Array (*vettore/matrice*)

Un dato strutturato.

Un insieme di variabili tutte dello stesso tipo identificate con un nome comune.
Uno specifico elemento è referenziato tramite un ***indice*** (*funzione di accesso*)

Pensiamo all'array come ad un insieme di cassetti numerati; per accedere al contenuto di uno specifico cassetto si deve specificare il numero d'ordine ad esso associato (*indice*)

Accesso diretto

Gli elementi dell' array vengono allocati in celle di memoria adiacenti. Per tale motivo è necessario dichiarare l' array prima del suo utilizzo in modo che il compilatore possa riservare in memoria lo spazio per conservarlo

Sintassi dichiarazione di array monodimensionale :

identificatore-tipo ***nome_variabile*** **[cardinalità] ;**

int numero[10];

Dichiarazione di un array chiamato **numero** contenente un massimo di 10 elementi interi.

Il C++ assegna gli indici agli elementi di un array a partire da 0; gli elementi dell'array di interi appena dichiarato vanno da **numero[0] a numero[9].**

```
int vettore[ ]={1,2,3,4,5}
```

→

Dichiarazione valida

Dalla lista degli inizializzatori ricava la cardinalità di vettore

Inizializzazione elementi array

- **Per singola inizializzazione direttamente nel programma;**

  a[0]=5; …… a[9]=9;

- **Per inizializzazione in fase di esecuzione (da tastiera)**

```
for (i=0;i<riemp;i++)
cin >> a[i];
```

Riemp <=cardinalità

- **Direttamente alla dichiarazione della struttura:**

```
int a[10]={ 5,8,3,6,2,0,4,1,7,9 }
```

int a[ ]={1,2,3,4,5} → **Dichiarazione di array valida**

Dalla lista degli inizializzatori ricava la cardinalità del vettore

```
#include <iostream>
#include <stdlib.h>
// prova325.cpp  ERRORE- Numero inizializzatori > cardinalità
int main()
{
    int vettore[10]={11,32,11,12,13,14,15,16,17,18,19,20};
    system("PAUSE");
    return 0;
}
```

ERRORE

Numero elementi inizializzati **>** Cardinalità dichiarata

Per inizializzazione in fase di esecuzione (da tastiera)

Scrittura array

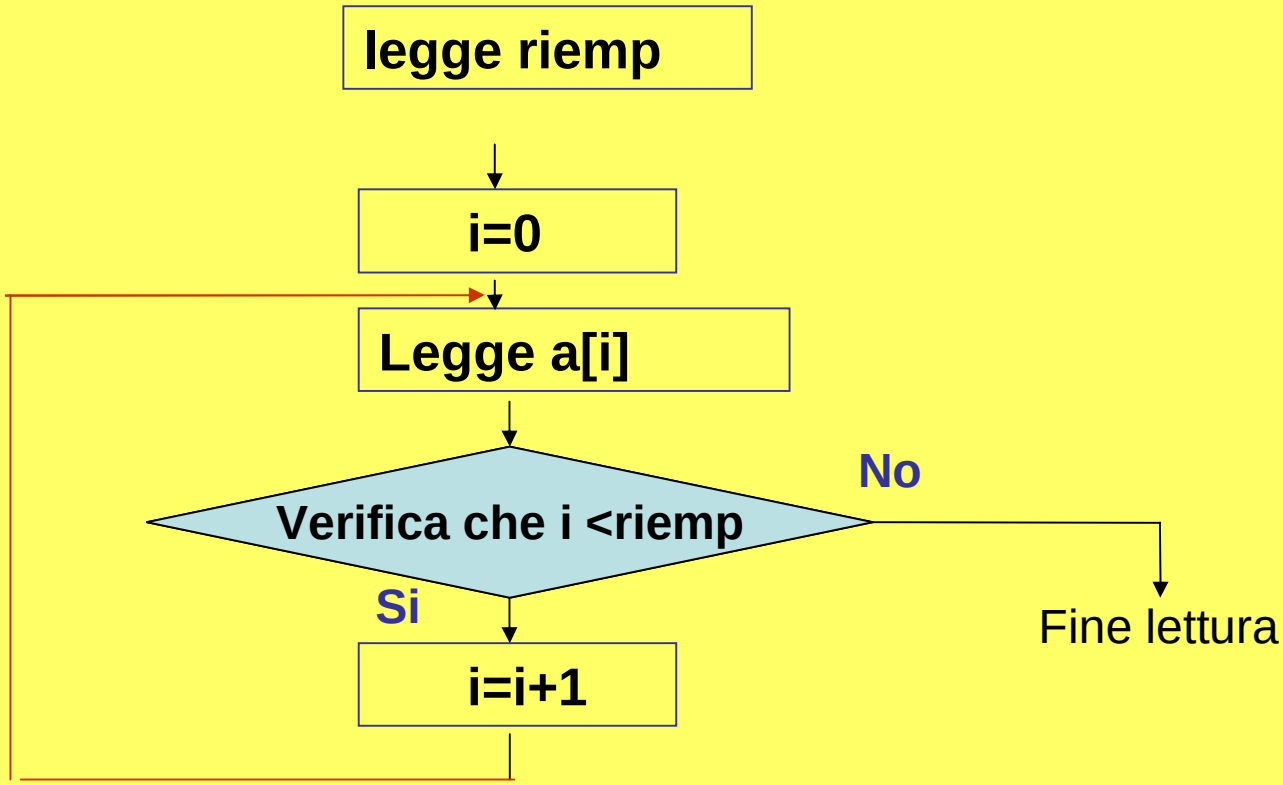
```
for (i=0; i < riemp; i++)
cin >> a[ i ];
```

5

Riemp

7	4	3	2	1	0	0	0	0	0

a [10]



```
/* prova13.cpp  Generazione di un vettore di max 10 interi e calcolo
 della media aritmetica dei suoi elementi */
void  main()
{
int numero[10], conta, somma, N;
float media;
somma=0;
cout << "inserire nro elementi"; cin >> N;
if (N <=10)
{
for (conta=0;conta<=N-1;conta++)
{
cin>>(numero[conta]);  somma+=numero[conta];
}
for (conta=0;conta<=N-1;conta++) cout <<(numero[conta]);
cout <<endl <<("somma calcolata =") <<somma <<endl;
media=(float (somma) / N);
cout <<"media =" <<media;
}
else
cout << "attenzione alla cardinalità del vettore ";
system("PAUSE");
}
```

- **Inizializzazione elementi mediante operazione di lettura da tastiera;**

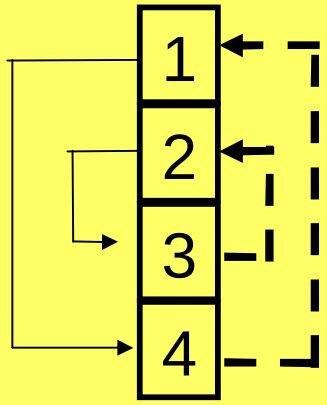
Esercizio 107

Assegnato un vettore di interi di riempimento N, produrre la trasposizione del vettore

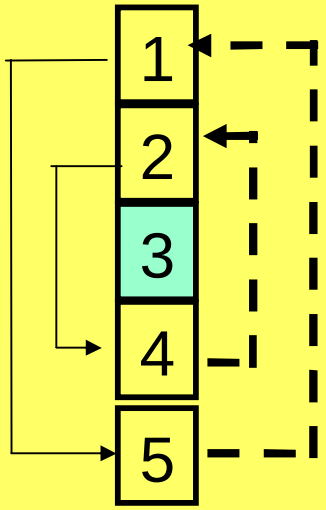
N=7

Vett = 1 2 3 4 5 6 7 ⟶ 7 6 5 4 3 2 1

a) N pari



b) N dispari




```
dimez=N/2;
for (int i=0; i<dimez; i++)
        {
          temp = vett [ N-i-1 ];
          vett [ N-i-1 ] = vett[ i ];
          vett[ i ] = temp;
        }
```

```
#include <iostream>
//  Esercizio107 Trasposizione di un vettore
using namespace std;
int main()
{
      int vett[ 10 ];
      int  N,dimez,temp;
      cout <<" Riempimento =";cin >>N;cout<<endl;
      cout <<"Lettura N valori del vettore"<<endl;
      for ( int i=0; i<N; i++)
      cin>>vett[ i ];
      dimez=N/2;
```

```
// Effettua scambi
    for (int i=0; i<dimez; i++)
      {
        temp=vett[ N-i-1 ];
        vett[ N-i-1 ]=vett[ i ];
        vett[ i ]=temp;
      }
// Produce stampa
    for (int i=0;i<N;i++)
    cout<<vett[ i ]<<"   ";
    cout<<endl;
    system("PAUSE");
    return 0;
}
```

Esercizio 2

Assegnato un vettore di max 10 valori interi, determinare **valore max** e **valore min** e rispettive posizioni (***indice***)

7 2 10 1 5 4

Max = 10
Min = 1
Posmax =3
Posmin =4

5 7 2 10 1 4 3 8 9 5 6 3

Riempimento eccedente cardinalità dichiarata

Descrizione algoritmo

Si parte supponendo che gli elementi MAX e MIN coincidano con il primo elemento del vettore; di conseguenza Posmax e Posmin sono localizzate in posizione indice 0 del vettore

Gli elementi MAX e MIN vengono confrontati in sequenza con tutti gli elementi del vettore a partire dall'elemento in seconda posizione (indice 1)

Durante i confronti se si trova un valore maggiore di MAX si provvede a caricarlo in MAX e ad aggiornare Posmax con l'indice dell'elemento trovato.
Stesso aggiornamento qualora si trovi un valore minore di MIN

5 7 2 10 1 4

Maxcorrente=5
Posmax=0
Indice vettore=1
Max corrente =5 < vett[1]=7 → Maxcorrente =7 Posmax=1; (Si)
Indice vettore=2
Max corrente =7 < vett[2]=2 → Maxcorrente =7 Posmax=1;
Indice vettore=3
Max corrente =7 < vett[3]=10 → Maxcorrente =10 Posmax=3; (Si)
Indice vettore=4
Max corrente =10 < vett[4]=1 → Maxcorrente =10 Posmax=3;
Indice vettore=5
Max corrente =10 < vett[5]=4 → Maxcorrente =10 Posmax=3;

Analisi delle informazioni

➢Informazioni di ingresso

Nome	Tipo	Descrizione significato
Vettore	vettore di interi di cardinalità 10	vettore su cui operare
N	variabile intera	riempimento del vettore

➢Informazioni di uscita

Nome	Tipo	Descrizione significato
Max	Variabile intera	Max corrente
Min	variabile intera	Min corrente
Posmax	Variabile intera	Posizione max corrente
Posmin	Variabile intera	Posizione min corrente

➢Informazioni di algoritmo

Nome	Tipo	Descrizione significato
Valore	**Variabile intera**	**Elemento del vettore che viene letto**
I	**variabile intera**	**Indice del vettore**

```
Inizio
Leggi riempimento del vettore;
Se riempimento  <= 10  allora
   inizio blocco1
      Leggi vettore;
      Assegna a maxcorrente primo valore del vettore;
      Assegna a mincorrente primo valore del vettore;
      Assegna a posizione-maxcorrente 0;
      Assegna a posizione-mincorrente  0;
      per posizione indice corrente da 1 a riempimento-1 incrementandosi di 1
          inizio blocco2
              Se maxcorrente < elemento corrente vettore  allora
                 inizio blocco3
                     Aggiorna maxcorrente con elemento corrente;
                     Aggiorna posmaxcorrente con indice corrente;
                 fine blocco3
              Se mincorrente > elemento corrente vettore  allora
                 inizio blocco4
                     Aggiorna mincorrente con elemento corrente;
                     Aggiorna posmincorrente con indice corrente;
                 fine blocco4
          fine blocco2
```

Produci risultati

fine blocco1

altrimenti

Produci frase “ Riempimento eccedente cardinalità del vettore”:

Fine

```
#include <iostream>
using namespace std;
// Assegnato un vettore max di 10 valori interi,
// determinare valore max e min e rispettive posizioni (indice)
int main()
{
    int vettore [10];
    int max, min, posmax,posmin,i,n, valore;
    cout << "riempimento del vettore";
    cin >> n;
    if ( n <= 10)  {
              for (i=0; i <= n-1; i++) {
                                  cout << "vettore[" <<i  <<" ]=" ;
                                  cin >> valore;
                                  vettore[i]=valore;
                                  cout << endl;
                                   }
               min=vettore[0];
               max=vettore[0];
               posmax=0;
               posmin=0;
```

```
                for (i=1; i <= n-1; i++)
                                {
                                if ( max < vettore[i])
                                    {
                                      max=vettore[i];
                                      posmax=i;
                                    }
                                if ( min > vettore[i])
                                    {
                                      min=vettore[i];
                                      posmin=i;
                                    }
                                }
                        cout <<max;
                        cout << posmax <<endl;
                        cout <<min;
                        cout << posmin;
                         }
                          else
                         cout << "Riempimento eccedente cardinalità dichiarata";
    system("PAUSE");
}
```

Esercizio 109

Assegnato un valore decimale intero N, produrne la rappresentazione in una base b (binaria, ottale o esadecimale)

```
#include <iostream>
using namespace std;
//  Esercizio 109  Conversione di base (2-8-16)
int main()
{
  int numero, base, dimez, temp, appo;
  char caratt[16]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
  int cifre[10];
  cout<<"\nDammi il numero in base 10 da convertire: ";
  cin>>numero;
  cout<<"Dammi la base: ";
  cin>>base;
  int i=0;
  while (numero>=base)
  {
    cifre[ i ] = (numero % base);
    numero /= base;
    i++;
  }
```

```
    cifre[ i ]=numero;
    dimez = i/2;
    for (int j=0; j<=dimez; j++)
    {
       temp=cifre[ j ];
       cifre[ j ]=cifre[ i-j ];
       cifre[ i-j ]=temp;
       }
    for (int j=0; j<=i; j++)
    {
    appo=cifre[ j ];
    cout <<caratt[appo]<<" ";
}
system("PAUSE");
return EXIT_SUCCESS;
}
```

Esercizio 147

Sia assegnato un vettore A di interi di riempimento N.

Nella ipotesi che il primo elemento A[0] sia minore del secondo elemento A[1] si eliminino dal vettore tutti gli elementi che risultano inferiori ad A[0] oppure superiori ad A[1].

riempimento N=10
vettore ingresso A → 2 10 8 7 6 13 1 4 11 20
vettore uscita A → 2 10 8 7 6 4

```
#include <iostream.h>
#include <stdlib.h>
using namespace std;
// Esercizio 147 eliminazione elementi del vettore
int elimina (int *A, int I, int K)
 {
  int J;
  for (J=I+1 ; J<=K; J++)
     A[ J-1 ] = A[ J ];
  return 0;
 }
int main()
{
  int  A[ 10 ];
  int N, I ,Z;
  cout << "Riempimento =";cin >>N; cout <<endl;
  for (I=0; I <=N-1 ; I++)
  {
  cout <<"A[ " <<I<<"]=";cin >>A[ I ];cout<<endl;
  }
```

```
if (A[ 0 ] < A[ 1 ])
{
   I=2;
   while (I <=N-1)
    {
      Z=0;
      if ((A[ I ] <A[ 0 ]) || (A[ I ] >A[ 1 ]) )
      {
        Z=1;
        elimina(&A[0], I, N-1);
        N=N-1;
      }
      if (Z <1)
      {
        I=I+1;
        Z=0;
      }
    }
```

```
    cout <<" Riempimento ridotto a -->" <<N <<endl <<endl;
    for (I=0; I<N; I++)
      cout <<A[ I ] <<"  ";
    cout <<endl;
  }
  else
    cout <<"non verificato A[0] < A[1]";
system("PAUSE");
return 0;
}
```

Esercizio 103

Assegnato in ingresso un vettore di N elementi interi, individuare tutte le possibili terne di valori consecutivi ordinate in senso crescente.

N=12

vettore ingresso = 1 10 12 14 4 5 -6 8 9 1 2 3

Terne

1	10	12
10	12	14
-6	8	9
1	2	3

Terna maggiore somma pos 2 valore 36

```
#include <iostream.h>
using namespace std;
//  Esercizio103  terne valori consecutivi in vewttore
int main()
{
      int vett[ 20 ];
      int N;
      int SommaP, SommaC, pos;
      cout <<"Riempimento vettore =";cin >>N;cout<<endl;
      cout <<"Lettura vettore"<<endl;
      for ( int i=0; i<N; i++)
      cin >>vett[ i ];
      cout <<endl;
      int  k=0;
      SommaP=SommaC=vett[0];
```

```
    for ( int i=0; i<N-2; i++)
    if ((vett[ i ]<vett[ i+1 ]) && (vett[ i+1 ]<vett[ i+2]) )
      {
      k=k+1;
      cout << k<<"-->"<<vett[ i ]<<"  "<<vett[ i+1 ]<<"  "<<vett[ i+2]<<endl;
      SommaC=vett[ i ]+vett[ i+1]+vett[ i+2];
      if ( SommaP < SommaC)
      {
      pos=k;
      SommaP=SommaC;
      }
      }
    cout<<" Terna di somma maggiore ";
    cout<<":posizione-->"<<pos<<" valore -->"<<SommaP<<endl;
    system("PAUSE");
    return 0;
}
```

E se non sono individuate terne?

Esercizio 170

E’ assegnata in input una sequenza di cifre decimali terminata dalla coppia 0 0.

Per ogni coppia di cifre consecutive non includenti lo zero si calcoli il valore del numero costituito da quelle cifre e si generi una tabella che per ogni valore riporti il numero di volte che è stato trovato

8 4 5 8 0 4 5 5 8 4 2 0 5 8 0 8 5 5 8 0 8 4 2 2 0 6 8 4 5 0 0

1	4	42	2
2	3	85	1
3	4	22	1
4	2	68	1

```
#include <iostream.h>
#include <stdlib.h>
using namespace std;
//  Esercizio170
int main()
{
      int  A[ 80 ],val[ 80 ],tot[ 80 ];
      int valore,conta;
      int  j,i,K;
      bool flag;
      for (int j=0; j<80; j++)
      {
      tot[ j ]=0;
      val[ j ]=0;
      }
      conta=1;
      //  Lettura anticipata dei primi due elementi
      cout <<"primi due elementi  "; cin >>A[ 0 ]; cin>>A[ 1 ];
```

```
// Determina numero elementi costituenti la sequenza
while ((A[ conta ] != 0) || (A[ conta-1 ] !=0 ) )
{
  conta=conta+1;
  cin >>A[ conta ];
}
conta=conta-1;
cout <<"Nro cifre lette "<<conta<<endl;
i=0;
K=0;
if (A[ i ] ==0)
   i= i+1;
while ( i < conta)
{
  if (A[ i ] !=0)
    if (A[ i+1 ] !=0)
    {
    valore=A[ i ]*10+A[ i+1 ];
    i=i+1;
```

```
//  Forza primo valore in vettore val
if (K == 0)
  {
  tot[ K ]=1;
  val[ K ] = valore;
  K=K+1;
  }
else
  {
  flag=false;
  // Se valore gia in totalizzazone solo incremento contatore
  for (int j=0; j<K; j++)
   if (valore == val[ j ])
   {
      flag=true;
      tot[ j ]=tot[ j ]+1;
   }
```

```
        // Valore non già in totalizzazione
        if (! flag)
          {

            val[ K ]=valore;
            tot[ K ]=1;
            K=K+1;
          }
         }
        }
        else
        // caso di coppia  0  #0
        i=i+2;
    }
    for (int m=0; m<K; m++)
    cout <<val[m] <<"  "<<tot[m]<<endl;
    system("PAUSE");
    return 0;
}
```

Esercizio 140

Assegnata in ingresso una sequenza di caratteri, individuare i possibili **caratteri → cifra** e generare , con i suddetti caratteri, il valore numerico numero procedendo nella escursione della sequenza da sinistra verso destra.

La sequenza in input è terminata dal carattere “ * “

ABC1DFG23HIL5FGH5KK*

12355

Nella codifica ASCII i caratteri da “0” a “9” sono rappresentati con i valori interi da 48 a 57.

Pertanto le cifre da 0 a 9 sono date dalla loro rappresentazione carattere - 48.

Ipotesi di algoritmo

Leggi primo elemento
Mentre elemento letto è diverso da *
blocco
    Se è una possibile cifra
        Aggiorna numero
    Leggi elemento successivo
fine blocco

```
#include <iostream>
using namespace std;
//  Esercizio140
int main()
{
    char seq[20];
    int i=0;
    int numero=0;
    cout <<"sequenza di ingresso =";
    cin >>seq[ i ];
    while (seq[ i ] != '*')
    {
        int cifra=seq[ i ];
        if ((cifra > 47) && (cifra < 60))
        numero=numero*10+(cifra-48);
        i=i+1;
        cin >>seq[ i ];
}
     if (i != 0)
     cout <<"Numero estrapolato ="<<numero<<endl;
     else
     cout <<"sequenza vuota" <<endl;
system("PAUSE");
return EXIT_SUCCESS;
```

```
sequenza di ingresso =abc1dfg23hil5fgh5kk*
Numero estrapolato =12355
Premere un tasto per continuare . . .
```

Esercizio 139

Assegnati N valori interi ed un valore di riferimento, individuare tutte le coppie di valori la cui somma superi il valore di riferimento

1 5 7 2 3 8 4

Riferimento =11

9	5
9	7
9	3
9	8
9	4
5	7
5	8
7	8
8	4

Metodo di soluzione

Ogni elemento (dal primo al penultimo) viene relazionato nella somma con tutti i rimanenti elementi

```
for (int i=0;i<N-1;i++)
    for (int j=i+1;j<N;j++)
```

Se la somma supera il valore di riferimento la coppia individuata viene memorizzata in un vettore di appoggio

```
if ((vett[ i ]+vett [ j ]) > rifer)
    {
      k=k+1;
      appo[ k ][ 0 ]=vett[ i ];
      appo[ k] [ 1]=vett[ j ];
      }
```

```
#include <iostream>
using namespace std;
// Esercizio139
int main()
{
    int vett[10];
    int appo[20][3];
    int N,rifer,k;
    cout <<"Nro valori = ";cin >>N;cout <<endl;
    cout <<"valore riferimento = ";cin >>rifer;
    for (int i=0;i<N;i++)
     cin >>vett[ I ];
     k=-1;
    for (int i=0;i<N-1;i++)
     for (int j=i+1;j<N;j++)
       if ((vett[ i ]+vett [ j ]) > rifer)
       {
          k=k+1;
          appo[ k ][ 0 ]=vett[ i ];
          appo[ k] [ 1]=vett[ j ];
         }
    for (int i=0; i<=k; i++)
       cout <<appo[ i ][ 0 ] <<"   "<<appo[ i ][ 1 ]<<endl;
system("PAUSE");
return EXIT_SUCCESS;
}
```

```
Nro valori = 7

valore riferimento = 11
9 5 7 2 3 8 4
9   5
9   7
9   3
9   8
9   4
5   7
5   8
7   8
8   4
Premere un tasto per continuare . . .
```

Esercizio126

Assegnato un vettore di N elementi interi, individuare nel vettore la più lunga sequenza di numeri pari, producendo:

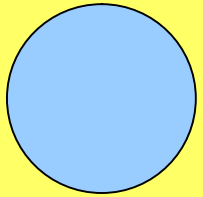
- la lunghezza
- la sequenza
- la posizione di inizio nel vettore

1 12 3 12 3 65 4 6 65 6 12 8

Lunghezza =3
Sequenza = 6 12 8
Posizione =9

Ipotesi algoritmo

Indice vettore =0
Contatore_max_seq=0
Contatore_nuova_seq=0
Mentre indice vettore < riempimento vettore
blocco 0
  Se elemento corrente è pari
      incrementa contatore_nuova_seq
  altrimenti
      blocco1
    Se contatore_nuova_seq >contatore_max_seq
        blocco2
          aggiorna contatore_max_seq
          determina posizione partenza
        fine blocco2
    azzera contatore_nuova_seq
    fine blocco1
fine blocco 0
Tratta ultimo elemento



Tratta ultimo elemento

Se ultimo elemento del vettore non è dispari non si transita nel *blocco2* (aggiornamenti) e potrebbe esistere una sequenza di pari non trattata.

Se ultimo elemento è pari
  Se contatore_nuova_seq >contatore_max_seq
    Aggiorna contatore_max_seq
    Aggiorna posizione

```
#include <iostream>
using namespace std;
//  Esercizio126 Determinazione sequenza max di pari
int main()
{
    int  vettore[20];
    int conta,contap,N,pos;
    cout <<"Riempimento-->";cin >>N; cout <<endl;
    conta=0;contap=0,pos;
    for (int i=0; i<N; i++)
    {
    cout <<"Vettore[ " <<i<<" ]=";cin >>vettore[ i ];cout <<endl;
    }
```

```
for (int i=0; i<N; i++)
 if ((vettore[ i ] %2) ==0)
      conta=conta+1;
 else
 {
     if (conta >contap)
     {
         contap=conta;
         pos=i-conta;
     }
    conta=0;
 }
// tratta ultimo elemento se pari
// potrebbe esserci sequenza di pari finale non chiusa
 if ((vettore[N-1] %2) ==0)
     if(conta>contap)
     {
         contap=conta;
         pos=N-conta;
     }
```

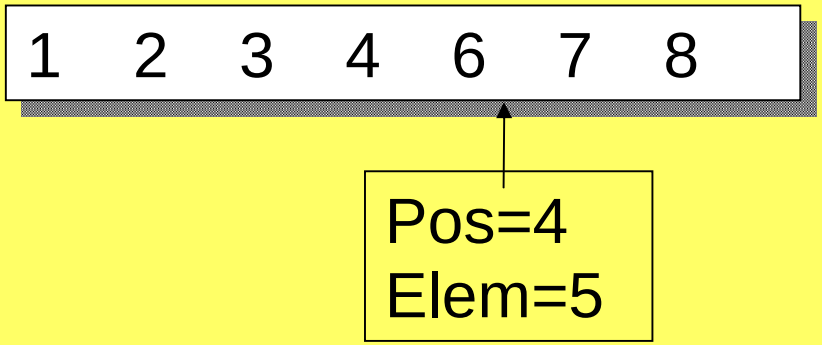
1
2
4
5
6
4
8

```
    cout<< "lunghezza sequenza max-->"<<contap<<endl;
    cout<< "posizione partenza-->"<<pos<<endl;
    cout<< "sequenza individuata-->";
    for (int i=pos;i<pos+contap;i++)
            cout <<vettore[ i ]<<"  ";
    cout<<endl;
system("PAUSE");
return EXIT_SUCCESS;
}
```

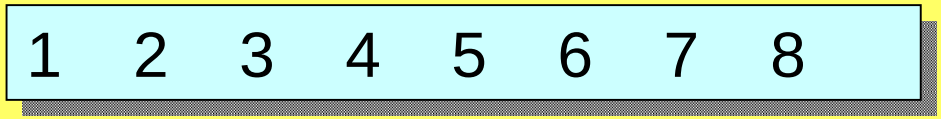
```
lunghezza sequenza max-->3
posizione partenza-->9
sequenza individuata-->6   12   8
Premere un tasto per continuare . . .
```

Esercizio 127

Dato un vettore di N elementi interi non ordinato, si vuole inserire un assegnato valore in una determinata posizione del vettore



N=7



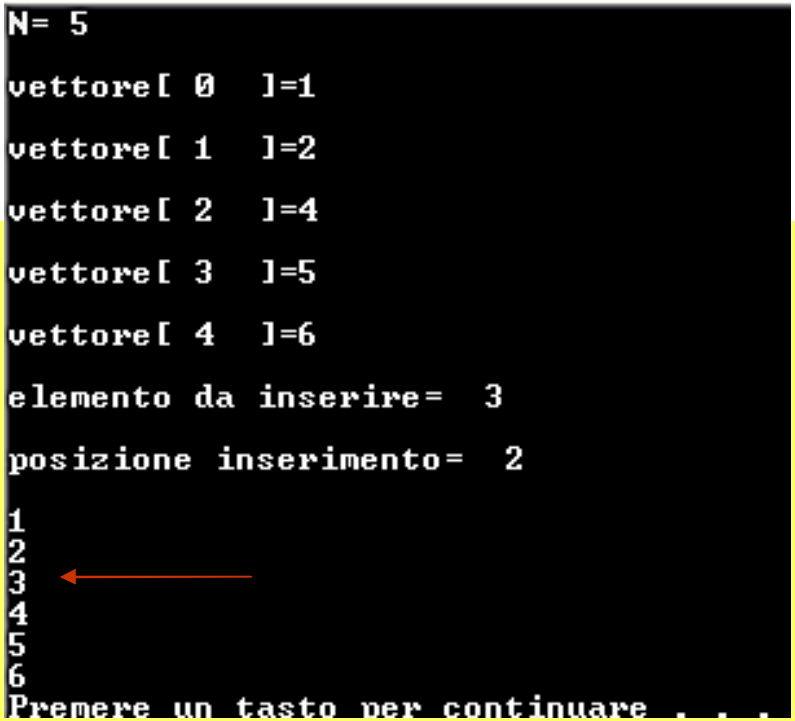
Problema : Il riempimento del vettore deve essere inferiore alla cardinalità del vettore

L’inserimento nel vettore richiede prioritariamente lo spostamento di una posizione a destra di tutti gli elementi a partire dalla posizione di inserimento per evitare perdita di informazioni

Numero di elementi da spostare = N - pos

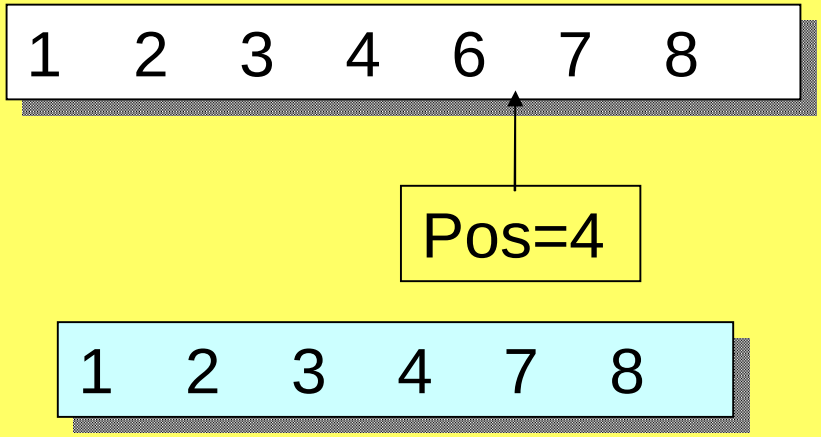
```
#include <iostream>
using namespace std;
// Esercizio127
int main()
{
    int vettore[10];
    int elem,pos;
    int N,nrosposta;
    cout <<"N= ";cin >>N;cout <<endl;
    if (N<10)
    {
     for (int i=0; i<N; i++)
     {
     cout <<"vettore[ "<<i<<"  ]=";cin >>vettore[i];cout<<endl;
     }
     cout << "elemento da inserire=  ";cin>>elem;cout<<endl;
     cout << "posizione inserimento=  ";cin>>pos;cout<<endl;
```

```
// Produce N-pos spostamenti a destra
     nrosposta=N-pos;
     for (int j=1; j<=nrosposta; j++)
       vettore[N-j+1]=vettore[N-j];
// Inserisce elemento in giusta posizione
     vettore[pos]=elem;
     for (int j=0; j<=N; j++)
     cout <<vettore[ j ]<<endl;
   }
   else cout<< "ERRORE- N eguale o maggiore della cardinalità del
vettore"<<endl;
system( "PAUSE");
return EXIT_SUCCESS;
```



Esercizio 128

Dato un vettore di N elementi interi non ordinato, si vuole eliminare il valore esistente in una determinata posizione del vettore



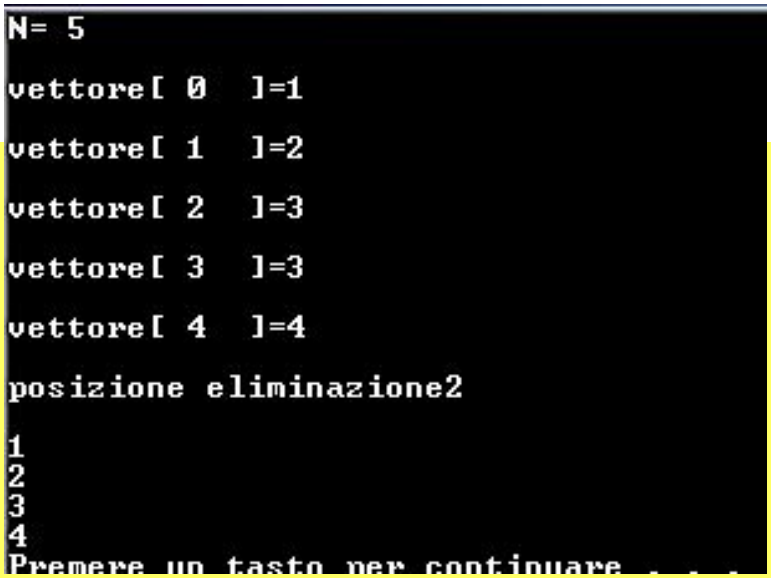
Gli elementi alla destra della posizione di cancellazione vanno semplicemente spostati di un posto a sinistra

Numero di elementi da spostare = N – pos-1

Necessario decrementare il riempimento del vettore

```
#include <iostream>
using namespace std;
// Esercizio128  Elimina elemento da un vettore
int main()
{
    int vettore[10];
    int elem,pos;
    int N,nrosposta;
    cout <<"N= ";cin >>N;cout <<endl;
    if (N<10)
    {
     for (int i=0; i<N; i++)
     {
     cout << "vettore[ "<<i<<"  ]=";cin >>vettore[ i ];cout<<endl;
     }
     cout << "posizione eliminazione";cin>>pos;cout<<endl;
```

```
// Produce N-pos-1 spostamenti sinistra
     nrosposta=N-pos-1;
     for (int j=1; j<=nrosposta; j++)
        vettore[pos+j-1]=vettore[pos+j];
     for (int j=0; j<=N-2; j++)
     cout <<vettore[ j ]<<endl;
   }
   else cout<< "ERRORE- N maggiore della cardinalità del vettore"<<endl;
system( "PAUSE");
return EXIT_SUCCESS;
}
```



Nuovo esercizio

Riprogettare gli esercizi 127 e 128 nel caso di vettore ordinato

Esercizio 132

Dato un vettore di N elementi interi ordinato in senso crescente, si vuole inserire un assegnato valore mantenendo l'ordinamento del vettore

10 20 30 50 60 70

Elemento da inserire 40

10 20 30 40 50 60 70

Ipotesi di algoritmo

Valuta se elemento va in prima posizione (in tal caso provvedere prima a N spostamenti a destra di una posizione)

Valuta se elemento va in ultima posizione (in tal caso nessun spostamento; accodare l'elemento)

Trova posizione di inserimento (posizione interna) e determina quanti spostamenti a destra produrre prima di inserire l'elemento nella giusta posizione

Controlli:
Valutare di non eccedere con l'inserimento la cardinalità definita per il vettore

```
#include <iostream.h>
#include <stdlib.h>
using namespace std;
// Esercizio132  Inserimento valore in vettore ordinato
int main()
{
      bool trovato;
      float vettore[20];
      int sposta, N, posinser;
      float elem;
      cout << "Riempimento =";cin >>N;cout<<endl;
      for (int j=0; j<N; j++)
      {
         cout << "vettore[ "<<j<<"  ]=";cin >>vettore[ j ];cout<<endl;
      }
      cout << "elemento da inserire=  ";cin>>elem; cout<<endl;
      trovato=false;
      int  i=0;
```

```
//   Controllo se elem < vettore[0]
     if (elem < vettore[0])
       {
       sposta=N;
       trovato=true;
       posinser=0;
       }
//   Controllo se elem > vettore[N-1]
     if (elem > vettore[N-1])
       {
       vettore[N]=elem;
       trovato=true;
       sposta=0;
       }
     if (! trovato)
       {
         i=1;
         while ((i<N) && (elem > vettore[i]))
             i=i+1;
         sposta=N-i;
         posinser=i;
       }
```

```
    if (sposta != 0)
      {
         for (int k=1;k<=sposta;k++)
            vettore[N+1-k]=vettore[N-k] ;
         vettore[posinser]=elem;
      }
    for (int i=0;i<=N;i++)
       cout <<vettore[i] <<endl;
    system("PAUSE");
    return 0;
}
```

Ricerca binaria

Esercizio30

Assegnato un valore intero K verificare la sua esistenza in un vettore ordinato in senso crescente (*condizione necessaria*) di N elementi interi

3	5	7	8	9	12	15	8

Casi: K

- coincidente con primo elemento dl vettore
- coincidente con ultimo elemento del vettore
- coincidente con elemento interno del vettore
- non coincidente con nessun elemento del vettore

1a soluzione

Confronto del valore K con tutti gli N elementi del vettore procedendo dal primo all'ultimo

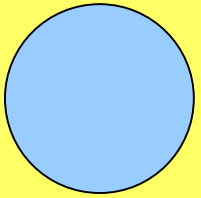
Si parte dal presupposto di valore *inesistente* che si aggiorna ad *esistente* se l'elemento realmente esiste nel vettore

Necessari N confronti

Non necessario vettore ordinato

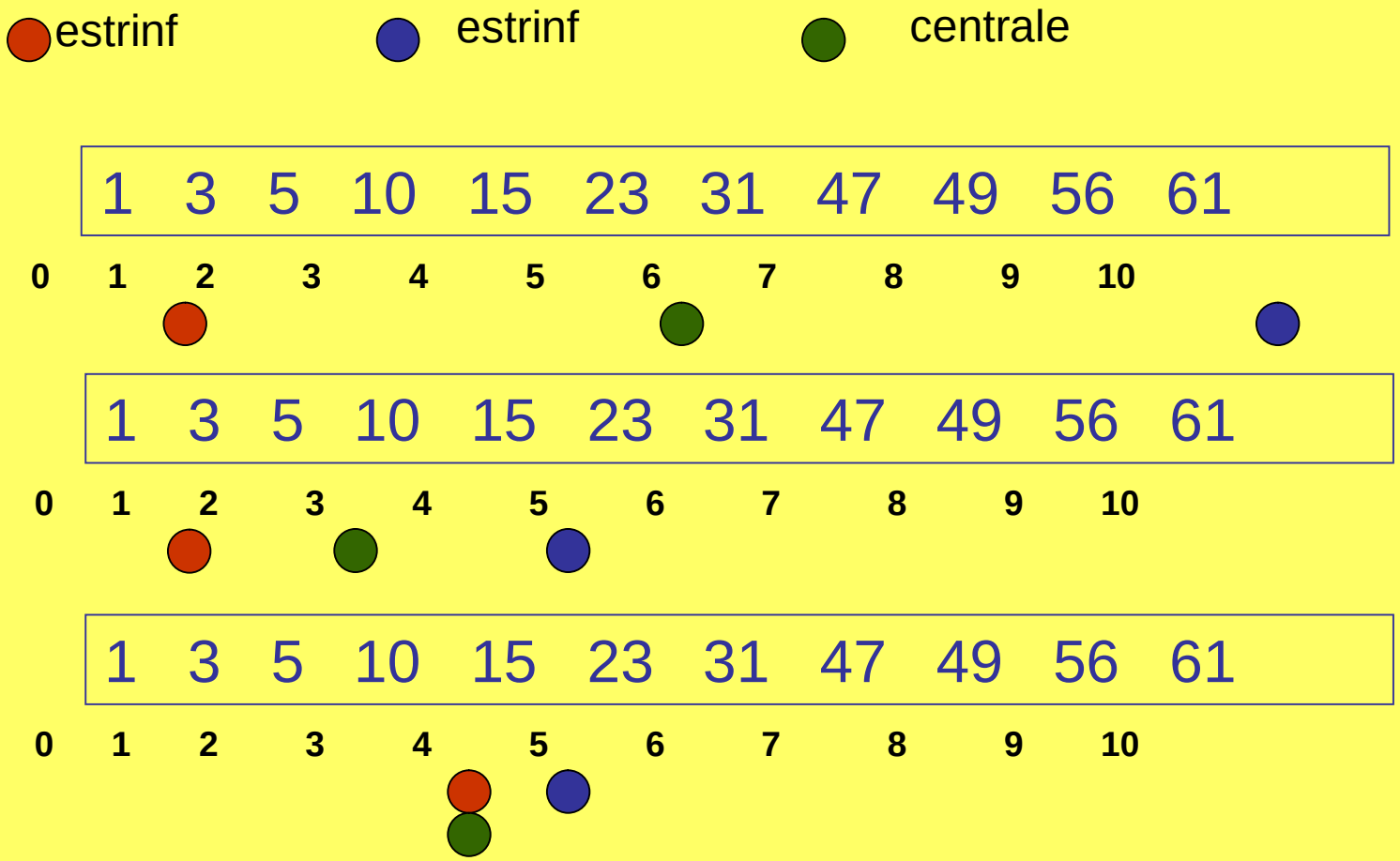
```
#include <iostream.h>
#include <stdlib.h>
using namespace std;
//Esercizio29  Ricerca in vettore
int main()
{
     bool trovato=false;
     int k,i,n;
     int vettore[10];
     // lettura elemento da cercare
     cout << "elemento da cercare --> ";cin >> k; cout <<endl;
     // lettura del vettore;
     cout << "riempimento del vettore --> ";cin >> n; cout <<endl;
     cout << " lettura vettore " <<endl;
     for (i=0;i<n;i++)
     {
     cout <<i <<"-->";cin >> vettore[i];cout <<endl;
     }
```

```
    for (i=0;i<n;i++)
      if (vettore[i]==k)
          {
          trovato=true;
          break;
          }
    if (trovato) cout << "trovato in posizione " <<i <<endl;
    else cout << "inesistente" <<endl;
    system( "PAUSE");
}
```



2a soluzione : Ricerca binaria

Valore da trovare : 10

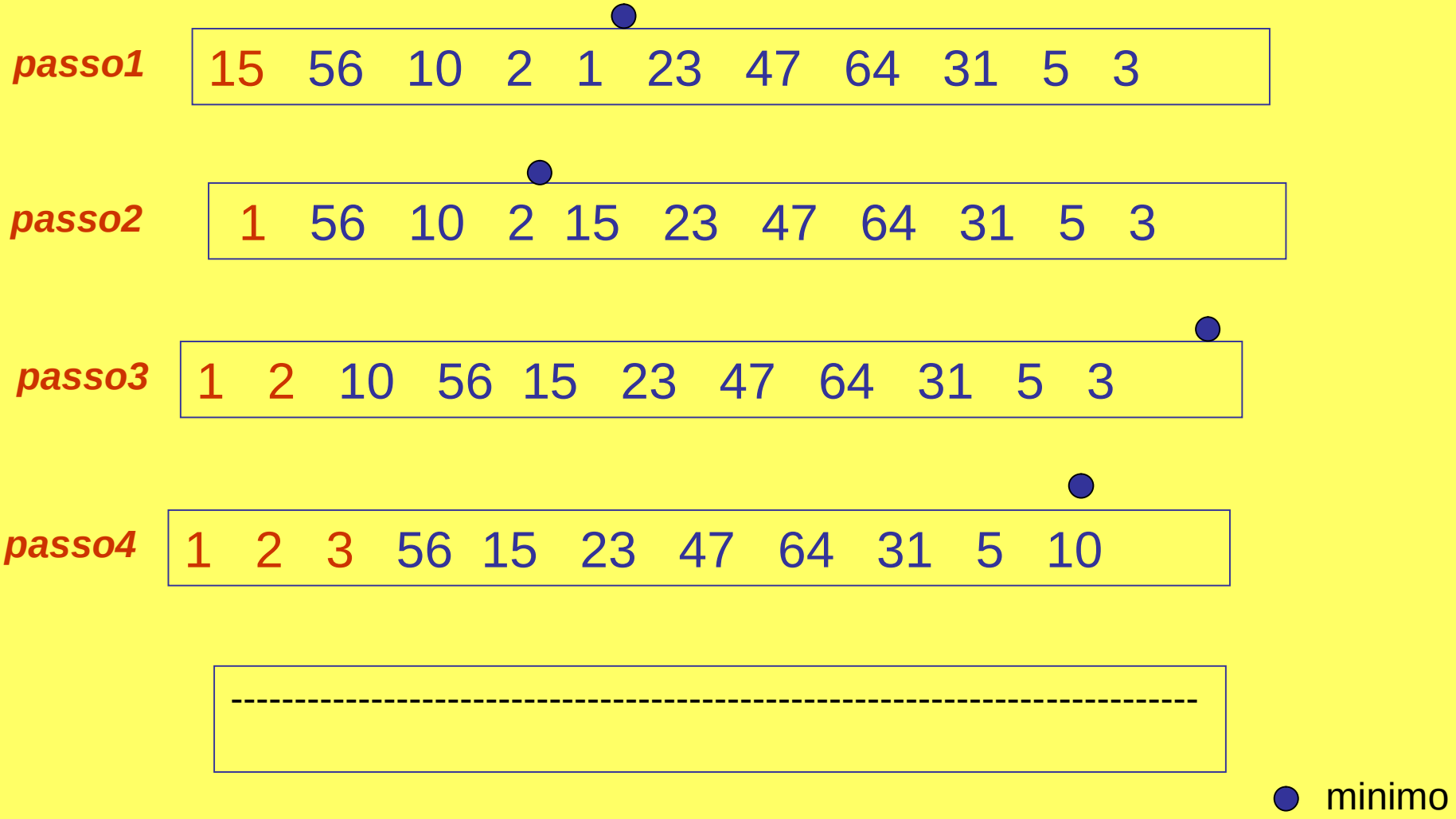


```
#include <iostream.h>
#include <stdlib.h>
using namespace std;
//Esercizio30  Ricerca binaria in vettore ordinato
int main()
{
     bool trovato=false;
     int k,i,n,estrinf,estrsup,centrale;
     int vettore[10];
     // lettura elemento da cercare
     cout << "elemento da cercare --> ";cin >> k; cout <<endl;
     // lettura del vettore;
     cout << "riempimento del vettore --> ";cin >> n; cout <<endl;
     cout << " lettura vettore " <<endl;
     for (i=0;i<n;i++)
     {
     cout <<i <<"-->";cin >> vettore[i];cout <<endl;
     }
```

```
    estrinf=0; estrsup=n-1;
    while ((!trovato) && (estrinf <=estrsup))
        {
        centrale = (estrsup + estrinf)/2;
        if (k ==vettore[centrale]) trovato=true;
        if (!trovato)
          if (k <vettore[centrale]) estrsup=centrale-1;
          else estrinf=centrale+1;
        }
    if (trovato) cout << "trovato in posizione " <<centrale <<endl;
    else cout << "inesistente" <<endl;
    system( "PAUSE");
}
```

Esercizio 32

Ordinamento di un vettore di N interi con metodo MINIMI SUCCESSIVI



In sintesi:

(passo1) Fissa il minimo al primo elemento del vettore e cerca, se esiste,un nuovo minimo fra gli elementi dalla posizione 2 ad N. Se esiste scambia di posizione l'elemento trovato con quello in prima posizione

(passo2) Fissa il minimo al secondo elemento del vettore e cerca, se esiste,un nuovo minimo fra gli elementi dalla posizione 3 ad N
Se esiste scambia di posizione l'elemento trovato con quello in seconda posizione

(passoN-1) Fissa il minimo all'elemento N-1del vettore e confronta con elemento N

Necessari N-1 passi

```
#include <iostream>
#include <cstdlib>
using namespace std;
// Esercizio32  ordinamento con minimi successivi
int main( )
{
    int vettore[10]={15,3,1,2,5,47,64,31,56,10};
    const int riemp=10;
    int pos,min;
    for (int k=0;k<riemp-1;k++)
    {
       //presunto valore minimo
       min=vettore[k];
       pos=k;
       for (int j=k+1;j<riemp;j++)
        if(vettore[j]<min)
        {
          min=vettore[j];pos=j;
        }
        if (pos !=k)
        {
           vettore[pos]=vettore[k];
           vettore[k]=min;
        }
        cout <<vettore[k] <<"  ";
    }
     cout  <<vettore[riemp-1] <<endl;
     system ("PAUSE");
}
```

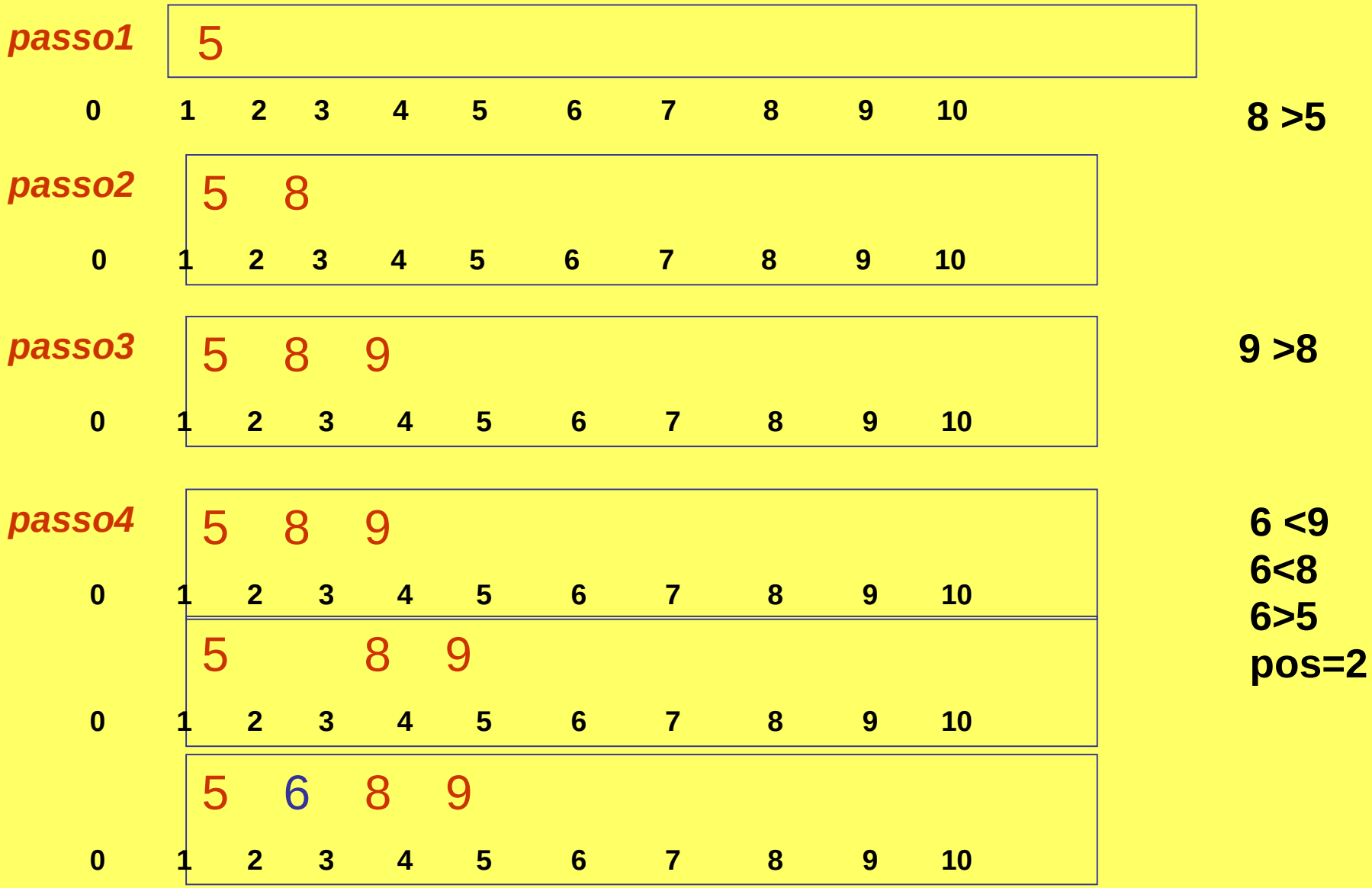
Esercizio 33

Ordinamento dinamico di un vettore durante il caricamento

Ipotesi algoritmo

Lettura primo elemento-Inserimento in prima posizione
Per i rimanenti valori
  Lettura elemento da inserire
  Se risulta > ultimo inserito lo si accoda nel vettore
  altrimenti
    si determina posizione di inserimento
    da questa posizione si spostano gli elementi di un posto a dx
    nella posizione si inserisce elemento letto

Elementi da inserire 5 8 9 6



```
#include <iostream>
#include <cstdlib>
using namespace std;
// Esercizio 33 Ordinamento dinamico di un
// vettore all'inserimento
int main()
{
    int vettore[10];
    bool flag;
    int riemp,elem,i=0,pos=-1;
    cout << "Riempimento del vettore ="; cin >>riemp;cout <<endl;
    cout << "vettore[ " <<i <<" ]";cin >>elem;cout <<endl;
    vettore[i]=elem;
```

```
for(i=1;i<riemp;i++)
{
   pos=-1; flag=true;
cout << "vettore[ " <<i <<" ]=";cin >>elem;cout <<endl;
if (elem >vettore[i-1])
 {
  vettore[i]=elem;
  flag=false;
 }
else
 {
  for (int j=0;j<i;j++)
    if (vettore[j] < elem)  pos=j;
    if (flag)
    {
      for (int k=i;k>pos;k--)
        vettore[k]=vettore[k-1];
      vettore[pos+1]=elem;
    }
 }
}
for (int k=0;k<riemp;k++)
cout << vettore[k] <<endl;
system( "PAUSE");
}
```

Riemp cicli nel caso migliore

Riemp*(Riemp-1)/2 nel caso peggiore

Esercizio 129

Dati due vettori vett1 e vett2 ordinati in senso crescente di riempimento vario riemp1 < = > riemp2 produrre un vettore vett3, effetto della fusione dei due vettori iniziali, anch'esso ordinato in senso crescente

Vett1	20	40	80					
Vett2	10	30	50	55	70			
Vett3	10	20	30	40	50	55	60	80

Mentre non tutti gli elementi di vett1 e vett2 sono stati inseriti in vett3 …….

…..preleva da vett1 o vett2 l'elemento che, nell'ordinamento crescente di vett3 segue l'ultimo elemento già inserito

```
If ( vett1[ i ]  <  vett2[ j ])
{
    vett3[ k ] = vett1[ i ]
        i=i+1
}
else
{
   vett3[ k ] = vett2[ j ]
        j=j+1
}
k=k+1
```

```
#include <iostream.h>
#include <stdlib.h>
using namespace std;
//Esercizio129  Fusione di due vettori ordinati
int main()
{
     int vett1[10],vett2[10],vett3[20];;
     int riemp1,riemp2,riemp3=0;
     cout<< "Riempimento1=";cin >>riemp1;cout <<endl;
     for (int j=0; j<riemp1; j++)
        {
        cout << "vettore1[ "  <<j <<"]=";cin >>vett1[j];
        cout <<endl;
        }
     cout<< "Riempimento2=";cin >>riemp2;cout <<endl;
     for (int j=0;j<riemp2;j++)
        {
        cout << "vettore2[ "<<j <<"]=";cin >>vett2[j];
        cout <<endl;
        }
```

```
int i=0,j=0;
while ((i<riemp1) && (j<riemp2))
  if (vett1[ i ] < vett2[ j ])
       {
       vett3[riemp3]=vett1[ i ];
       riemp3=riemp3+1;
       i=i+1;
       }
  else
       {
       vett3[riemp3]=vett2 [j ];
       riemp3=riemp3+1;
       j=j+1;
       }
```

```
// Inserimento elementi vettore non terminato
    while (i<riemp1)
        {
        vett3[riemp3]=vett1 [i ];
        riemp3=riemp3+1;
        i=i+1;
        }
    while (j<riemp2)
        {
        vett3[riemp3]=vett2[ j ];
        riemp3=riemp3+1;
        j=j+1;
        }
    for (int k=0;k<riemp3;k++)
        cout <<vett3[k] <<"   ";
    system("PAUSE");
    return 0;
}
```

Esercizio 173

Assegnata una sequenza di N cifre intere (0,..9) individuare quale relazione d'ordine rispettano

a)Identità
b)Crescente in senso stretto
c)Crescente
d)Simmetria alla cifra centrale
e)Nessuna relazione

cifre tutte eguali
Ci <Ci+1 se i<i+1
Ci <=Ci+1 se i<i+1

N=5
55555 →a
1 →b
2 →c
12321 →d
14309 →e

```
#include <iostream>
using namespace std;
// Esercizio 173
int main()
{
    int vett[10];
    int N;
    bool identita,crescstretti,crescenti,simmetria;
    cout << "N = ";cin>>N;cout<<endl;
    cout <<"Sequenza cifre -->";
    for (int i=0;i<N;i++)
    cin >>vett[i];
    cout<<endl;
    identita=crescstretti=crescenti=simmetria=true;
```

```
int j=0;
while (( j<N-1 ) && identita)
  {
  if (vett[ j ] !=vett[ j+1 ])
  identita=false;
  j=j+1;
  }
 j=0;
while ( (j<N-1 ) && crescstretti)
  {
  if ((vett[ j+1 ] -vett[ j ]) !=1)
  crescstretti=false;
  j=j+1;
  }
  j=0;
while (( j<N-1) && crescenti)
  {
  if (( vett[ j+1 ] -vett[ j ]) <0)
  crescenti=false;
  j=j+1;
  }
```

```
        while (j<((N-1)/2) )
          {
          if (vett[ j ] != vett[ N-j-1])
          simmetria=false;
          j=j+1;
          }
        if (identita && crescstretti)
        cout <<"uguali" ;
        if (crescstretti )
        cout<<"crescenti stretti"<<endl;
        if (crescenti && !crescstretti)
        cout <<"crescenti"<<endl;
        if (!identita && !crescstretti && !crescenti && !simmetria)
        cout <<"nessuna relazione"<<endl;
        if (simmetria)
        cout<<"simmetria)"<<endl;
        system("PAUSE");
        return 0;
}
```

Esercizio 175

Assegnata una parola, determinare le occorrenze di ciascun carattere

precipitevolissimevolmente

N=26

p	2
r	1
e	5
c	1
i	4
t	2
v	2
o	2
l	2
s	2
m	2
n	1

```
#include <iostream>
using namespace std;
//  Esercizio 175
int main()
{
    char word[50];
    int conta[26];
    char car;
    for (int j=0; j<26; j++)
    conta[ j ]=0;
    int I,K,N,valore;
    cout <<"lunghezza del testo =";cin >>N;cout <<endl;
    I=0;
    K=0;
```

```
    while (K < N)
        {
          cin >> word[ K ];
          valore=word[ K ];
          for (int J=97;J<=120;J++)
            {

             if ( valore ==J)
               conta[ J-97 ]=conta[ J-97 ]+1;
            }
          K=K+1;
          }
          for (int L=97;L<=120;L++)
          if (conta[ L-97 ] != 0)
          {
          car=L;
          cout <<L <<"   " <<car<<"   "<<conta[L-97]<<endl;
          }
system("PAUSE");
return EXIT_SUCCESS;
}
```

Esercizio 95

Assegnati il riempimento ed i valori di un vettore di interi, si cerchi nel vettore la più lunga sequenza di numeri pari producendo:

- la sequenza
- la posizione da cui parte
- la lunghezza

Nel caso di inesistenza di valori pari si produca opportuna segnalazione

Riemp=12
Vettore=[1, 12, 3 ,12 ,3 ,655 ,6 ,4 ,655 ,6 ,12 , 8]

Sequenza= 6 12 8
Parte da = 10
Lunghezza =3

Analisi delle informazioni

➢Informazioni di ingresso

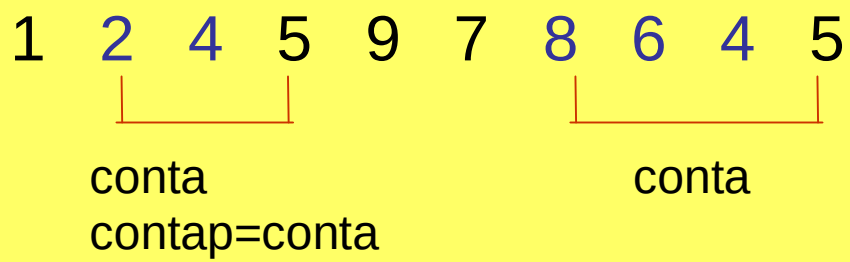
Nome	Tipo	Descrizione significato
Vettore	**vettore di interi di cardinalità 20**	**vettore su cui operare**
Riemp	**variabile intera**	**riempimento del vettore**

➢Informazioni di uscita

Nome	Tipo	Descrizione significato
contp	**Variabile intera**	**Lunghezza sequenza dei pari. Per ogni ricerca di nuova sequenza conserva lunghezza sequenza precedentemente trovata**
pos	**variabile intera**	**Posizione da cui parte sequenza**
Vettore	**Come sopra**	**Come sopra**

➢Informazioni di algoritmo

Nome	Tipo	Descrizione significato
conta	**Variabile intera**	**Lunghezza sequenza corrente**
I	**variabile intera**	**Indice del vettore**



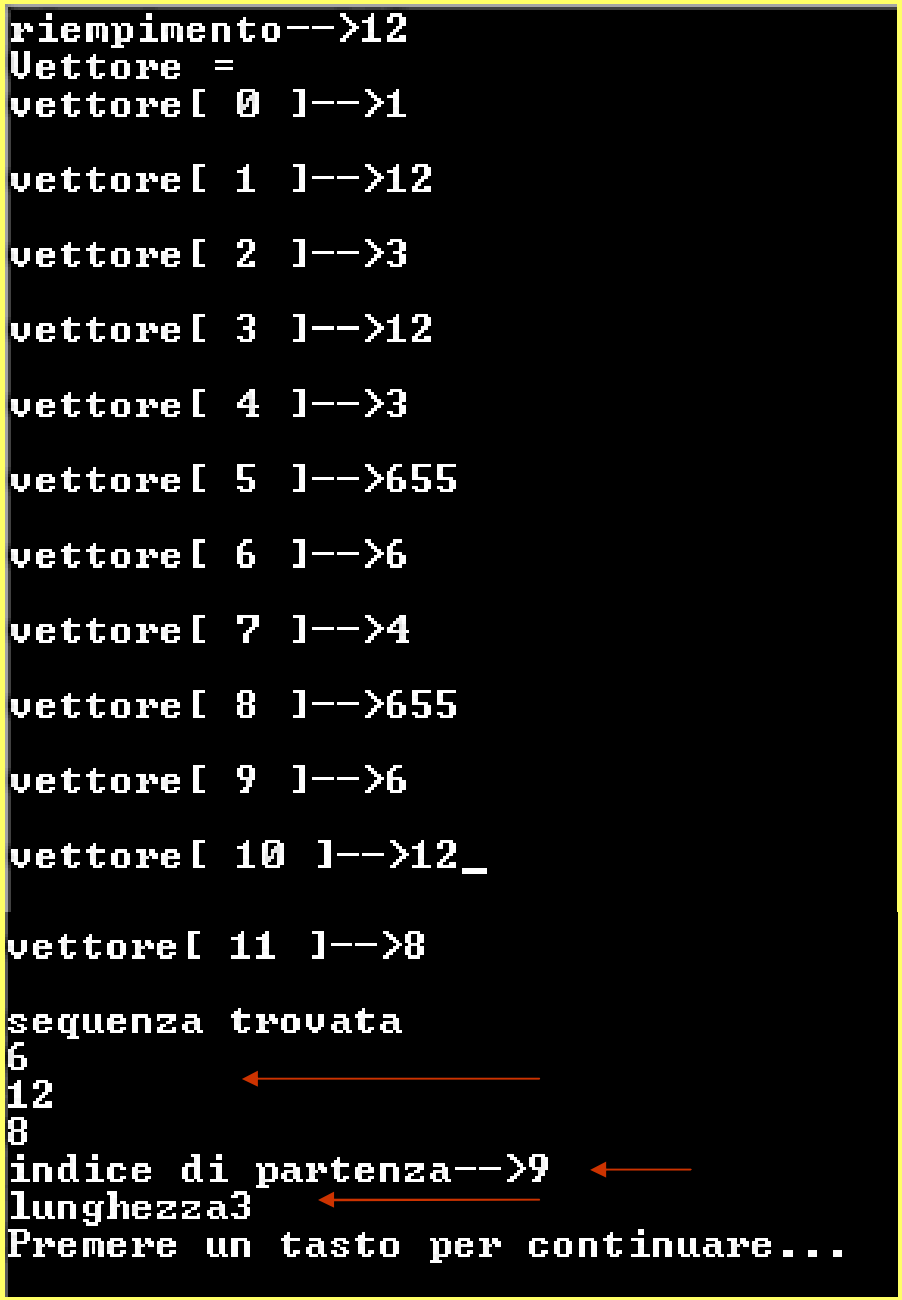
Inizio
Leggi riempimento del vettore;
Leggi elementi del vettore;
Azzera indice vettore;
Azzera contatore precedente e contatore attuale;
Mentre (indice vettore <riempimento)
    Se (elemento corrente è pari)
        incrementa contatore attuale;
    altrimenti
      Se contatore attuale>contatore precedente
        inizio blocco
         contatore precedente=contatore attuale;
         salva posizione inizio sequenza trovata;
         azzera contatore attuale;
        fine blocco
     incrementa indice vettore ;
fine-mentre

Se (contatore attuale >= contatore precedente)
inizio blocco
  contatore precedente = contatore attuale
  salva posizione inizio sequenza
fine blocco
Se contatore precedente >0
  produci stampe richieste
altrimenti
  produci “Sequenza non trovata”;
Fine

```
#include <iostream.h>
#include <stdlib.h>
// Esercizio 95  Ricerca max sequenza numeri pari
// in vettore di numeri interi
using namespace std;
int main()
{
      int riemp;
      int conta=0,contp=0,pos,indice;
      int vettore[20];
      cout << "riempimento-->";cin >>riemp;
      cout << "Vettore = " <<endl;
      for (int i=0;i<=riemp-1;i++)
      {
        cout<< "vettore[ " <<i<< " ]-->";
        cin >>vettore[i]; cout <<endl;
      }
      indice=0;
```

```
while (indice <riemp)
 {
  if ((vettore[indice] %2) ==0)
  conta=conta+1;
  else
   if (conta >=contp)
   {
   contp=conta;
   pos=indice-conta;
   conta=0;
   }
 indice=indice+1;
 }
```

```
   if (conta >= contp)
   {
    contp=conta;
    pos=indice-conta;
   }
   if (contp>0)
    {
    cout << "sequenza trovata"<<endl;
    for ( int j=pos;j<pos+contp;j++)
    cout <<vettore[j] <<endl;
    cout << "indice di partenza-->" <<pos <<endl;
    cout<< "lunghezza" <<contp <<endl;
    }
    else
    cout <<"sequenza non trovata" <<endl;
  system("PAUSE");
  return 0;
}
```


riempimento-->12
Vettore =
vettore[0]-->1
vettore[1]-->12
vettore[2]-->3
vettore[3]-->12
vettore[4]-->3
vettore[5]-->655
vettore[6]-->6
vettore[7]-->4
vettore[8]-->655
vettore[9]-->6
vettore[10]-->12_
vettore[11]-->8
sequenza trovata
6
12
8
indice di partenza-->9
lunghezza3
Premere un tasto per continuare...

Esercizio 161

Assegnato un vettore **V** di interi di riempimento ***N***, se è verificato che V[0] < V[1], a partire dall'elemento V[2] eliminare dal vettore tutti gli elementi che siano o < di V[0] o > di V[1].
Gli elementi eliminati siano inseriti in un vettore **W**

N=10

V → 2 10 8 7 6 13 1 4 11 20

output

N=6

V → 2 4 6 7 8 10

N=4

W → 1 11 13 20

```
#include <iostream.h>
#include <stdlib.h>
using namespace std;
// Esercizio 161
int elimina (int *A, int I, int K)
 {
  int J;
  for (J=I+1; J<=K; J++)
     A[ J-1 ]=A[ J ];
  return 0;
 }
int carica ( int *S,  int K,  int a)
{
  S[ K ]=a;
  return 0;
}
```

```
int ordina ( int *S, int L)
{
  bool fatto;
  fatto=false;
  int indice=0,salva;
  while (( indice < L) ||  (fatto))
     {
     indice=indice+1;
     fatto=false;
     for (int K=0; K<L-1; K++)
       if (S[ K ] > S[ K+1 ])
       {
       salva=S[ K ];
       S[ K ]=S[ K+1 ];
       S[ K+1 ]=salva;
       fatto=true;
       }
 }
  return 0;
}
```

```
int main()
{
  int V[ 50 ],W[ 50 ];
  int  N, K, I, appo;
  cout <<"Riempimento =";cin >>N; cout <<endl;
  for (int L=0; L <=N-1; L++)
  {
  cout <<"V[ " <<L<<"]=";cin >>V[ L ];cout<<endl;
  }
  if (V[ 0 ] < V[ 1 ])
  {
     I=2;
     K=0;
     while ( I <N )
     {
        if (( V[ I ] <V[ 0 ])  || (V[ I ] >V[ 1 ]) )
        {
          appo=V[ I ];
          elimina(&V[ 0 ], I ,N-1);
          carica(&W[ 0 ], K, appo);
          N=N-1;
          K=K+1;
        }
```

```
        else
        I=I+1;
     }
     ordina(&V[ 0 ], N);
     ordina(&W[ 0 ], K);
    cout <<" Riempimento di V =" <<N <<endl <<endl;
    for (I=0; I<N; I++)
      cout <<V[ I ] <<"  ";
    cout <<endl;
    cout <<"Riempimento di W  =" <<K<<endl;
    for (I=0;I<K;I++)
      cout <<W[ I ] <<"  ";
    cout <<endl;
  }
  else
    cout <<"non verificato V[0] < V[1]";
system("PAUSE");
return 0;
}
```

Esercizio 99

Assegnato un vettore di interi (max 20) determinare le occorrenze di ciascun elemento producendo per ogni valore una rappresentazione grafica (asterischi)

Riemp = 12

Vettore = 10 50 50 40 10 10 10 0 40 60 40 60

1	4	****
2	2	**
3	3	***
4	2	**
0	1	*

Una ipotesi

Se il Riempimento del vettore è >0 sicuramente almeno un elemento esiste nella struttura vettore e pertanto "*in partenza*" avrà occorrenza 1.

Bozza di procedimento

- Posizionamento sul primo elemento del vettore e confronto con i rimanenti elementi totalizzando le occorrenze
- Posizionamento sul secondo elemento del vettore e confronto con i rimanenti elementi totalizzando le occorrenze
- ……

Errata

10 50 50 40 10 10 10 0 40 60 40 60

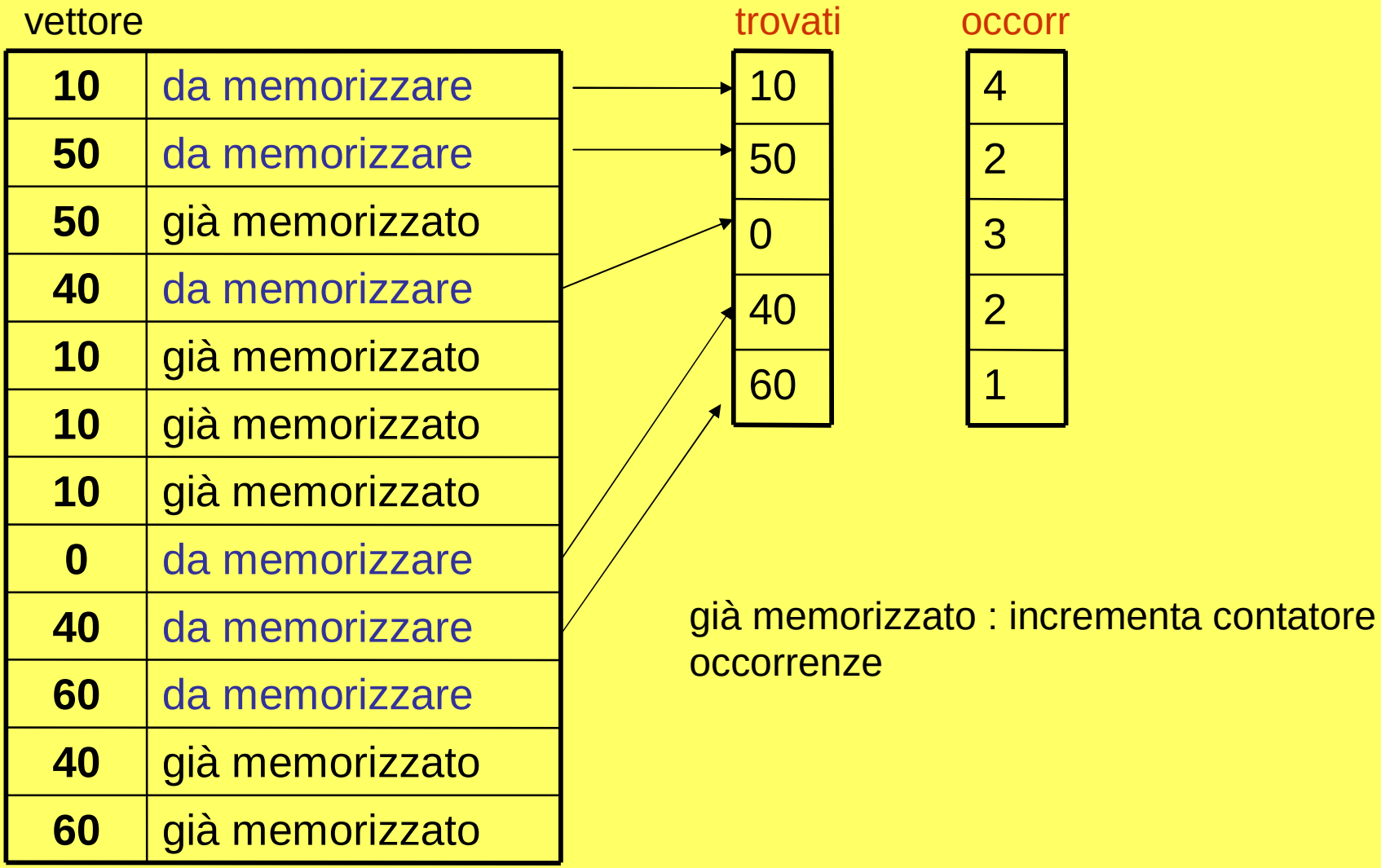
10	4	
50	2	
50	1	???
40	3	
10	3	
10	2	???
10	1	???
0	1	
40	2	
60	2	
40	1	???
60	1	???

Come risolvere le anomalie ***???***

Necessario per ogni ricerca verificare se l'elemento è stato già considerato, ovvero avere in linea ricordo degli elementi già trattati

Se già incontrato si tratterà solo di incrementare nella struttura di ricordo il contatore delle occorrenze

Se nuovo lo si inserisce nella struttura di ricordo ponendo a 1 il valore della occorrenza


vettore
trovati
occorr
10 da memorizzare
50 da memorizzare
50 già memorizzato
40 da memorizzare
10 già memorizzato
10 già memorizzato
10 già memorizzato
0 da memorizzare
40 da memorizzare
60 da memorizzare
40 già memorizzato
60 già memorizzato
10
50
0
40
60
4
2
3
2
1
già memorizzato : incrementa contatore occorrenze

Analisi delle informazioni

➢Informazioni di ingresso

Nome	Tipo	Descrizione significato
Vettore	**vettore di interi di cardinalità 20**	**vettore su cui operare**
Riemp	**variabile intera**	**riempimento del vettore**

➢Informazioni di uscita

Nome	Tipo	Descrizione significato
trovati	**vettore di interi di cardinalità 20**	**Vettore ricordo elementi trovati**
occorr	**vettore di interi di cardinalità 20**	**Vettore con occorrenze elementi trovati**
Vettore	**Come sopra**	**Come sopra**

➢Informazioni di algoritmo

Nome	Tipo	Descrizione significato
contatrovati	**Variabile intera**	**Riempimento dinamico del vettore trovati**
flag	**variabile intera**	**Indicatore di elemento esistente/inesistente**

Bozza di procedimento (corretta)

Con un ciclo di scorrimento su tutti gli elementi ……

Per ogni elemento si verifica

se già esiste in ***trovati***; in tal caso incremento della relativa occorrenza

altrimenti lo si accoda a ***trovati***, (incrementando contatrovati)
e ponendo la relativa occorrenza ad 1
in ***occorr***

```
#include <iostream.h>
#include <stdlib.h>
using namespace std;
//  Esercizio99  Determinazione occorrenze in vettore
int main()
{
      int vettore[20];
      int trovati[20];
      int occorr[20];
      int riemp, contatrovati;
      int flag;
      cout << "Riempimento -->";cin >>riemp; cout<<endl;
      for (int m=0;m<riemp;m++)
      {
       cout<< "vettore[ " <<m<< " ]-->";
       cin >>vettore[m]; cout <<endl;
      }
```

```
if (riemp>0)
{
trovati[0]=vettore[0];
occorr[0]=1;
contatrovati=1;
for (int i=1;i<riemp;i++)
{
 flag=0;
for (int k=0;k<contatrovati;k++)
  if (vettore[i]== trovati[k])
   {
  occorr[k]=occorr[k]+1;
  flag=1;
   }
  if (flag==0)
  {
  contatrovati=contatrovati+1;
  trovati[contatrovati-1]=vettore[i];
  occorr[contatrovati-1]=1;
  }
}
}
```

Posizionamenti iniziali

```
for (int j=0;j<contatrovati;j++)
{
cout <<trovati[j] <<"   " <<occorr[j] <<"   " ;
for (int i=1;i<=occorr[j];i++)
cout <<"* ";
cout <<endl;
}
system("PAUSE");
return 0;
}
```