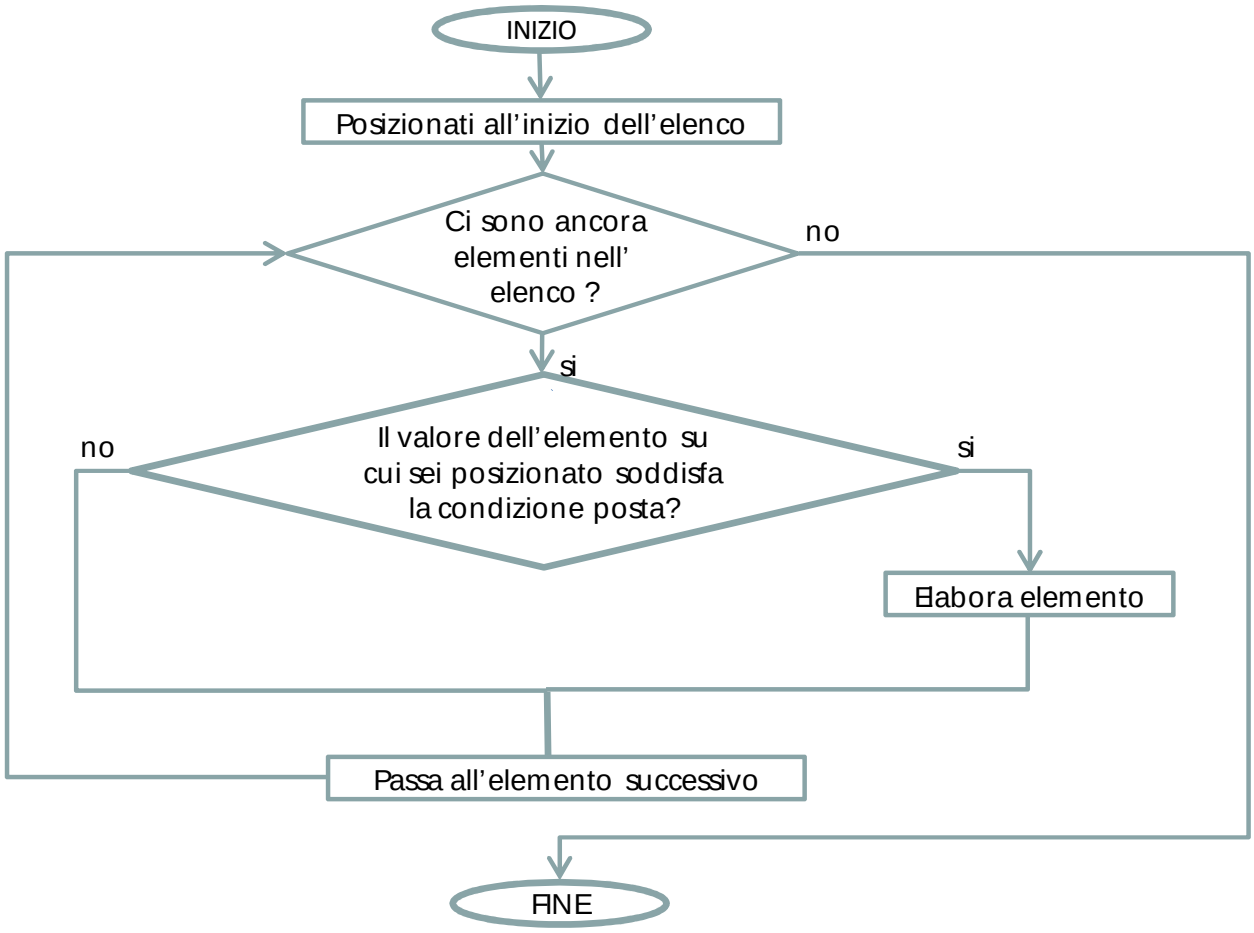
## 4.3.3. Elaborazioni classiche con i vettori

*Si propongono qui alcuni problemi di base fornendo soluzioni classiche che, seppur semplici, si ritengono significative per la formazione di base di uno studente che si occupa di informatica.*

### 4.3.3.1. Selezione

Problema→ selezionare tutti gli elementi di un elenco che soddisfano una certa condizione. Ad esempio in un insieme di voti selezionare quelli sufficienti.

Il problema è formulato in termini generali, in particolare non è specificato cosa si intende fare degli elementi selezionati: potremmo voler contare quanti sono, oppure visualizzarli, o quant'altro. Si propone un metodo di soluzione generale, che può essere utilizzato con qualsiasi tipo di dato e con qualsiasi numero di valori.




Mostriamo l'implementazione di un caso particolare, scegliendo di visualizzare i valori che soddisfano la condizione posta. Dobbiamo inoltre pensare a come memorizzare i dati: ci si riferisce come esempio ad un vettore costituito da 10 elementi di tipo intero.

Leggi il codice

```
using System;
class Program
{  static void Main( )
   {   int[] v = { 6,7,5,8,4,6,6,5,8,4,};
       for (int i = 0; i < 10; i++)
       {
           if (v[i] >=6)
           {
               Console.WriteLine("voto sufficiente: {0}", v[i]);
           }
       }
   }
}
```

### 4.3.3.2. Ricerca sequenziale

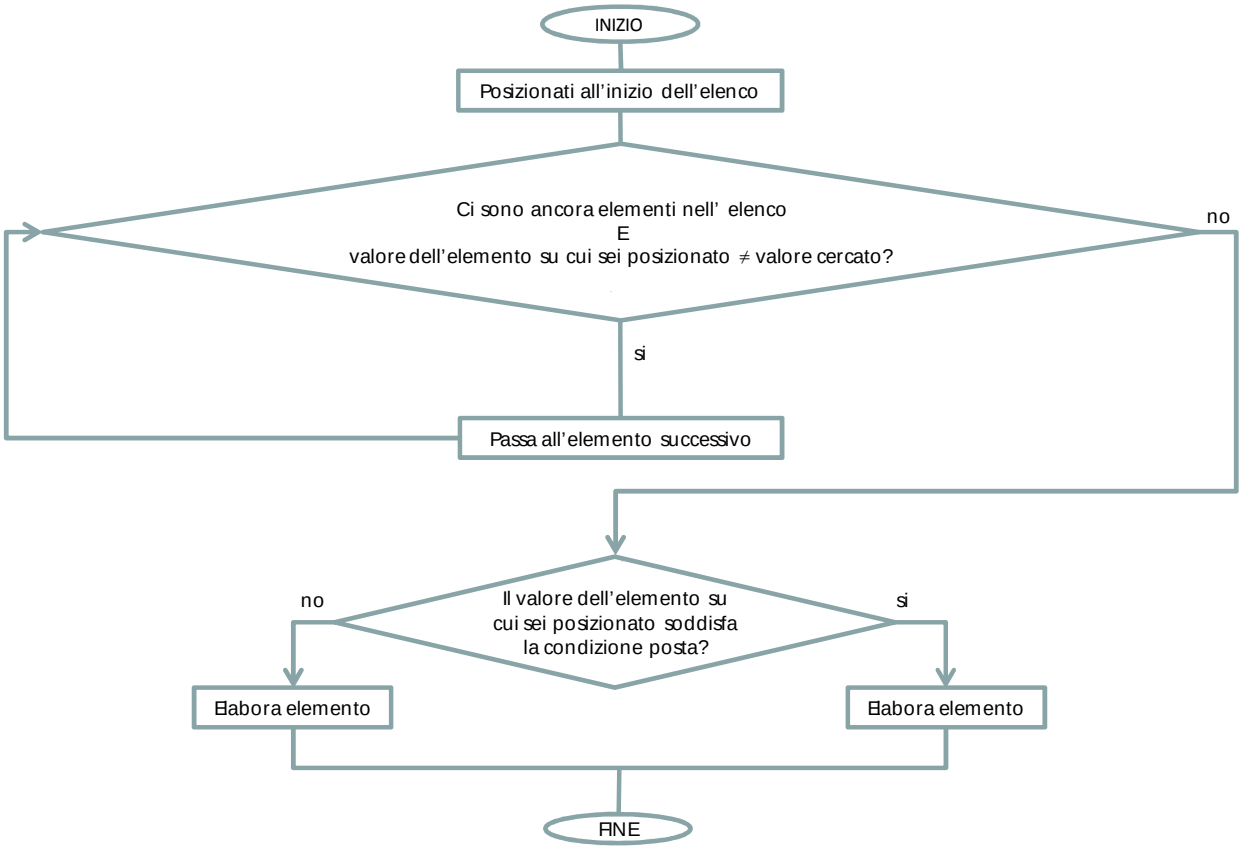
Problema → Verificare se un valore X è presente in un elenco. Ad esempio in un insieme di nomi cerca “Matteo”.

Osserviamo la differenza fra **selezione** e **ricerca**: nel primo caso abbiamo individuato tutti gli elementi che soddisfano una condizione, ora si tratta di individuare, se è presente, un elemento, quello ricercato, in un insieme dove gli elementi sono tutti diversi fra loro.

Proponiamo una soluzione generale, considerando che dovremo esaminare ad uno ad uno gli elementi e che dovremo interrompere la ricerca quando si verifica uno dei due seguenti casi:

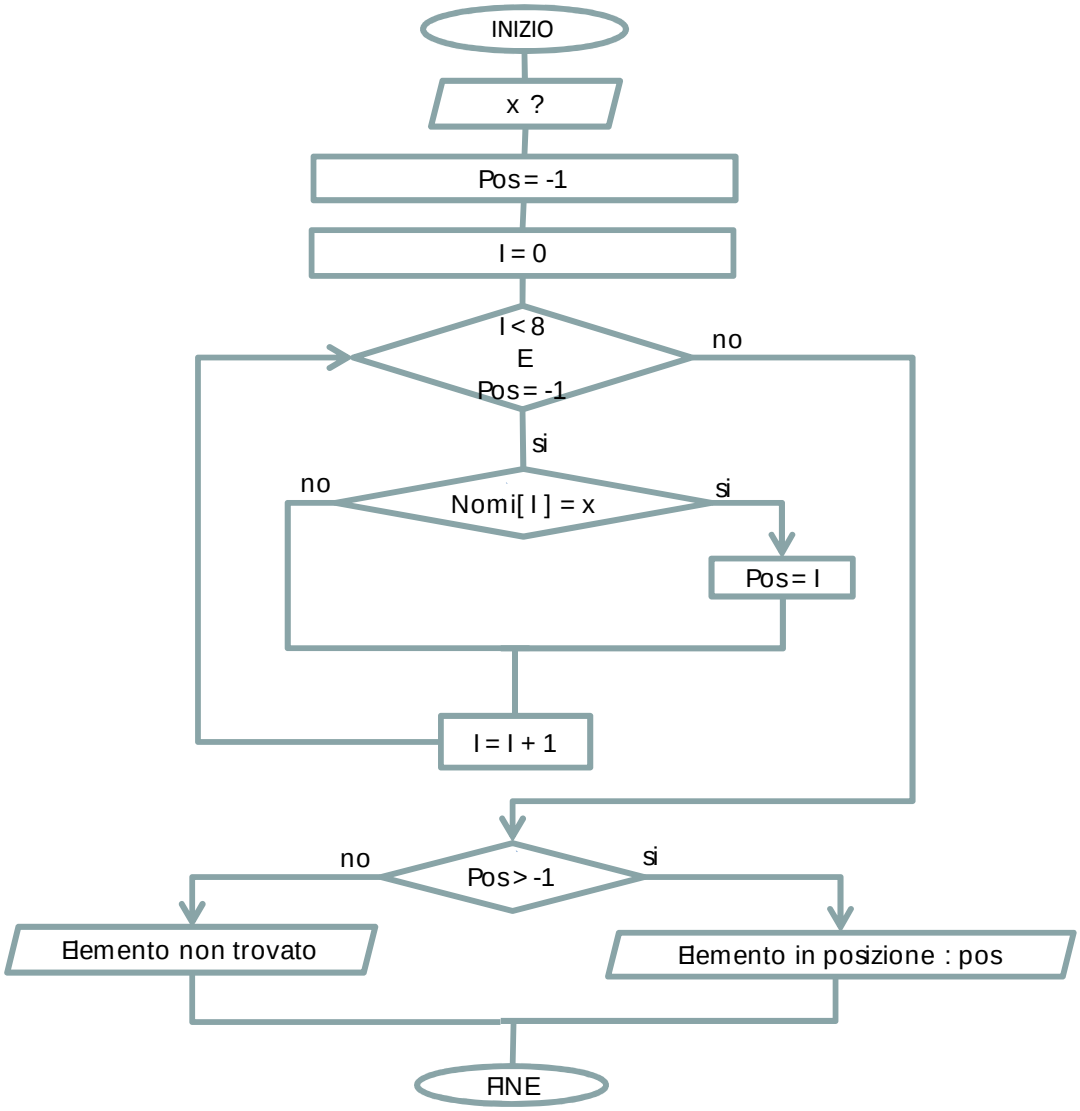
- l’elemento cercato è stato trovato
- tutti gli elementi dell’insieme sono stati controllati.

Naturalmente la ricerca può avere esito positivo, ma anche negativo, nel caso in cui l’elemento che sui sta cercando non sia presente nell’insieme dato.



Questo algoritmo è molto generale, consideriamo ora un caso più specifico: **ricerchiamo l'elemento e visualizziamo la posizione in cui esso si trova**; qualora l'elemento non venga trovato si visualizza un messaggio informativo.

Supponiamo che i valori siano costituiti da un elenco di 8 nomi memorizzati su un vettore di stringhe, indichiamo con x il nome della variabile che memorizza il valore da cercare e con Pos la posizione dell'elemento trovato. La variabile Pos viene inizializzata a -1 ad indicare che, prima di iniziare la ricerca, l'elemento non è stato trovato!



Osserva e leggi il codice corrispondente



```
using System;
class Program
{
   static void Main(string[] args)
   {
      string[] Nomi = {"Giuseppe", "Elisa", "Piero", "Maria", "Gianni", "Elisabetta", "Matteo", "Manuela"};
      string x;
      Console.Write("Quale nome stai cercando ? ");
      x = Console.ReadLine();
      int Pos = -1;
      int I = 0;
      while ( (I<8) && (Pos == -1) )
      {
         if (Nomi[I] == x)
         {
            Pos = I; //ho trovato l'elemento in posizione I
         }
         I = I + 1;
      }
      if ( Pos > -1)
      {
         Console.WriteLine("{0} è in posizione {1}",x,Pos);
      }
      else
      {
         Console.WriteLine("{0} non è presente nell'eleno",x);
      }

      Console.WriteLine("\nPremi INVIO per chiudere il programma");
      Console.ReadLine();

   }
}
```

### 4.3.3.3. Ricerca binaria

Problema→ Cercare un valore X in un vettore ordinato.

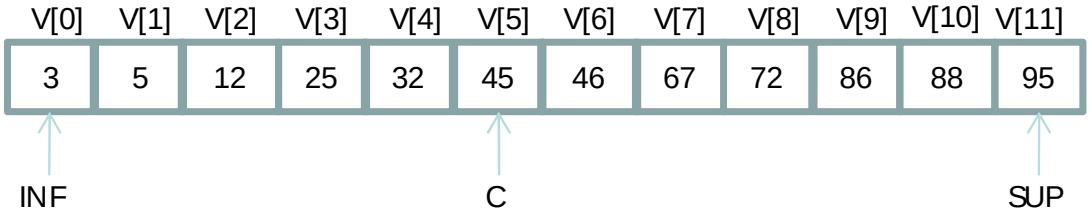
Consideriamo ad esempio un vettore con dati di tipo intero in ordine crescente, i valori potrebbero essere quelli rappresentati in figura, e proviamo a cercare in esso un dato qualsiasi, ad esempio 88.

Rispetto al problema tradizionale della ricerca qui c'è una variante interessante: sappiamo che i dati sono ordinati, quindi è ragionevole nella ricerca di una soluzione del problema avvalerci della nuova informazione al fine di rendere più efficiente l'algoritmo.

È chiaro che i dati proposti sono solo esemplificativi, dobbiamo trovare una soluzione generale, che possa essere valida qualsiasi siano i valori del vettore, e qualunque sia il valore da cercare.

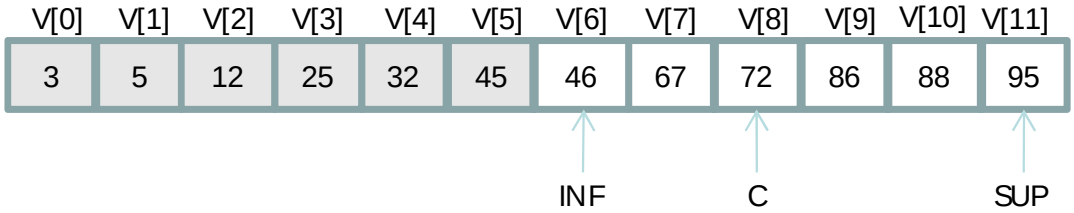
Quindi in generale noi non abbiamo alcuna indicazione relativa a dove si trova il valore, potrebbe essere in qualsiasi posizione.

Proviamo a vedere se per caso si trova nella posizione centrale (indicata con C), che calcolo come posizione intermedia fra l'estremo inferiore (INF=0) del vettore che sto considerando e l'estremo superiore (SUP = 11), scegliendo di arrotondare per difetto (questa scelta non è significativa, potrei arrotondare per eccesso); provo quindi a vedere se il valore cercato è nella posizione 5.

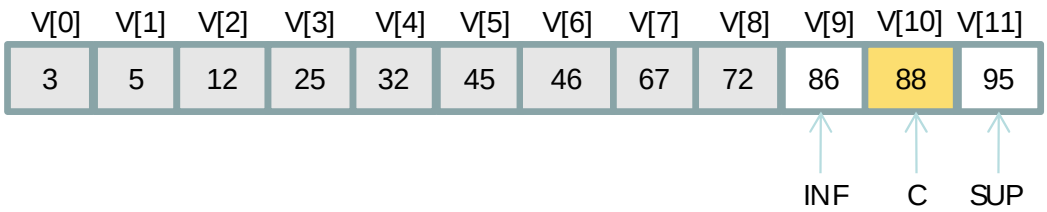



Osserva che in posizione 5 c'è il valore 45 che è minore del valore cercato: siccome i dati sono in ordine crescente, il valore 88, se è presente nel vettore, si trova sicuramente nella parte destra, dalla posizione 6 in avanti.

Nella figura qui di seguito viene rappresentata graficamente questa situazione: gli elementi nelle prime posizioni non sono più da prendere in considerazione (per questo sono stati oscurati), l'indicazione INF che rappresenta la posizione inferiore da cui cominciare a ricercare è stata spostata in corrispondenza dell'elemento in posizione 6. L'estremo superiore dell'intervallo di ricerca rimane invariato, per quanto ne sappiamo fin qui il valore 88 potrebbe benissimo trovasi proprio nell'ultima posizione del vettore. Come prima, iniziamo a cercare in mezzo, cioè individuiamo la posizione centrale fra gli elementi ancora da considerare (il valor medio fra 6 e 11, le due posizioni estreme di ricerca) che risulta essere la posizione 8.



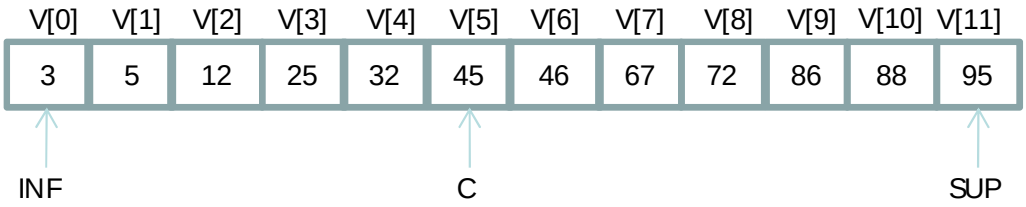

Come sopra, il procedimento è sempre lo stesso! Nella posizione 8 c'è il valore 72, quindi il valore 88, se c'è, si trova oltre la posizione 6. La figura rappresenta la nuova situazione.

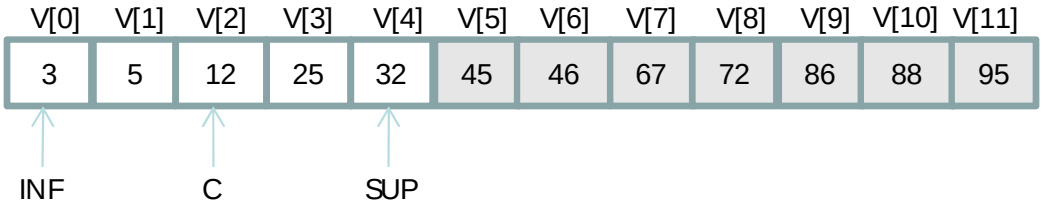


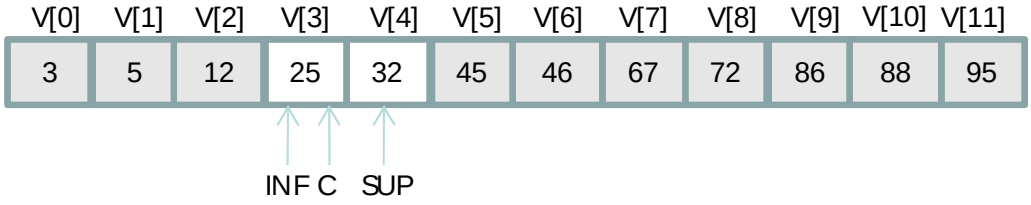

Aggiorniamo la posizione degli indicatori INF e SUP, quindi calcoliamo la nuova posizione centrale, che risulta essere la posizione 10.
Verifichiamo se troviamo l'elemento che stiamo cercando: questa volta l'esito è positivo, la ricerca si è conclusa con successo.

**Da dove deriva il nome di ricerca binaria? Rifletti!**
il vettore è costituito da 12 elementi, e fossimo stati molto fortunati avremmo potuto trovare il valore cercato al primo tentativo.
Da notare che però, pur avendo eseguito una sola prova, siamo sicuri che l'elemento, se c'è, si trova da una sola parte, quindi possiamo già scartare metà degli elementi! La lunghezza del vettore su cui dobbiamo ancora effettuare la ricerca si è dimezzata!
Questo avviene ad ogni tentativo: il secondo lo effettuo considerando solo più 6 elementi, non sono ancora fortunato, ma di nuovo metà degli elementi può essere scartata.
È chiaro che questo metodo è più efficiente della ricerca sequenziale (ha un costo però nemmeno tanto nascosto: i dati devono essere ordinati).

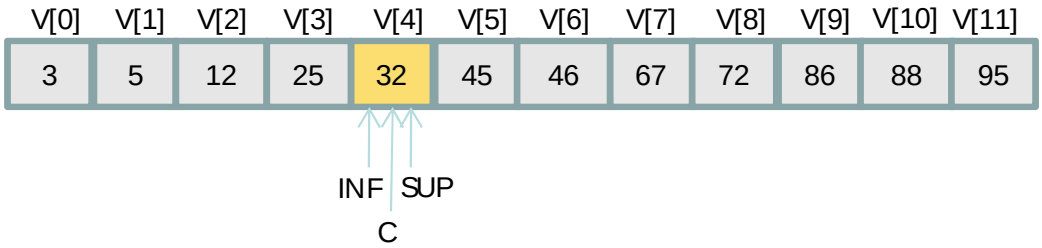
Effettuiamo un'altra prova ricercando il valore 32: si riporta solo la simulazione grafica, il ragionamento è quello precedente.



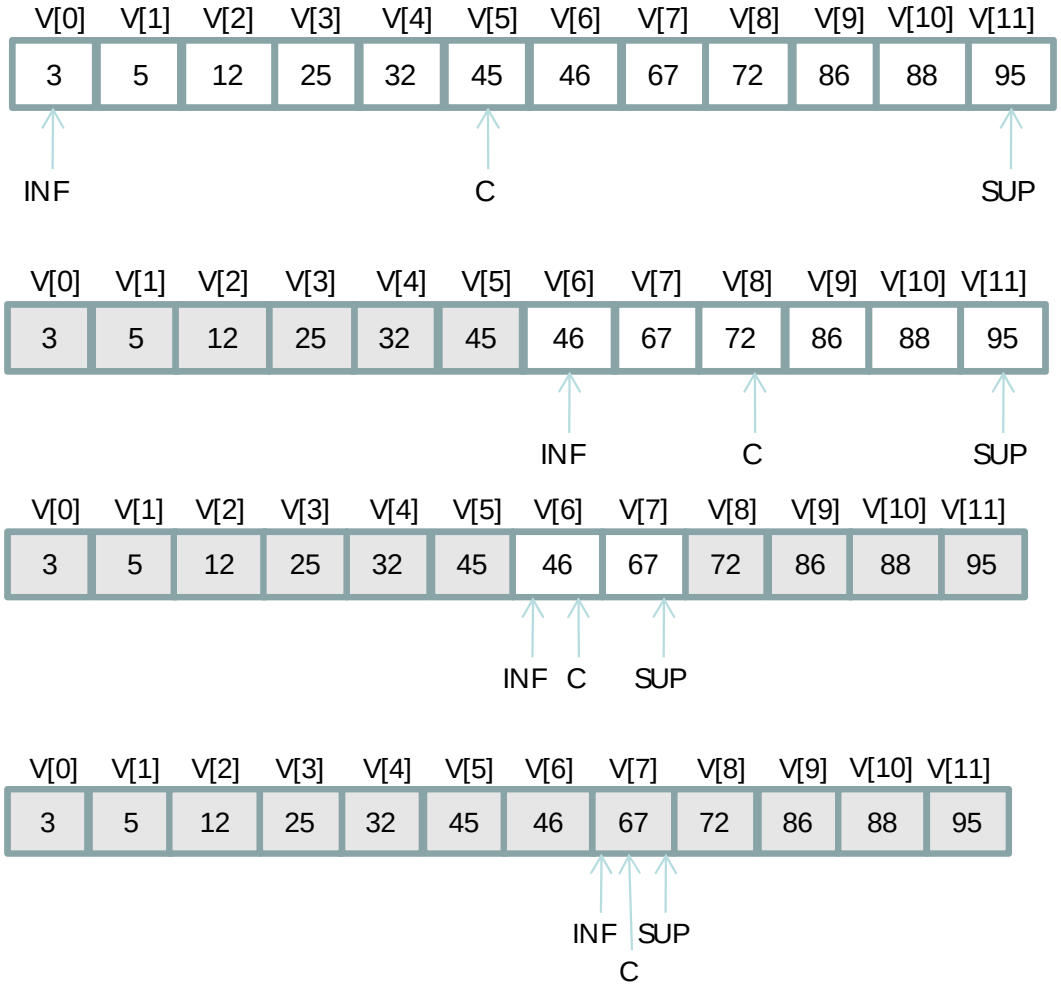




Ancora non abbiamo trovato il nostro elemento, andiamo avanti con il solito metodo: spostiamo l'estremo inferiore a sinistra (il valore se c'è è maggiore di 25), e calcoliamo la nuova posizione del massimo.




Anche questa volta la ricerca si conclude con un successo.
Cosa succede quando cerchiamo un valore che non è presente? Non possiamo sempre essere fortunati!
Proviamo a seguire il metodo della ricerca binaria per vedere se è presente il valore 70.




Ancora una volta la posizione dell'estremo inferiore e superiore di ricerca coincidono, cioè la ricerca è ormai limitata ad un solo elemento. Neanche in questa posizione abbiamo trovato il valore 70 che stiamo cercando, allora possiamo affermare con sicurezza che esso non è presente nel vettore.

Osserva e leggi il codice corrispondente

```
1   class RicercaBinaria
2   {
3     static void Main()
4     {
5       int[]v={3,5,12,25,32,45,46,67,72,86,88,95};
6       int INF=0,SUP=11,C;
7       int X=88;
8       C=(INF+SUP)/2;
9       while((v[C]!=X)&& (INF<SUP))
10      {
11        if(v[C]>X)
12        {
13          SUP=C-1;
14        }
15        else
16        {
17          INF=C+1;
18        }
19        C=(INF+SUP)/2;
20      }
21      if(v[C]==X)
22      {
23        Console.WriteLine("\ntrovato il valore {0} in posizione {1}",X,C);
24      }
25      else
26      {
27        Console.WriteLine("\nil valore {0} non è presente nel vettore",X);
28      }
29      Console.ReadLine();
30    }
31  }
```

Analizziamo in dettaglio il listato, indicando man mano le istruzioni che vengono eseguite

| | |
|---|---|
| riga 9 | la ricerca del valore prosegue finché non lo abbiamo ancora trovato (v[C] != X) purché ci siano ancor posizioni del vettore da esaminare (INF < SUP); |
| riga 21 | all'uscita dal blocco ripetizione non possiamo sapere se abbiamo trovato l'elemento, o se esso non è proprio presente, quindi dobbiamo subito effettuare questo controllo. |

**Rifletti**

Se raddoppia il numero di elementi del vettore, di quanto aumenta il numero di tentativi da effettuare?[Aumenta di uno, perché
Se ho ad esempio 1000 elementi, qual è il numero minimo di tentativi da effettuare? Quale il numero massimo? [Minimo 1, massimo 10: perché?]

### 4.3.3.4. Ordinamento

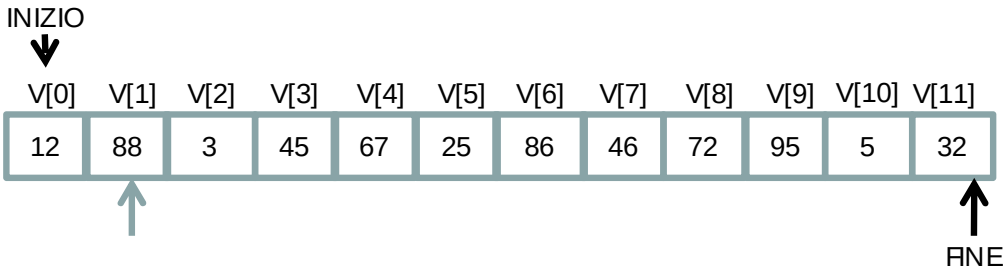
Problema→ Disporre in ordine crescente i valori di un vettore.

Consideriamo ad esempio il vettore di interi del paragrafo precedente in cui però i dati non sono disposti in ordine. Nel descrivere il procedimento risolutivo si di riferisce ad elementi del vettore in posizioni particolari, che descriviamo nella tabella seguente.

*Tabella di descrizione delle variabili*

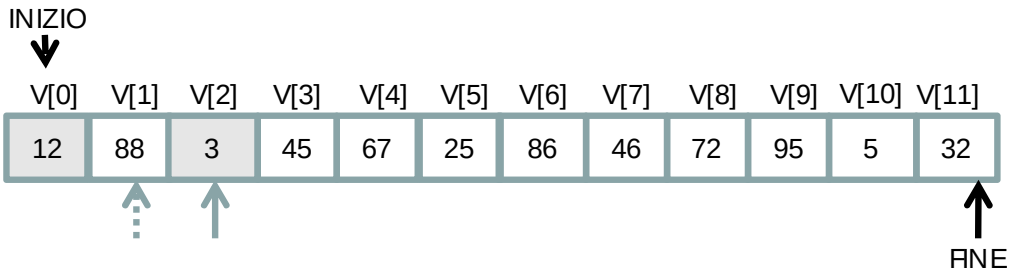
| Nome | Descrizione |
|---|---|
| V[12] | vettore da ordinare |
| INIZIO | posizione del primo elemento del vettore da ordinare |
| FINE | posizione dell'ultimo elemento del vettore da ordinare |
| I | posizione dell'elemento corrente del vettore, cioè dell' elemento che via via stiamo elaborando. |

Per risolvere il problema possiamo operare nel modo descritto di seguito: confrontiamo man mano ogni elemento del vettore con l'elemento nella posizione iniziale e, se necessario, effettuiamo uno scambio.

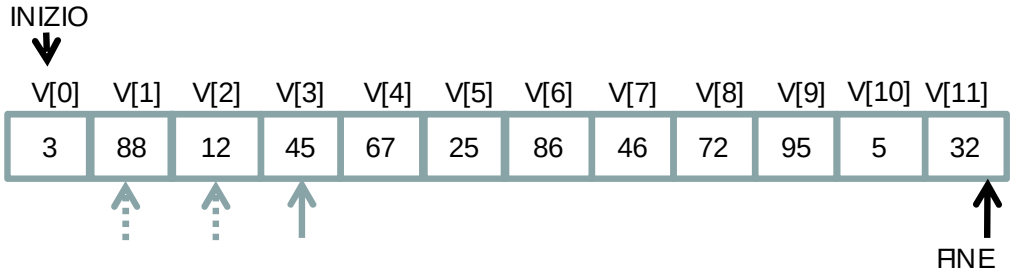



Osservo che l'elemento in posizione iniziale è V[0] e vale 12: confronto ogni elemento successivo con tale valore. Il primo confronto sarà fra V[0] e V[1]: il 12 è minore i 88, quindi i valori risultano essere in ordine crescente fa loro.

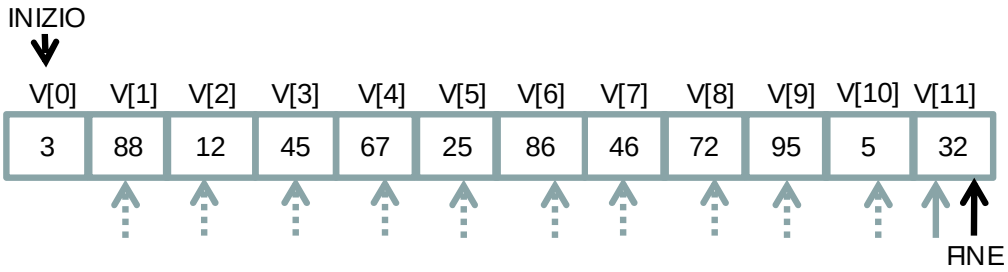
Passo all'elemento successivo, graficamente indicato dalla freccia che si sposta:




Ora i valori devono essere scambiati! Se voglio ottenere il vettore con i dati in ordine crescente i valori più piccoli si devono spostare all'inizio: effettuo lo scambio fra gli elementi evidenziati e passo a considerare l'elemento in posizione successiva.

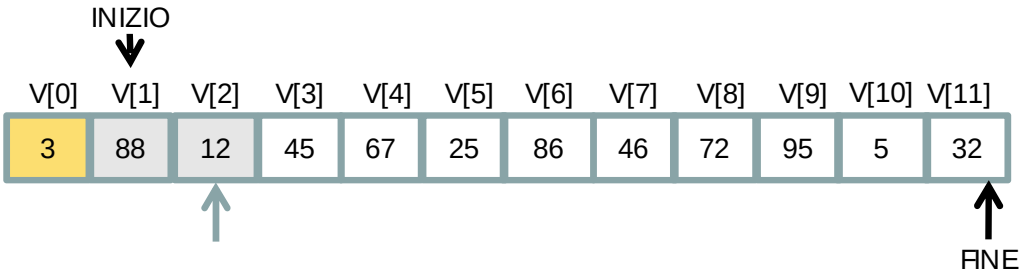



Confronto man mano tutti gli elementi con quello iniziale, se necessario effettuo gli scambi, fino ad arrivare all'ultimo elemento come indicato nella figura seguente.

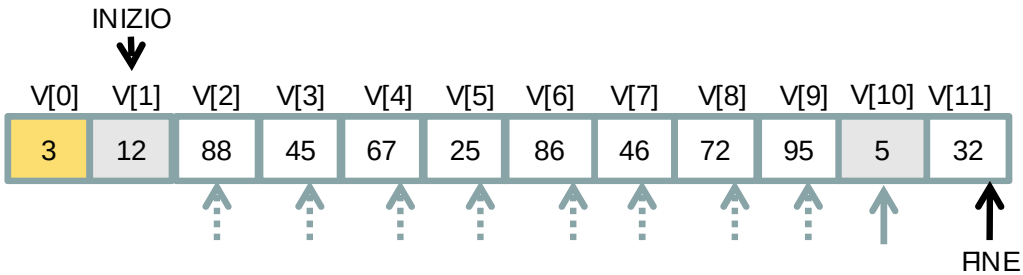



Riflettiamo su quanto è successo: ho confrontato tutti gli elementi con il primo, quando ho trovato un elemento minore di quello nella posizione iniziale ho effettuato lo scambio, quindi nella posizione iniziale si trova sicuramente il valore più piccolo. Il vettore non è ancora tutto ordinato, ma una parte sì: sono sulla buona strada, anche se il compito è piuttosto impegnativo.

Riprendo considerando non più tutti gli elementi del vettore, ma solo quelli ancora da mettere in ordine. Graficamente mettiamo in evidenza la parte del vettore già ordinata (solo il primo elemento per ora), e spostiamo la freccia inizio che indica la posizione dell'elemento iniziale del vettore (ancora) da ordinare.

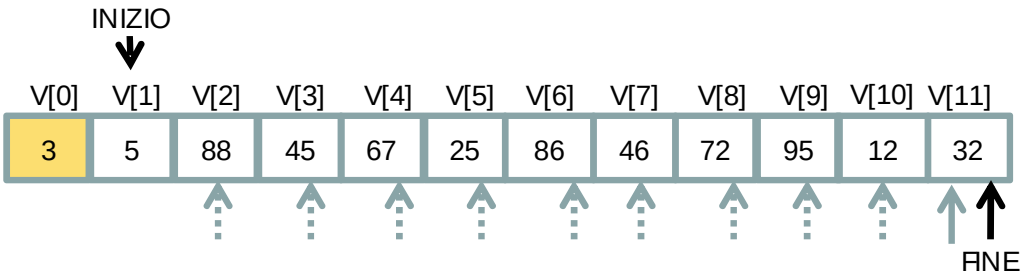



Come prima, confronto tutti gli elementi del vettore da ordinare, ad uno ad uno , con quello iniziale (che ora è quello in posizione 1): il valore 12 è minore di 88, quindi effettuo subito uno scambio! Poi vado avanti man mano.

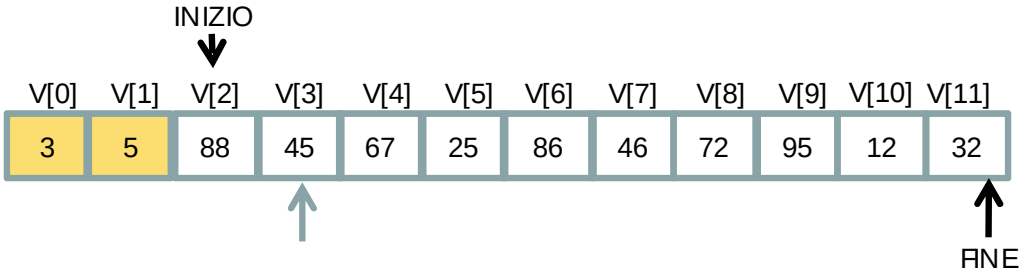



o.

Ho trovato un altro valore da scambiare: il 5, essendo più piccolo del 12, passa all'inizio, quindi si continua la scansione degli elementi che restano fino ad arrivare alla fine.[61]




Abbiamo terminato la seconda scansione, il risultato ottenuto è rappresentato in figura:



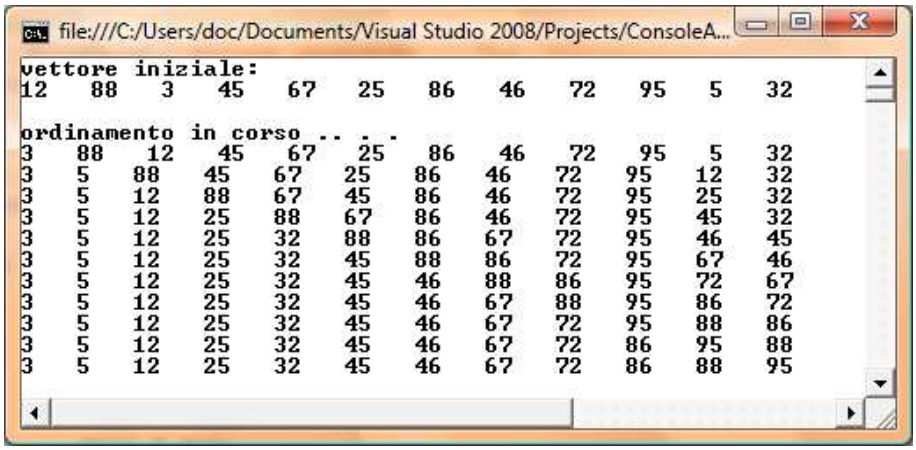

È chiaro che ogni volta che si fa una scansione completa si ottiene che il più piccolo degli elementi passa nella posizione iniziale, così alla scansione successiva si riduce il numero di elementi del vettore ancora da ordinare.

L'algoritmo risolutivo avrà quindi un blocco ripetizione corrispondente ad ogni singola scansione nella quale si elaborano tutti gli elementi, a partire dalla posizione iniziale fino a quella finale, effettuando i confronti e gli eventuali scambi.
Ciascuna di queste scansioni dovrà però essere a sua volta ripetuta, quindi avremo un altro blocco ripetizione (i due blocchi risultano annidati).

Cosa cambia fra una scansione completa e la successiva? Si modificano le dimensioni della parte di vettore già ordinato e di quella ancora da ordinare. Più specificatamente si sposterà l'indicatore che abbiamo chiamato INIZIO, che rappresenta la posizione iniziale del vettore ancora da ordinare. Inizio parte dalla posizione zero, e man mano si sposta verso la fine.

[61] Un metodo di ordinamento simile a quello qui illustrato è noto con il nome di "bubble sort" ad indicare proprio i valori più piccoli che *salgono* verso le prime posizioni come fossero delle bolle.

Osserva e leggi il codice corrispondente




```
using System;
   class Ordinamento
   {
      static void Main()
      {
         int[] v = { 12,88,3,45,67,25,86,46,72,95,5,32};
         int FINE = 12;
         int xyz;
         //visualizzo il vettore iniziale
         Console.WriteLine("vettore iniziale:    ");
         for (int x = 0; x < FINE; x++)
            Console.Write("{0}   ", v[x]);
         Console.WriteLine();

         Console.WriteLine("\nordinamento in corso .. . . ");
         for (int INIZIO = 0; INIZIO < FINE-1; INIZIO++)
         {
            for (int I = INIZIO + 1; I < FINE; I++)
            {
               if (v[INIZIO] > v[I])
               {   //scambio
                  xyz = v[INIZIO];
                  v[INIZIO] = v[I];
                  v[I] = xyz;
               }
            }
            //visualizzo il vettore dopo ogni scansione
            for (int x = 0; x < 12; x++)
               Console.Write("{0}   ", v[x]);
            Console.WriteLine();
         }
         Console.ReadLine();
      }
   }
```