# Vettori

# Esercizi risolti

## 1 Esercizio: “Ricerca di un elemento in vettore”

Scrivere un programma in linguaggio C che riceve in ingresso una sequenza di N numeri interi. I numeri sono memorizzati in un vettore. Il valore N è inserito dall’utente, ma il vettore può contenere al massimo 30 numeri. Terminato l’inserimento della sequenza di numeri, l’utente inserisce un valore di riferimento. Il programma deve indicare se tale valore di riferimento è contenuto nel vettore.

### Soluzione

```c
/* PROGRAMMAZIONE IN C */

/* File: ricerca_elemento.c */
/* Soluzione proposta esercizio "Ricerca di un elemento in un vettore" */

#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    const int MAXN = 30 ;   /* dimensione massima del vettore */

    int N ;                 /* occupazione effettiva del vettore */
    int vet[MAXN] ;         /* sequenza di numeri interi */
    int i ;                 /* indice dei cicli */
    int numero ;            /* numero da ricercare nella sequenza */
    int trovato ;           /* flag per indicare se la sequenza contiene
                               il numero inserito */

    /* LEGGI LE DIMENSIONI DEL VETTORE */
    do
    {
        printf("Quanti numeri saranno inseriti? ") ;
        scanf("%d",&N) ;

        /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
        if ( N > MAXN || N <=0 )
            printf("Errore: il numero deve essere compreso tra %d e 0\n",
                   MAXN) ;
    }
    while ( N > MAXN || N <=0 ) ;

    /* LEGGI UNA SEQUENZA DI N NUMERI INTERI, MEMORIZZANDOLI IN UN VETTORE */
    printf("Inserisci una sequenza di %d numeri\n", N) ;
    for ( i=0; i<N; i++ )
    {
        printf("Elemento %d: ", i+1) ;
        scanf("%d", &vet[i]) ;
    }
    printf("\n") ;

```

```
42      /* STAMPA IL VETTORE DI INTERI */
43      printf("La␣sequenza␣inserita␣e'␣la␣seguente\n") ;
44      for ( i=0; i<N; i++ )
45          printf("Elemento␣%d:␣%d\n", i+1, vet[i]) ;
46      printf("\n") ;
47
48      /* LEGGI IL NUMERO DA RICERCARE NELLA SEQUENZA */
49      printf("Inserisci␣il␣numero␣da␣cercare␣nella␣sequenza:␣") ;
50      scanf("%d",&numero) ;
51
52      /* VERIFICA SE LA SEQUENZA DI NUMERI CONTIENE IL NUMERO INSERITO */
53
54      /* INIZIALIZZA IL FLAG "trovato". IL FLAG ASSUME I VALORI
55      -- "trovato" E' UGUALE A 0 SE IL VETTORE "vet" NON CONTIENE IL VALORE "numero"
56      -- "trovato" E' UGUALE A 1 SE IL VETTORE "vet" CONTIENE IL VALORE "numero" */
57      trovato = 0 ;
58
59      /* IL CICLO FOR SCANDISCE IL VETTORE "vet" E VERIFICA SE CONTIENE
60      IL VALORE "numero".
61
62      LA RICERCA TERMINA QUANDO SI TROVA UNA CELLA "vet[i]"
63      UGUALE A "numero" O QUANDO SONO STATE CONSIDERATE TUTTE LE CELLE DEL VETTORE */
64
65      for ( i=0; i<N && trovato==0; i++ )
66      {
67          if ( vet[i] == numero )
68              /* SE "vet" CONTIENE IL VALORE IN "numero", AGGIORNA IL FLAG "trovato" */
69              trovato = 1 ;
70      }
71
72      /* STAMPA IL RISULTATO */
73      if ( trovato == 0 )
74          printf("Il␣numero␣%d␣non␣e'␣contenuto␣nella␣sequenza␣inserita\n", numero) ;
75      else
76          printf("Il␣numero␣%d␣e'␣contenuto␣nella␣sequenza␣inserita\n", numero) ;
77
78      exit(0) ;
79  }
```

# 2 Esercizio: "Verificare se un vettore contiene tutti elementi tra loro uguali"

Scrivere un programma in linguaggio C che riceve in ingresso una sequenza di N numeri interi. I numeri sono memorizzati in un vettore. Il valore N è inserito dall'utente, ma il vettore può contenere al massimo 30 numeri. Terminato l'inserimento della sequenza di numeri, il programma deve verificare se gli elementi del vettore sono tutti uguali tra loro.

## Soluzione

```
1   /* PROGRAMMAZIONE IN C */
2
3   /* File: tutti_uguali.c */
4   /* Soluzione proposta esercizio "Verificare se un vettore contiene tutti
5   elementi tra loro uguali" */
6
```

```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    const int MAXN = 30 ;   /* dimensione massima del vettore */

    int N ;                 /* occupazione del vettore */
    int vet[MAXN] ;         /* sequenza di numeri interi */
    int i ;                 /* indice dei cicli */
    int uguali ;            /* flag per indicare se la sequenza contiene numeri
                               tutti uguali */

    /* LEGGI LE DIMENSIONI DEL VETTORE */
    do
    {
        printf("Quanti numeri saranno inseriti? ") ;
        scanf("%d",&N) ;

        /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
        if ( N > MAXN || N <=0 )
            printf("Errore: il numero deve essere compreso tra %d e 0\n",
                   MAXN) ;
    }
    while ( N > MAXN || N <=0 ) ;


    /* LEGGI UNA SEQUENZA DI N NUMERI INTERI, MEMORIZZANDOLI IN UN VETTORE */
    printf("Inserisci una sequenza di %d numeri\n", N) ;
    for ( i=0; i<N; i++ )
    {
        printf("Elemento %d: ", i+1) ;
        scanf("%d", &vet[i]) ;
    }
    printf("\n") ;

    /* STAMPA IL VETTORE DI INTERI */
    printf("La sequenza inserita e' la seguente\n") ;
    for ( i=0; i<N; i++ )
        printf("Elemento %d: %d\n", i+1, vet[i]) ;
    printf("\n") ;

    /* VERIFICA SE TUTTI I NUMERI DELLA SEQUENZA SONO UGUALI */

    /* INIZIALIZZA IL FLAG "uguali". IL FLAG ASSUME I VALORI
    -- "uguali" E' UGUALE A 0 SE ALMENO DUE CELLE DEL VETTORE NON CONTENGONO
       LO STESSO VALORE
    -- "uguali" E' UGUALE A 1 SE TUTTE LE CELLE DEL VETTORE CONTENGONO
       LO STESSO VALORE */
    uguali = 1 ;

    /* IL CICLO FOR SCANDISCE IL VETTORE "vet" E VERIFICA SE TUTTE LE COPPIE DI
    CELLE ADIACENTI CONTENGONO LO STESSO VALORE. LA RICERCA TERMINA QUANDO
    SI TROVANO ALMENO DUE CELLE ADIACENTI CHE NON CONTENGONO LO STESSO VALORE O
    QUANDO SONO STATE CONSIDERATE TUTTE LE CELLE DEL VETTORE */

```

```
63      /* NEL CICLO FOR SI CONFRONTA OGNI CELLA DEL VETTORE CON LA CELLA PRECEDENTE.
64      SI OSSERVA CHE LA CELLA CON INDICE 0 (VET[0]) NON PUO' ESSERE CONFRONTATA
65      CON LA CELLA PRECEDENTE (CON INDICE -1). PERTANTO L'INDICE "i" DEL CICLO
66      ASSUME I VALORI TRA 1 E N-1 */
67      for ( i=1; i < N && uguali==1; i++ )
68      {
69          if ( vet[i] != vet[i-1] )
70              /* SE LE DUE CELLE NON CONTENGONO LO STESSO VALORE, AGGIORNA IL
71              FLAG "uguali" */
72              uguali = 0 ;
73      }
74
75      /* STAMPA IL RISULTATO */
76      if ( uguali == 0 )
77          printf("La␣sequenza␣non␣contiene␣numeri␣tutti␣uguali\n") ;
78      else
79          printf("La␣sequenza␣contiene␣numeri␣tutti␣uguali\n") ;
80
81      exit(0) ;
82  }
```

# 3 Esercizio: "Verificare se un vettore di interi è ordinato"

Scrivere un programma in linguaggio C che riceve in ingresso una sequenza di N numeri interi. I numeri sono memorizzati in un vettore. Il valore N è inserito dall'utente, ma il vettore può contenere al massimo 30 numeri. Terminato l'inserimento della sequenza di numeri, il programma deve verificare se il vettore contiene una sequenza di numeri ordinata in modo strettamente crescente.

## Soluzione

```
1   /* PROGRAMMAZIONE IN C */
2
3   /* File: vettore_ordinato.c */
4   /* Soluzione proposta esercizio "Verificare se un vettore di interi e' ordinato" */
5
6   #include <stdio.h>
7   #include <stdlib.h>
8
9   int main(void)
10  {
11
12      const int MAXN = 30 ;    /* dimensione massima del vettore */
13
14      int N ;                  /* occupazione del vettore */
15      int vet[MAXN] ;          /* sequenza di numeri interi */
16      int i ;                  /* indice dei cicli */
17      int crescente ;          /* flag per indicare se la sequenza e' crescente */
18
19      /* LEGGI LE DIMENSIONI DEL VETTORE */
20      do
21      {
22          printf("Quanti␣numeri␣saranno␣inseriti?␣") ;
23          scanf("%d",&N) ;
24
25          /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
26          if ( N > MAXN || N <=0 )
```

```
27              printf("Errore: il numero deve essere compreso tra %d e 0\n",
28                    MAXN) ;
29      }
30      while ( N > MAXN || N <=0 ) ;
31
32      /* LEGGI UNA SEQUENZA DI N NUMERI INTERI, MEMORIZZANDOLI IN UN VETTORE */
33      printf("Inserisci una sequenza di %d numeri\n", N) ;
34      for ( i=0; i<N; i++ )
35      {
36          printf("Elemento %d: ", i+1) ;
37          scanf("%d", &vet[i]) ;
38      }
39      printf("\n") ;
40
41      /* STAMPA IL VETTORE DI INTERI */
42      printf("La sequenza inserita e' la seguente\n") ;
43      for ( i=0; i<N; i++ )
44          printf("Elemento %d: %d\n", i+1, vet[i]) ;
45      printf("\n") ;
46
47      /* VERIFICA SE LA SEQUENZA DI NUMERI E' ORDINATA IN MODO CRESCENTE */
48
49      /* INIZIALIZZA IL FLAG "crescente". IL FLAG ASSUME I VALORI
50      -- "crescente"  E' UGUALE A 1 SE LA SEQUENZA E' CRESCENTE
51      -- "crescente"  E' UGUALE A 0 SE LA SEQUENZA NON E' CRESCENTE */
52      crescente = 1 ;
53
54      /* IL CICLO FOR SCANDISCE IL VETTORE "vet" E CONTROLLA SE LA SEQUENZA
55      MEMORIZZATA NEL VETTORE E' CRESCENTE. LA RICERCA TERMINA QUANDO SI VERIFICA
56      CHE LA SEQUENZA NON E' CRESCENTE O QUANDO SONO STATE CONSIDERATE TUTTE
57      LE CELLE DEL VETTORE */
58
59      /* NEL CICLO FOR SI CONFRONTA OGNI CELLA DEL VETTORE CON LA CELLA PRECEDENTE.
60      SI OSSERVA CHE LA CELLA CON INDICE 0 (VET[0]) NON PUO' ESSERE CONFRONTATA
61      CON LA CELLA PRECEDENTE (CON INDICE -1). PERTANTO L'INDICE "i" DEL CICLO
62      ASSUME I VALORI TRA 1 E N-1 */
63      for ( i=1; i < N && crescente==1; i++ )
64      {
65          if ( vet[i] <= vet[i-1] )
66              /* SEQUENZA NON CRESCENTE, AGGIORNA IL FLAG "crescente" */
67              crescente = 0 ;
68      }
69
70      /* STAMPA IL RISULTATO */
71      if ( crescente == 0 )
72          printf("La sequenza non e' crescente\n") ;
73      else
74          printf("La sequenza e' crescente\n") ;
75
76      exit(0) ;
77  }
```

# 4 Esercizio: “Stampa istogrammi”

Scrivere un programma in linguaggio C che riceve in ingresso una sequenza di N numeri interi. Il valore N è inserito dall’utente. I numeri sono memorizzati in un vettore. Terminato l’inserimento della sequenza

di numeri, il programma deve visualizzare una riga di asterischi per ogni numero inserito. Il numero di asterischi nella riga è pari al valore del numero inserito. Ad esempio, dato il vettore `9 4 6` il programma deve visualizzare:

```
Elemento 1: 9 *********
Elemento 2: 4 ****
Elemento 3: 6 ******
```

## Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: istogrammi.c */
/* Soluzione proposta esercizio "Stampa istogrammi" */

#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    const int MAXN = 200 ;  /* dimensione massima del vettore */

    int N ;                 /* occupazione del vettore */
    int vet[MAXN] ;         /* sequenza di numeri interi */
    int i, j ;              /* indici dei cicli */

    /* LEGGI LE DIMENSIONI DEL VETTORE */
    do
    {
        printf("Quanti numeri saranno inseriti? ") ;
        scanf("%d",&N) ;

        /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
        if ( N > MAXN || N <=0 )
            printf("Errore: il numero deve essere compreso tra %d e 0\n",
                   MAXN) ;
    }
    while ( N > MAXN || N <=0 ) ;

    /* LEGGI UNA SEQUENZA DI N NUMERI INTERI, MEMORIZZANDOLI IN UN VETTORE */
    printf("Inserisci una sequenza di %d numeri\n", N) ;
    for ( i=0; i<N; i++ )
    {
        printf("Elemento %d: ", i+1) ;
        scanf("%d", &vet[i]) ;
    }
    printf("\n") ;

    /* STAMPA IL VETTORE DI INTERI */
    printf("La sequenza inserita e' la seguente\n") ;
    for ( i=0; i<N; i++ )
        printf("Elemento %d: %d\n", i+1, vet[i]) ;
    printf("\n") ;

    /* STAMPA GLI ISTOGRAMMI */
    printf("Stampa degli istogrammi\n") ;
```

```
47      for ( i=0; i<N; i++ )
48      {
49          /* STAMPA IL NUMERO IN POSIZIONE "i" NEL VETTORE "vet" (OSSIA vet[i]) */
50          printf("Elemento␣%d:␣%d␣", i+1, vet[i]) ;
51
52          /* STAMPA L'ISTOGRAMMA PER IL NUMERO "vet[i]", OSSIA STAMPA UN
53          NUMERO DI "*" UGUALE A vet[i] */
54          for ( j=0; j < vet[i]; j++ )
55              printf("*") ;
56          printf("\n") ;
57      }
58      exit(0) ;
59  }
```

# 5 Esercizio: "Calcolo dell'opposto di un numero binario rappresentato in complemento a 2 su N bit"

Scrivere un programma che riceve in ingresso un numero binario rappresentato in complemento a 2 su N bit. Inizialmente l'utente inserisce il numero N di bit. Quindi inserisce le cifre del numero binario un bit alla volta, partendo dal bit più significativo (MSB). Terminato l'inserimento del numero, il programma esegue le seguenti operazioni:

1. visualizza il numero inserito partendo dal bit più significativo
2. calcola l'opposto del numero binario ricevuto in ingresso
3. visualizza l'opposto del numero binario ricevuto in ingresso partendo dal bit più significativo (MSB).

Per poter effettuare il calcolo del risultato, utilizzare il metodo secondo il quale si considerano le cifre del numero binario in complemento a due a partire dalla meno significativa (LSB) alla più significativa (MSB) (ossia da destra verso sinistra). Si ricopiano in uscita tutti gli zeri fino al primo 1 compreso. Dopo si invertono i restanti bit.
*Suggerimento.* utilizzare come punto di partenza il programma sviluppato nell'esercizio di ugual nome nell'Unità "Cicli ed Iterazioni"

## Soluzione

```
1   /* PROGRAMMAZIONE IN C */
2
3   /* File: opposto_ca2_vettori_v1.c */
4   /* Soluzione proposta esercizio "Calcolo dell'opposto di un
5   numero binario rappresentato in complemento a 2 su N bit" */
6
7   #include <stdio.h>
8   #include <stdlib.h>
9
10  int main(void)
11  {
12      const int MAXN = 200 ;  /* dimensione massima del vettore */
13
14      int N ;                 /* numero di cifre del numero binario */
15      int bit[MAXN] ;         /* numero binario */
16      int opposto[MAXN] ;     /* opposto del numero binario */
17
18      int inverti ;           /* flag per indicare se le cifre binarie devono
```

```
19                                  essere invertite */
20      int i ;                  /* indice dei cicli */
21
22      /* LEGGI IL NUMERO DI CIFRE BINARIE */
23      do
24      {
25          printf("Quanti␣bit␣saranno␣inseriti?␣") ;
26          scanf("%d", &N) ;
27
28          if ( N > MAXN || N <=0 )
29              printf("Errore:␣il␣numero␣deve␣essere␣compreso␣tra␣%d␣e␣0\n",
30                     MAXN) ;
31      }
32      while ( N > MAXN || N <=0 ) ;
33
34      /* LEGGI LE CIFRE BINARIE E MEMORIZZALE NEL VETTORE. L'ELEMENTO "bit[0]"
35      CONTIENE IL BIT PIU' SIGNIFICATIVO. L'ELEMENTO "bit[N-1]" CONTIENE IL BIT
36      MENO SIGNIFICATIVO */
37
38      printf("Inserisci␣le␣cifre␣binarie␣partendo␣dalla␣piu'␣significativa\n") ;
39      for ( i=0; i<N; i++ )
40      {
41          printf("Inserisci␣il␣bit␣di␣peso␣%d:␣", N-1-i) ;
42          scanf("%d",&bit[i]) ;
43      }
44
45      /* STAMPA IL NUMERO BINARIO INSERITO */
46      printf("Il␣numero␣binario␣inserito␣e'␣il␣seguente:\n") ;
47      for ( i=0; i<N; i++ )
48          printf("Bit␣di␣peso␣%d:␣%d\n", N-1-i, bit[i]) ;
49      printf("\n") ;
50
51      /* LEGGI LE CIFRE DEL NUMERO BINARIO A PARTIRE DALLA CIFRA MENO SIGNIFICATIVA
52      ("bit[N-1]") A QUELLA PIU' SIGNIFICATIVA ("bit[0]") ED ESEGUI
53      LA CONVERSIONE */
54
55      /* INIZIALIZZA IL FLAG "inverti":
56      -- SE "inverti" E' UGUALE a 1: si invertono tutte le cifre binarie successive
57      -- SE "inverti" E' UGUALE A 0: si ricopiano in uscita i bit successivi
58      "inverti" E' INIZIALIZZATO A 0 ED ASSEGNATO A 1 QUANDO SI TROVA IL
59      PRIMO BIT UGUALE A 1 */
60      inverti = 0 ;
61
62      for ( i=N-1; i>=0; i-- )
63      {
64          /* CALCOLA IL VALORE OPPOSTO */
65          if ( inverti == 0 )
66          {
67              /* RICOPIA IN USCITA LA CIFRA BINARIA INSERITA */
68              opposto[i] = bit[i] ;
69
70              /* SE HAI TROVATO LA PRIMA CIFRA BINARIA AD 1, AGGIORNA "inverti" */
71              if ( bit[i] == 1 )
72                  inverti = 1 ;
73          }
74          else
```

```
75          {
76              /* RICOPIA IN USCITA L'INVERSO DELLA CIFRA BINARIA INSERITA */
77              if ( bit[i] == 1 )
78                  opposto[i] = 0 ;
79              else
80                  opposto[i] = 1 ;
81          }
82      }
83
84      /* STAMPA IL RISULTATO A PARTIRE DALLA CIFRA PIU' SIGNIFICATIVA */
85      printf("Il␣numero␣binario␣risultante␣e'␣il␣seguente:\n");
86      for ( i=0; i<N; i++ )
87          printf("bit␣di␣peso␣%d:␣%d\n", N-1-i, opposto[i]) ;
88      printf("\n") ;
89
90      exit(0) ;
91  }
```

# 6 Esercizio: "Operazione di shift di un vettore"

Scrivere un programma in linguaggio C che riceve in ingresso una sequenza di N numeri interi. Il valore N è inserito dall'utente. I numeri sono memorizzati in un vettore. Il programma esegue le seguenti operazioni:

1. visualizza il vettore
2. esegue uno spostamento (shift) a sinistra di una posizione del contenuto del vettore. Pertanto ogni elemento del vettore deve assumere il valore dell'elemento immediatamente successivo all'interno del vettore. L'elemento di indice N-1 deve assumere il valore zero.
   Ad esempio dato il vettore 1 10 15 18
   Il programma deve generare il vettore 10 15 18 0
   Il programma visualizza il vettore ottenuto.
3. esegue uno spostamento (shift) a destra di una posizione del contenuto del vettore ottenuto nel passo precedente. Pertanto ogni elemento del vettore deve assumere il valore dell'elemento immediatamente precedente all'interno del vettore. L'elemento di indice 0 deve assumere il valore zero.
   Ad esempio dato il vettore 10 15 18 0
   Il programma deve generare il vettore 0 10 15 18
   Il programma visualizza il vettore ottenuto.

*Nota.* Nella definizione di "destra" e "sinistra" si immagini il vettore stampato orizzontalmente, a partire dalla cella di indice 0.

## Soluzione

```
1   /* PROGRAMMAZIONE IN C */
2
3   /* File: shift_vettore.c */
4   /* Soluzione proposta esercizio "Operazione di shift di un vettore" */
5
6   #include <stdio.h>
7   #include <stdlib.h>
8
9   int main(void)
10  {
11      const int MAXN = 200 ;  /* dimensione massima del vettore */
12
```

```
13      int N ;                 /* dimensione del vettore */
14      int vet[MAXN] ;         /* sequenza di numeri interi */
15      int i ;                 /* indice dei cicli */
16
17      /* LEGGI LE DIMENSIONI DEL VETTORE */
18      do
19      {
20          printf("Quanti numeri saranno inseriti? ") ;
21          scanf("%d",&N) ;
22
23          /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
24          if ( N > MAXN || N <=0 )
25              printf("Errore: il numero deve essere compreso tra %d e 0\n",
26                     MAXN) ;
27      }
28      while ( N > MAXN || N <=0 ) ;
29
30      /* LEGGI UNA SEQUENZA DI N NUMERI INTERI, MEMORIZZANDOLI IN UN VETTORE */
31      printf("Inserisci una sequenza di %d numeri\n", N) ;
32      for ( i=0; i<N; i++ )
33      {
34          printf("Elemento %d: ", i+1) ;
35          scanf("%d", &vet[i]) ;
36      }
37      printf("\n") ;
38
39      /* STAMPA IL VETTORE DI INTERI */
40      printf("La sequenza inserita e' la seguente\n") ;
41      for ( i=0; i<N; i++ )
42          printf("Elemento %d: %d\n", i+1, vet[i]) ;
43      printf("\n") ;
44
45      /* ESEGUI UNO SPOSTAMENTO (SHIFT) A SINISTRA DI UNA POSIZIONE DEL CONTENUTO
46      DEL VETTORE. ASSEGNA IL VALORE 0 ALLA CELLA vet[N-1] */
47      for ( i=0; i<N-1; i++ )
48          /* COPIA NELLA CELLA vet[i] IL CONTENUTO DELLA CELLA SUCCESSIVA vet[i+1] */
49          vet[i] = vet[i+1] ;
50
51      /* ASSEGNA IL VALORE 0 ALLA CELLA vet[N-1]. NOTA: QUESTA ASSEGNAZIONE DEVE
52      ESSERE FATTA AL TERMINE DEL CICLO FOR. INFATTI SE VIENE FATTA PRIMA DEL CICLO
53      FOR SI PERDEREBBE IL VALORE INIZIALMENTE CONTENUTO NELLA CELLA vet[N-1].
54      QUESTO VALORE DEVE INVECE ESSERE ASSEGNATO ALLA CELLA vet[N-2] */
55      vet[N-1] = 0 ;
56
57      /* STAMPA IL VETTORE DI INTERI */
58      printf("Stampa del vettore dopo l'operazione di shift a sinistra\n");
59      for ( i=0; i<N; i++ )
60          printf("Elemento %d: %d\n", i+1, vet[i]) ;
61      printf("\n") ;
62
63      /* ESEGUI UNO SPOSTAMENTO (SHIFT) A DESTRA DI UNA POSIZIONE DEL CONTENUTO
64      DEL VETTORE. ASSEGNA IL VALORE 0 ALLA CELLA vet[0] */
65      for ( i=N-1; i>0; i-- )
66          /* COPIA NELLA CELLA vet[i] IL CONTENUTO DELLA CELLA PRECEDENTE vet[i-1] */
67          vet[i] = vet[i-1] ;
68
```

```
69      /* ASSEGNA IL VALORE 0 ALLA CELLA vet[0]. NOTA: QUESTA ASSEGNAZIONE DEVE
70      ESSERE FATTA AL TERMINE DEL CICLO FOR. INFATTI SE VENISSE FATTA PRIMA DEL
71      CICLO FOR SI PERDE IL VALORE INIZIALMENTE CONTENUTO NELLA CELLA vet[0].
72      QUESTO VALORE DEVE INVECE ESSERE ASSEGNATO ALLA CELLA vet[1] */
73      vet[0] = 0 ;
74
75      /* STAMPA IL VETTORE DI INTERI */
76      printf("Stampa del vettore dopo l'operazione di shift a destra\n");
77      for ( i=0; i<N; i++ )
78          printf("Elemento %d: %d\n", i+1, vet[i]) ;
79      printf("\n") ;
80
81      exit(0) ;
82  }
```

# 7 Esercizio: "Compattazione di un vettore"

Scrivere un programma in linguaggio C che legge N numeri interi da tastiera e li memorizza in un vettore. Il numero N viene inserito dall'utente ed è minore di 20. Il programma deve generare un secondo vettore che compatta i numeri contenuti nel primo vettore. In particolare:

- ogni numero che compare ripetuto nel primo vettore, deve comparire una sola volta nel secondo vettore
- ogni numero uguale a zero presente nel primo vettore non deve comparire nel secondo vettore.

Il programma deve visualizzare il contenuto del secondo vettore.

Ad esempio, si supponga N=8 e si consideri la sequenza di numeri 1 18 3 0 24 3 6 0 inseriti da tastiera. Il programma deve visualizzare 1 18 3 24 6 .

## Soluzione

```
1   /* PROGRAMMAZIONE IN C */
2
3   /* File: compattazione.c */
4   /* Soluzione proposta esercizio "Compattazione di un vettore" */
5
6   #include <stdio.h>
7   #include <stdlib.h>
8
9   int main(void)
10  {
11
12      const int MAXN = 20 ;                  /* dimensione massima del vettore */
13
14      int vet[MAXN] ;                        /* sequenza di numeri interi */
15      int compatto[MAXN] ;                   /* sequenza compatta di numeri interi */
16      int N ;                                /* dimensione del vettore "vet" */
17      int N_compatto ;                       /* dimensione del vettore "compatto" */
18      int i, j ;                             /* indici dei cicli */
19      int trovato ;                          /* flag per la ricerca */
20
21      /* LEGGI LE DIMENSIONI DEL VETTORE */
22      do
23      {
24          printf("Quanti numeri saranno inseriti? ") ;
```

```
25          scanf("%d",&N) ;
26
27          /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
28          if ( N > MAXN || N <=0 )
29              printf("Errore: il numero deve essere compreso tra %d e 0\n",
30                     MAXN) ;
31      }
32      while ( N > MAXN || N <=0 ) ;
33
34      /* LEGGI UNA SEQUENZA DI N NUMERI INTERI, MEMORIZZANDOLI IN UN VETTORE */
35      printf("Inserisci una sequenza di %d numeri\n", N) ;
36      for ( i=0; i<N; i++ )
37      {
38          printf("Elemento %d: ", i+1) ;
39          scanf("%d", &vet[i]) ;
40      }
41      printf("\n") ;
42
43      /* STAMPA IL VETTORE DI INTERI */
44      printf("La sequenza inserita e' la seguente\n") ;
45      for ( i=0; i<N; i++ )
46          printf("Elemento %d: %d\n", i+1, vet[i]) ;
47      printf("\n") ;
48
49      /* AGGIORNA IL VETTORE "compatto" */
50
51      /* INIZIALMENTE IL VETTORE "compatto" NON CONTIENE NESSUN NUMERO */
52      N_compatto = 0 ;
53
54      /* IL CICLO FOR SCANDISCE IL VETTORE "vet" */
55      for ( i=0; i< N; i++ )
56      {
57          /* CONSIDERA SOLO LE CELLE IN "vet" CON VALORE DIVERSO DA 0 */
58          if ( vet[i] != 0 )
59          {
60              /* INIZIALIZZA IL FLAG "trovato". IL FLAG ASSUME I VALORI
61              -- "trovato" E' UGUALE A 0 SE IL VETTORE "compatto" NON CONTIENE
62                 IL VALORE IN "vet[i]"
63              -- "trovato" E' UGUALE A 1 SE IL VETTORE "compatto" CONTIENE
64                 IL VALORE IN "vet[i]" */
65              trovato=0;
66
67              /* IL CICLO FOR SCANDISCE IL VETTORE "compatto"  E VERIFICA SE
68              IL VALORE IN "vet[i]" E' PRESENTE NEL VETTORE "compatto".
69
70              LA RICERCA TERMINA QUANDO SI TROVA ALMENO UNA CELLA "compatto[j]"
71              CHE HA LO STESSO VALORE DI "vet[i]" O QUANDO SONO STATE CONSIDERATE
72              TUTTE LE CELLE DEL VETTORE "compatto" */
73
74              for ( j=0; j < N_compatto && trovato == 0; j++ )
75              {
76                  /* SE "compatto" CONTIENE "vet[i]", AGGIORNA IL FLAG "trovato" */
77                  if ( compatto[j] == vet[i] )
78                      trovato=1 ;
79              }
80
```

```
81              if ( trovato == 0 )
82              {
83                  /* SE "trovato" E' UGUALE A 0, IL VETTORE "compatto" NON CONTIENE
84                  IL VALORE IN "vet[i]". ACCODA NEL VETTORE "compatto" IL VALORE IN
85                  "vet[i]" E INCREMENTA LE DIMENSIONI DEL VETTORE "compatto" */
86                  compatto[N_compatto] = vet[i] ;
87                  N_compatto = N_compatto + 1 ;
88              }
89          }
90      }
91
92      /* STAMPA DEL VETTORE RISULTANTE (VETTORE "compatto") */
93      printf("Stampa del vettore risultante\n");
94      if (N_compatto == 0)
95          printf("Il vettore risultante non contiene nessun elemento \n") ;
96      else
97      {
98          printf("Il vettore risultante contiene %d elementi \n", N_compatto) ;
99          for ( i=0; i< N_compatto; i++ )
100             printf("Elemento %d: %d\n", i+1, compatto[i]) ;
101         printf("\n") ;
102     }
103     exit(0) ;
104 }
```

# 8 Esercizio: “Intersezione di due vettori”

Siano dati due vettori di interi inseriti da tastiera. La lunghezza dei due vettori è inserita dall’utente da tastiera. I due vettori possono avere lunghezza diverse, ma possono contenere al massimo 30 numeri. Si scriva un programma in linguaggio C per generare un terzo vettore che contiene l’intersezione tra due vettori. Tale vettore deve contenere i numeri presenti in entrambi i vettori dati.

Ad esempio, si assuma che siano stati inseriti i due vettori:
1 6 15 20 25
2 20 18 6
Il programma deve visualizzare la sequenza 6 20.

## Soluzione

```
1  /* PROGRAMMAZIONE IN C */
2
3  /* File: intersezione_vettori.c */
4  /* Soluzione proposta esercizio "Intersezione di due vettori" */
5
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 int main(void)
11 {
12     const int MAXN = 30 ;         /* dimensione massima dei vettori */
13
14     int vet1[MAXN], vet2[MAXN] ; /* vettori di interi */
15     int N1, N2 ;                  /* dimensione dei vettori */
16
17     int intersezione[MAXN] ;      /* intersezione tra i due vettori di interi */
```

```
18      int N_intersezione ;          /* dimensione del vettore intersezione */
19
20      int i, j ;                    /* indici dei cicli */
21      int trovato ;                 /* flag per la ricerca */
22
23      /* LEGGI LE DIMENSIONI DEL PRIMO VETTORE */
24      do
25      {
26          printf("Quanti␣numeri␣saranno␣inseriti␣nel␣primo␣vettore?␣") ;
27          scanf("%d", &N1) ;
28
29          /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
30          if ( N1 > MAXN || N1 <=0 )
31              printf("Errore:␣il␣numero␣deve␣essere␣compreso␣tra␣0␣e␣%d\n",MAXN) ;
32      }
33      while ( N1 > MAXN || N1 <= 0 ) ;
34
35      /* LEGGI IL PRIMO VETTORE */
36      printf("Inserisci␣il␣primo␣vettore␣di␣%d␣elementi\n", N1) ;
37      for ( i=0; i< N1; i++ )
38      {
39          printf("Elemento␣%d:␣", i+1) ;
40          scanf("%d", &vet1[i]) ;
41      }
42      printf("\n") ;
43
44      /* STAMPA DEL PRIMO VETTORE */
45      printf("Stampa␣del␣primo␣vettore\n");
46      for ( i=0; i< N1; i++ )
47          printf("Elemento␣%d:␣%d\n", i+1, vet1[i]) ;
48      printf("\n") ;
49
50      /* LEGGI LE DIMENSIONI DEL SECONDO VETTORE */
51      do
52      {
53          printf("Quanti␣numeri␣saranno␣inseriti␣nel␣secondo␣vettore?␣") ;
54          scanf("%d", &N2) ;
55
56          /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
57          if ( N2 > MAXN || N2 <=0 )
58              printf("Errore:␣il␣numero␣deve␣essere␣compreso␣tra␣0␣e␣%d\n", MAXN) ;
59      }
60      while ( N2 > MAXN || N2 <= 0 ) ;
61
62      /* LEGGI IL SECONDO VETTORE */
63      printf("Inserisci␣il␣secondo␣vettore␣di␣%d␣elementi\n", N2) ;
64      for ( i=0; i< N2; i++ )
65      {
66          printf("Elemento␣%d:␣", i+1) ;
67          scanf("%d", &vet2[i]) ;
68      }
69      printf("\n") ;
70
71      /* STAMPA DEL SECONDO VETTORE */
72      printf("Stampa␣il␣secondo␣vettore\n");
73      for ( i=0; i< N2; i++ )
```

```
74          printf("Elemento␣%d:␣%d\n",i+1, vet2[i]) ;
75      printf("\n") ;
76
77      /* AGGIORNA IL VETTORE "intersezione" */
78
79      /* INIZIALMENTE IL VETTORE "intersezione" NON CONTIENE NESSUN NUMERO */
80      N_intersezione = 0 ;
81
82      /* IL CICLO FOR SCANDISCE IL VETTORE "vet1" */
83      for ( i=0; i<N1; i++ )
84      {
85          /* INIZIALIZZA IL FLAG "trovato". IL FLAG ASSUME I VALORI
86            -- "trovato" E' UGUALE A 0 SE IL VETTORE "vet2" NON CONTIENE
87               IL VALORE IN "vet1[i]"
88            -- "trovato" E' UGUALE A 1 SE IL VETTORE "vet2" CONTIENE IL
89               VALORE IN "vet1[i]" */
90          trovato = 0;
91
92          /* PER OGNI ELEMENTO "vet1[i]" DI "vet1", IL CICLO FOR SCANDISCE IL
93          VETTORE "vet2" E VERIFICA SE "vet2" CONTIENE IL VALORE IN "vet1[i]"
94
95          LA RICERCA TERMINA QUANDO SI TROVA UNA CELLA "vet2[j]" UGUALE A "vet1[i]"
96          O QUANDO SONO STATE CONSIDERATE TUTTE LE CELLE DEL VETTORE "vet2" */
97
98          for ( j=0; j<N2 && trovato==0; j++ )
99          {
100             if ( vet2[j] == vet1[i] )
101             {
102                 /* SE "vet2" CONTIENE IL VALORE IN "vet1[i]", QUESTO
103                 VALORE E' INSERITO NEL VETTORE "intersezione" */
104                 intersezione[N_intersezione] = vet1[i] ;
105
106                 /* INCREMENTA LA DIMENSIONE DEL VETTORE "intersezione" */
107                 N_intersezione = N_intersezione + 1 ;
108
109                 /* AGGIORNA IL FLAG "trovato" */
110                 trovato = 1 ;
111             }
112         }
113     }
114
115     /* STAMPA DEL VETTORE "intersezione" */
116     printf("Stampa␣del␣vettore␣intersezione\n");
117     if (N_intersezione == 0)
118         printf("Il␣vettore␣intersezione␣non␣contiene␣nessun␣elemento␣\n") ;
119     else
120     {
121         printf("Il␣vettore␣intersezione␣contiene␣%d␣elementi␣\n",
122                N_intersezione) ;
123         for ( i=0; i< N_intersezione; i++ )
124             printf("Elemento␣%d:␣%d\n", i+1, intersezione[i]) ;
125         printf("\n") ;
126     }
127 }
```

## Soluzione alternativa

Nella soluzione precedente, un elemento comune ai due vettori e presente più volte nel primo vettore viene ripetuto anche nel vettore risultato. Ad esempio se sono stati inseriti i vettori 4 1 6 4 e 5 4 7 1, il programma genera la sequenza 4 1 4. Nella soluzione successiva, la sequenza risultato non contiene invece ripetizioni.

```
/* PROGRAMMAZIONE IN C */

/* File: intersezione_vettori_v2.c */
/* Soluzione proposta esercizio "Intersezione di due vettori" */

#include <stdio.h>
#include <stdlib.h>

int main(void)
{

    const int MAXN = 30 ;          /* dimensione massima del vettore */

    int vet1[MAXN], vet2[MAXN] ;   /* vettori di interi */
    int N1, N2 ;                   /* dimensione dei vettori */

    int intersezione[MAXN] ;       /* intersezione tra i due vettori di interi */
    int N_intersezione ;           /* dimensione del vettore intersezione */

    int i, j ;                     /* indici dei cicli */
    int trovato, presente ;        /* flag per la ricerca */

    /* LEGGI LE DIMENSIONI DEL PRIMO VETTORE */
    do
    {
        printf("Quanti numeri saranno inseriti nel primo vettore? ") ;
        scanf("%d", &N1) ;

        /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
        if ( N1 > MAXN || N1 <=0 )
            printf("Errore: il numero deve essere compreso tra %d e 0\n",MAXN) ;
    }
    while ( N1 > MAXN || N1 <= 0 ) ;

    /* LEGGI IL PRIMO VETTORE */
    printf("Inserisci il primo vettore di %d elementi\n", N1) ;
    for ( i=0; i< N1; i++ )
    {
        printf("Elemento %d: ", i+1) ;
        scanf("%d", &vet1[i]) ;
    }
    printf("\n") ;

    /* STAMPA DEL PRIMO VETTORE */
    printf("Stampa del primo vettore\n");
    for ( i=0; i< N1; i++ )
        printf("Elemento %d: %d\n", i+1, vet1[i]) ;
    printf("\n") ;

    /* LEGGI LE DIMENSIONI DEL SECONDO VETTORE */
    do
```

```
52      {
53          printf("Quanti␣numeri␣saranno␣inseriti␣nel␣secondo␣vettore?␣") ;
54          scanf("%d", &N2) ;
55
56          /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
57          if ( N2 > MAXN || N2 <=0 )
58              printf("Errore:␣il␣numero␣deve␣essere␣compreso␣tra␣%d␣e␣0\n", MAXN) ;
59      }
60      while ( N2 > MAXN || N2 <= 0 ) ;
61
62      /* LEGGI IL SECONDO VETTORE */
63      printf("Inserisci␣il␣secondo␣vettore␣di␣%d␣elementi\n", N2) ;
64      for ( i=0; i< N2; i++ )
65      {
66          printf("Elemento␣%d:␣", i+1) ;
67          scanf("%d", &vet2[i]) ;
68      }
69      printf("\n") ;
70
71      /* STAMPA DEL SECONDO VETTORE */
72      printf("Stampa␣il␣secondo␣vettore\n");
73      for ( i=0; i< N2; i++ )
74          printf("Elemento␣%d:␣%d\n",i+1, vet2[i]) ;
75      printf("\n") ;
76
77      /* AGGIORNAMENTO DEL VETTORE "intersezione" */
78
79      /* INIZIALMENTE IL VETTORE "intersezione" NON CONTIENE NESSUN NUMERO */
80      N_intersezione = 0 ;
81
82      /* IL CICLO FOR SCANDISCE IL VETTORE "vet1" */
83      for ( i=0; i<N1; i++ )
84      {
85          /* INIZIALIZZA IL FLAG "presente". IL FLAG ASSUME I VALORI
86             -- "presente" E' UGUALE A 0 SE IL VETTORE "intersezione" NON C
87                CONTIENE IL VALORE IN "vet1[i]"
88             -- "presente" E' UGUALE A 1 SE IL VETTORE "intersezione"
89                CONTIENE IL VALORE IN "vet1[i]" */
90          presente = 0 ;
91
92          /* IL CICLO FOR SCANDISCE IL VETTORE "intersezione" E VERIFICA SE IL
93          VALORE IN "vet1[i]" E' GIA' PRESENTE NEL VETTORE "intersezione"
94
95          LA RICERCA TERMINA QUANDO SI TROVA UNA CELLA "intersezione[j]"
96          UGUALE A "vet1[i]" O QUANDO SONO STATE CONSIDERATE TUTTE LE CELLE
97          DEL VETTORE "intersezione" */
98
99          for ( j=0; j<N_intersezione && presente==0; j++ )
100         {
101             /* SE "intersezione" CONTIENE "vet1[i]", AGGIORNA IL FLAG
102             "presente" */
103             if ( intersezione[j] == vet1[i] )
104                 presente=1 ;
105         }
106
107         /* SE IL VETTORE "intersezione" NON CONTIENE IL VALORE IN "vet1[i]",
```

```
108         VERIFICA SE VETTORE "vet2" CONTIENE IL VALORE IN "vet1[i]" */
109         if ( presente == 0 )
110         {
111             /* INIZIALIZZA IL FLAG "trovato". IL FLAG ASSUME I VALORI
112             -- "trovato" E' UGUALE A 0 SE IL VETTORE "vet2" NON CONTIENE
113             IL VALORE IN "vet1[i]"
114             -- "trovato" E' UGUALE A 1 SE IL VETTORE "vet2" CONTIENE
115             IL VALORE IN "vet1[i]" */
116             trovato = 0 ;
117
118             /* PER OGNI ELEMENTO vet1[i] DI vet1, IL CICLO FOR SCANDISCE IL
119             VETTORE "vet2" E VERIFICA SE "vet2" CONTIENE IL VALORE IN "vet1[i]"
120
121             LA RICERCA TERMINA QUANDO SI TROVA UNA CELLA "vet2[j]" UGUALE
122             A "vet1[i]" O QUANDO SONO STATE CONSIDERATE TUTTE LE CELLE DEL
123             VETTORE "vet2" */
124
125             for ( j=0; j<N2 && trovato==0; j++ )
126             {
127                 if ( vet2[j] == vet1[i] )
128                 {
129                     /* SE "vet2" CONTIENE IL VALORE IN "vet1[i]", QUESTO
130                     VALORE E' INSERITO NEL VETTORE "intersezione" */
131                     intersezione[N_intersezione] = vet1[i] ;
132
133                     /* INCREMENTA LA DIMENSIONE DEL VETTORE "intersezione" */
134                     N_intersezione = N_intersezione + 1 ;
135
136                     /* AGGIORNA IL FLAG "trovato" */
137                     trovato = 1 ;
138                 }
139             }
140         }
141     }
142
143     /* STAMPA DEL VETTORE "intersezione" */
144     printf("Stampa␣del␣vettore␣intersezione\n");
145     if (N_intersezione == 0)
146         printf("Il␣vettore␣intersezione␣non␣contiene␣nessun␣elemento␣\n") ;
147     else
148     {
149         printf("Il␣vettore␣intersezione␣contiene␣%d␣elementi␣\n", N_intersezione) ;
150         for ( i=0; i< N_intersezione; i++ )
151             printf("Elemento␣%d:␣%d\n", i+1, intersezione[i]) ;
152         printf("\n") ;
153     }
154     exit(0) ;
155 }
```

# 9 Esercizio: “Calcolo di occorrenze”

Scrivere un programma in linguaggio C che legge N numeri interi da tastiera e li memorizza in un vettore. Il numero N viene inserito dall’utente ed è minore di 20. Il programma deve visualizzare, per ogni cifra contenuta nel vettore, il numero di occorrenze.

Ad esempio, si supponga N=7 e si consideri la sequenza di numeri 1 6 15 6 2 15 15. Il programma deve visualizzare:

```
numero 1 occorrenze 1
numero 6 occorrenze 2
numero 15 occorrenze 3
numero 2 occorrenze 1
```

*Suggerimento.* Per ogni numero presente nel vettore, il numero di occorrenze deve essere visualizzato una sola volta (ad esempio per i numeri 6 e 15). Utilizzare un vettore di supporto per poter tenere traccia dei numeri nel vettore per cui sono già state calcolate le occorrenze. Gestire questo vettore di supporto in modo analogo al vettore per la compattazione di una sequenza, visto nell'esercizio 7 "Compattazione di un vettore".

## Soluzione

```c
/* PROGRAMMAZIONE IN C */

/* File: num_occorrenze.c */
/* Soluzione proposta esercizio "Calcolo di occorrenze" */


#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    const int MAXN = 20 ;   /* dimensione massima del vettore */

    int vet[MAXN] ;         /* serie di numeri interi */
    int compatto[MAXN] ;    /* serie compatta di numeri interi:
                               contiene, senza ripetizione, i valori del
                               vettore "vet" */
    int N ;                 /* dimensione del vettore "vet" */
    int N_compatto ;        /* dimensione del vettore "compatto" */
    int i, j, t ;           /* indici dei cicli */
    int trovato ;           /* flag per la ricerca */
    int occorrenze;         /* numero di occorrenze */

    /* LEGGI LE DIMENSIONI DEL VETTORE */
    do
    {
        printf("Quanti numeri saranno inseriti? ") ;
        scanf("%d",&N) ;

        /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
        if ( N > MAXN || N <=0 )
            printf("Errore: il numero deve essere compreso tra %d e 0\n",
                   MAXN) ;
    }
    while ( N > MAXN || N <=0 ) ;

    /* LEGGI UNA SERIE DI N NUMERI INTERI, MEMORIZZANDOLI IN UN VETTORE */
    printf("Inserisci il vettore di %d elementi\n", N) ;
    for ( i=0; i< N; i++ )
    {
```

```
41          printf("Elemento␣%d:␣", i+1) ;
42          scanf("%d", &vet[i]) ;
43      }
44      printf("\n") ;
45
46      /* STAMPA IL VETTORE DI INTERI */
47      printf("Stampa␣del␣vettore␣inserito\n") ;
48      for ( i=0; i<N; i++ )
49          printf("Elemento␣%d:␣%d\n", i+1, vet[i]) ;
50      printf("\n") ;
51
52      /* AGGIORNA IL VETTORE "compatto" E CALCOLA IL NUMERO DI OCCORRENZE */
53
54      /* INIZIALMENTE IL VETTORE "compatto" NON CONTIENE NESSUN NUMERO */
55      N_compatto = 0 ;
56
57      /* IL CICLO FOR SCANDISCE IL VETTORE "vet1" */
58      for ( i=0; i< N; i++ )
59      {
60          /* INIZIALIZZA IL FLAG "trovato". IL FLAG ASSUME I VALORI
61          -- "trovato" E' UGUALE A 0 SE IL VETTORE "compatto" NON CONTIENE
62             IL VALORE IN "vet[i]"
63          -- "trovato" E' UGUALE A 1 SE IL VETTORE "compatto" CONTIENE
64             IL VALORE IN "vet[i]" */
65          trovato=0 ;
66
67          /* PER OGNI ELEMENTO vet1[i] DI vet1, IL CICLO FOR SCANDISCE IL VETTORE
68          "compatto" E VERIFICA SE "compatto" CONTIENE IL VALORE IN "vet1[i]"
69
70          LA RICERCA TERMINA QUANDO SI TROVA UNA CELLA "compatto[j]"
71          UGUALE A "vet1[i]" O QUANDO SONO STATE CONSIDERATE TUTTE LE CELLE
72          DEL VETTORE "compatto" */
73
74          for ( j=0; j< N_compatto && trovato==0; j++ )
75          {
76              /* SE "compatto" CONTIENE "vet1[i]", AGGIORNA IL FLAG "trovato" */
77              if ( compatto[j] == vet[i] )
78                  trovato = 1 ;
79          }
80
81          if ( trovato == 0 )
82          {
83              /* SE "trovato" E' UGUALE A 0, COPIA NEL VETTORE "compatto" IL
84              VALORE IN "vet[i]" */
85              compatto[N_compatto] = vet[i] ;
86              N_compatto = N_compatto + 1 ;
87
88              /* CALCOLA IL NUMERO DI OCCORRENZE DI "vet[i]" NEL VETTORE "vet".
89              IL CICLO FOR SCANDISCE IL VETTORE "vet" E CONTA QUANTE VOLTE
90              IL VALORE IN "vet[i]" E' PRESENTE NEL VETTORE "vet" */
91              occorrenze = 0 ;
92              for ( t=0; t< N; t++ )
93              {
94                  if ( vet[t] == vet[i] )
95                      occorrenze = occorrenze + 1 ;
96              }
```

```
97
98              /* STAMPA DELLE OCCORRENZE */
99              printf("Elemento␣%d:␣%d,␣occorrenze␣%d\n", i+1, vet[i], occorrenze) ;
100         }
101     }
102     exit(0) ;
103 }
```

## Soluzione alternativa

In questa soluzione non viene utilizzato un vettore di supporto per tenere traccia dei numeri nel vettore per cui sono già state calcolate le occorrenze.

```
1  /* PROGRAMMAZIONE IN C */
2
3  /* File: num_occorrenze_v2.c */
4  /* Soluzione proposta esercizio "Calcolo di occorrenze" */
5
6  /* In questa soluzione non viene utilizzato un vettore di supporto
7  per tenere traccia dei numeri nel vettore per cui sono già state calcolate
8  le occorrenze*/
9
10 #include <stdio.h>
11 #include <stdlib.h>
12
13 int main(void)
14 {
15     const int MAXN = 20 ;              /* dimensione massima del vettore */
16
17     int vet[MAXN] ;                    /* serie di numeri interi */
18     int N ;                            /* dimensione del vettore "vet" */
19     int i, j, t ;                      /* indici dei cicli */
20     int trovato ;                      /* flag per la ricerca */
21     int occorrenze;                    /* numero di occorrenze */
22
23     /* LEGGI LE DIMENSIONI DEL VETTORE */
24     do
25     {
26         printf("Quanti␣numeri␣saranno␣inseriti?␣") ;
27         scanf("%d",&N) ;
28
29         /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
30         if ( N > MAXN || N <=0 )
31             printf("Errore:␣il␣numero␣deve␣essere␣compreso␣tra␣%d␣e␣0\n",
32                    MAXN) ;
33     }
34     while ( N > MAXN || N <=0 ) ;
35
36     /* LEGGI UNA SERIE DI N NUMERI INTERI, MEMORIZZANDOLI IN UN VETTORE */
37     printf("Inserisci␣il␣vettore␣di␣%d␣elementi\n", N) ;
38     for ( i=0; i< N; i++ )
39     {
40         printf("Elemento␣%d:␣", i+1) ;
41         scanf("%d", &vet[i]) ;
42     }
43     printf("\n") ;
44
45     /* STAMPA IL VETTORE DI INTERI */
```

```
46      printf("Stampa␣del␣vettore␣inserito\n") ;
47      for ( i=0; i<N; i++ )
48          printf("Elemento␣%d:␣%d\n", i+1, vet[i]) ;
49      printf("\n") ;
50
51      /* CALCOLA IL NUMERO DI OCCORRENZE */
52
53      /* IL CICLO FOR SCANDISCE IL VETTORE "vet1".
54      PER OGNI CELLA "vet[i]", VERIFICA SE ESISTE UNA CELLA IN UNA DELLE POSIZIONI
55      PRECEDENTI, CHE CONTIENE UN VALORE UGUALE A "vet[i]" */
56      for ( i=0; i< N; i++ )
57      {
58          /* INIZIALIZZA IL FLAG "trovato". IL FLAG ASSUME I VALORI
59          -- "trovato" E' UGUALE A 0 SE IL VETTORE "vet" NON CONTIENE
60          UN'ALTRA CELLA CON LO STESSO VALORE DI "vet[i]"
61          -- "trovato"  E' UGUALE A 1 SE IL VETTORE "vet" CONTIENE
62          UN'ALTRA CELLA CON LO STESSO VALORE DI "vet[i]" */
63          trovato=0 ;
64
65          /* IL CICLO FOR SCANDISCE TUTTE LE CELLE DEL VETTORE "vet"
66          CHE PRECEDONO "vet[i]" */
67          for ( j = 0; j < i && trovato==0; j++ )
68          {
69              /* SE SE ESISTE UNA CELLA IN UNA DELLE POSIZIONI PRECEDENTI,
70              CHE CONTIENE UN VALORE UGUALE A "vet[i]", AGGIORNA "trovato" */
71              if ( vet[j] == vet[i] )
72                  trovato = 1 ;
73          }
74
75          if ( trovato==0 )
76          {
77              /* SE "trovato" E' UGUALE A 0, IL VALORE IN "vet[i]" E' CONSIDERATO
78              PER LA PRIMA VOLTA. SI CALCOLANO LE OCCORRENZE DI "vet[i]" */
79
80              /* IL CICLO FOR SCANDISCE IL VETTORE "vet" E CONTA QUANTE VOLTE
81              IL VALORE IN "vet[i]" E' PRESENTE NEL VETTORE "vet" */
82
83              occorrenze = 0 ;
84              for ( t=0; t<N; t++ )
85              {
86                  if ( vet[t] == vet[i] )
87                      occorrenze = occorrenze + 1 ;
88              }
89
90              /* STAMPA DELLE OCCORRENZE */
91              printf("Valore␣%d,␣occorrenze␣%d\n", vet[i], occorrenze) ;
92          }
93      }
94      exit(0) ;
95  }
```

# 10 Esercizio: "Fusione di due vettori ordinati"

Scrivere un programma in linguaggio C che esegue la fusione di due vettori di interi ordinati in modo crescente. Il programma deve eseguire le seguenti operazioni:

1. leggere due vettori di N interi. Il numero N viene inserito dall'utente ed è minore di 20. I due vettori possono avere lunghezza diversa. I due vettori si suppongono già ordinati in maniera crescente.
2. creare un terzo vettore di lunghezza pari alla somma delle lunghezze dei due vettori dati. Il vettore dovrà contenere i numeri contenuti nei due vettori di partenza. I numeri nel vettore devono essere ordinati in modo crescente.
3. stampare il vettore generato.

Ad esempio, si assuma che siano stati inseriti i due vettori
1 6 15 20 25
2 8 18 19.
Il programma dovrà visualizzare la sequenza 1 2 6 8 15 18 19 20 25

## Soluzione

```c
/* PROGRAMMAZIONE IN C */

/* File: fusione.c */
/* Soluzione proposta esercizio "Fusione di due vettori ordinati" */

#include <stdio.h>
#include <stdlib.h>

int main(void)
{

    const int MAXN = 20 ;              /* dimensione massima del vettore */

    int vet1[MAXN], vet2[MAXN] ;       /* vettori di interi */
    int N1, N2 ;                       /* dimensione dei vettori */

    int fusione[2*MAXN] ;              /* risultato fusione di vet1 e vet2 */
    int N_fusione ;                    /* dimensione del vettore "fusione" */

    int i, j, t ;                      /* indici dei cicli */

    /* LEGGI LE DIMENSIONI DEL PRIMO VETTORE */
    do
    {
        printf("Quanti numeri saranno inseriti nel primo vettore? ") ;
        scanf("%d", &N1) ;

        /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
        if ( N1 > MAXN || N1 <=0 )
            printf("Errore: il numero deve essere compreso tra 0 e %d\n",
                   MAXN) ;
    }
    while ( N1 > MAXN || N1 <= 0 ) ;

    /* LEGGI IL PRIMO VETTORE */
    printf("Inserisci il primo vettore di %d elementi\n", N1) ;
    for ( i=0; i< N1; i++ )
    {
        printf("Elemento %d: ", i+1) ;
        scanf("%d", &vet1[i]) ;
```

```
41      }
42      printf("\n") ;
43
44      /* STAMPA DEL PRIMO VETTORE */
45      printf("Stampa del primo vettore\n");
46      for ( i=0; i< N1; i++ )
47          printf("Elemento %d: %d\n", i+1, vet1[i]) ;
48      printf("\n") ;
49
50      /* LEGGI LE DIMENSIONI DEL SECONDO VETTORE */
51      do
52      {
53          printf("Quanti numeri saranno inseriti nel secondo vettore? ") ;
54          scanf("%d", &N2) ;
55
56          /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
57          if ( N2 > MAXN || N2 <=0 )
58              printf("Errore: il numero deve essere compreso tra %d e 0\n", MAXN) ;
59      }
60      while ( N2 > MAXN || N2 <= 0 ) ;
61
62      /* LEGGI IL SECONDO VETTORE */
63      printf("Inserisci il secondo vettore di %d elementi\n", N2) ;
64      for ( i=0; i< N2; i++ )
65      {
66          printf("Elemento %d: ", i+1) ;
67          scanf("%d", &vet2[i]) ;
68      }
69      printf("\n") ;
70
71      /* STAMPA DEL SECONDO VETTORE */
72      printf("Stampa il secondo vettore\n");
73      for ( i=0; i< N2; i++ )
74          printf("Elemento %d: %d\n",i+1, vet2[i]) ;
75      printf("\n") ;
76
77      /* AGGIORNA IL VETTORE "fusione" */
78
79      /* IL VETTORE "fusione" HA DIMENSIONE PARI ALLA SOMMA DELLE
80      DIMENSIONI DI "vet1" E "vet2" */
81      N_fusione = N1 + N2 ;
82
83      /* I VETTORI "vet1", "vet2" E "fusione" SONO VISITATI RISPETTIVAMENTE
84      CON GLI INDICI "j", "t", E "i" */
85      for ( i=0, j=0, t=0; i< N_fusione && j<N1 && t< N2; i++ )
86      {
87          if ( vet1[j] <= vet2[t] )
88          {
89              /* GLI ELEMENTI DI "vet1" SONO ACCODATI NEL VETTORE "fusione" */
90              /* SE "vet1[j]" E' MINORE O UGUALE DI "vet2[t]", ALLORA "vet1[j]"
91              E' COPIATO IN "fusione[i]" PER PRIMO. VIENE INCREMENTATO "j",
92              MENTRE "i" E' INCREMENTATO DAL CICLO FOR */
93              fusione[i] = vet1[j] ;
94              j = j + 1 ;
95          }
96          else /* vet1[j] > vet2[t] */
```

```
97          {
98              /* GLI ELEMENTI DI "vet2" SONO ACCODATI NEL VETTORE "fusione" */
99              /* SE "vet1[t]" E' MAGGIORE DI "vet2[j]", ALLORA "vet2[t]"
100             E' COPIATO IN "fusione[i]" PER PRIMO. VIENE INCREMENTATO "t", MENTRE
101             "i" E' INCREMENTATO DAL CICLO FOR */
102             fusione[i] = vet2[t] ;
103             t = t + 1 ;
104         }
105     }
106
107     if ( i < N_fusione )
108     {
109         /* IL VETTORE "fusione" DEVE ESSERE ANCORA COMPLETATO INSERENDO
110         GLI ELEMENTI FINALI DI "vet1" O "vet2" */
111
112         if ( j == N1 )
113         {
114             /* TUTTI GLI ELEMENTI DI "vet1" SONO STATI COPIATI IN "fusione".
115             "fusione" VIENE ORA COMPLETATO CON GLI ELEMENTI DI "vet2" NON ANCORA
116             CONSIDERATI */
117
118             for (  ; i< N_fusione; i++, t++ )
119                 fusione[i] = vet2[t] ;
120         }
121         else
122         {
123             /* TUTTI GLI ELEMENTI DI "vet2" SONO STATI COPIATI IN "fusione".
124             "fusione" VIENE ORA CON GLI ELEMENTI DI "vet1" NON ANCORA
125             CONSIDERATI */
126             for (  ; i< N_fusione; i++, j++ )
127                 fusione[i] = vet1[j] ;
128         }
129     }
130
131     /* STAMPA DEL VETTORE "fusione"*/
132     printf("Il␣vettore␣risultante␣contiene␣contiene␣%d␣elementi\n", N_fusione);
133     for ( i=0; i< N_fusione; i++ )
134         printf("Elemento␣%d:␣%d\n",i+1,fusione[i]);
135     printf("\n");
136     exit(0) ;
137 }
```