Vettori

Indice Lezioni Linguaggio C

√ Tipi di dato primitivi e Operatori

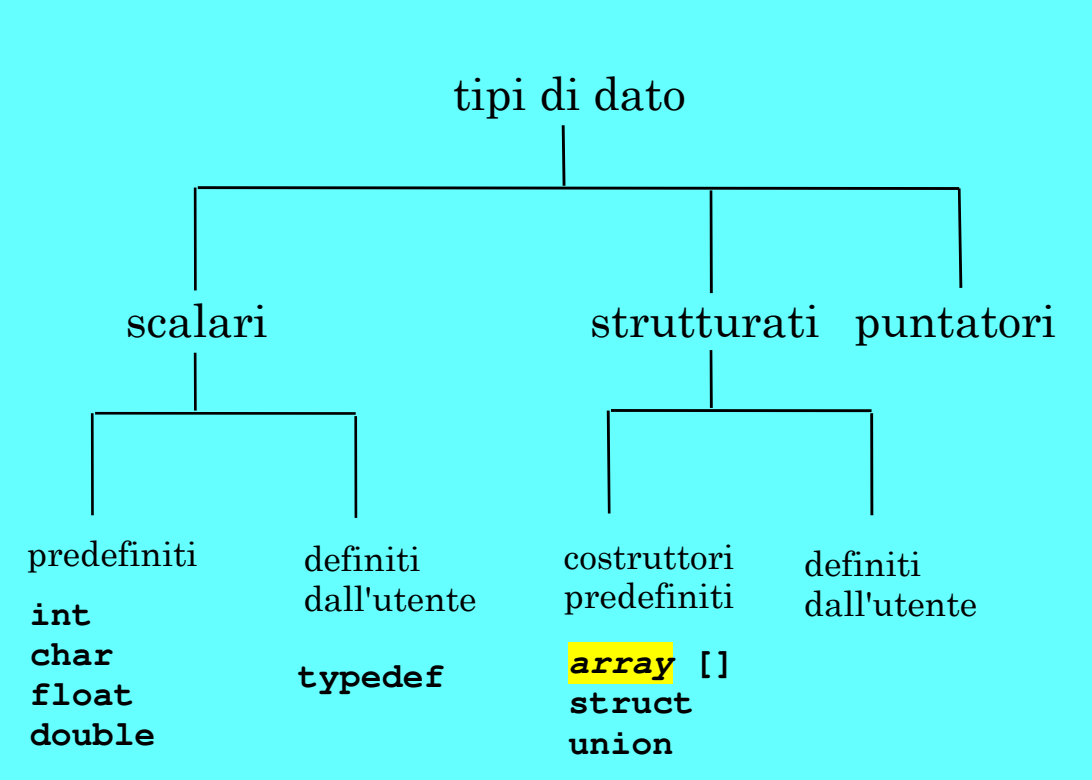
√ Dichiarazioni

√ Espressioni

√ Istruzioni

- **Tipi di dato strutturati (*vettori*)**

Tipi di Dato



Tipi strutturati

- Il linguaggio C fornisce due *costruttori* fondamentali:
  - **[ ]** (***array:*** **vettori, matrici)**
  - **struct** (***strutture***)

VETTORI

Un vettore di dimensione *N* è un insieme finito di *N* variabili dello stesso tipo, ognuna identificata da un ***indice intero*** compreso fra 0 e *N*-1

STRUTTURE

Una struttura è un insieme finito di variabili non necessariamente dello stesso tipo, ognuna identificata da un ***nome***

Vettori

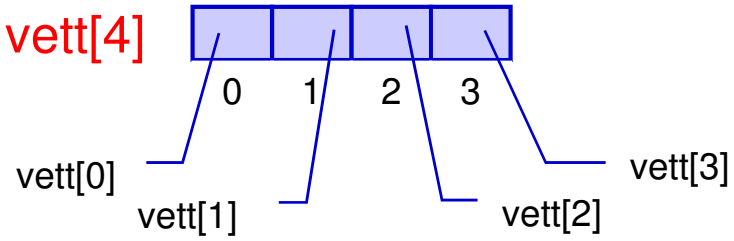
SINTASSI della *definizione* di una *variabile* di tipo vettore:

<tipo> <identificatore> [<espr-costante>] ;

Importante!

Vettore di 4 variabili <u>dello stesso tipo</u>, ognuna identificata da un *indice intero* compreso fra 0 e 3:

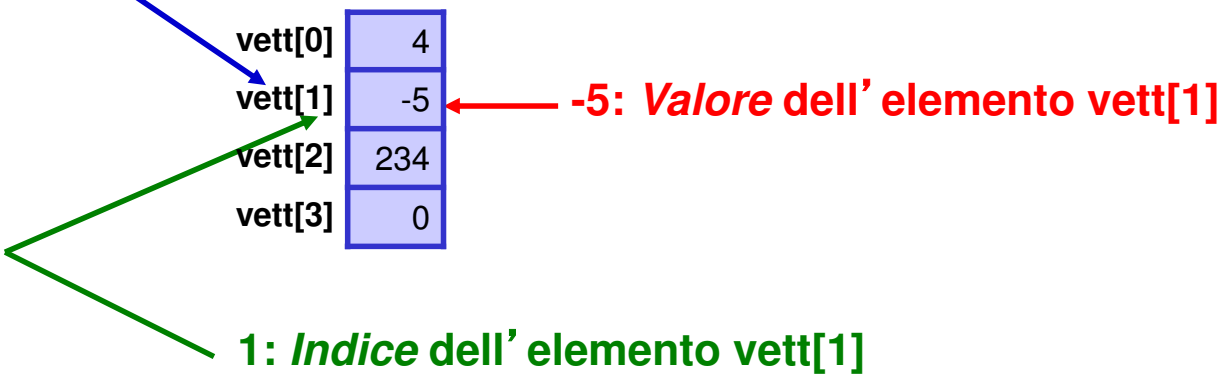
```
int vett[4];
```



Vettori

vett: Nome del vettore

vett[1]: *Nome* del secondo elemento di vett



Manipolazioni di vettori

- Un vettore **non può essere acceduto** “nel suo insieme”

Esempi

~~scanf(“%d”,&vett);~~

~~vett = ...;~~

~~a = vett + 10;~~

- L’**accesso** è sui singoli elementi
  - vett[x], con x variabile intera oppure costante

Esempi

```
int vett[10];
vett[0] = 10;
scanf(“%d”, &vett[i]);
a = vett[3] + 10;
```

Assegno un valore al primo elemento

Assegno un valore all’i-esimo elemento

Uso il valore del quarto elemento

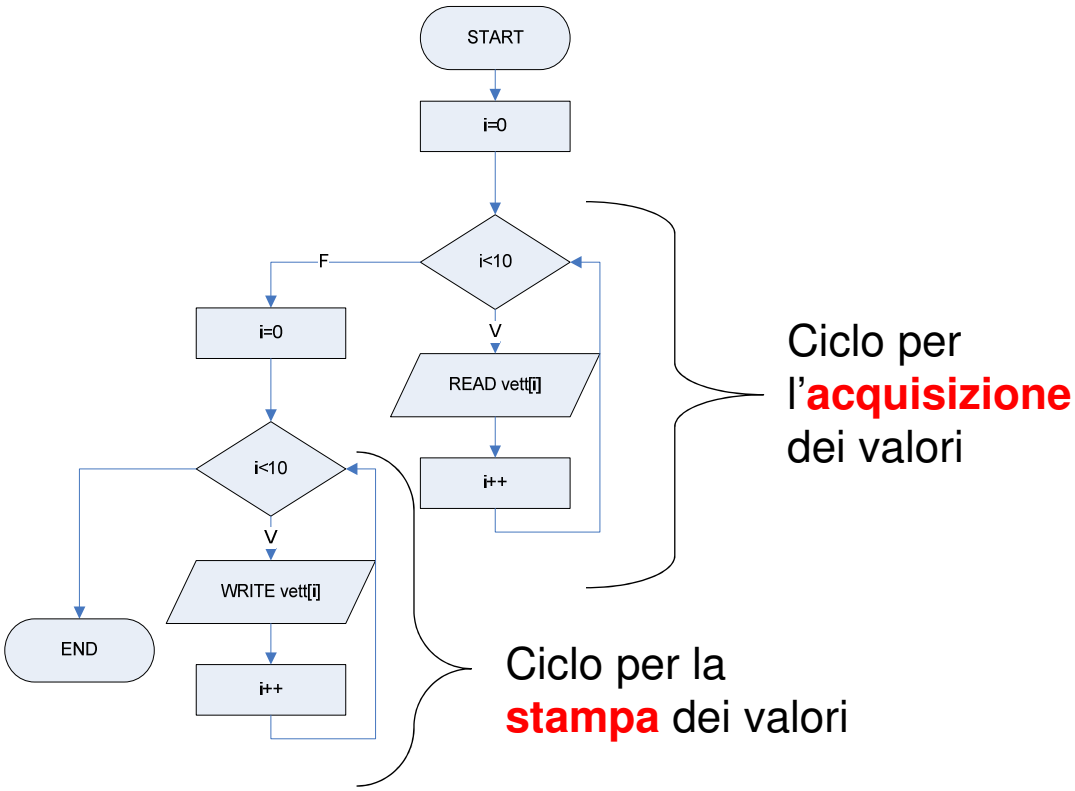
Esercizio 1 (*Specifica*)

- Scrivere un programma che inizializzi un vettore di 10 interi con valori inseriti da tastiera e quindi ne stampi il contenuto.

OSSERVAZIONE

- È possibile acquisire un elemento alla volta quindi per inizializzare un vettore è necessario eseguire un ciclo sugli elementi del vettore
- È possibile stampare un elemento alla volta quindi per stampare il vettore è necessario eseguire un ciclo sugli elementi del vettore

Esercizio 1 (*Algoritmo*)



Esercizio 1 (*Rappresentazione informazioni*)

- Serve una **costante** (***int***) per denotare la dimensione del vettore: **N**
  - In questo modo modificando la definizione della costante e ricompilando il programma è possibile gestire vettori di dimensioni diverse
- Serve un vettore di ***int*** di dimensione pari a **N:** **vett [N]**
- Servono, poi, una variabile ausiliaria (***int***) come contatore della scansione del vettore (**i**)

Esercizio 1 (*Programma*)

```
#include <stdio.h>
#define N 10
main()
 {
  int i;
  int vett [N];

  for (i=0; i<N; i++)
      scanf("%d", &vett[i]);
  for (i=0; i<N; i++)
      printf("vett[%d]: %d\n", i, vett[i]);
 }
```

Per gestire vettori di dimensione 100 basta sostituire la vecchia dimensione (10) con la nuova (100)

Esercizio 2 (*Specifica*)

- Scrivere un programma che dopo aver acquisito un vettore di 10 interi con valori inseriti da tastiera, ne determini il valore massimo e lo stampi.

Esercizio 2 (*Idea*)

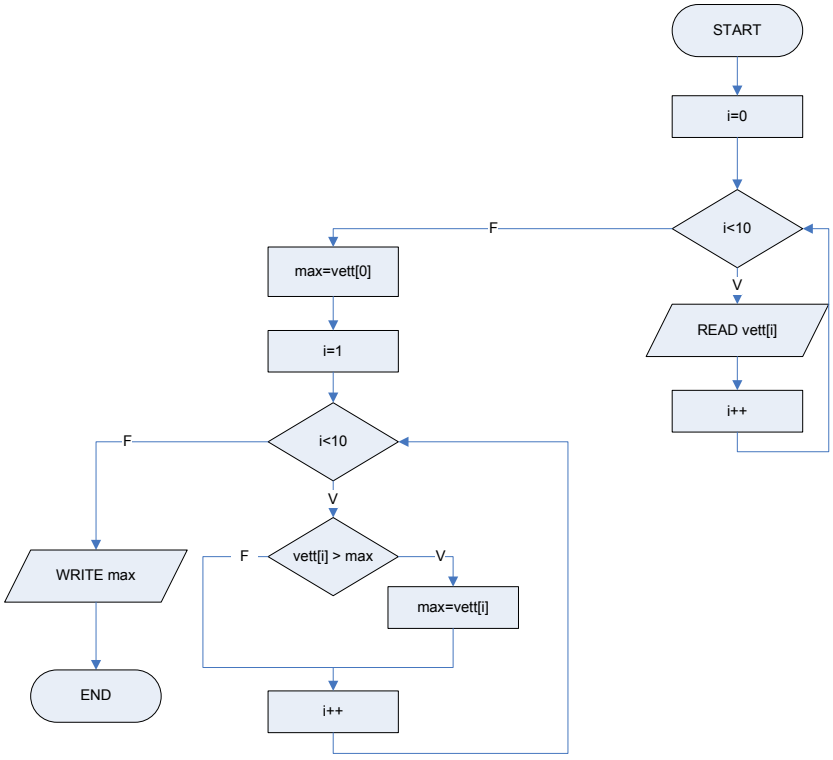
- Ancora una volta la definizione è per induzione:

CASO BASE: Il massimo fino al primo valore del vettore, vett[0], è vett[0]

CASO INDUTTIVO: Il massimo fino a vett[i] è il valore massimo tra vett[i] e il massimo fino a vett[i-1]

- Dopo aver controllato <u>tutti</u> gli elementi del vettore, il massimo registrato corrisponderà al massimo del vettore

Esercizio 2 (*Algoritmo*)



Esercizio 2 (*Rappresentazione informazioni*)

- Serve una costante (***int***) per denotare la dimensione massima del vettore: **N**
- Serve un vettore di ***int*** di dimensione pari a **N:** **vett[N]**
- Servono, poi, due variabili ausiliarie (***int***): un contatore della scansione del vettore (**i**) e un contenitore del massimo momentaneo (**max**)

Esercizio 2 (*Programma*)

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

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

  max=vett[0];
  for (i=1; i<N; i++)
      if (vett[i]>max)
          max=vett[i];
  printf("Il massimo del vettore è %d\n", max);
 }
```

Esercizio 2bis (*Specifica*)

- Scrivere un programma che, dato un vettore di 10 interi, ne determini il valore massimo e stampi sia il massimo sia la posizione nel vettore in cui questo compare.

Esercizio 2bis (*Rappresentazione informazioni*)

- Serve una costante (***int***) per denotare la dimensione massima del vettore: **N**
- Serve un vettore di ***int*** di dimensione pari a **N:** **vett[N]**
- Servono, poi, tre variabili ausiliarie (***int***): un contatore della scansione del vettore (**i**), un contenitore del massimo momentaneo (**max**) e l'indice della posizione del massimo (**pos**)

Esercizio 2bis (*Programma*)

```
#include <stdio.h>
#define N 100
main()
 {
  int i, max, pos;
  int vett[N];

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

  max=vett[0];
  pos=0;
  for (i=1; i<N; i++)
      if (vett[i]>max)
          {  max=vett[i];
             pos=i;   }
  printf("Il massimo del vettore è %d in posizione %d\n",
   max, pos);
 }
```


Osservazioni

- A meno di non settare opportunamente i parametri del compilatore, tipicamente i compilatori del linguaggio C *non effettuano il controllo* sul rispetto degli indici (inferiore e superiore) del vettore!

  Quindi, un'istruzione del tipo **`v[N]=54`** viene accettata dal compilatore senza segnalazione di errori, che si verificheranno in modo impredicibile a tempo di esecuzione.

Osservazioni (2)

- Un vettore è un insieme finito di **max_M** variabili dello stesso tipo, *ognuna* identificata da un *indice* compreso fra 0 e max_M-1

- La dimensione massima del vettore è specificata da un valore costante (e *non può* essere determinata o modificata a tempo di esecuzione)

1. Il vettore in C non è un tutt'uno e non si debbono per forza *usare* tutte le celle disponibili di un vettore di **max_M** elementi

2. Spesso, la porzione di vettore realmente utilizzata *dipende dai dati di ingresso*. La dimensione **M** della porzione del vettore realmente utilizzata (**M<= max_M**) può essere stabilita a tempo di esecuzione

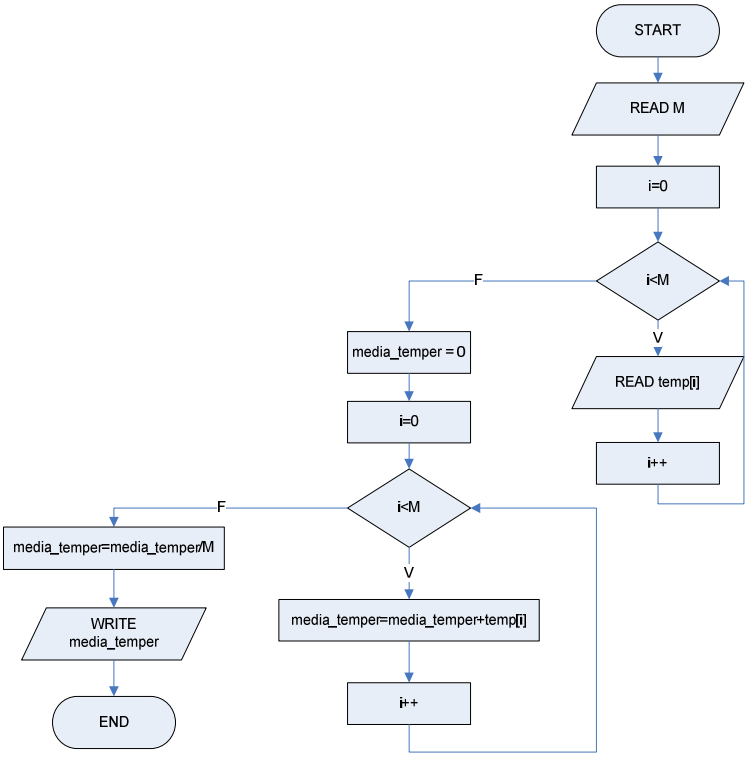
Esercizio 3 (*Specifica*)

- Scrivere un programma che, data una serie di rilevazioni di *al più* 100 temperature espresse in gradi Kelvin memorizzate in un vettore, calcoli la media delle temperature fornite.

Esercizio 3 (*Idea*)

- Chiedi in input il numero di valori **M** effettivamente rilevati e leggi tutti questi valori
- Somma tutti i valori inseriti (ovvero i valori nelle posizioni 0…M-1)
- La media cercata è data dalla somma precedente divisa per **M**

Esercizio 3 (*Algoritmo*)



Esercizio 3 (*Rappresentazione informazioni*)

- Serve una costante (***int***) per denotare la dimensione massima del vettore: **max_M=100 ***
- Serve un vettore di ***float*** di dimensione pari a **max_M** (**temp[max_M]**)
- Serve una variabile (***int***) per indicare il numero di valori di temperature effettivamente letti da input: **M** (0<M<=max_M)
- Serve una variabile ausiliaria (***int***) come contatore della scansione del vettore (**i**)
- Serve, infine, una variabile ausiliaria (***float***) con il doppio ruolo di accumulatore delle somme parziali e di media (**media_temper**)

* **Il vettore dovrà essere dimensionato a 100 celle, perché le temperature possono essere fino a 100, anche se le celle realmente utilizzate potranno essere meno**

Esercizio 3 (*Programma*)

```
#include <stdio.h>
#define max_M 100

main()
 {
  int M, i;
  float temp[max_M];
  float media_temper=0.;
  do  scanf("%d", &M);
  while ((M<=0) || (M>max_M));
  for (i=0; i<M; i++)
     { do  scanf("%d", &temp[i]);  while (temp[i]<0);
     }
  for (i=0; i<M; i++)
     media_temper=media_temper+temp[i];
  media_temper= media_temper / M;
  printf("Media delle temperature Kelvin inserite =
   %f\n", media_temper);
 }
```

Esercizio 3bis (*Specifica*)

- Scrivere un programma che, data una serie di rilevazioni di massimo 100 temperature espresse in gradi Kelvin memorizzata in un vettore (*la serie delle temperature lette termina quando viene inserito –50*), calcoli la media delle temperature fornite.

NOTA: La sequenza di valori da leggere non è data, così come non è noto il numero totale dei valori da sommare

Esercizio 4 (*Specifica*)

- Scrivere un programma che, dato un vettore di al più 1000 elementi reali a doppia precisione non nulli, copi in un altro vettore solo gli elementi compresi tra 10 e 500. Al termine, stampi il contenuto del vettore risultante.

Esercizio 4 (*Idea*)

- Chiedi in input i valori da inserire nel vettore e tieni traccia del numero di valori inseriti (ovvero della dimensione del vettore effettivamente utilizzata)
- Scandisci tutti gli elementi inseriti nel primo vettore utilizzando un contatore
- Mentre scandisci il vettore, copia ogni valore nel range nel nuovo vettore ed incrementa un *contatore diverso da quello utilizzato per scandire il primo vettore*
- Stampa il valore finale del contatore perché esso rappresenta la dimensione effettiva del secondo vettore ovvero il numero di valori copiati

Esercizio 4 (*Rappresentazione informazioni*)

- Serve una costante (***int***) per denotare la dimensione massima dei due vettori: **max_M=1000** *
- Serve una variabile logica (***int***) come flag che rappresenta l'intenzione dell'utente di terminare l'inserimento o continuarlo: **continua**
- Servono due vettori di ***double*** di dimensione pari a **max_M**
- Servono, poi, tre variabili ausiliarie (***int***) per denotare la dimensione del primo vettore (**dim**), come contatore della scansione del primo vettore (**i**) e come contatore dei valori copiati (**conta**)

* **Probabilmente, il secondo vettore avrà meno valori ammissibili del primo, ma perché il programma funzioni in tutti i casi, entrambi i vettori devono essere dimensionati per contenere fino a 1000 elementi**

Esercizio 4 (*Programma*)

```
#include <stdio.h>
#define max_M 1000
main()
 {
  int i, conta=0, dim;
  int continua;
  double vett_uno[max_M], vett_due[max_M];

  i=0;
  do{   scanf("%d",&vett_uno[i]);
        i++;
        printf("Ancora? (0/1)"); scanf("%d",&continua);}
  while( continua && i<max_M )
  dim=i;
  conta = 0;
  for (i=0; i<dim ; i++)
      if ((vett_uno[i]>=10.)&&(vett_uno[i]<=500.))
          { vett_due[conta]=vett_uno[i];
            conta++;
          }
  printf("Sono stati copiati %d elementi nel vettore = %d\n", conta);
 }
```


Esercizio 5 (*Specifica*)

- Scrivere un programma che, letto da input un valore intero, cerchi se il valore dato è presente o meno in un vettore di 500 elementi interi.

Esercizio 5 (*Idea*)

- Confronta il valore dato con tutti gli elementi del vettore
  - Se trovi un elemento del vettore uguale al valore dato allora puoi stampare che l'elemento è contenuto nel vettore
- Solo dopo aver verificato che ***tutti*** gli elementi sono diversi dal valore dato si può stampare che l'elemento non è contenuto nel vettore

Esercizio 5 (*Algoritmo - versione 1*)

- Leggi un valore da input e inizializza una *variabile "logica"* a "false"
- Scandisci tutto il vettore da 0 ad N-1 confrontando ciascun elemento con il valore dato
- Se trovi un elemento del vettore uguale al valore dato, cambia il valore della variabile "logica"
- Stampa una scritta opportuna che indichi se l'elemento è contenuto o meno nel vettore

Esercizio 5 (*Rappresentazione informazioni*)

- Serve una costante (***int***) per denotare la dimensione massima del vettore: **max_N**
- Serve un vettore di ***int*** di dimensione pari a **max_N:** **vettore[ ]**
- Servono, poi, due variabili ausiliarie (***int***), come contatore della scansione del vettore (**i**) e come contenitore del valore di input dato (**valore**)
- Serve, infine, una variabile "logica" (***int***) inizializzata a "false"**:** **trovato**

Esercizio 5 (*Programma - versione 1*)

```
#include <stdio.h>
#define max_N 500
main()
 {
  int i, valore, trovato=0;
  int vettore[max_N];
  scanf("%d", &valore);
  for(i=0;i<max_N;i++)
   scanf("%d",&vettore[i]);
  for (i=0; i<max_N; i++)
      if (vettore[i]==valore)
          trovato=1;
  if (trovato) printf("L'elemento %d è contenuto nel
   vettore\n", valore);
    else printf("L'elemento %d non è contenuto nel
   vettore\n", valore);
 }
```

E se max_N fosse uguale a 100000000?

Esercizio 5 (*Algoritmo - versione 2*)

- Leggi un valore da input e inizializza una variabile "logica" a "false"
- Scandisci tutto il vettore da 0 ad N-1 confrontando ciascun elemento con il valore dato
- Non appena trovi un elemento del vettore uguale al valore dato, cambia il valore della variabile "logica" ed <u>interrompi la scansione del vettore</u>
- Stampa una scritta opportuna che indichi se l'elemento è contenuto o meno nel vettore

Esercizio 5 (*Programma - versione 2*)

```
#include <stdio.h>
#define max_N 500
main()
 {
  int i, valore, trovato=0;
  int vettore[max_N];
  scanf("%d", &valore);
  for(i=0;i<max_N;i++)
   scanf("%d",&vettore[i]);


  for (i=0; (i<max_N)&&(! trovato); i++)
      if (vettore[i]==valore)
          trovato=1;
  if (trovato)
   printf("L'elemento %d è contenuto nel vettore\n",
   valore);
  else printf("L'elemento %d non è contenuto nel
   vettore\n", valore);
 }
```

Esercizio 6 (*Specifica*)

- Scrivere un programma che, letto da input un valore intero, cancelli tutte le occorrenze del valore da un vettore di 50 elementi interi.

Esercizio 6 (*Algoritmo – versione 1*)

- Scorri tutto il vettore
- Per ogni elemento uguale al valore cercato
  - Sostituisci l'elemento con l'ultimo elemento del vettore
  - Decrementa la dimensione del vettore

Esercizio 6 (*Rappresentazione informazioni*)

- Serve una costante (***int***) per denotare la dimensione massima del vettore: **max_N**
- Serve un vettore di ***int*** di dimensione pari a **max_N:** **vettore[ ]**
- Servono, poi, due variabili ausiliarie (***int***), come contatore della scansione del vettore (**i**) e come contenitore del valore di input dato (**valore**)

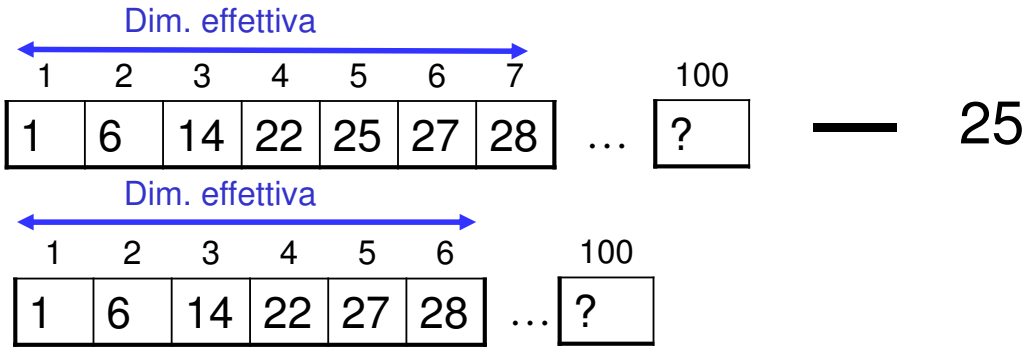
Esercizio 6 (*Programma - versione 1*)

```
#include <stdio.h>
#define max_N 50
main()
 {
  int i, valore;
  int vettore[max_N], dim;
for(i=0;i<max_N;i++)
   scanf("%d",&vettore[i]);
scanf("%d", &valore);
dim=max_N;
  for (i=0; i<dim; i++)
      if (vettore[i]==valore){
              vettore[i]=vettore[dim-1];
              dim=dim-1;
              i=i-1;}
   }
```

Esercizio 6 (*Algoritmo - versione 2*)

PROBLEMA: La versione 1 dell'Algoritmo cambia l'ordine degli elementi. E se volessi mantenere lo stesso ordine?

ESEMPIO



Esercizio 6 (*Algoritmo - versione 2*) (*Idea*)

Dato l'elemento da eliminare VELIM e la dimensione DIM del vettore

1. Cerco la posizione I dove si trova VELIM

Se il valore appartiene al vettore

2. Aggiorno la dimensione a DIM-1
3. Elimino il valore dal vettore
   - Copio ogni valore nelle posizioni I+1…DIM nella posizione precedente, I…DIM-1

Esercizio 6 (*Codifica*)

```
#include <stdio.h>
#define N 100
void main ( )
{
int VALORI[N],VELIM,I,K,DIM;

printf("QUANTI VALORI NEL VETTORE ? ");
scanf("%d",&DIM);
for  (I=0; I<DIM; I++){
printf("introdurre il val. %d:",I+1);
scanf("%d", &VALORI[I]); }
printf("valore da eliminare :");
scanf("%d", &VELIM);
```


Esercizio 6 (*Codifica*)

```
/* Cerca il valore e lo elimina */
I = 0;
while (I < DIM )
     if (VELIM != VALORI[I]) I= I + 1;
     else {
            DIM = DIM-1;
            for (K = I; K<DIM; K++)
               VALORI[K] = VALORI[K+1];
          }}
```

Altri esercizi

- Scrivere un programma che dato un vettore di al più 50 elementi di tipo reale, consenta di aggiungere uno o più elementi al vettore
- Scirvere un programma che dopo aver acquisito un vettore di al più 50 elementi di tipo reale, presenta all'utenete il seguente menù per la gestione di vettori:

  1 – aggiungi un elemento al vettore

  2 – cerca un elemento nel vettore

  3 – elimina un elemento dal vettore

  4 – mostra il vettore attuale

  5 – esci

Il programma deve consentire all'utente di eseguire una o più operazioni fino a quando non digita l'opzione di uscita (4). Al termine di ogni operazione eseguita il programma deve stampare il contenuto del vettore.