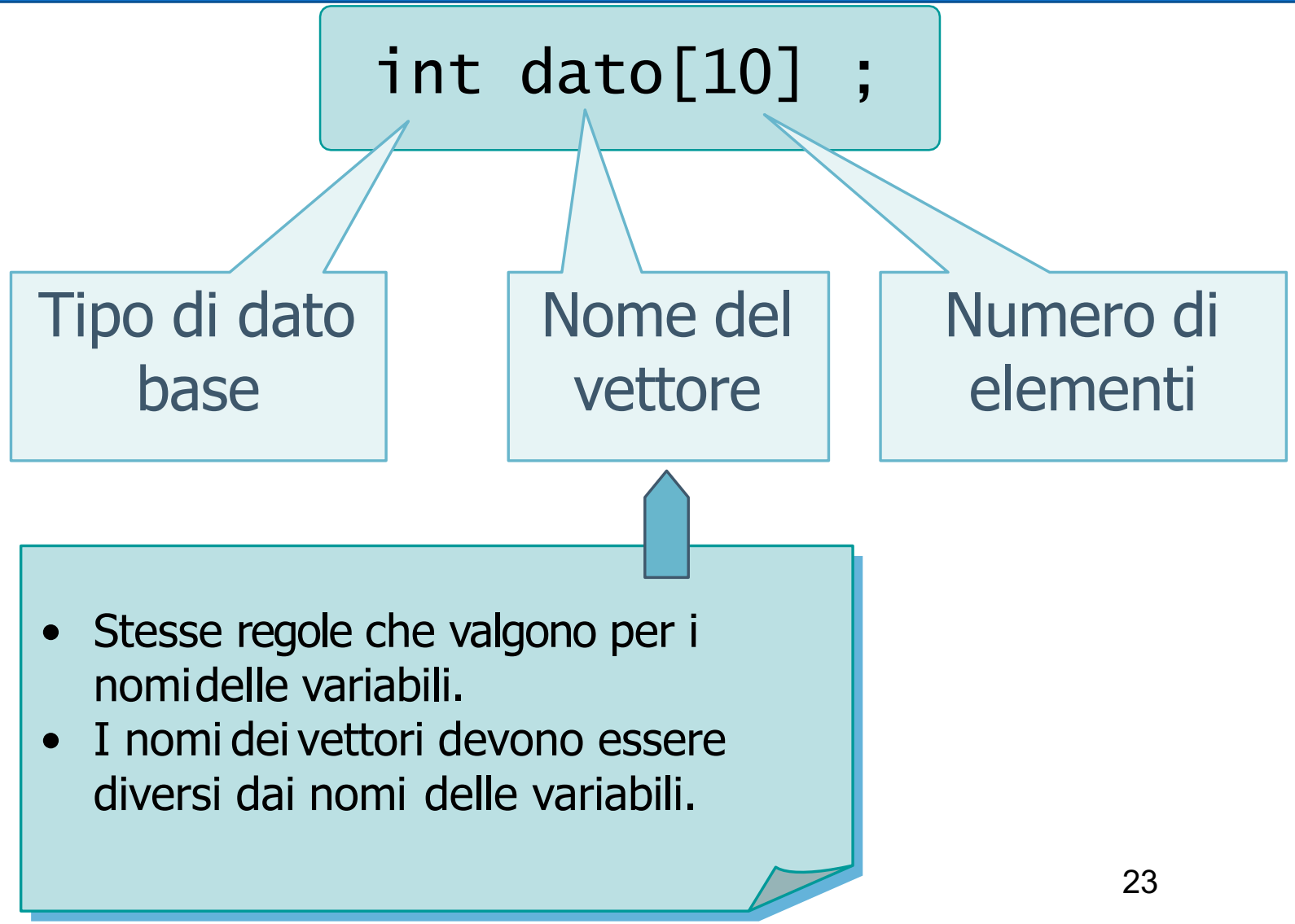
- Non si può effettuare alcuna operazione sul vettore
  - `dato = 0`
  - `printf("%d", dato)`
- Occorre operare sui singoli elementi
  - Solitamente all'interno di un ciclo `for`



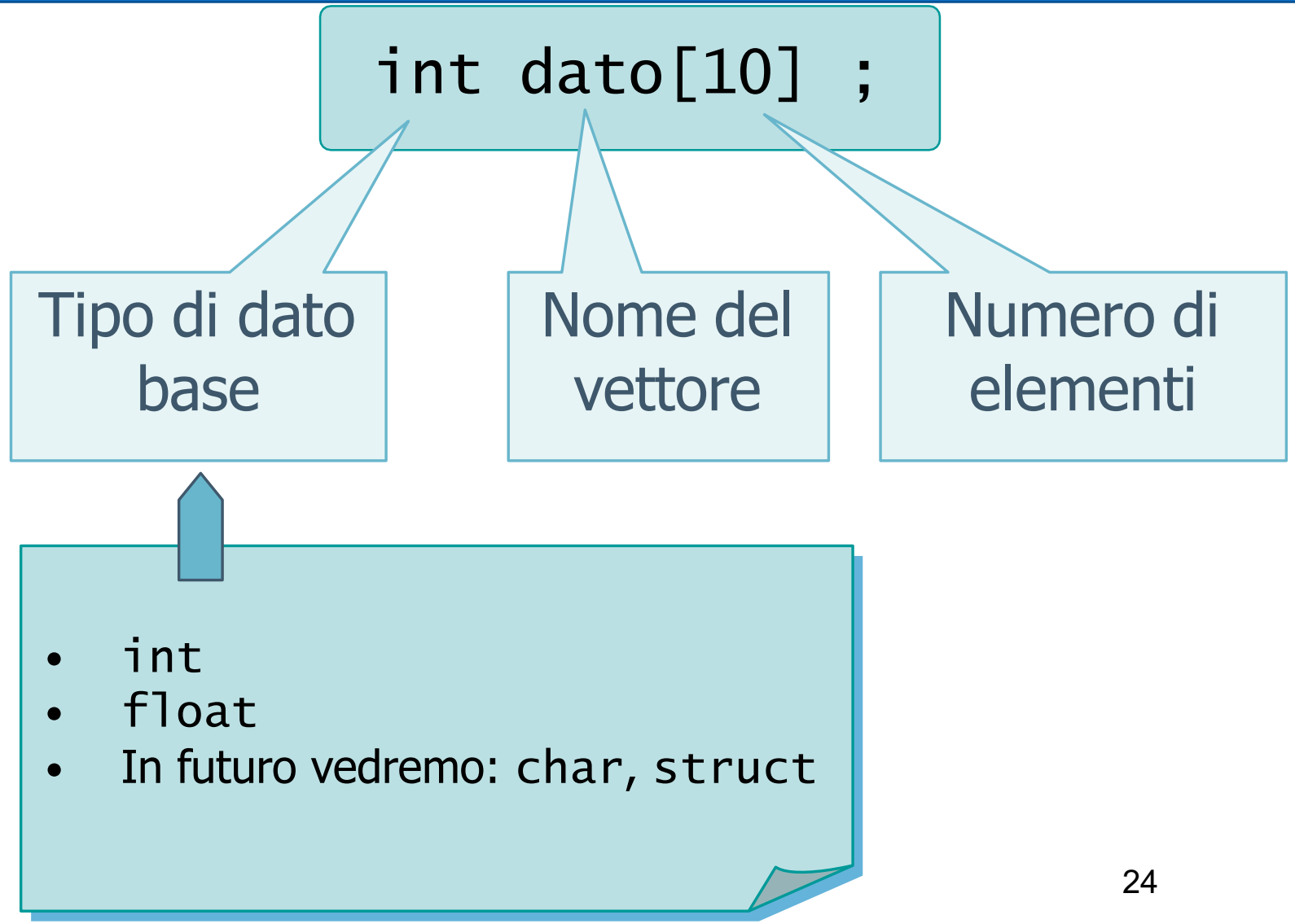
Definizione di vettori in C



Definizione di vettori in C



Definizione di vettori in C



Definizione di vettori in C


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

Tipo di dato base

Nome del vettore

Numero di elementi

- Intero positivo
- Costante nota a tempo di compilazione
- Impossibile cambiarla in seguito

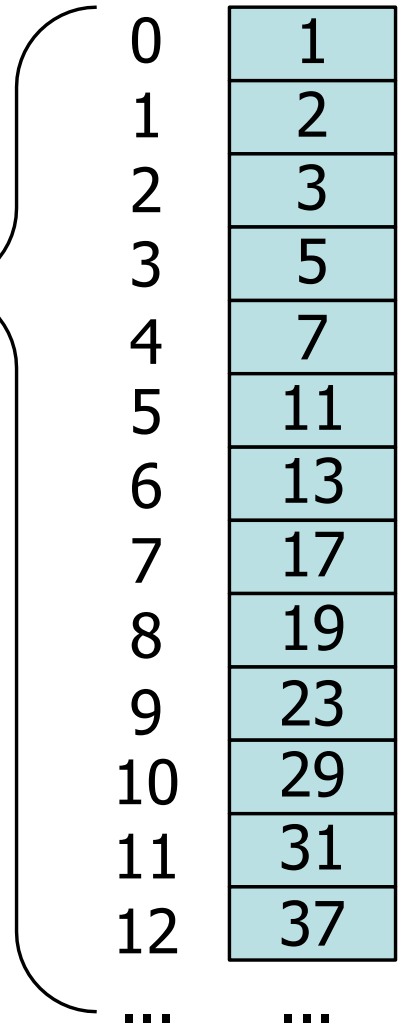
Esempi

UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

- `int dato[10] ;`
- `float lati[8] ;`
- `int numeriprimi[100] ;`
- `int is_primo[1000] ;`

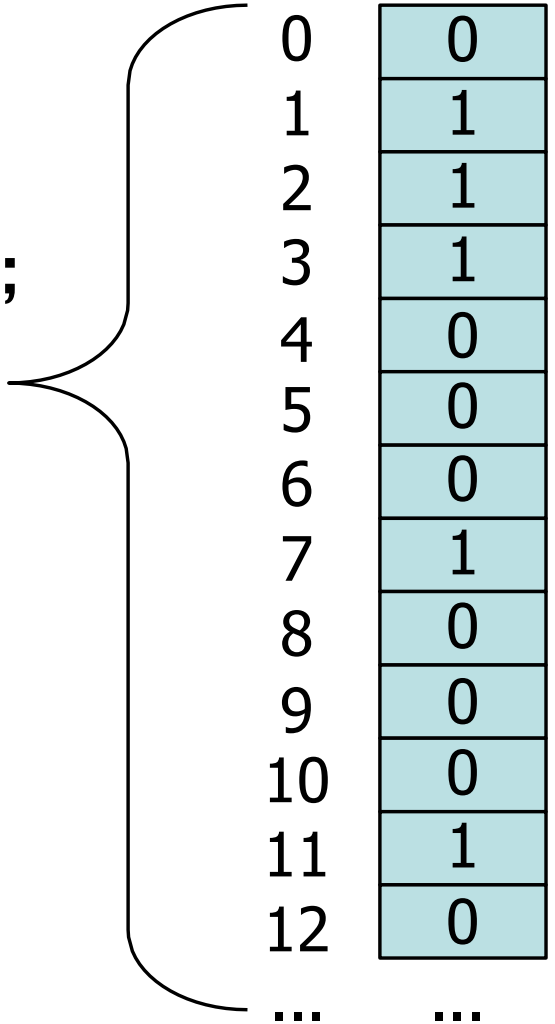
Esempi

- `int dato[10] ;`
- `float lati[8] ;`
- `int numeriprimi[100] ;`
- `int is_primo[1000] ;`



Esempi

- `int dato[10] ;`
- `float lati[8] ;`
- `int numeriprimi[100] ;`
- `int is_primo[1000] ;`



Errore frequente

- Dichiarare un vettore usando una variabile anziché una costante


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

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

Errore frequente

- Dichiarare un vettore usando una variabile non ancora inizializzata


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

. . .

scanf("%d",&N) ;
```

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

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] ;
```

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] ;
```

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?

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 tutti questi punti

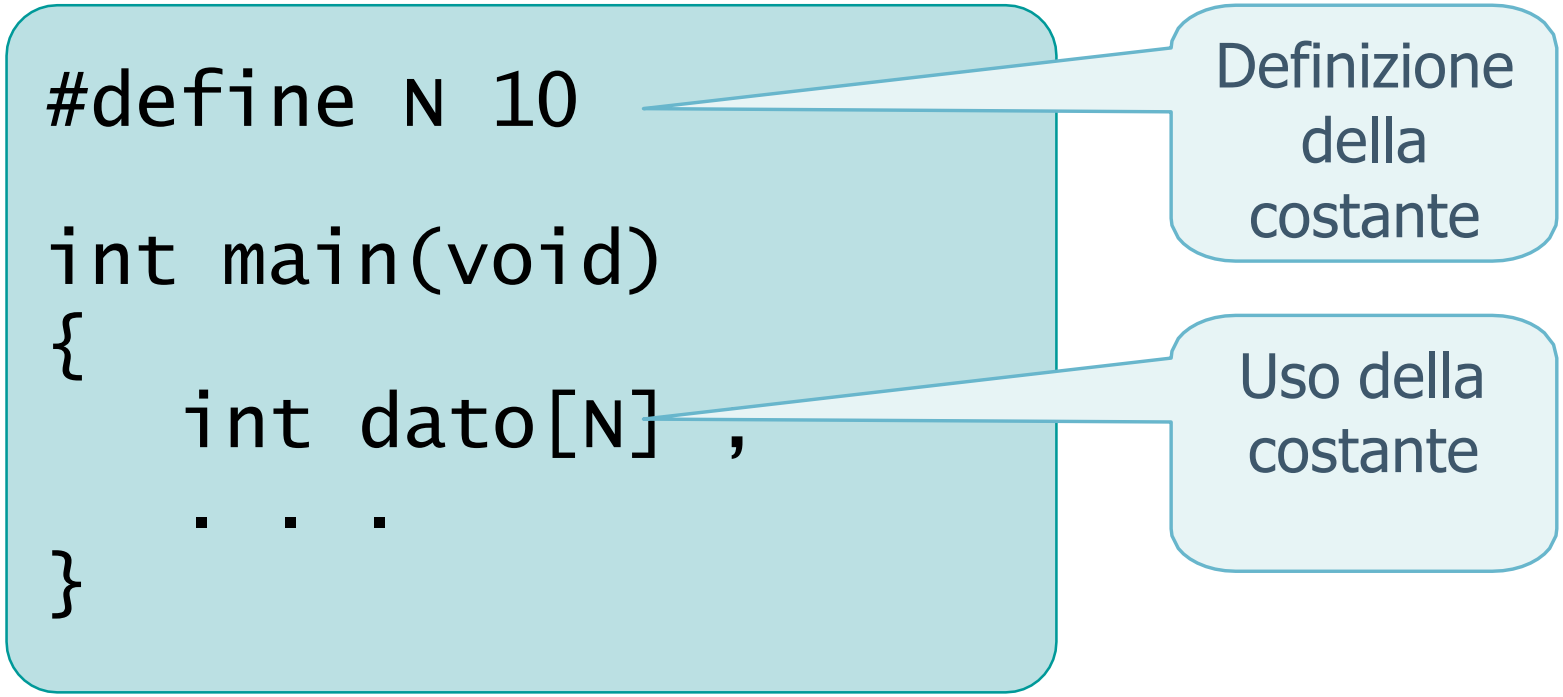
Soluzione

- 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

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

Costrutto #define



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
```

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
```

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]) ;
}
```

Modificatore const


```
int main(void)
{
    const int N = 10 ;

    int dato[N] ;

    . . .
}
```

Definizione della costante

Uso della costante

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 ;
```

Esempio

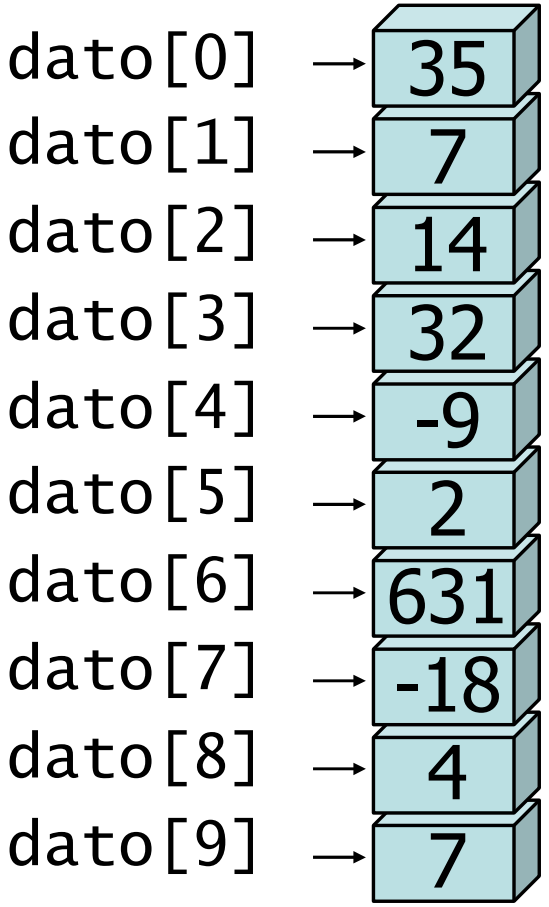

```
int main(void)
{
  const int MAX = 10 ;
  int i ;
  int dato[MAX] ;


  . . .


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

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

Suggerimenti

- Utilizzare sempre il modificatore `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

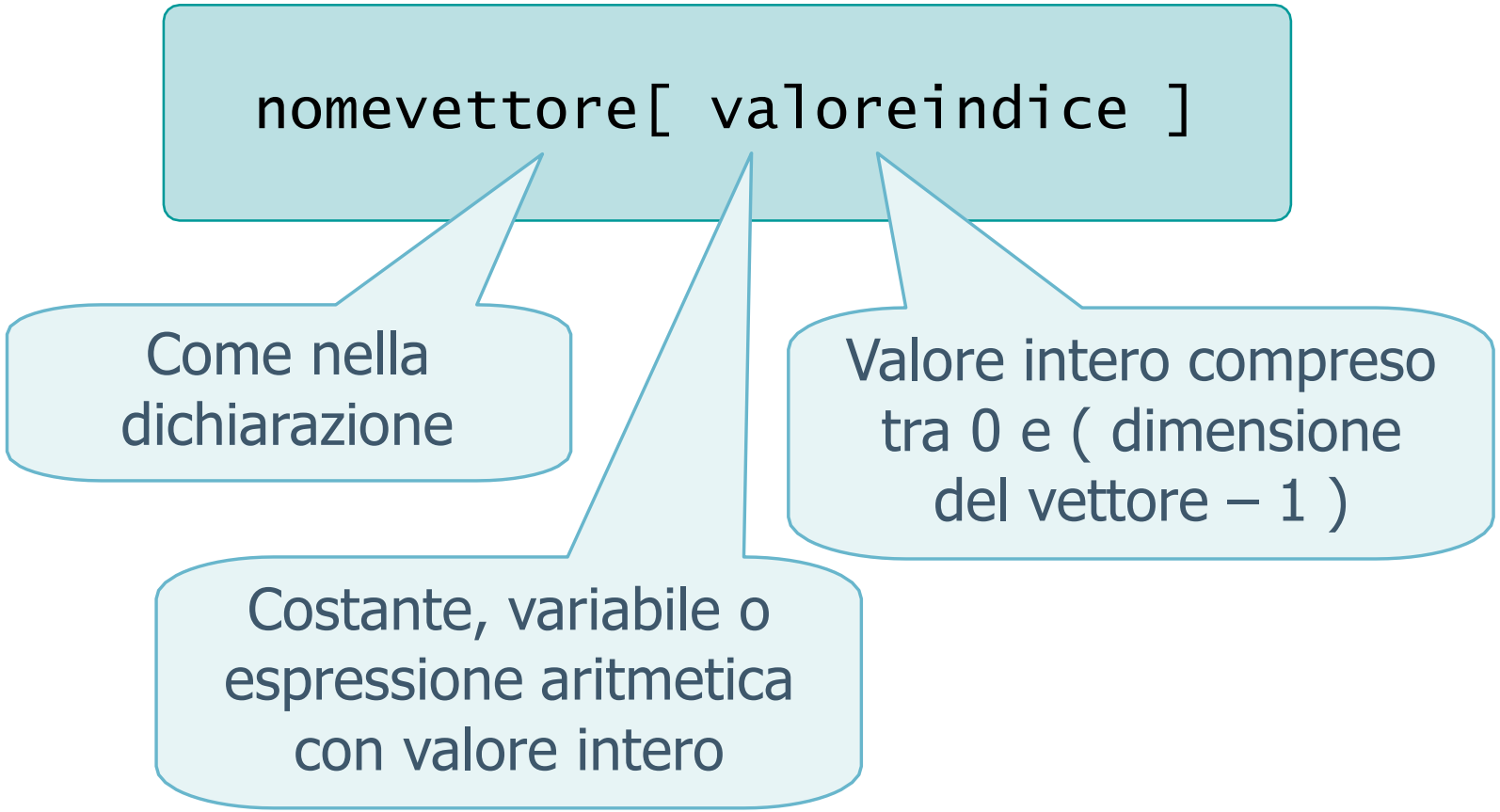
Accesso ai valori di un vettore

- 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: `[ ]`




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

Sintassi


nomevettore[valoreindice]
Come nella dichiarazione
Costante, variabile o espressione aritmetica con valore intero
Valore intero compreso tra 0 e (dimensione del vettore – 1)

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

Uso di una cella 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]) ;`

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

Operazioni elementari sui vettori

UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

- Definizioni
- Stampa di un vettore
- Lettura di un vettore
- Copia di un vettore
- Ricerca di un elemento
- Ricerca del massimo o minimo
- Vettori ad occupazione variabile

Definizioni (1/2)


```
const int 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 */
```

Definizioni (2/2)


```
const int 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

UNIVERSITÀ DEGLI STUDI DI BRESCIA

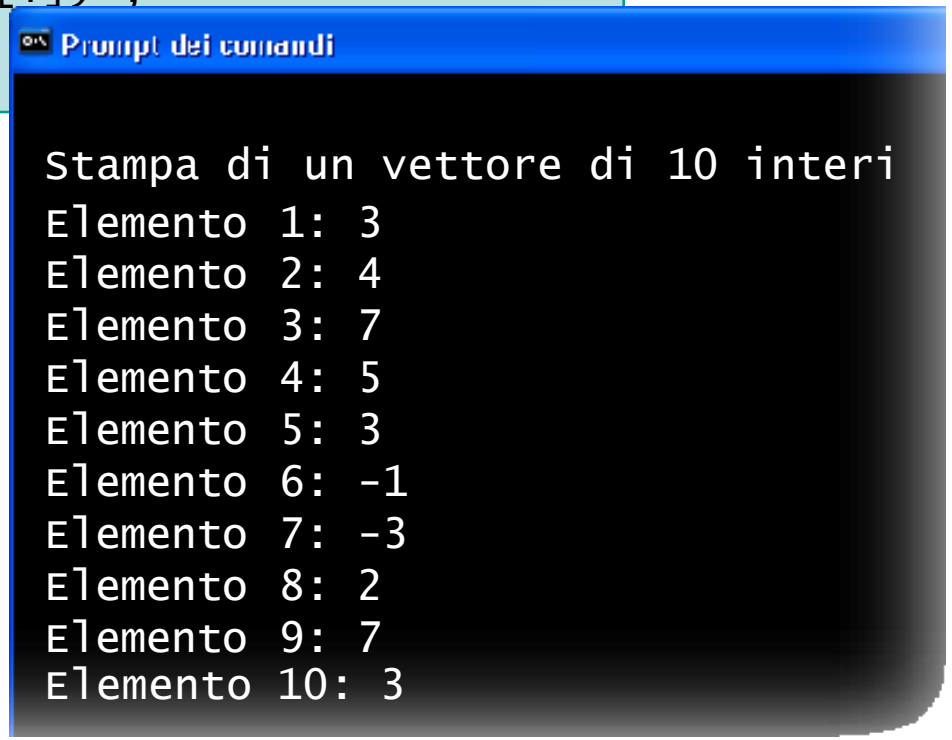
- Occorre stampare un elemento per volta, all'interno di un ciclo `for`
- Ricordare che
  - gli indici del vettore variano tra 0 e `N-1`
  - gli utenti solitamente contano tra 1 e `N`
  - `v[i]` è l'elemento `(i+1)`-esimo

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]) ;
}
```

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]) ;
}
```

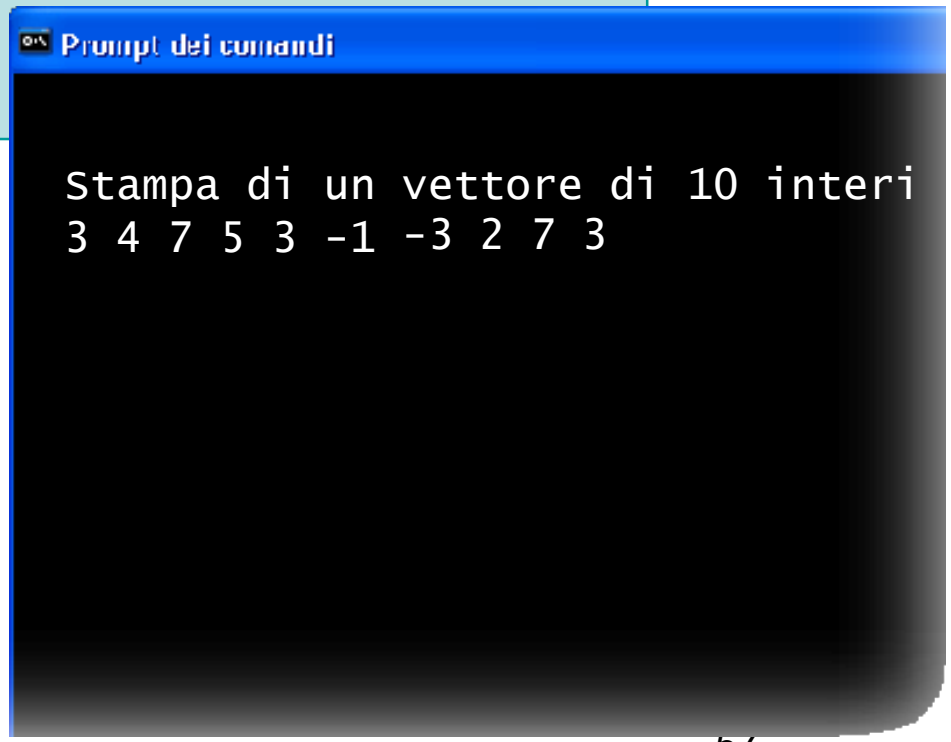


Stampa in linea


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

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

Stampa in linea


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

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



Stampa vettore di reali

UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

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

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

Avvertenze

- Anche se il vettore è di reali, l'indice è sempre intero
- Separare sempre i vari elementi almeno con uno spazio (o un a capo)

Lettura di un vettore

- Occorre leggere un elemento per volta, all'interno di un ciclo `for`
- Ricordare l'operatore & nella `scanf`

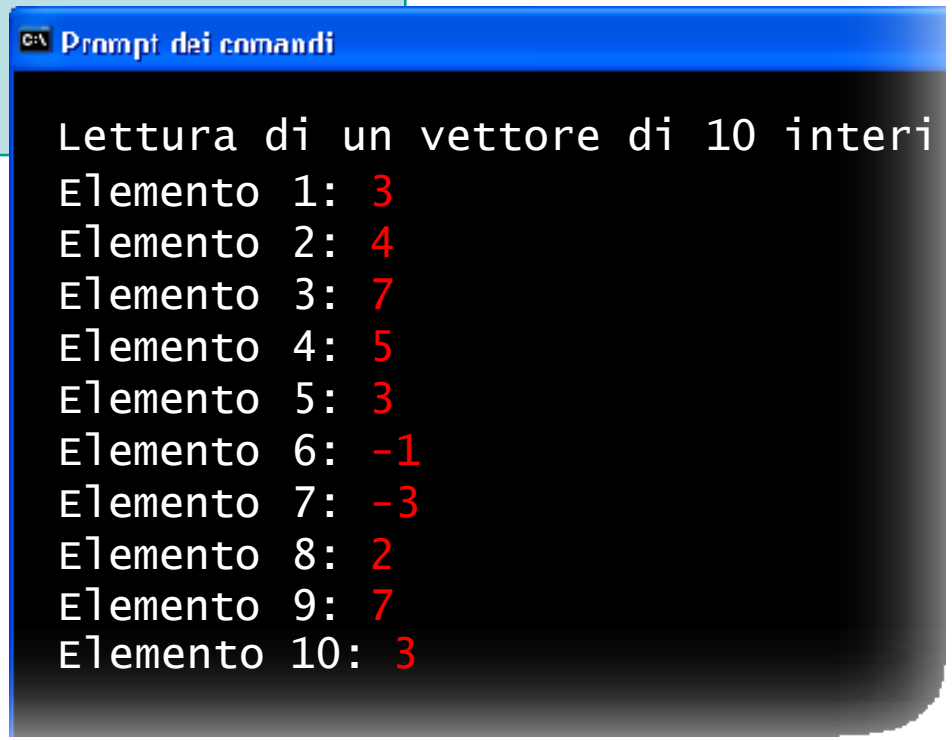
Lettura vettore di interi

UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

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

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

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]) ;
}
```



Lettura vettore di reali


```
printf("Lettura di %d reali\n", N) ;

for( i=0; i<N; i++ )
{
    printf("Elemento %d: ", i+1) ;
    scanf("%f", &r[i]) ;
}
```

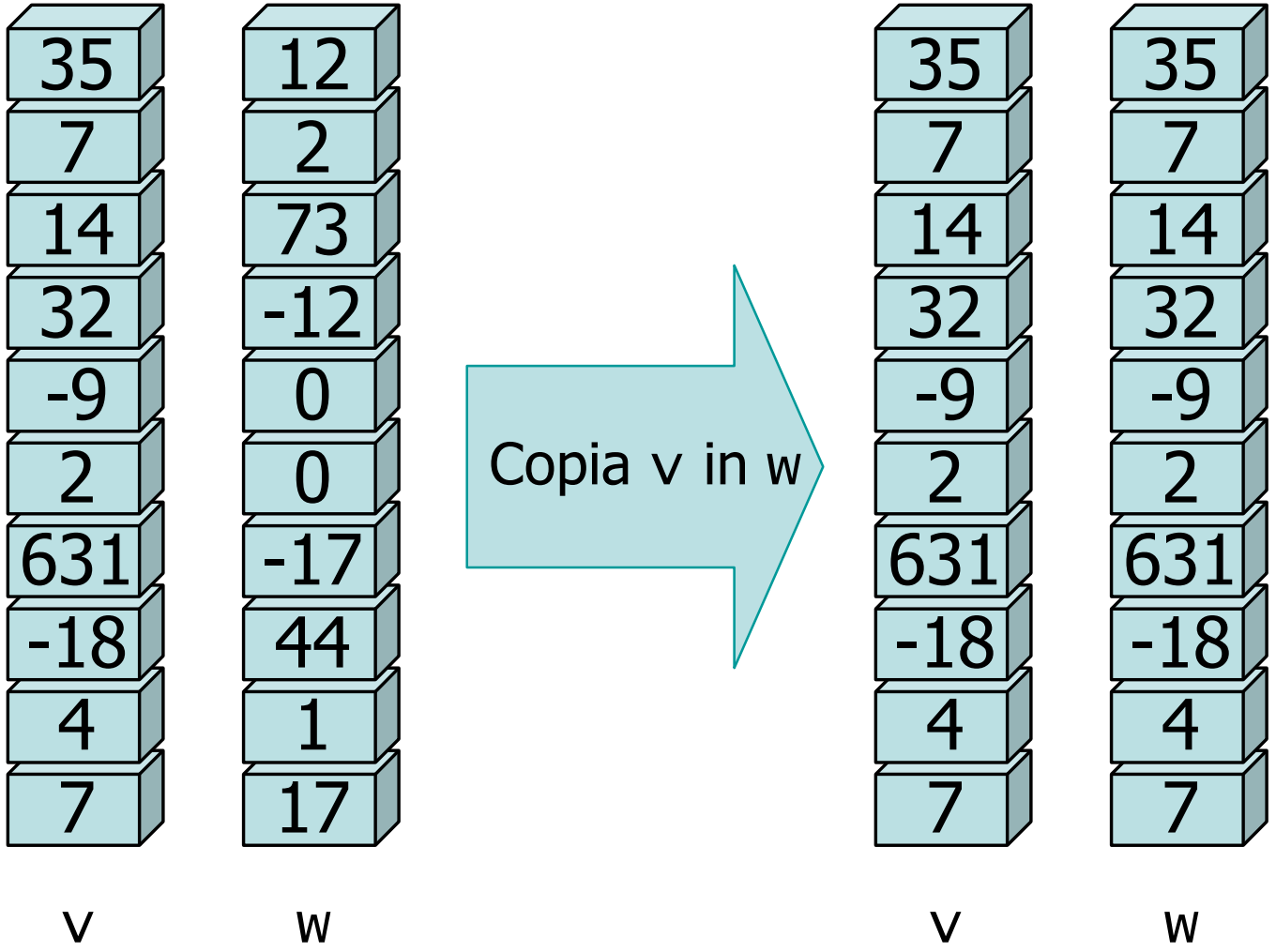
Avvertenze

- Anche se il vettore è di reali, l'indice è sempre intero
- Fare precedere sempre ogni lettura da una `printf` esplicativa

Copia di un vettore

- Più correttamente, si tratta si 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

Copia di un vettore



Copia di un vettore

UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

```
/* copia il contenuto di v[] in w[] */

for( i=0; i<N; i++ )
{
    w[i] = v[i] ;
}
```

Avvertenze

- Nonostante siano coinvolti due vettori, occore un solo ciclo for, e un solo indice per accedere agli elementi di entrambi i vettori
- Assolutamente non tentare di fare la copia in una sola istruzione!
- 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"

Ricerca di un elemento (1/3)


```
int dato ; /* dato da ricercare */
int trovato ; /* flag per ricerca */
int pos ;     /* posizione elemento */

...

printf("Elemento da ricercare? ");
scanf("%d", &dato) ;
```

Ricerca di un elemento (2/3)


```
trovato = 0 ;
pos = -1 ;

for( i=0 ; i<N ; i++ )
{
    if( v[i] == dato )
    {
        trovato = 1 ;
        pos = i ;
    }
}
```

Ricerca di un elemento (3/3)


```
if( trovato==1 )
{
    printf("Elemento trovato "
       "alla posizione %d\n", pos+1) ;
}
else
{
    printf("Elemento non trovato\n");
}
```

Varianti

- Altri tipi di ricerche
  - Contare quante volte è presente l'elemento cercato
  - Cercare se esiste almeno un elemento maggiore (o minore) del valore specificato
  - Cercare se esiste un elemento approssimativamente uguale a quello specificato
  - ...

Ricerca del massimo

- Dato un vettore (di interi o reali), determinare
  - quale sia l'elemento di valore massimo
  - quale sia la posizione in cui si trova tale elemento
- Conviene applicare la stessa tecnica per l'identificazione del massimo già vista in precedenza
  - Conviene inizializzare il max al valore del primo elemento

Ricerca del massimo (1/2)


```
float max ;  /* valore del massimo */
int posmax ; /* posizione del max */

...
max = r[0] ;
posmax = 0 ;

for( i=1 ; i<N ; i++ )
{
    if( r[i]>max )
    {
        max = r[i] ;
        posmax = i ;
    }
}
```

Ricerca del massimo (2/2)


```
printf("Il max vale %f e si ", max) ;
printf("trova in posiz. %d\n", posmax) ;
```

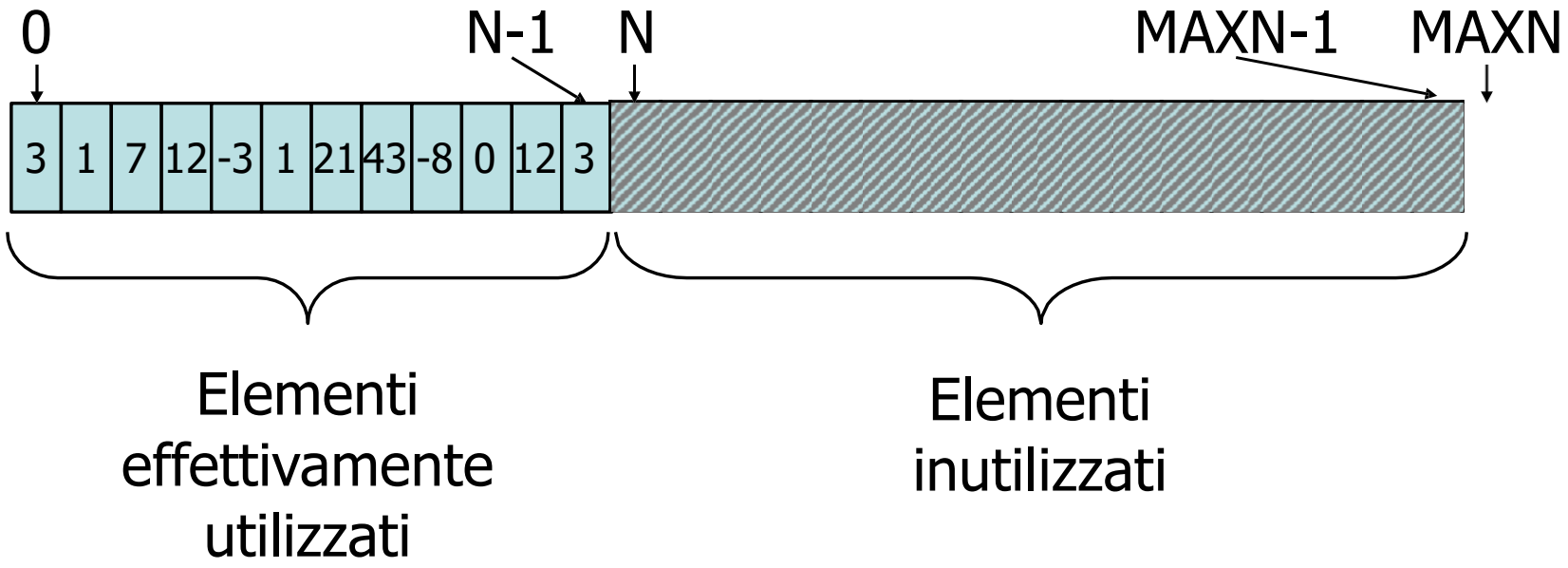
Occupazione variabile

- La principale limitazione dei vettori è la loro dimensione fissa, definita come costante al tempo di compilazione del programma
- Molto spesso non si conosce l'effettivo numero di elementi necessari fino a quando il programma non andrà in esecuzione
- Occorre identificare delle tecniche che ci permettano di lavorare con vettori di dimensione fissa, ma occupazione variabile

Tecnica adottata

- Dichiarare un vettore di dimensione sufficientemente ampia da contenere il massimo numero di elementi nel caso peggiore
  - Esempio: MAXN
- La parte iniziale del vettore sarà occupata dagli elementi, la parte finale rimarrà inutilizzata
- Dichiarare una variabile che tenga traccia dell'effettiva occupazione del vettore
  - Esempio: N

Tecnica adottata



Esempio


```
/* dimensione massima */
const int MAXN = 100 ;

int v[MAXN] ; /* vettore di dim. max. */

int N ; /* occupazione effettiva
           del vettore */

...

N = 0 ; /* inizialmente “vuoto” */
```

Regole di utilizzo

- All'inizio del programma si inizializza N al numero effettivo di elementi
  - Esempio: `scanf("%d", &N);`
- Verificare sempre che `N<=MAXN`
- Durante l'esecuzione, utilizzare sempre `N`, e mai `MAXN`
  - Esempio: `for(i=0; i<N; i++)`
- Gli elementi da `v[N]` a `v[MAXN-1]` vengono ignorati (costituiscono memoria "sprecata")

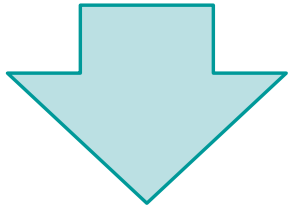
Crescita del vettore

- Un vettore ad occupazione variabile può facilmente crescere, aggiungendo elementi "in coda"

```
v[N] = nuovo_elemento ;
N++ ;
```

Esempio

- Acquisire da tastiera una serie di numeri reali, e memorizzarli in un vettore.
- La serie di numeri è terminata da un valore uguale a 0



Il valore di N non è noto all'inizio del programma, ma viene aggiornato via via che si leggono gli elementi

Soluzione (1/3)


```
const int MAXN = 100 ;

float v[MAXN] ;
float dato ;

int N ;
int i ;

printf("Inserisci gli elementi\n")
; printf("(per terminare 0)\n") ;
```

Soluzione (2/3)


```
N = 0 ; /* vettore inizialm. vuoto */

/* leggi il primo dato */
i = 0 ;
printf("Elemento %d: ", i+1) ;
scanf("%f", &dato) ;
i++ ;

/* aggiungi al vettore */
if( dato != 0.0 )
{
    v[N] = dato ;
    N++ ;
}
```

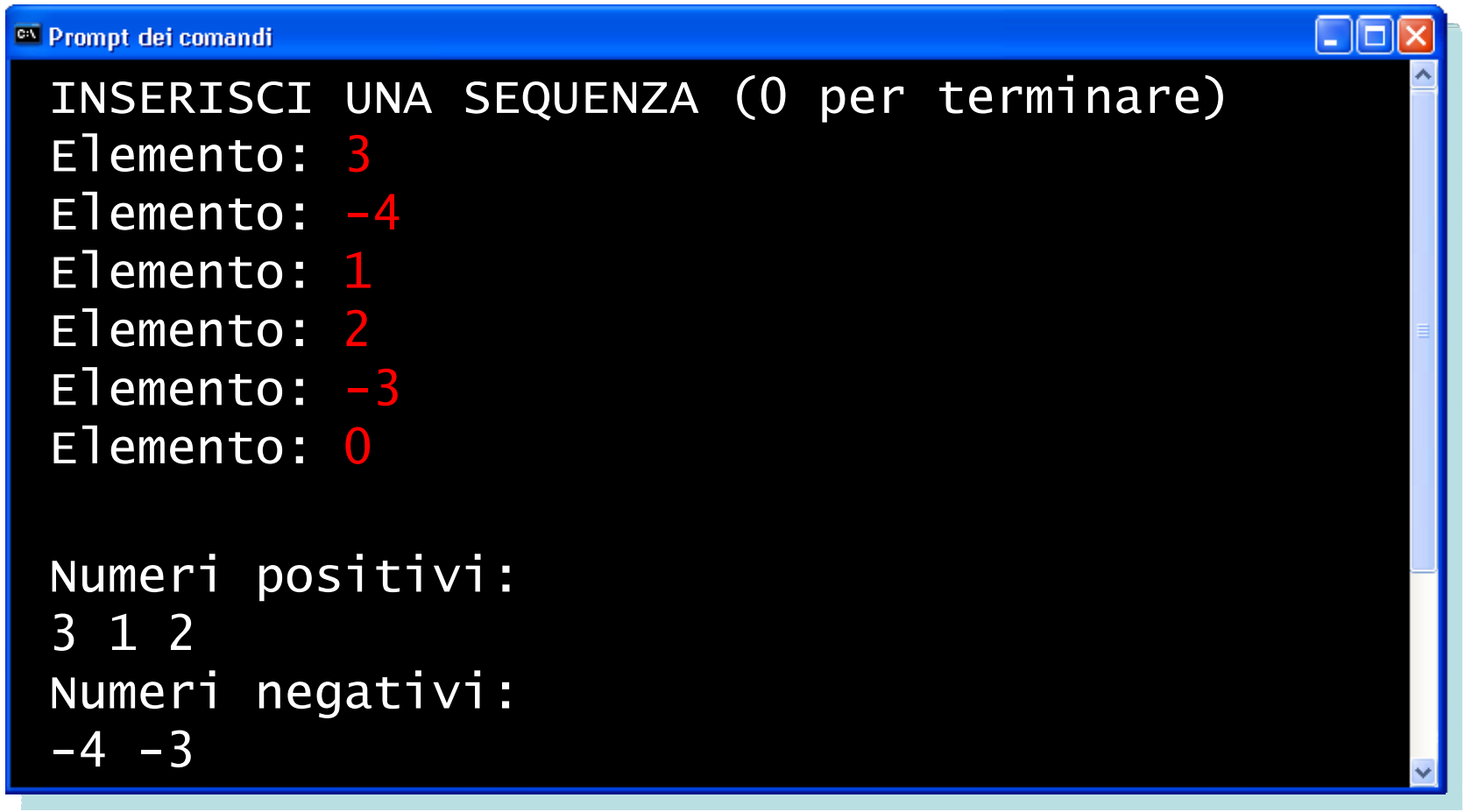
Soluzione (3/3)


```
while(dato!= 0.0)
{
    /* leggi il dato successivo */
    printf("Elemento %d: ", i+1) ;
    scanf("%f", &dato) ;
    i++ ;

    /* aggiungi al vettore */
    if( dato != 0.0 )
    {
        v[N] = dato ;
        N++ ;
    }
}
```

Esercizio “Positivi e Negativi”

- Si realizzi un programma che legga da tastiera una sequenza di numeri interi (terminata da 0), e che successivamente stampi
  - tutti i numeri positivi presenti nella sequenza, nello stesso ordine
  - tutti i numeri negativi presenti nella sequenza, nello stesso ordine

Analisi


Prompt dei comandi
INSERISCI UNA SEQUENZA (0 per terminare)
Elemento: 3
Elemento: -4
Elemento: 1
Elemento: 2
Elemento: -3
Elemento: 0
Numeri positivi:
3 1 2
Numeri negativi:
-4 -3