Capitolo 2

# Primo programma in C

*printf("hello, world");*
*Kernighan & Ritchie, 1978*

## 2.1 Somma di due numeri

Si scriva un programma in linguaggio C che legga due valori interi e visualizzi la loro somma.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: somma.c */
/* Soluzione proposta esercizio "Somma di due numeri" */

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

int main(void)
{
    int a, b ; /* addendi */
    int c ; /* somma */

    /* STAMPA COSA ESEGUIRA' IL PROGRAMMA */
    printf("Somma due numeri\n\n") ;

    /* LEGGI GLI ADDENDI */
    printf("Immetti il primo numero: ") ;
    scanf("%d", &a) ;

    printf("Immetti il secondo numero: ") ;
    scanf("%d", &b) ;

    /* CALCOLA LA SOMMA */
    c = a + b ;

    /* STAMPA IL RISULTATO */
    printf("\n") ;
    printf("La somma %d + %d e' uguale a %d\n", a, b, c) ;

    exit(0) ;
}
```

## 2.2 Precedente e successivo

Si scriva un programma in linguaggio C che legga un valore intero e visualizzi il valore intero precedente e il successivo.

### Soluzione

```c
/* PROGRAMMAZIONE IN C */

/* File: precedente_successivo.c */
/* Soluzione proposta esercizio "Precedente e successivo" */

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

int main(void)
{
    int a ; /* numero inserito */
    int prec, succ ; /* numero precedente e numero successivo */

    /* LEGGI IL NUMERO */
    printf("Immetti␣il␣numero:␣") ;
    scanf("%d", &a) ;

    /* CALCOLA IL NUMERO PRECEDENTE */
    prec = a - 1 ;

    /* CALCOLA IL NUMERO SUCCESSIVO */
    succ = a + 1 ;

    /* STAMPA IL RISULTATO */
    printf("\n") ;
    printf("Il␣numero␣inserito␣e'␣%d\n", a) ;
    printf("Il␣numero␣precedente␣a␣%d␣e'␣%d\n", a, prec) ;
    printf("Il␣numero␣successivo␣a␣%d␣e'␣%d\n", a, succ) ;

    exit(0) ;
}
```

## 2.3 Media tra due numeri

Si scriva un programma in linguaggio C che legga due valori interi e visualizzi la loro media aritmetica.

### Soluzione

```c
/* PROGRAMMAZIONE IN C */

/* File: media.c */
/* Soluzione proposta esercizio "Media tra due numeri" */

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

int main(void)
{
    int a, b ; /* numeri inseriti */
    float somma ; /* somma dei due numeri */
    float media ; /* media dei due numeri */

    /* STAMPA COSA ESEGUIRA' IL PROGRAMMA */
    printf("Calcolo␣della␣media␣di␣due␣numeri\n\n") ;

    /* LEGGI I DUE NUMERI */
    printf("Immetti␣il␣primo␣numero:␣") ;
    scanf("%d", &a) ;

    printf("Immetti␣il␣secondo␣numero:␣") ;
    scanf("%d", &b) ;

    /* CALCOLA LA SOMMA DEI DUE NUMERI */
    somma = a + b ;
```

```
    /* CALCOLA LA MEDIA DEI DUE NUMERI */
    media = somma / 2 ;

    /* SOLUZIONE ALTERNATIVA PER IL CALCOLO DELLA MEDIA DEI DUE NUMERI.
       LA MEDIA E' CALCOLATA SENZA UTILIZZARE LA VARIABILE SOMMA:
       media = ( a + b ) / 2 ;
    */

    /* STAMPA IL RISULTATO */
    printf("\n") ;
    printf("La media aritmetica di %d e %d e' %f\n", a, b, media);

    exit(0) ;
}
```

## 2.4 Semplice Calcolatrice

Si scriva un programma in linguaggio C capace di compiere le 4 operazioni (somma, sottrazione, moltiplicazione e divisione) tra due numeri reali inseriti da tastiera. Dopo che sono stati inseriti i due numeri, detti A e B, il programma dovrà visualizzare i quattro valori A+B, A-B, A*B, A/B. Si ipotizzi che sia B≠0.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: calcolatrice.c */
/* Soluzione proposta esercizio "Semplice calcolatrice" */

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

int main(void)
{
    float a, b ; /* numeri inseriti */
    float somma, differenza, prodotto, quoziente ;

    /* STAMPA COSA ESEGUIRA' IL PROGRAMMA */
    printf("Programma: Calcolatrice\n\n") ;

    /* LEGGI I DUE NUMERI */
    printf("Inserisci il primo numero: ") ;
    scanf("%f", &a) ;

    printf("Inserisci il secondo numero: ") ;
    scanf("%f", &b) ;

    /* CALCOLA LA SOMMA */
    somma = a + b ;

    /* CALCOLA LA DIFFERENZA */
    differenza = a - b ;

    /* CALCOLA IL PRODOTTO */
    prodotto = a * b ;

    /* CALCOLA LA DIVISIONE */
    quoziente = a / b ;

    /* STAMPA IL RISULTATO */
    printf("\n") ;
    printf("Numeri inseriti %f e %f\n", a, b) ;
    printf("La somma e' %f\n", somma) ;
    printf("La differenza e' %f\n", differenza) ;
    printf("Il prodotto e' %f\n", prodotto) ;
    printf("La divisione e' %f\n", quoziente) ;

    exit(0) ;
```

```
}
```

## 2.5 Calcolo di aree

Si scriva un programma in linguaggio C che, dato un numero reale $D$ immesso da tastiera, calcoli e stampi:

1. l'area del quadrato di lato $D$
2. l'area del cerchio di diametro $D$
3. l'area del triangolo equilatero di lato $D$

### Soluzione

```
/* PROGRAMMAZIONE IN C */

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

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

int main(void)
{
   float d ;            /* numero inserito */
   float aq, ac, at;    /* area quadrato, cerchio, triangolo */
   float r ;            /* raggio del cerchio */

   float rad3_4 ;       /* costante pari a radice(3)/4 */

   rad3_4 = sqrt(3) / 4 ;

   /* STAMPA COSA ESEGUIRA' IL PROGRAMMA */
   printf("Calcolo di aree\n\n") ;

   /* LEGGI IL NUMERO */
   printf("Immetti il valore di D: ") ;
   scanf("%f", &d) ;

   /* CALCOLA L'AREA DEL QUADRATO DI LATO D */
   aq = d * d ;

   /* soluzione alternativa per il calcolo dell'area del quadrato utilizzando
   la funzione pow(base, esponente) definita in math.h
   aq = pow(d, 2) ;
   */

   /* CALCOLA L'AREA DEL CERCHIO DI DIAMETRO D */
   /* calcola il raggio del cerchio */
   r = d/2 ;

   /* calcola l'area del cerchio */
   ac = M_PI * ( r * r ) ;
   /* nota: il valore di PI greco e' definito in math.h come M_PI */

   /* soluzione alternativa per il calcolo dell'area del cerchio
   ac = M_PI * pow(r, 2) ;
   */

   /* CALCOLA L'AREA DEL TRIANGOLO EQUILATERO DI LATO D */
   at = rad3_4 * ( d * d ) ;

   /* soluzione alternativa per il calcolo dell'area del triangolo equilatero
   at = rad3_4 * pow( d, 2 ) ;
   */
```

```
    /* STAMPA IL RISULTATO */
    printf("\n") ;
    printf("Le␣aree␣calcolate␣sono:\n") ;
    printf("Area␣del␣quadrato␣di␣lato␣%f␣=␣%f\n", d, aq) ;
    printf("Area␣del␣cerchio␣di␣diametro␣%f␣=␣%f\n",d, ac) ;
    printf("Area␣del␣triangolo␣equilatero␣di␣lato␣%f␣=␣%f\n", d, at) ;

    exit(0) ;
}
```

*Capitolo* $3$

# Scelte ed alternative

## 3.1 Indovina cosa...

Determinare che cosa fa il seguente frammento di programma in linguaggio C:

```
int a, b, c;
scanf("%d", &a);
scanf("%d", &b);
if( a>b )
{
    c = a ;
    a = b ;
    b = c ;
}
printf("%d\n", b) ;
```

### Soluzione

Il programma, se $a \leq b$, stampa $b$. Viceversa, se $a > b$, scambia tra di loro i valori di $a$ e $b$ ("passando" attraverso una variabile di comodo $c$), e poi stampa $b$. In definitiva, se $b$ è più grande, stampa $b$. Se $a$ è più grande, scambia $a$ con $b$ e stampa $b$ (ossia quello che prima era $a$).

Conclusione: il programma stampa il maggiore dei due numeri inseriti.

Un modo alternativo per fare la stessa cosa (senza "toccare" il valore di $a$ e $b$) sarebbe:

```
if( a>b )
{
    printf("%d\n", a) ;
}
else
{
    printf("%d\n", b);
}
```

## 3.2 Segno del numero

Si realizzi un programma in linguaggio C che acquisisca da tastiera un numero e stampi un messaggio che indichi se tale numero sia positivo oppure negativo.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: es-posneg.c */
/* Soluzione proposta esercizio "Segno del numero" */
```

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

int main(void)
{
    int a ; /* numero inserito */

    /* LEGGI IL NUMERO */
    printf("Immetti un numero: ") ;
    scanf("%d", &a) ;

    /* VERIFICA SE IL NUMERO E' POSITIVO O NEGATIVO */
    if ( a >= 0 )
    {
        /* IL NUMERO E' POSITIVO O NULLO */
        printf("Il numero %d e' positivo\n", a) ;
    }
    else
    {
        /* IL NUMERO E' NEGATIVO */
        printf("Il numero %d e' negativo \n", a) ;
    }

    exit(0) ;
}
```

## 3.3 Valore assoluto

Si realizzi un programma in linguaggio C che acquisisca da tastiera un numero e stampi il valore assoluto di tale numero.

### Soluzione

```c
/* PROGRAMMAZIONE IN C */

/* File: es-valabs.c */
/* Soluzione proposta esercizio "Valore assoluto" */

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

int main(void)
{
    int a, b ; /* numero inserito ed il corrispondente valore assoluto */

    /* LEGGI IL NUMERO */
    printf("Immetti un numero: ") ;
    scanf("%d", &a) ;

    /* VERIFICA SE IL NUMERO E' POSITIVO O NEGATIVO */
    if ( a >= 0 )
    {
        /* IL NUMERO E' POSITIVO */
        printf("Il numero %d e' positivo\n", a) ;

        /* ASSEGNA A b IL VALORE DI a */
        b = a ;
    }
    else
    {
        /* IL NUMERO E' NEGATIVO */
        printf("Il numero %d e' negativo\n", a) ;

        /* ASSEGNA A b IL VALORE DI a CAMBIANDO IL SEGNO */
        b = -a ;
    }

    /* STAMPA IL RISULTATO */
```

```
    printf("Il␣valore␣assoluto␣di␣%d␣e'␣%d\n", a, b) ;

    exit(0) ;
}
```

### Soluzione alternativa

In questa soluzione viene utilizzata una sola variabile per memorizzare prima il numero inserito e poi il suo valore assoluto.

```
/* PROGRAMMAZIONE IN C */

/* File: es-valabs2.c */
/* Soluzione alternativa proposta esercizio "Valore assoluto" */

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

int main(void)
{
    int a ; /* numero inserito ed il corrispondente valore assoluto*/

    /* LEGGI IL NUMERO */
    printf("Immetti␣un␣numero:␣") ;
    scanf("%d", &a) ;

    /* STAMPA IL NUMERO */
    printf("Il␣numero␣inserito␣e'␣%d\n", a) ;

    /* VERIFICA SE IL NUMERO E' NEGATIVO */
    if ( a < 0 )
    {
       /* SE IL NUMERO E' NEGATIVO, IL VALORE ASSOLUTO E' OTTENUTO CAMBIANDO
        IL SEGNO DEL NUMERO */
        a = -a ;
    }

    /* STAMPA IL RISULTATO */
    printf("Il␣valore␣assoluto␣del␣numero␣inserito␣e'␣%d\n", a) ;

    exit(0) ;
}
```

## 3.4 Controlla A e B

Si scriva un programma in linguaggio C che legga due numeri da tastiera, detti $A$ e $B$, e determini le seguenti informazioni, stampandole a video:

1. determini se $B$ è un numero positivo o negativo
2. determini se $A$ è un numero pari o dispari
3. calcoli il valore di $A + B$
4. determini quale scelta dei segni nell'espressione $(\pm A) + (\pm B)$ porta al risultato massimo, e quale è questo valore massimo.

*Suggerimento.* Nel punto 4., il valore massimo della somma di A e B si può ottenere sommando il valore assoluto di A e di B.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: controlla-ab-v2.c */
/* Soluzione proposta esercizio "Controlla A e B" */
```

```
5
#include <stdio.h>
#include <stdlib.h>

int main(void)
10 {
    int a, b ; /* numeri inseriti A e B */
    int c ; /* somma A+B */

    /* LEGGI A e B */
15  printf("Immetti␣A:␣") ;
    scanf("%d", &a) ;

    printf("Immetti␣B:␣") ;
    scanf("%d", &b) ;
20
    /* CONTROLLA IL SEGNO DI B E STAMPA IL MESSAGGIO OPPORTUNO */
    if( b >= 0 )
    {
        printf("B␣e'␣positivo\n") ;
25  }
    else
    {
        printf("B␣e'␣negativo\n") ;
    }
30
    /* CONTROLLA LA PARITA' DI A E STAMPA IL MESSAGGIO OPPORTUNO */
    /* A e' pari se il resto della divisione di A per 2 e' uguale a zero */
    if( a%2 == 0 )
    {
35      printf("A␣e'␣pari\n") ;
    }
    else
    {
        printf("A␣e'␣dispari\n") ;
40  }

    /* CALCOLA A+B E STAMPA IL RISULTATO */
    c = a + b ;
    printf("La␣somma␣%d␣+␣%d␣e'␣uguale␣a␣%d\n", a, b, c) ;
45
    /* CALCOLA IL VALORE MASSIMO DELLA SOMMA (+- A) + (+- B) E STAMPA IL RISULTATO*/
    /* Il valore massimo e' ottenuto sommando il valore assoluto di A e di B */
    /* Calcola il valore assoluto di A */
    if( a < 0 )
50      a = -a ;

    /* Calcola il valore assoluto di B */
    if( b < 0 )
        b = -b ;
55
    printf("Il␣valore␣massimo␣della␣somma␣+-A␣+␣+-B␣e'␣uguale␣a␣%d\n", a+b ) ;

    exit(0) ;
}
```

## 3.5 Classificazione triangolo

Si scriva un programma in linguaggio C che legga da tastiera i valori delle lunghezze dei tre lati di un triangolo (detti $A$, $B$ e $C$), e determini:

- se il triangolo è equilatero
- se il triangolo è isoscele
- se il triangolo è scaleno
- se il triangolo è rettangolo.

### Soluzione parziale

In questa prima soluzione si assume, per il momento, che i valori $A$, $B$, $C$ descrivano correttamente un triangolo.
*Nota.* Per il calcolo del quadrato, è da preferirsi l'espressione `a*a` piuttosto che `pow(a,2)` in quanto è affetta da errori di approssimazione molto minori.

```
/* PROGRAMMAZIONE IN C */

/* File: triangolo-v1.c */
/* Soluzione proposta esercizio "Classificazione triangolo" (soluzione parziale) */

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

int main(void)
{
    float a, b, c ; /* lati del triangolo */

    /* LEGGI A, B e C */
    printf("Immetti A: ") ;
    scanf("%f", &a) ;

    printf("Immetti B: ") ;
    scanf("%f", &b) ;

    printf("Immetti C: ") ;
    scanf("%f", &c) ;

    printf("Verifico le proprieta' del triangolo di lati: %f, %f, %f\n", a, b, c) ;

    /* VERIFICA SE E' EQUILATERO (3 LATI UGUALI)*/
    if( a==b && b==c )
        printf("Il triangolo e' equilatero\n");
    else
        printf("Il triangolo non e' equilatero\n");

    /* VERIFICA SE E' ISOSCELE (2 LATI UGUALI)*/
    if( a==b || b==c || a==c )
        printf("Il triangolo e' isoscele\n") ;
    else
        printf("Il triangolo non e' isoscele\n") ;

    /* VERIFICA SE E' SCALENO (3 LATI DIVERSI)*/
    if( a!=b && b!=c && a!=c )
        printf("Il triangolo e' scaleno\n") ;
    else
        printf("Il triangolo non e' scaleno\n") ;

    /* VERIFICA SE E' RETTANGOLO (TEOREMA DI PITAGORA) */
    /* verifica se il lato A e' l'ipotenusa */
    if( a*a == b*b + c*c )
        printf("Il triangolo e' rettangolo (ipotenusa A)\n") ;
    else
        printf("Il triangolo non e' rettangolo (ipotenusa A)\n") ;

    /* verifica se il lato B e' l'ipotenusa */
    if ( b*b == a*a + c*c )
        printf("Il triangolo e' rettangolo (ipotenusa B)\n") ;
    else
        printf("Il triangolo non e' rettangolo (ipotenusa B)\n") ;

    /* verifica se il lato C e' l'ipotenusa */
    if( c*c == b*b + a*a )
        printf("Il triangolo e' rettangolo (ipotenusa C)\n") ;
    else
        printf("Il triangolo non e' rettangolo (ipotenusa C)\n") ;

    /* verifica se il triangolo e' rettangolo */
```

```
    if ( ( a*a == b*b + c*c ) ||
         ( b*b == a*a + c*c ) ||
         ( c*c == b*b + a*a ) )
        printf("Il triangolo e' rettangolo\n") ;
    else
        printf("Il triangolo non e' rettangolo\n") ;

    exit(0) ;
}
```

## Soluzione parziale alternativa

Anche in questa soluzione si assume, per il momento, che i valori $A$, $B$, $C$ descrivano correttamente un triangolo.

```
/* PROGRAMMAZIONE IN C */

/* File: triangolo-v2.c */
/* Soluzione proposta esercizio "Classificazione triangolo"
(soluzione parziale alternativa) */

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

int main(void)
{
    float a, b, c ;             /* lati del triangolo */
    float quadA, quadB, quadC ; /* quadrati costruiti sui cateti */

    /* LEGGI A, B e C */
    printf("Immetti A: ") ;
    scanf("%f", &a) ;

    printf("Immetti B: ") ;
    scanf("%f", &b) ;

    printf("Immetti C: ") ;
    scanf("%f", &c) ;

    printf("Verifico le proprieta' del triangolo di lati: %f, %f, %f\n", a, b, c) ;

    /* VERIFICA SE E' EQUILATERO (3 LATI UGUALI) */
    if( a==b && b==c )
    {
        printf("Il triangolo e' equilatero\n");

        /* IL TRIANGOLO EQUILATERO NON PUO' ESSERE RETTANGOLO */
        printf("Il triangolo non e' rettangolo\n") ;
    }
    else
    {
        /* SE IL TRIANGOLO NON E' EQUILATERO VERIFICA SE E' ISOSCELE O SCALENO */
        printf("Il triangolo non e' equilatero\n") ;

        /* VERIFICA SE E' ISOSCELE (2 LATI UGUALI)*/
        if( a==b || b==c || a==c )
            printf("Il triangolo e' isoscele\n") ;
        else
        {
            printf("Il triangolo non e' isoscele\n") ;

            /* IL TRIANGOLO E' SCALENO POICHE' NON E' NE' EQUILATERO NE' ISOSCELE */
            printf("Il triangolo e' scaleno\n") ;
        }


        /* SE IL TRIANGOLO NON E' EQUILATERO PUO' ESSERE RETTANGOLO */
        /* verifica se il triangolo e' rettangolo (teorema di Pitagora) */
```

```
55          /* calcola il valore dei quadrati costruiti sui cateti */
            quadA = a*a ;
            quadB = b*b ;
            quadC = c*c ;

60          if( quadA == quadB + quadC )
                printf("Il␣triangolo␣e'␣rettangolo␣(ipotenusa␣A)\n") ;
            else
            {
                if( quadB == quadA + quadC )
65                  printf("Il␣triangolo␣e'␣rettangolo␣(ipotenusa␣B)\n") ;
                else
                {
                   if( quadC == quadA + quadB )
                      printf("Il␣triangolo␣e'␣rettangolo␣(ipotenusa␣C)\n") ;
70                 else
                       printf("Il␣triangolo␣non␣e'␣rettangolo␣\n") ;
                }
            }
        }
75      exit(0);
    }
```

## Soluzione finale

In questa soluzione il programma prima di classificare il triangolo, controlla se i numeri $A$, $B$, $C$ rappresentano correttamente un triangolo.

```
    /* PROGRAMMAZIONE IN C */

    /* File: triangolo-v3.c */
    /* Soluzione proposta esercizio "Classificazione triangolo" (soluzione finale) */
5
    #include <stdio.h>
    #include <stdlib.h>
    #include <math.h>

10  int main(void)
    {
        float a, b, c ; /* lati del triangolo */
        float quadA, quadB, quadC ; /* quadrati costruiti sui cateti */

15      /* LEGGI A, B e C */
        printf("Immetti␣A:␣") ;
        scanf("%f", &a) ;

        printf("Immetti␣B:␣") ;
20      scanf("%f", &b) ;

        printf("Immetti␣C:␣") ;
        scanf("%f", &c) ;

25      printf("Verifico␣le␣proprieta'␣del␣triangolo␣di␣lati:␣%f,␣%f,␣%f\n",
               a, b, c) ;

        /* CONTROLLA SE E' UN TRIANGOLO:
           - I LATI DEVONO ESSERE POSITIVI
30         - OGNI LATO DEVE ESSERE MINORE DELLA SOMMA DEGLI ALTRI DUE
           - OGNI LATO DEVE ESSERE MAGGIORE DELLA DIFFERENZA DEGLI ALTRI DUE */
        if( a<=0 || b<=0 || c<=0 )
            printf("Errore:␣i␣lati␣devono␣essere␣positivi\n") ;
        else
35      {
             if( a>=b+c || b>=a+c || c>=a+b )
               printf("Errore:␣ogni␣lato␣deve␣essere␣minore␣della␣somma"
                      "␣degli␣altri␣due\n") ;
             else
40           {
                 if( ( b>c  && a <= b-c ) ||
```

```
                ( b<=c && a <= c-b ) ||
                ( a>c  && b <= a-c ) ||
                ( a<=c && b <= c-a ) ||
                ( a>b  && c <= b-a ) ||
                ( a<=b && c <= a-b ) )
            /*oppure if ((a <= fabs(b-c)) || (b <=fabs(a-c)) || (c <=fabs(a-b)))*/
                printf("Errore: ogni lato deve essere maggiore della "
                       "differenza degli altri due\n");
            else
            {
                /* A QUESTO PUNTO SONO SICURO CHE SIA UN TRIANGOLO!*/

                /* VERIFICA SE E' EQUILATERO (3 LATI UGUALI) */
                if( a==b && b==c )
                {
                    printf("Il triangolo e' equilatero\n");

                    /* IL TRIANGOLO EQUILATERO NON PUO' ESSERE RETTANGOLO */
                    printf("Il triangolo non e' rettangolo\n") ;
                }
                else
                {
                     /* SE IL TRIANGOLO NON E' EQUILATERO VERIFICA
                     SE E' ISOSCELE O SCALENO */
                     printf("Il triangolo non e' equilatero\n") ;

                     /* VERIFICA SE E' ISOSCELE (2 LATI UGUALI)*/
                     if( a==b || b==c || a==c )
                         printf("Il triangolo e' isoscele\n") ;
                     else
                     {
                         printf("Il triangolo non e' isoscele\n") ;

                         /* IL TRIANGOLO E' SCALENO POICHE' NON E'
                         NE' EQUILATERO NE' ISOSCELE */
                         printf("Il triangolo e' scaleno\n") ;
                     }

                     /* SE IL NON E' EQUILATERO PUO' ESSERE RETTANGOLO */
                     /* verifica se e' rettangolo (teorema di Pitagora) */
                     /* calcola il valore dei quadrati costruiti sui cateti */
                     quadA = a*a ;
                     quadB = b*b ;
                     quadC = c*c ;

                     if( quadA == quadB + quadC )
                         printf("E' rettangolo (ipotenusa A)\n") ;
                     else
                     {
                         if( quadB == quadA + quadC )
                             printf("E' rettangolo (ipotenusa B)\n") ;
                         else
                         {
                             if( quadC == quadA + quadB )
                                 printf("E' rettangolo (ipotenusa C)\n") ;
                             else
                                 printf("Il triangolo non e' rettangolo \n") ;
                         }
                      }
                }
            }
        }
    }
    exit(0);
}
```

## 3.6 Equazioni di primo grado

Data l'equazione

$$ax + b = 0$$

con $a$ e $b$ inseriti da tastiera, scrivere un programma in linguaggio C per determinare il valore di $x$, se esiste, che risolve l'equazione.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: primogrado.c */
/* Soluzione proposta esercizio "Equazione di primo grado" */

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

int main(void)
{
    float a, b ; /* coefficienti a e b */
    float x ;    /* valore di x che risolve l'equazione */

    printf("Risoluzione equazioni di primo grado\n") ;
    printf("Equazione nella forma: ax + b = 0\n") ;

    /* LEGGI a e b */
    printf("Immetti coefficiente a: ") ;
    scanf("%f", &a) ;

    printf("Immetti coefficiente b: ") ;
    scanf("%f", &b) ;

    /* x VIENE CALCOLATO COME x=-b/a. SI DEVONO VERIFICARE I VALORI DI a E b */
    if( a != 0 )
    {
        x = - b / a ;
        printf("La soluzione e' x = %f\n", x) ;
    }
    else
    {
        /* CASO a==0 */
        if( b==0 )
        {
            printf("Equazione indeterminata (ammette infinite soluzioni)\n");
        }
        else
        {
            printf("Equazione impossibile (non ammette soluzioni)\n");
        }
    }
    exit(0) ;
}
```

## 3.7 Stampa dei mesi

Dato un numero intero tra 1 e 12, che rappresenta il mese corrente, stampare il nome del mese per esteso ("Gennaio" ... "Dicembre").

### Soluzione (con `if` annidati)

```
/* PROGRAMMAZIONE IN C */

/* File: mesi.c */
/* Soluzione (con if annidati) proposta esercizio "Stampa dei mesi" */

```

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

int main(void)
{
    int mese ; /* mese inserito */

    /* LEGGI IL NUMERO DEL MESE */
    printf("Inserisci␣il␣numero␣del␣mese:␣") ;
    scanf("%d", &mese) ;

    /* VISUALIZZA IL NOME DEL MESE CORRISPONDENTE AL NUMERO INSERITO*/
    if( mese == 1 )
        printf("Gennaio\n") ;
    else
    {
        if( mese == 2 )
            printf("Febbraio\n") ;
        else
        {
            if( mese == 3 )
                printf("Marzo\n") ;
            else
            {
                if( mese == 4 )
                    printf("Aprile\n") ;
                else
                {
                    if( mese == 5 )
                        printf("Maggio\n") ;
                    else
                    {
                        if( mese == 6 )
                            printf("Giugno\n") ;
                        else
                        {
                            if( mese == 7 )
                                printf("Luglio\n") ;
                            else
                            {
                                if( mese == 8 )
                                    printf("Agosto\n") ;
                                else
                                {
                                    if( mese == 9 )
                                        printf("Settembre\n") ;
                                    else
                                    {
                                        if( mese == 10 )
                                            printf("Ottobre\n") ;
                                        else
                                        {
                                            if( mese == 11 )
                                                printf("Novembre\n") ;
                                            else
                                            {
                                                if( mese == 12 )
                                                    printf("Dicembre\n") ;
                                                else
                                                    printf("MESE␣ERRATO!\n") ;
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

```
    }
    exit(0) ;
}
```

### Soluzione (con catene `if - else if`)

```
/* PROGRAMMAZIONE IN C */

/* File: mesi2.c */
/* Soluzione (con catene if - else if) proposta esercizio "Stampa dei mesi" */

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

int main(void)
{
    int mese ; /* mese inserito */

    /* LEGGI IL NUMERO DEL MESE */
    printf("Inserisci␣il␣numero␣del␣mese:␣") ;
    scanf("%d", &mese) ;

    /* VISUALIZZA IL NOME DEL MESE CORISPONDENTE AL NUMERO INSERITO*/
    if( mese == 1 )
        printf("Gennaio\n") ;
    else if( mese == 2 )
        printf("Febbraio\n") ;
    else if( mese == 3 )
        printf("Marzo\n") ;
    else if( mese == 4 )
        printf("Aprile\n") ;
    else if( mese == 5 )
        printf("Maggio\n") ;
    else if( mese == 6 )
        printf("Giugno\n") ;
    else if( mese == 7 )
        printf("Luglio\n") ;
    else if( mese == 8 )
        printf("Agosto\n") ;
    else if( mese == 9 )
        printf("Settembre\n") ;
    else if( mese == 10 )
        printf("Ottobre\n") ;
    else if( mese == 11 )
        printf("Novembre\n") ;
    else if( mese == 12 )
        printf("Dicembre\n") ;
    else
        printf("MESE␣ERRATO!\n") ;

    exit(0) ;
}
```

### Soluzione (con istruzione `switch`)

```
/* PROGRAMMAZIONE IN C */

/* File: mesi3.c */
/* Soluzione (con istruzione switch) proposta esercizio "Stampa dei mesi" */

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

int main(void)
{
    int mese ; /* mese inserito */

    /* LEGGI IL NUMERO DEL MESE */
    printf("Inserisci␣il␣numero␣del␣mese:␣") ;
    scanf("%d", &mese) ;
```

```
    /* VISUALIZZA IL NOME DEL MESE CORISPONDENTE AL NUMERO INSERITO*/
    switch( mese )
    {
        case 1:
            printf("Gennaio\n") ;
            break ;
        case 2:
            printf("Febbraio\n") ;
            break ;
        case 3:
            printf("Marzo\n") ;
            break ;
        case 4:
            printf("Aprile\n") ;
            break ;
        case 5:
            printf("Maggio\n") ;
            break ;
        case 6:
            printf("Giugno\n") ;
            break ;
        case 7:
            printf("Luglio\n") ;
            break ;
        case 8:
            printf("Agosto\n") ;
            break ;
        case 9:
            printf("Settembre\n") ;
            break ;
        case 10:
            printf("Ottobre\n") ;
            break ;
        case 11:
            printf("Novembre\n") ;
            break ;
        case 12:
            printf("Dicembre\n") ;
            break ;
        default:
            printf("MESE␣ERRATO!\n") ;
    }
    exit(0) ;
}
```

## 3.8 Semplice calcolatrice

Si scriva un programma in linguaggio C che implementi una semplice calcolatrice in grado di compiere le 4 operazioni ($+ - \times \div$) tra numeri interi.

Il programma presenti un semplice menù da cui l'utente indichi (con un numero tra 1 e 4) l'operazione da svolgere. In seguito il programma acquisirà da tastiera i due operandi e stamperà il risultato dell'operazione.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: calcola.c */
/* Soluzione proposta esercizio "Semplice calcolatrice" */

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

int main(void)
{
    int op ; /* operazione richiesta */
    int a, b, c ; /* numeri inseriti ( a e b ) e risultato operazione  ( c ) */
```

```
    int err ; /* condizione di errore */

    /* STAMPA LE POSSIBILI OPERAZIONI SVOLTE DALLA CALCOLATRICE */
    printf("Semplice␣calcolatrice\n\n") ;

    printf("Inserisci␣1␣per␣la␣somma\n");
    printf("Inserisci␣2␣per␣la␣sottrazione\n");
    printf("Inserisci␣3␣per␣la␣moltiplicazione\n");
    printf("Inserisci␣4␣per␣la␣divisione\n");

    /* LEGGI QUALE OPERAZIONE DEVE ESSERE SVOLTA */
    printf("La␣tua␣scelta:") ;
    scanf("%d", &op) ;

    /* LEGGI  I NUMERI INSERITI */
    printf("Immetti␣il␣primo␣operando:␣");
    scanf("%d", &a) ;

    printf("Immetti␣il␣secondo␣operando:␣");
    scanf("%d", &b) ;

    /* LA CONDIZIONE DI ERRORE VIENE INIZIALIZZATA */
    err = 0 ;

    /* ESEGUI L'OPERAZIONE RICHIESTA */
    switch( op )
    {
        case 1:
            c = a + b ;
            break ;
        case 2:
            c = a - b ;
            break ;
        case 3:
            c = a * b ;
            break ;
        case 4:
            if( b == 0 )
            {
                printf("Impossibile␣dividere␣per␣zero!\n");
                err = 1 ;
            }
            else
            {
                c = a / b ;
            }
            break ;
        default:
            printf("Operazione␣errata\n") ;
            err = 1 ;
    }

    /* SE NON SI E' VERIFICATA NESSUNA CONDIZIONE DI ERRORE,
       STAMPA IL RISULTATO */
    if( err == 0 )
        printf("Il␣risultato␣vale:␣%d\n", c) ;

    exit(0) ;
}
```

## 3.9 Calcolo del massimo

Si scriva un programma in linguaggio C che acquisisca due numeri interi da tastiera e:

- determini, stampando un messaggio opportuno quale dei due numeri (il primo o il secondo) sia maggiore
- stampi il valore di tale numero.

## Soluzione

Si trascuri il caso in cui i due numeri siano uguali.

```
/* PROGRAMMAZIONE IN C */

/* File: massimo.c */
/* Soluzione proposta esercizio "Calcolo del massimo" */

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

int main(void)
{
    int a, b ; /* numeri inseriti */

    /* LEGGI I NUMERI */
    printf("Immetti il primo numero: ") ;
    scanf("%d", &a) ;

    printf("Immetti il secondo numero: ") ;
    scanf("%d", &b) ;

    /* VERIFICA SE a E' MAGGIORE DI b */
    if ( a > b )
    {
        printf("Il primo numero %d e' maggiore del secondo numero %d\n", a, b) ;
        printf("Il valore massimo e' %d\n", a) ;
    }
    else
    {
         /* CASO a < b (SI E' TRASCURATO IL CASO IN CUI I NUMERI SIANO UGUALI) */
         printf("Il secondo numero %d e' maggiore del primo numero %d\n", b, a) ;
         printf("Il valore massimo e' %d\n", b) ;
    }
    exit(0) ;
}
```

## Soluzione

Si consideri il caso in cui i due numeri siano uguali.

```
/* PROGRAMMAZIONE IN C */

/* File: massimov2.c */
/* Soluzione alternativa proposta esercizio "Calcolo del massimo" */

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

int main(void)
{
    int a, b ; /* numeri inseriti */

    /* LEGGI I NUMERI */
    printf("Immetti il primo numero: ") ;
    scanf("%d", &a) ;

    printf("Immetti il secondo numero: ") ;
    scanf("%d", &b) ;

    /* VERIFICA SE a E' MAGGIORE DI b */
    if ( a > b )
    {
        printf("Il primo numero %d e' maggiore del secondo numero %d\n", a, b) ;
        printf("Il valore massimo e' %d\n", a) ;
    }
    else
    {
         /* CASO a <= b */
```

```
        /* VERIFICA SE b E' MAGGIORE DI a */
        if ( a < b )
        {
          printf("Il secondo numero %d e' maggiore del primo numero %d\n",
                 b, a) ;
          printf("Il valore massimo e' %d\n", b) ;
        }
        else
          /* CASO a = b */
          printf("Il primo numero %d ed il secondo numero %d sono uguali\n",
                  a, b) ;
    }
    exit(0) ;
}
```

## 3.10 Calcolo del massimo a 3

Si scriva un programma in linguaggio C che acquisisca tre numeri interi da tastiera e:

- determini, stampando un messaggio opportuno quale dei tre numeri (il primo, il secondo o il terzo) sia maggiore
- stampi il valore di tale numero.

Si trascuri il caso in cui i numeri siano uguali.

### Soluzione (con `if` annidate)

```
/* PROGRAMMAZIONE IN C */

/* File: massimo3.c */
/* Soluzione proposta esercizio "Calcolo del massimo a 3" */

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

int main(void)
{
    int a, b, c ; /* numeri inseriti */

    /* LEGGI I NUMERI */
    printf("Immetti il primo numero: ") ;
    scanf("%d", &a) ;

    printf("Immetti il secondo numero: ") ;
    scanf("%d", &b) ;

    printf("Immetti il terzo numero: ") ;
    scanf("%d", &c) ;

    /* VERIFICA SE a E' MAGGIORE DI b */
    if ( a > b )
    {
       /* CASO a > b */
       /* VERIFICA SE a E' MAGGIORE DI c */
       if ( a > c )
       {
            /* CASO a > c */
            /* a E' IL MASSIMO POICHE' a > b E a > c */
            printf("Il primo numero %d e' maggiore del secondo %d e del terzo %d\n",
                    a, b, c) ;
            printf("Il valore massimo e' %d\n", a) ;
       }
       else
       {
            /* CASO a < c (si e' trascurato il caso in cui i numeri siano uguali) */
```

```
            /* c E' IL MASSIMO POICHE' a > b E c > a */
            printf("Il terzo numero %d e' maggiore del primo %d e del secondo %d\n",
                   c, a, b) ;
            printf("Il valore massimo e' %d\n", c) ;
        }
    }
    else
    {
        /* CASO a < b */
        /* VERIFICA SE b E' MAGGIORE DI c */
        if ( b > c )
        {
            /* CASO b > c */
            /* b E' IL MASSIMO POICHE' a < b E b > c */
            printf("Il secondo numero %d e' maggiore del primo %d e del terzo %d\n",
                   b, a, c) ;
            printf("Il valore massimo e' %d\n", b) ;
        }
        else
        {
            /* CASO c < b */
            /* c E' IL MASSIMO POICHE' a < b E b < c
            (si e' trascurato il caso in cui i numeri siano uguali) */
            printf("Il terzo numero %d e' maggiore del primo %d e del secondo %d\n",
                   c, a, b) ;
            printf("Il valore massimo e' %d\n", c) ;
        }
    }
    exit(0) ;
}
```

## Soluzione (con condizioni complesse)

```
/* PROGRAMMAZIONE IN C */

/* File: massimo3v2.c */
/* Soluzione alternativa proposta esercizio "Calcolo del massimo a 3" */

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

int main(void)
{
    int a, b, c ; /* numeri inseriti */

    /* LEGGI I NUMERI */
    printf("Immetti il primo numero: ") ;
    scanf("%d", &a) ;

    printf("Immetti il secondo numero: ") ;
    scanf("%d", &b) ;

    printf("Immetti il terzo numero: ") ;
    scanf("%d", &c) ;

    /* VERIFICA SE a E' MAGGIORE DI b E DI c*/
    if ( a > b && a > c)
    {
        /* a E' IL MASSIMO */
        printf("Il primo numero %d e' maggiore del secondo %d e del terzo %d\n",
               a, b, c) ;
        printf("Il valore massimo e' %d\n", a) ;
    }
    else
    {
        /* VERIFICA SE b E' MAGGIORE DI a E DI c*/
        if ( b > a && b > c)
        {
            /* b E' IL MASSIMO */
            printf("Il secondo numero %d e' maggiore del primo %d e del terzo %d\n",
```

```
                b, a, c) ;
            printf("Il␣valore␣massimo␣e'␣%d\n", b) ;
        }
        else
        {
            /* POICHE' a E b NON SONO IL MASSIMO, ALLORA c E' IL MASSIMO */
            /* ATTENZIONE: SI E' TRASCURATO IL CASO IN CUI I NUMERI SIANO UGUALI */
            printf("Il␣terzo␣numero␣%d␣e'␣maggiore␣del␣primo␣%d␣e␣del␣secondo␣%d\n",
                c, a, b) ;
            printf("Il␣valore␣massimo␣e'␣%d\n", c) ;
        }
    }
    exit(0) ;
}
```

## 3.11 Equazione di secondo grado

Si realizzi un programma in linguaggio C per risolvere equazioni di secondo grado. In particolare, data una generica equazione di secondo grado nella forma

$$ax^2 + bx + c = 0$$

dove $a$, $b$, $c$ sono coefficienti reali noti e $x$ rappresenta l'incognita, il programma determini le due radici $x_1$ ed $x_2$ dell'equazione data, ove esse esistano.

Si identifichino tutti i casi particolari ($a = 0$, $\Delta \leq 0$, ...) e si stampino gli opportuni messaggi informativi.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: secondogrado.c */
/* Soluzione proposta esercizio "Equazione di secondo grado" */

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

int main(void)
{
    float a, b, c ; /* coefficienti a, b e c */
    float delta ;   /* discriminante */
    float x1, x2 ;  /* valori di x che risolvono l'equazione */

    printf("Risoluzione␣equazioni␣di␣secondo␣grado\n") ;
    printf("Equazione␣nella␣forma:␣ax^2␣+␣bx+␣c␣=␣0\n") ;

    /* LEGGI a e b */
    printf("Immetti␣coefficiente␣a:␣") ;
    scanf("%f", &a) ;

    printf("Immetti␣coefficiente␣b:␣") ;
    scanf("%f", &b) ;

    printf("Immetti␣coefficiente␣c:␣") ;
    scanf("%f", &c) ;

    /* Se a==0, allora in realtà è un'equazione di primo grado */

    if ( a==0 )
    {
        /* x VIENE CALCOLATO COME x=-c/b.
           SI DEVONO VERIFICARE I VALORI DI b E c */
        if ( b != 0 )
        {
            x1 = - c / b ;
            printf("Una␣soluzione:␣x␣=␣%f\n", x1) ;
```

```
        }
        else
        {
            /* CASO b==0 */
            if ( b==0 )
            {
                printf("Equazione␣indeterminata␣(ammette␣infinite␣soluzioni)\n");
            }
            else
            {
                printf("Equazione␣impossibile␣(non␣ammette␣soluzioni)\n");
            }
        }
    }
    else  /* a != 0, quindi è una 'vera' equazione di secondo grado */
    {
        /* Calcoliamo il discriminante 'delta' */
        delta = b*b - 4*a*c ;
        // printf("Il discriminante vale: %f\n", delta) ;

        if ( delta<0 )
        {
            printf("Non␣ci␣sono␣soluzioni␣in␣campo␣reale\n") ;
        }
        else if ( delta == 0 )
        {
            x1 = -b / (2*a) ;
            printf("Una␣soluzione␣doppia:␣x␣=␣%f\n", x1) ;
        }
        else /* delta > 0 */
        {
            /* caso normale */
            x1 = ( -b - sqrt(delta) ) / ( 2 * a ) ;
            x2 = ( -b + sqrt(delta) ) / ( 2 * a ) ;

            printf("Due␣soluzioni:␣x␣=␣%f␣e␣x␣=␣%f\n", x1, x2 ) ;
        }
    }

    exit(0) ;
}
```

*Capitolo* 4

# Cicli ed iterazioni

## 4.1 Indovina cosa...

Determinare il valore visualizzato dai seguenti programma nel caso in cui `num=4` e per i seguenti valori della variabile `conta`: `conta=5`, `conta=0`, `conta=1`, `conta= -5`.

```
int conta, num;
scanf("%d", &conta);
scanf("%d", &num);

while (conta != 0)
{
    num = num * 10;
    conta = conta - 1;
}
printf("%d\n", num) ;
```

```
int conta, num;
scanf("%d", &conta);
scanf("%d", &num);

while (conta > 0)
{
    num = num * 10;
    conta = conta - 1;
}
printf("%d\n", num) ;
```

### Soluzione

Si nota innanzitutto come i due programmi siano identici tranne per la condizione `conta!=0`, che diviene `conta>0` nel secondo. Ciò significa che i due programmi si comporteranno in modo identico ogniqualvolta `conta` sarà un valore positivo o nullo (perché in tal caso le due condizioni `conta!=0` e `conta>0` si equivalgono), mentre si potranno comportare diversamente quando `conta<0`.

Analizzando il ciclo, si nota che l'operazione principale eseguita è `num=num*10`, che viene ripetuta `conta` volte. In pratica il programma calcola un valore finale pari a $num * 10^{conta}$.

In definitiva il valore calcolato (e stampato) sarà:

| | Programma di sinistra | Programma di destra |
|---|---|---|
| `num=4, conta=5` | 400000 | 400000 |
| `num=4, conta=0` | 4 | 4 |
| `num=4, conta=1` | 40 | 40 |
| `num=4, conta=-5` | (*) | 4 |

(*) in questo caso il programma esibisce un comportamento anomalo, dovuto ad un errore di programmazione (non ci si è "protetti" contro un dato errato, ossia negativo, inserito dall'utente). Il ciclo viene eseguito un'enormità di volte (dell'ordine di $2^{32}$ volte), finché il valore di `conta`, che parte da -5 e viene decrementato ripetutamente fino a quando la sottrazione non andrà in overflow, e poi nuovamente finché non arriverà a zero. In tal caso `num` viene moltiplicato per 10 un'enormità di volte, andando ripetutamente in overflow... il risultato ottenuto sarà quindi totalmente imprevedibile (e tra l'altro dipendente dall'implementazione degli `int` nel compilatore utilizzato). A titolo di esempio, nel caso del compilatore Dev-C++ su piattaforma Windows, dopo circa 20 secondi (durante i quali il programma decrementa `conta` all'impazzata) viene stampato il valore 0.

## 4.2 Conversione Binario-Decimale

Si scriva un programma in linguaggio C che converta un numero binario in un numero decimale. Il numero binario è rappresentato su N bit, e il valore di N è inserito da tastiera. L'utente inserisce le cifre del numero binario un bit alla volta, partendo dal bit meno significativo (ossia dal bit di peso $2^0$). Il programma visualizzerà il numero decimale corrispondente.
*Suggerimento.* Per calcolare le potenze di 2 utilizzare la funzione `pow`, includendo la libreria `math.h`. Ad esempio per calcolare $2^5$, si scriverà `pow(2,5)`. In generale, data una base $a$, per calcolare $y = a^b$, si scrive `y = pow(a,b)` includendo la libreria `math.h`.

### Soluzione

Questa soluzione è "generalizzabile" facilmente ad altre basi pur di cambiare il valore della costante `BASE`.

```
/* PROGRAMMAZIONE IN C */

/* File: bindec1.c */
/* Soluzione proposta esercizio "Conversione Binario-Decimale" */

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

#define BASE 2

int main(void)
{
    int N ;         /* numero di cifre binarie */
    int bit ;       /* cifra binaria */
    int peso ;      /* peso della cifra binaria */
    int numero ;    /* valore decimale del numero binario */

    /* INIZIALIZZA LE VARIABILI */
    peso = 0 ;   /* LA PRIMA CIFRA BINARIA LETTA AVRA' PESO ZERO */
    numero = 0 ; /* IL VALORE DECIMALE DEL NUMERO BINARIO E'
                    INIZIALIZZATO A ZERO */

    /* LEGGI IL NUMERO DI CIFRE BINARIE */
    printf("Immetti␣il␣numero␣di␣bit␣del␣numero␣binario:␣") ;
    scanf("%d", &N) ;

    /* LEGGI IL NUMERO BINARIO */
    printf("\nImmetti␣il␣numero␣binario␣partendo␣dal␣bit␣meno␣significativo␣\n") ;

    while ( peso < N )
    {
        /* LEGGI LA CIFRA BINARIA SUCCESSIVA */
        printf("Immetti␣la␣cifra␣binaria␣2^%d:", peso) ;
        scanf("%d", &bit) ;

        /* CALCOLA IL VALORE IN DECIMALE DELLA CIFRA BINARIA INSERITA
        E AGGIUNGILO ALLA CIFRA DECIMALE CALCOLATA FINO AD ORA*/
        numero = numero + bit * pow(BASE, peso) ;

        /* AGGIORNA IL PESO DELLA CIFRA BINARIA */
        peso = peso + 1 ;
    }

    /* STAMPA IL RISULTATO */
    printf("\n") ;
    printf("La␣cifra␣decimale␣calcolata␣e':␣%d\n", numero) ;
    exit(0) ;
}
```

## Soluzione alternativa

Viene proposta una seconda soluzione, che non usa la funzione `pow` ma calcola la potenza mediante ripetute moltiplicazioni ed inoltre controlla se le cifre inserite sono corrette.

```
/* PROGRAMMAZIONE IN C */

/* File: bindec2.c */
/* Soluzione proposta esercizio "Conversione Binario-Decimale" */
/* Versione 2 */

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

#define BASE 2

int main(void)
{
    int N ;         /* numero di cifre binarie */
    int bit ;       /* cifra binaria */
    int peso ;      /* peso della cifra binaria */
    int potenza;    /* potenza BASE^peso */
    int numero ;    /* valore decimale del numero binario */

    /* INIZIALIZZA LE VARIABILI */
    peso = 0 ;       /* LA PRIMA CIFRA BINARIA IMMESSA AVRA' PESO 0 */
    numero = 0 ;     /* IL VALORE DECIMALE DEL NUMERO BINARIO E'
                        INIZIALIZZATO A 0 */
    potenza = 1 ;    /* POICHE' PESO=0, BASE^PESO E' UGUALE A 1 */

    /* LEGGI IL NUMERO DI CIFRE BINARIE */
    printf("Immetti il numero di bit del numero binario: ") ;
    scanf("%d", &N) ;

    while ( peso < N )
    {
        /* LEGGI LA CIFRA BINARIA SUCCESSIVA */
        printf("Immetti la cifra binaria 2^%d:", peso) ;
        scanf("%d", &bit) ;

        /* CONTROLLA SE IL VALORE DELLA CIFRA BINARIA E' CORRETTO */
        if( bit >= 0 && bit < BASE)
        {
          /* CALCOLA IL VALORE IN DECIMALE DELLA CIFRA BINARIA INSERITA
          E AGGIUNGILO ALLA CIFRA DECIMALE CALCOLATA FINO AD ORA*/
          numero = numero + bit*potenza ;

           /* AGGIORNA IL PESO DELLA CIFRA BINARIA */
            peso = peso + 1 ;

            /* AGGIORNA LA POTENZA */
            potenza = potenza * BASE ;
        }
        else
            /* SE IL VALORE DELLA CIFRA BINARIA NON E' CORRETTO
               STAMPA UN MESSAGGIO */
            printf("Dato errato - reinseriscilo\n") ;
    }

    /* STAMPA IL RISULTATO */
    printf("\n") ;
    printf("La cifra decimale calcolata e': %d\n", numero) ;

    exit(0) ;
}
```

## 4.3 Media dei numeri

Si scriva un programma in linguaggio C per calcolare la media aritmetica di una serie di numeri inseriti da tastiera. L'introduzione di un valore particolare pari a "0" indica il termine del caricamento dei dati.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: media_numeri.c */
/* Soluzione proposta esercizio "Media dei numeri" */

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

int main(void)
{
    int numero ;         /* numero inserito */
    int conta ;          /* conta quanti numeri sono inseriti */
    float somma ;        /* somma dei numeri inseriti */
    float media ;        /* media dei numeri inseriti */

    /* "somma" e "media" sono di tipo float per calcolare la media
    come valore decimale con la virgola*/

    /* INIZIALIZZA LE VARIABILI */
    somma = 0 ;
    conta = 0 ;

    /* LEGGI UN NUMERO */
    printf("Inserire una serie di numeri. La condizione di terminazione "
           "e' il numero zero.\n") ;
    printf("Inserisci numero: ") ;
    scanf ("%d", &numero) ;

     /* LEGGI UNA SERIE DI NUMERI, FINO A QUANDO NON E' INSERITO IL NUMERO 0 */
     while ( numero != 0 )
     {
        /* AGGIORNA LA SOMMA DEI NUMERI INSERITI */
        somma = somma + numero ;

        /* INCREMENTA IL CONTATORE DEI NUMERI INSERITI FINO AD ORA */
        conta = conta + 1 ;

        /* LEGGI UN NUMERO */
        printf("Inserisci numero: ") ;
        scanf ("%d", &numero);
     }

    /* CALCOLA LA MEDIA DEI NUMERI INSERITI */
    media = somma/conta ;

    /* STAMPA IL RISULTATO */
    printf("\n") ;
    printf("Numeri inseriti %d, Somma %f, Media %f \n", conta, somma, media);
    exit(0) ;
}
```

## 4.4 Massimo e minimo

Si scriva un programma in linguaggio C per calcolare il valore massimo e minimo di un insieme di N numeri inseriti da tastiera. Il programma deve leggere il valore di N, ed in seguito deve leggere una sequenza di N numeri. A questo punto il programma deve stampare il massimo ed il minimo tra i numeri inseriti.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: massimo_minimo.c */
/* Soluzione proposta esercizio "Massimo e minimo" */

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

int main(void)
{
    int numero ;         /* numero inserito */
    int N ;              /* quanti numeri saranno inseriti */
    int max, min ;       /* valore massimo e minimo tra i numeri inseriti */

    /* LEGGI QUANTI NUMERI SARANNO INSERITI */
    printf("Indica quanti numeri saranno inseriti: ") ;
    scanf("%d", &N) ;

    /* VERIFICA CHE LA SEQUENZA INSERITA CONTENGA ALMENO UN NUMERO*/
     if ( N <= 0 )
         printf("Errore: non sara' inserito nessun numero \n") ;
     else
     {
         /* LEGGI UN NUMERO */
         printf("Inserisci un numero: ") ;
         scanf ("%d", &numero) ;

         /* N VIENE DECREMENTATO POICHE' E' STATO INSERITO UN NUMERO */
         N = N - 1 ;

         /* INIZIALIZZA "max" e "min" CON IL PRIMO NUMERO INSERITO */
         max = numero ;
         min = numero ;

         /* LEGGI GLI ALTRI NUMERI DELLA SEQUENZA */
         while ( N > 0 )
         {
              /* LEGGI UN NUMERO */
              printf("Inserisci un numero: ") ;
              scanf ("%d", &numero) ;

              /* AGGIORNA IL VALORE MASSIMO "max" */
              if ( numero > max )
                 max = numero ;
              else
              {
                 /* AGGIORNA IL VALORE MINIMO "min" */
                 if ( numero < min )
                    min = numero ;
              }

             /* N VIENE DECREMENTATO POICHE' E' STATO INSERITO UN NUMERO */
             N = N - 1 ;
         }

         /* STAMPA IL RISULTATO */
         printf("\n") ;
         printf("Valore massimo %d, Valore minimo %d\n", max, min) ;
     }
     exit(0) ;
}
```

## 4.5 Quadrati perfetti

Si scriva un programma in linguaggio C per il calcolo dei quadrati perfetti per una sequenza di numeri. Il programma deve prima leggere un numero inserito da tastiera, e quindi stampare i primi quadrati perfetti sino al quadrato del numero.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: quadrati_perfetti.c */
/* Soluzione proposta esercizio "Quadrati perfetti" */

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

int main(void)
{
    int numero_finale ; /* numero inserito */
    int N ;             /* numero compreso tra 0 e "numero_finale" */
    int quadrato ;      /* quadrato del numero "N" */

    /* LEGGI UN NUMERO */
    printf("Inserisci un numero intero e positivo: ") ;
    scanf("%d", &numero_finale) ;

    /* VERIFICA CHE IL NUMERO INSERITO SIA POSITIVO */
    if ( numero_finale < 0 )
        printf("Errore: il numero deve essere positivo\n") ;
    else
    {
        /* INIZIALIZZA IL NUMERO "N" CON IL VALORE 0 */
        N = 0 ;

        /* CONSIDERA TUTTI I NUMERI TRA 0 E "numero_finale"
        E PER OGNI NUMERO CALCOLA IL QUADRATO */
        while ( N <= numero_finale )
        {
            /* CALCOLA IL QUADRATO DEL NUMERO "N" */
            quadrato = pow(N,2) ;

            /* IN ALTERNTIVA E' POSSIBILE CALCOLARE IL
            QUADRATO di "N" COME quadrato = N * N ; */

            /* STAMPA IL RISULTATO */
            printf("\n") ;
            printf("Numero %d, Quadrato %d\n", N, quadrato) ;

            /* INCREMENTA IL VALORE DEL NUMERO "N" */
            N = N + 1 ;
        }
    }
    exit(0) ;
}
```

## 4.6 Fattoriale

Si scriva un programma in linguaggio C che acquisisca un numero intero positivo N da tastiera e stampi il valore del fattoriale di N.
*Suggerimento.* Si ricorda che il fattoriale di un numero è il prodotto di tutti i numeri compresi tra 1 ed N.

$$N! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (N-1) \cdot N$$

Inoltre $0! = 1$.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: fattoriale.c */
/* Soluzione proposta esercizio "Fattoriale" */

```

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

int main(void)
{
    int N ;             /* numero inserito */
    int fattoriale ;    /* fattoriale del numero */

    /* LEGGI UN NUMERO */
    printf("Inserisci un numero intero positivo: ") ;
    scanf("%d", &N) ;

    /* VERIFICA CHE IL NUMERO INSERITO SIA POSITIVO */
    if ( N < 0 )
        printf("Errore: il numero inserito deve essere positivo\n") ;
    else
    {
        /* INIZIALIZZA IL VALORE DEL FATTORIALE */
        fattoriale = 1 ;

        /* IL FATTORIALE E' CALCOLATO COME PRODOTTO
          TRA TUTTI I NUMERI COMPRESI TRA "N" E 1 */
        while( N > 1 )
        {
                /* AGGIORNA IL VALORE DEL FATTORIALE */
                fattoriale = fattoriale * N ;

                /* DECREMENTA IL VALORE DI "N" */
                N = N - 1 ;
        }

        /* STAMPA IL RISULTATO */
        printf("\n") ;
        printf("Il fattoriale e' %d\n", fattoriale) ;
     }
     exit(0) ;
}
```

## 4.7 Classificazione di sequenze

Si scriva un programma in linguaggio C per poter analizzare una sequenza di numeri. Dati N numeri interi letti da tastiera si vogliono calcolare e stampare su schermo diversi risultati:

- quanti sono i numeri positivi, nulli e negativi
- quanti sono i numeri pari e dispari
- se la sequenza dei numeri inseriti è crescente, decrescente oppure né crescente né decrescente.

*Suggerimento.* Una sequenza è crescente se ogni numero è maggiore del precedente, decrescente se ogni numero è minore del precedente, né crescente né decrescente in tutti gli altri casi.

### Soluzione

```c
/* PROGRAMMAZIONE IN C */

/* File: sequenzanumeri.c */
/* Soluzione proposta esercizio "Classificazione di sequenze" */

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

int main(void)
{
```

```
    int totale ;                    /* quanti numeri saranno inseriti */
    int numero ;                    /* ultimo numero inserito */
    int numero_precedente ;         /* penultimo numero inserito */
    int N ;                         /* contatore per scandire i
                                       numeri della sequenza */
    int positivi, negativi, nulli; /* contatori numeri postivi, negativi,
                                       o nulli */
    int pari, dispari;              /* contatori numeri pari o dispari */
    int crescente, decrescente ;    /* flag per indicare se la sequenza e'
                                       crescente o decrescente */

    /* LEGGI QUANTI NUMERI SARANNO INSERITI */
    printf("Quanti␣numeri␣saranno␣inseriti?␣") ;
    scanf("%d", &totale) ;

    /* INIZIALIZZA A ZERO I CONTATORI DI NUMERI POSITIVI, NEGATIVI, NULLI,
    PARI E DIPARI */
    positivi = 0 ;
    negativi = 0 ;
    nulli = 0 ;
    pari = 0 ;
    dispari = 0 ;

    /* INIZIALIZZA I FLAG PER INDICARE SE LA SEQUENZA E' CRESCENTE O DECRESCENTE
    -- SE "crescente" E' UGUALE a 1: SEQUENZA CRESCENTE
    -- SE "crescente" E' UGUALE a 0: SEQUENZA NON CRESCENTE
    -- SE "decrescente" E' UGUALE a 1: SEQUENZA DECRESCENTE
    -- SE "decrescente" E' UGUALE a 0: SEQUENZA NON DECRESCENTE
    INIZIALIZZA AD 1 ENTRAMBI I FLAG. ALL'INTERNO DEL CICLO WHILE
    ASSEGNA I FLAG A 0 SE VERIFICHI CHE LA SEQUENZA NON E' CRESCENTE O
    NON E' DECRESCENTE */
    crescente = 1 ;
    decrescente = 1 ;

    /* INIZIALIZZA IL CONTATORE DEI NUMERI GIA' INSERITI */
    N = 0 ;

    /* RIPETI IL SEGUENTE CICLO FINO A QUANDO NON SONO STATI INSERITI TUTTI
    I NUMERI DELLA SEQUENZA */
    while( N < totale )
    {
         /* LEGGI UN NUMERO */
         printf("Inserisci␣il␣numero␣%d:␣", N+1) ;
         scanf("%d", &numero) ;

         /* SE IL NUMERO E' UGUALE A ZERO INCREMENTA IL CONTATORE "nulli" */
         if ( numero == 0 )
                nulli = nulli + 1 ;
         else
         {
                /* IL NUMERO E' DIVERSO DA ZERO. SE NUMERO E' POSITIVO
                INCREMENTA IL CONTATORE "positivi" ALTRIMENTI INCREMENTA
                IL CONTATORE "negativi" */
                if ( numero > 0 )
                  positivi = positivi + 1 ;
                else
                  negativi = negativi + 1 ;
         }

         /* SE IL NUMERO E' PARI INCREMENTA IL CONTATORE "pari"
         ALTRIMENTI INCREMENTA IL CONTATORE "dispari" */
         if ( numero % 2 == 0 )
                 pari = pari + 1 ;
         else
                 dispari = dispari + 1 ;

         /* PER VERIFICARE SE LA SEQUENZA E' CRESCENTE O DECRESENTE
         CONFRONTA IL NUMERO CORRENTE CON IL PENULTIMO NUMERO INSERITO.
         LA VERIFICA PUO' ESSERE FATTA SOLO QUANDO SONO STATI INSERITI
         ALMENO DUE NUMERI DELLA SEQUENZA, OSSIA N>1. INFATTI,
```

```
            N==0 QUANDO VIENE INSERITO IL PRIMO NUMERO E N==1 QUANDO VIENE
            INSERITO IL SECONDO NUMERO */

            if ( N > 1 )
            {
                    /* SE IL NUMERO CORRENTE E' MAGGIORE DEL PRECEDENTE LA
                    SEQUENZA NON E' DECRESCENTE */
                    if ( numero > numero_precedente )
                         decrescente=0;
                    else
                    {
                         /* SE IL NUMERO CORRENTE E' MINORE DEL PRECEDENTE LA
                         SEQUENZA NON E' CRESCENTE */
                         if (numero < numero_precedente)
                              crescente=0;
                         else
                         {
                         /* SE IL NUMERO CORRENTE E' UGUALE AL PRECEDENTE LA
                         SEQUENZA NON E' STRETTAMENTE CRESCENTE NE'
                         STRETTAMENTE DECRESCENTE */
                         crescente=0;
                         decrescente=0;
                         }
                    }
            }

            /* IL NUMERO CORRENTE SARA' IL PENULTIMO NUMERO INSERITO NELLA PROSSIMA
            ITERAZIONE DEL CICLO */
            numero_precedente=numero;

            /* INCREMENTA IL CONTATORE DEI NUMERI INSERITI */
            N = N + 1 ;
        }

        /* STAMPA IL RISULTATO */
        printf("Hai␣inserito:␣%d␣positivi,␣%d␣negativi,␣%d␣uguali␣a␣zero\n",
        positivi, negativi, nulli) ;

        printf("Hai␣inserito:␣%d␣numeri␣pari␣e␣%d␣numeri␣dispari\n",
               pari, dispari) ;

        if ( crescente == 1 )
            printf("La␣sequenza␣e'␣crescente\n") ;
        else
        {
            if ( decrescente == 1 )
                    printf("La␣sequenza␣e'␣decrescente\n") ;
            else
                    printf("La␣sequenza␣non␣e'␣ne`␣crescente␣ne`␣decrescente\n") ;
        }

        exit(0) ;
}
```

## 4.8 Divisori di un numero

Sia dato un numero intero positivo $N$ inserito da tastiera. Si scriva un programma in linguaggio C che calcoli i numeri interi che sono divisori (con resto uguale a zero) di $N$. Dire inoltre se $N$ è un numero primo.
*Suggerimento.*

- Un numero $M$ è divisore di un numero $N$ se il resto della divisione $N/M$ è uguale a zero.
- Un numero è primo se è divisibile solo per 1 o per il numero stesso.

### Soluzione

```c
/* PROGRAMMAZIONE IN C */

/* File: divisori.c */
/* Soluzione proposta esercizio "Divisori di un numero" */

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

int main(void)
{
    int numero ;   /* numero inserito */
    int divisore ; /* divisore del numero. E' un contatore per scandire
                      tutti i valori tra 1 e "numero"  */
    int primo ;    /* flag per indicare se il numero inserito e' primo */

    /* LEGGI UN NUMERO */
    printf("Inserisci un numero intero positivo: ") ;
    scanf("%d", &numero) ;

    /* CONTROLLA SE IL NUMERO E' POSITIVO */
    if ( numero <= 0 )
        printf("Errore: hai inserito un numero nullo o negativo\n") ;
    else
    {
        /* PER CALCOLARE I DIVISORI CONSIDERA
        TUTTI I NUMERI COMPRESI TRA 1 E "numero" */
        divisore=1 ;

        /* INIZIALIZZA IL FLAG "primo":
        -- SE "primo" E' UGUALE a 1: "numero" E' PRIMO
        -- SE "primo" E' UGUALE A 0: "numero" NON E' PRIMO.
        IPOTIZZA CHE  "numero" SIA PRIMO ED INIZIALIZZA primo=1.
        ALL'INTERNO DEL CICLO ASSEGNA primo=0 SE VERIFICHI CHE
        "numero" NON E' PRIMO (OSSIA SE E' DIVISIBILE CON RESTO ZERO
        ALMENO PER UN VALORE DIVERSO DA 1 E DA "numero") */
        primo = 1 ;

        /* IL CICLO ANALIZZA TUTTI I VALORI DI "divisore"
        COMPRESI TRA 1 E "numero" */
        while ( divisore <= numero )
        {
            /* VERIFICA SE IL RESTO DELLA DIVISIONE E' UGUALE A ZERO */
            if ( numero%divisore == 0 )
            {
                /* STAMPA IL RISULTATO */
                printf("%d e' divisore di %d\n", divisore, numero) ;

                /* SE "divisore" E' DIVERSO SIA DA 1 CHE DA "NUMERO"
                ALLORA "numero" NON E' PRIMO*/
                if ( divisore != 1 && divisore != numero )
                    primo=0;
            }

            /* INCREMENTA IL VALORE DEL POSSIBILE DIVISORE DI "numero" */
            divisore = divisore + 1 ;
        }
    }

    /* STAMPA IL RISULTATO */
    if ( primo == 1 )
        printf("%d e' un numero primo \n", numero) ;
    else
        printf("%d non e' un numero primo \n", numero) ;

    exit(0) ;
}
```

## 4.9 Massimo comune divisore di 2 numeri

Si scriva un programma in linguaggio C per calcolare il massimo comun divisore (MCD) di due numeri interi positivi. Il MCD è definito come il massimo tra i divisori comuni ai due numeri.
*Suggerimento.* Si considerino due numeri interi N1 e N2. Il MCD di N1 e N2 è il massimo tra i numeri che sono divisori (con resto uguale a zero) sia di N2 che di N1. In particolare, si supponga che sia N1 minore di N2. Il MCD è il massimo tra i numeri compresi tra 1 e N1 che sono divisori (con resto uguale a zero) sia di N1 che di N2.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: massimo_comun_divisore.c */
/* Soluzione proposta esercizio "Massimo comune divisore di 2 numeri" */

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

int main(void)
{
    int numero1, numero2 ;   /* numeri inseriti */
    int minimo ;             /* valore minimo tra numero1 e numero2 */
    int divisore ;           /* divisore del numero. E' un contatore per
                                scandire tutti i valori tra 1 e "minimo" */
    int mcd ;                /* massimo comun divisore */

    /* LEGGI I DUE NUMERI */
    printf("Inserisci il primo numero: ") ;
    scanf("%d", &numero1) ;

    printf("Inserisci il secondo numero: ") ;
    scanf("%d", &numero2) ;

    /* CONTROLLA SE ENTRAMBI I NUMERI SONO POSITIVI */
    if ( numero1 <= 0 || numero2 <= 0 )
        printf("Errore: hai inserito un numero nullo o negativo\n") ;
    else
    {
        /* CALCOLA IL VALORE INFERIORE TRA I DUE NUMERI INSERITI*/
        if ( numero1 < numero2 )
            minimo = numero1 ;
        else
            minimo = numero2 ;

        /* PER CALCOLARE IL MASSIMO COMUN DIVISORE CONSIDERA
        TUTTI I NUMERI COMPRESI TRA 1 E "minimo". IL MASSIMO COMUN DIVISORE
        E' IL MASSIMO TRA I VALORI COMPRESI TRA 1 e "minimo" CHE E' DIVISORE
        SIA DI "numero1" CHE DI "numero2" */
        divisore=1;
        mcd=1;

        while ( divisore <= minimo )
        {
            /* VERIFICA SE IL NUMERO RAPPRESENTATO IN "divisore"
            E' DIVISORE, CON RESTO UGUALE A 0, SIA DI "numero1" CHE
            DI "numero2" */
            if ( numero1%divisore == 0  && numero2%divisore == 0 )
            {
                /* POICHE' IL RESTO E' UGUALE A 0, IL VALORE DI "divisore"
                E' UN POSSIBILE MASSIMO COMUN DIVISORE. AGGIORNA IL VALORE
                DEL MASSIMO COMUM DIVISORE */
                mcd = divisore ;
                printf("%d e' divisore \n", mcd) ;
            }
            /* INCREMENTA IL VALORE DI "divisore" */
```

```
            divisore = divisore + 1 ;
        }

        /* STAMPA IL RISULTATO */
        printf("\n") ;
        printf("Il massimo comun divisore per i numeri %d e %d e' %d\n",
               numero1, numero2, mcd) ;
    }
    exit(0) ;
}
```

## 4.10 Minimo comune multiplo di 2 numeri

Si scriva un programma in linguaggio C per calcolare il minimo comune multiplo (MCM) di due numeri interi positivi. Dati due numeri interi N1 e N2, il minimo comune multiplo è il più piccolo numero M che è divisibile (con resto pari a zero) sia per N1 che per N2.
*Suggerimento.* Si considerino due numeri interi N1 e N2. Sia N1 più grande di N2. Il MCM è il primo multiplo di N1 che è divisibile (con resto uguale a zero) per N2.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: minimo_comune_multiplo.c */
/* Soluzione proposta esercizio "Minimo comune multiplo di 2 numeri" */

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

int main(void)
{
    int numero1, numero2 ; /* numeri inseriti */
    int massimo, minimo ;  /* valore massimo e minimo tra numero1 e numero2 */
    int conta ;            /* contatore per generare i multipli di "massimo" */
    int fine ;             /* flag per indicare che e' stato trovato
                              il minimo comune multiplo */
    int mcm ;              /* valore del minimo comune multiplo */

    /* LEGGI I DUE NUMERI */
    printf("Inserisci il primo numero: ") ;
    scanf("%d", &numero1) ;

    printf("Inserisci il secondo numero: ") ;
    scanf("%d", &numero2) ;

    /* CONTROLLA SE ENTRAMBI I NUMERI SONO POSITIVI */
    if ( numero1<=0 || numero2<=0 )
        printf("Errore: hai inserito un numero nullo o negativo\n") ;
    else
    {
        /* CALCOLA IL VALORE MAGGIORE E INFERIORE TRA I DUE NUMERI INSERITI*/
        if ( numero1 > numero2 )
        {
            massimo = numero1 ;
            minimo = numero2 ;
        }
        else
        {
            massimo = numero2 ;
            minimo = numero1 ;
        }

        /* INIZIALIZZA "conta" e "mcm" */
        conta=1;
        mcm=0;

        /* INIZIALIZZA IL FLAG "fine" A 0. LA RICERCA TERMINA QUANDO "fine"
        ASSUME IL VALORE 1 */
```

```
        fine = 0 ;

        while ( fine == 0 )
        {
            /* CALCOLA IL SUCCESSIVO MULTIPLO DI "massimo". QUESTO VALORE E'
            UN CANDIDATO MINIMO COMUNE MULTIPLO */
            mcm = conta * massimo ;

            /* VERIFICA SE "minimo" E' DIVISORE DI "mcm" */
            if ( mcm % minimo == 0 )
            {
                /* LA RICERCA E' TERMINATA. AGGIORNA IL FLAG "fine" */
                fine = 1 ;
            }
            else
            {
                /* INCREMENTA LA VARIABILE "conta" */
                conta = conta + 1 ;
            }
        }

        /* STAMPA IL RISULTATO */
        printf("\n") ;
        printf("Il␣MCM␣per␣%d␣e␣%d␣e'␣%d\n",
               numero1, numero2, mcm);

    }
    exit(0) ;
}
```

## 4.11 Disegno figure geometriche

1. Si realizzi un programma in linguaggio C che legga un numero intero N e visualizzi un quadrato di asterischi di lato N (vedi esempio con N = 5).
2. Si realizzi una variante del programma per visualizzare solo i lati del quadrato (vedi esempio con N = 5).
3. Si realizzi una variante del programma per visualizzare un triangolo isoscele rettangolo di lato N (vedi esempio con N = 5).
4. Si realizzi una variante del programma per visualizzare un quadrato di lato N come nell'esempio del caso 4 (con N = 5).

```
Caso 1      Caso 2      Caso 3      Caso 4

*****       *****       *           *++++
*****       *   *       **          **+++
*****       *   *       ***         ***++
*****       *   *       ****        ****+
*****       *****       *****       *****
```

### Soluzione Caso 1

```
/* PROGRAMMAZIONE IN C */

/* File: quadasterisco.c */
/* Soluzione proposta esercizio "Disegno figure geometriche (Caso 1)" */

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

int main(void)
{
    int lato ;           /* lato del quadrato */
```

```
    int riga, colonna ;  /* riga e colonna del quadrato */

    /* LEGGI IL LATO DEL QUADRATO */
    printf("Inserisci il lato del quadrato: ") ;
    scanf("%d",&lato) ;

    /* CONTROLLA SE IL LATO DEL QUADRATO E' UN NUMERO MAGGIORE DI 0 */
    if ( lato <= 0 )
        printf("Errore, il lato deve essere maggiore di zero\n") ;
    else
    {
        /* IL CICLO PIU' ESTERNO SCANDISCE LA RIGHE DEL QUADRATO */

        /* INIZIALIZZA LA VARIABILE PER LA SCANSIONE DELLE RIGHE DEL QUADRATO */
        riga = 0 ;

        while ( riga < lato )
        {
            /* IL CICLO PIU' INTERNO SCANDISCE LE COLONNE DEL QUADRATO */
            /* PER OGNI RIGA STAMPA "*" PER OGNI COLONNA */

            /*INIZIALIZZA LA VARIABILE PER LA SCANSIONE DELLE COLONNE
            DEL QUADRATO */
            colonna = 0 ;

            while ( colonna < lato )
            {
                /* STAMPA "*" senza andare a capo */
                printf("*") ;

                /* INCREMENTA "colonna" PER PASSARE ALLA COLONNA SUCCESSIVA */
                colonna = colonna + 1 ;
            }

            /* TERMINATA LA STAMPA DI UNA RIGA SI DEVE RIPORTARE IL CURSORE
            AL MARGINE SINISTRO DELLO SCHERMO */
            printf("\n");

            /* INCREMENTA "riga" PER PASSARE ALLA RIGA SUCCESSIVA */
            riga = riga + 1 ;
        }
    }
    exit(0) ;
}
```

## Soluzione Caso 2

```
/* PROGRAMMAZIONE IN C */

/* File: quadasterisco2.c */
/* Soluzione proposta esercizio "Disegno figure geometriche (Caso 2)" */

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

int main(void)
{
    int lato ;           /* lato del quadrato */
    int riga, colonna ;  /* riga e colonna del quadrato */

    /* LEGGI IL LATO DEL QUADRATO */
    printf("Inserisci il lato del quadrato: ") ;
    scanf("%d",&lato) ;

    /* CONTROLLA SE IL LATO DEL QUADRATO E' UN NUMERO MAGGIORE DI 0 */
    if ( lato <= 0 )
        printf("Errore, il lato deve essere maggiore di zero\n") ;
    else
    {
        /* IL CICLO PIU' ESTERNO SCANDISCE LA RIGHE DEL QUADRATO */
```

```
25          /* INIZIALIZZA LA VARIABILE PER LA SCANSIONE DELLE RIGHE DEL QUADRATO */
            riga = 0 ;

            while ( riga < lato )
            {
30              /* IL CICLO PIU' INTERNO SCANDISCE LE COLONNE DEL QUADRATO */

                /*INIZIALIZZA LA VARIABILE PER LA SCANSIONE DELLE COLONNE
                DEL QUADRATO */
                colonna = 0 ;
35
                while ( colonna < lato )
                {
                    /* PER LA PRIMA E L'ULTIMA RIGA STAMPA "*" PER OGNI COLONNA */
                    if ( riga == 0 || riga == (lato-1) )
40                      printf("*") ;
                    else
                    {
                        /* PER LE ALTRE RIGHE STAMPA "*" SOLO PER LA PRIMA
                        E L'ULTIMA COLONNA */
45                      if ( colonna == 0 || colonna == (lato-1) )
                            printf("*") ;
                        else
                            /* IN TUTTI GLI ALTRI CASI STAMPA UNO SPAZIO*/
                            printf("␣") ;
50                  }

                    /* INCREMENTA "colonna" PER PASSARE ALLA COLONNA SUCCESSIVA */
                    colonna = colonna + 1 ;
                }
55
                /* TERMINATA LA STAMPA DI UNA RIGA SI DEVE RIPORTARE IL CURSORE
                AL MARGINE SINISTRO DELLO SCHERMO */
                printf("\n") ;

60              /* INCREMENTA "riga" PER PASSARE ALLA RIGA SUCCESSIVA */
                riga = riga + 1 ;
            }
        }
        exit(0) ;
65  }
```

## Soluzione Caso 3

```
    /* PROGRAMMAZIONE IN C */

    /* File: triangasterisco.c */
    /* Soluzione proposta esercizio "Disegno figure geometriche (Caso 3)" */
5
    #include <stdio.h>
    #include <stdlib.h>

    int main(void)
10  {
        int lato ;           /* lato del triangolo */
        int riga, colonna ;  /* riga e colonna del triangolo */

        /* LEGGI IL LATO DEL TRIANGOLO */
15      printf("Inserisci␣il␣lato␣del␣triangolo:␣") ;
        scanf("%d", &lato) ;

        /* CONTROLLA SE IL LATO DEL TRIANGOLO E' UN NUMERO MAGGIORE DI 0 */
        if ( lato <=0 )
20          printf("Errore,␣il␣lato␣deve␣essere␣maggiore␣di␣zero\n") ;
        else
        {
            /* IL CICLO PIU' ESTERNO SCANDISCE LA RIGHE DEL TRIANGOLO */

25          /* INIZIALIZZA LA VARIABILE PER LA SCANSIONE DELLE RIGHE DEL
```

```
        TRIANGOLO */
        riga = 0 ;

        while ( riga < lato )
        {
            /* IL CICLO PIU' INTERNO SCANDISCE LE COLONNE DEL TRIANGOLO */
            /* PER OGNI RIGA STAMPA "*" SOLO SE  colonna <= riga */

            /*INIZIALIZZA LA VARIABILE PER LA SCANSIONE DELLE COLONNE DEL
            TRIANGOLO */
            colonna = 0 ;

            while ( colonna <= riga )
            {
                /* STAMPA "*" senza andare a capo */
                printf("*") ;

                /* INCREMENTA "colonna" PER PASSARE ALLA COLONNA SUCCESSIVA */
                colonna = colonna + 1 ;
            }

            /* TERMINATA LA STAMPA DI UNA RIGA SI DEVE RIPORTARE IL CURSORE
            AL MARGINE SINISTRO DELLO SCHERMO */
            printf("\n") ;

            /* INCREMENTA "riga" PER PASSARE ALLA RIGA SUCCESSIVA */
            riga = riga + 1 ;
        }
    }
    exit(0) ;
}
```

## Soluzione Caso 4

```
/* PROGRAMMAZIONE IN C */

/* File: quadasterisco3.c */
/* Soluzione PROPOSTA esercizio "Disegno figure geometriche (Caso 4)" */

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

int main(void)
{
    int lato ;           /* lato del quadrato */
    int riga, colonna ;  /* riga e colonna del quadrato */

    /* LEGGI IL LATO DEL QUADRATO */
    printf("Inserisci␣il␣lato␣del␣quadrato:␣") ;
    scanf("%d",&lato) ;

    /* CONTROLLA SE IL LATO DEL QUADRATO E' UN NUMERO MAGGIORE DI 0 */
    if ( lato <= 0 )
        printf("Errore,␣il␣lato␣deve␣essere␣maggiore␣di␣zero\n") ;
    else
    {
        /* IL CICLO PIU' ESTERNO SCANDISCE LA RIGHE DEL QUADRATO */

        /* INIZIALIZZA LA VARIABILE PER LA SCANSIONE DELLE RIGHE DEL QUADRATO */
        riga = 0 ;

        while ( riga < lato )
        {
            /* IL CICLO PIU' INTERNO SCANDISCE LE COLONNE DEL QUADRATO */

            /*INIZIALIZZA LA VARIABILE PER LA SCANSIONE DELLE COLONNE
            DEL QUADRATO */
            colonna = 0 ;

            while ( colonna < lato )
```

```
            {
                /* SE colonna <= riga  STAMPA "*" ALTRIMENTI STAMPA "+" */
                if ( colonna <= riga )
                    printf("*") ;
                else
                    printf("+") ;

                /* INCREMENTA "colonna" PER PASSARE ALLA COLONNA SUCCESSIVA */
                colonna = colonna + 1 ;
            }

            /* TERMINATA LA STAMPA DI UNA RIGA SI DEVE RIPORTARE IL CURSORE
            AL MARGINE SINISTRO DELLO SCHERMO */
            printf("\n") ;

            /* INCREMENTA "riga" PER PASSARE ALLA RIGA SUCCESSIVA */
            riga = riga + 1 ;
        }
    }
    exit(0) ;
}
```

## 4.12 Rappresentazione del triangolo di Floyd

Scrivere un programma in linguaggio C per la rappresentazione del triangolo di Floyd. Il triangolo di Floyd è un triangolo rettangolo che contiene numeri naturali, definito riempiendo le righe del triangolo con numeri consecutivi e partendo da 1 nell'angolo in alto a sinistra.

Si consideri ad esempio il caso N=5. Il triangolo di Floyd e' il seguente:

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

Il programma riceve da tastiera un numero intero N. Il programma visualizza le prime N righe del triangolo di Floyd.
*Suggerimento.* Si osserva che il numero di valori in ogni riga corrisponde all'indice della riga: 1 valore sulla prima riga, 2 sulla seconda, 3 sulla terza.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: TriangoloFloyd.c */
/* Soluzione proposta esercizio "Rappresentazione del triangolo di Floyd" */

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

int main(void)
{
    int numero ;          /* numero inserito */
    int riga, colonna ;   /* riga e colonna del triangolo */
    int cifra ;           /* numero da stampare nel triangolo di Floyd */

    /* LEGGI UN NUMERO */
    printf("Inserisci␣il␣numero␣") ;
    scanf("%d",&numero) ;

    /* CONTROLLA SE IL NUMERO E' MAGGIORE DI 0 */
    if ( numero <=0 )
        printf("Errore,␣il␣lato␣deve␣essere␣maggiore␣di␣zero\n") ;
    else
    {
```

```
        /* IL CICLO PIU' ESTERNO SCANDISCE LA RIGHE DEL TRIANGOLO */

        /* INIZIALIZZA LA VARIABILE PER LA SCANSIONE DELLE RIGHE DEL
        TRIANGOLO */
        riga = 0 ;

        /* LA PRIMA CIFRA DA STAMPARE NEL TRIANGOLO E' 1 */
        cifra=1;

        while ( riga < numero )
        {
            /* IL CICLO PIU' INTERNO SCANDISCE LE COLONNE DEL TRIANGOLO */
            /* PER OGNI RIGA STAMPA IL VALORE IN "cifra" SOLO SE
            colonna <= riga */

            /*INIZIALIZZA LA VARIABILE PER LA SCANSIONE DELLE COLONNE DEL
            TRIANGOLO */
            colonna = 0 ;

            while ( colonna <= riga )
            {
                /* STAMPA "cifra" */
                printf("%d ", cifra) ;

                /* INCREMENTA "colonna" PER PASSARE ALLA COLONNA SUCCESSIVA */
                colonna = colonna + 1 ;

                /* INCREMENTA "cifra" */
                cifra=cifra+1;
            }

            /* TERMINATA LA STAMPA DI UNA RIGA SI DEVE RIPORTARE IL CURSORE
            AL MARGINE SINISTRO DELLO SCHERMO */
            printf("\n") ;

            /* INCREMENTA "riga" PER PASSARE ALLA RIGA SUCCESSIVA */
            riga = riga + 1 ;
        }
    }
    exit(0) ;
}
```

## 4.13 Calcolo dell'opposto di un numero binario rappresentato in complemento a 2 su N bit

Scrivere un programma in linguaggio C che riceva 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 meno significativo. Il programma calcola l'opposto del numero binario ricevuto in ingresso. Tale numero sarà visualizzato partendo dalla cifra meno significativa.
*Suggerimento.* 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 alla più significativa (ossia da destra verso sinistra). Si ricopiano in uscita tutti gli zeri fino al primo 1 compreso. Dopo si invertono i restanti bit.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: opposto_ca2.c */
/* Soluzione proposta esercizio "Calcolo dell'opposto di un numero binario
rappresentato in complemento a 2 su N bit" */

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

```
int main(void)
{
    int N ;          /* numero di cifre del numero binario */
    int bit ;        /* cifra binaria del numero binario */
    int opposto ;    /* cifra binaria dell'opposto del numero binario */
    int inverti ;    /* flag per indicare se le cifre binarie devono essere
                        invertite */
    int num_bits ;   /* contatore per scandire le cifre binarie */

    /* LEGGI IL NUMERO DI CIFRE BINARIE */
    printf("Quanti bit saranno inseriti? ") ;
    scanf("%d", &N) ;

    /* INIZIALIZZA IL FLAG "inverti":
    -- SE "inverti" E' UGUALE a 1: si invertono tutti i bit inseriti
       successivamente
    -- SE "inverti" E' UGUALE A 0: si ricopiano in uscita i bit inseriti
       successivamente
    "inverti" E' INIZIALIZZATO A 0 ED ASSEGNATO A 1 QUANDO VIENE INSERITO
    IL PRIMO BIT UGUALE A 1 */
    inverti = 0 ;

    /* LEGGI LE CIFRE DEL NUMERO BINARIO A PARTIRE DAL BIT MENO SIGNIFICATIVO */
    printf("Inserisci il numero binario dal bit meno significativo\n");

    /* INIZIALIZZA "num_bits" A 0*/
    num_bits = 0 ;

    while ( num_bits < N )
    {
        /* LEGGI LA CIFRA BINARIA */
        printf("Inserisci il bit di peso %d: ", num_bits) ;
        scanf("%d", &bit) ;

        /* CALCOLA IL VALORE OPPOSTO */
        if ( inverti == 0 )
        {
            /* RICOPIA IN USCITA LA CIFRA BINARIA INSERITA */
            opposto = bit ;

            /* SE HAI TROVATO LA PRIMA CIFRA BINARIA AD 1, AGGIORNA "inverti" */
            if ( bit == 1 )
                inverti = 1 ;
        }
        else
        {
            /* RICOPIA IN USCITA L'INVERSO DELLA CIFRA BINARIA INSERITA */
            if ( bit == 1 )
                opposto = 0 ;
            else
                opposto = 1 ;
        }

        /* STAMPA IL RISULTATO */
        printf("Risultato %d\n", opposto) ;

        /* INCREMENTA IL CONTATORE "num_bits" */
        num_bits = num_bits + 1 ;
    }
    exit(0) ;
}
```

## 4.14 Somma di numeri binari

Si considerino due numeri binari rappresentati in binario puro su N bit. Il valore di N viene inserito da tastiera. I due numeri sono inseriti da tastiera un bit alla volta a partire dal bit meno significativo (LSB). Si scriva un programma in linguaggio C per eseguire la somma dei due numeri. Il programma deve visualizzare il risultato delle somma, ed indicare se si è verificata la condizione di overflow.

## Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: somma_binario.c */
/* Soluzione proposta esercizio "Somma di numeri binari" */

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

int main(void)
{
    int N ;             /* numero di cifre binarie */
    int bit_numero1 ;   /* cifra binaria del primo numero */
    int bit_numero2 ;   /* cifra binaria del secondo numero */
    int bit_risultato ; /* cifra binaria risultato dell'operazione di somma */
    int riporto ;       /* riporto */
    int num_bits ;      /* contatore per scandire le cifre binarie */

    /* LEGGI IL NUMERO CIFRE BINARIE */
    printf("Inserisci il numero di bit: ") ;
    scanf("%d", &N) ;

    /* INIZIALIZZA IL RIPORTO A 0 */
    riporto = 0;

    /* LEGGI LE CIFRE BINARIE A PARTIRE DAL BIT MENO SIGNIFICATIVO */
    printf("\nInserisci i due numeri binari partendo dal bit meno significativo\n");

    /* INIZIALIZZA "num_bits" A 0 */
    num_bits = 0 ;

    while ( num_bits < N )
    {
        /* LEGGI LA CIFRA BINARIA DEL PRIMO NUMERO */
        printf("\n");
        printf ("Inserisci la cifra %d di peso 2^%d del primo numero: ",
                num_bits+1, num_bits) ;
        scanf("%d", &bit_numero1) ;

        /* LEGGI LA CIFRA BINARIA DEL SECONDO NUMERO */
        printf ("Inserisci la cifra %d di peso 2^%d del secondo numero: ",
                num_bits+1, num_bits) ;
        scanf("%d", &bit_numero2) ;

        /* SOMMA LE DUE CIFRE BINARIE */
        bit_risultato = bit_numero1 + bit_numero2 + riporto ;

        /* VERIFICA CHE IL RISULTATO DELLA SOMMA SIA 0 O 1 */
        /* ASSEGNA IL RIPORTO A 1 SE IL RISULTATO DELLA SOMMA E' DIVERSO
        DA 0 O 1, ASSEGNA IL RIPORTO A ZERO ALTRIMENTI */
        if ( bit_risultato >= 2 )
        {
            bit_risultato = bit_risultato - 2 ;
            riporto = 1 ;
        }
        else
            riporto = 0 ;

        /* STAMPA IL RISULTATO */
        printf("Il risultato per la cifra %d di peso %d e' %d e il riporto e' %d\n",
               num_bits+1, num_bits, bit_risultato, riporto) ;

        /* INCREMENTA IL CONTATORE "num_bits" */
        num_bits = num_bits + 1 ;
    }

    /* STAMPA L'INFORMAZIONE SULLA CONDIZIONE DI OVERFLOW */
    printf("\n") ;
    if ( riporto == 1 )
```

```
        printf("La somma ha generato overflow\n") ;
    else
        printf("La somma non ha generato overflow\n") ;

    exit(0) ;
}
```

## Soluzione alternativa

```
/* PROGRAMMAZIONE IN C */

/* File: somma_binario2.c */
/* Soluzione proposta esercizio "Somma di numeri binari" */

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

int main(void)
{
    int N ;             /* numero di cifre binarie */
    int bit_numero1 ;   /* cifra binaria del primo numero */
    int bit_numero2 ;   /* cifra binaria del secondo numero */
    int bit_risultato ; /* cifra binaria risultato dell'operazione di somma */
    int riporto ;       /* riporto */
    int num_bits ;      /* contatore per scandire le cifre binarie */

    /* LEGGI IL NUMERO DI CIFRE BINARIE */
    printf("Inserisci il numero di bit: ") ;
    scanf("%d", &N) ;

    /* INIZIALIZZA IL RIPORTO A 0 */
    riporto = 0;

    /* LEGGI LE CIFRE BINARIE A PARTIRE DAL BIT MENO SIGNIFICATIVO */
    printf("\nInserisci i due numeri binari partendo dal bit meno significativo\n");

    /* INIZIALIZZA "num_bits" A 0 */
    num_bits = 0 ;

    while ( num_bits < N )
    {
        /* LEGGI LA CIFRA BINARIA DEL PRIMO NUMERO */
        printf("\n");
        printf ("Inserisci la cifra %d di peso %d del primo numero: ",
                num_bits+1, num_bits) ;
        scanf("%d", &bit_numero1) ;

        /* LEGGI LA CIFRA BINARIA DEL SECONDO NUMERO */
        printf ("Inserisci la cifra %d di peso %d del secondo numero: ",
                num_bits+1, num_bits) ;
        scanf("%d", &bit_numero2) ;

        /* SOMMA LE DUE CIFRE BINARIE */

        /* CASO 1: IL RIPORTO OTTENUTO DALLA SOMMA DEll E DUE CIFRE BINARIE
        PRECEDENTI E' 0 */
        if ( riporto == 0 )
        {
            /* VERIFICA SE LE DUE CIFRE BINARIE SONO DIVERSE
            (1 e 0 oppure 0 e 1) */
            if ( bit_numero1 != bit_numero2 )
            {
                /* SE LE DUE CIFRE BINARIE SONO DIVERSE LA SOMMA
                E' 1 E IL RIPORTO E' 0 */
                bit_risultato = 1 ;
                riporto = 0 ;
            }
            else
            {
                /* SE LE DUE CIFRE BINARIE SONO UGUALI (ENTRAMBE 1 OPPURE 0)
```

```
                LA SOMMA E' 0 */
                bit_risultato = 0 ;

65              /* SE LE DUE CIFRE BINARIE SONO UGUALI A 1 IL RIPORTO E' 1 */
                if ( bit_numero1 == 1 ) /* OPPURE bit_numero2 == 1 */
                    riporto = 1 ;
                else
                    /* SE LE DUE CIFRE BINARIE SONO UGUALI A 0 IL RIPORTO E' 0 */
70                  riporto = 0 ;
            }
        }
        else
        {
75          /* CASO 2: IL RIPORTO OTTENUTO DALLA SOMMA DELLE DUE CIFRE
            BINARIE PRECEDENTI E' 1 */

            /* VERIFICA SE LE DUE CIFRE BINARIE SONO DIVERSE
            (1 e 0 oppure 0 e 1) */
80          if (bit_numero1 != bit_numero2 )
            {
                /* SE LE DUE CIFRE BINARIE SONO DIVERSE
                LA SOMMA E' 0 E IL RIPORTO E' 1 */
                bit_risultato = 0 ;
85              riporto = 1 ;
            }
            else
            {
                /* SE LE DUE CIFRE BINARIE SONO UGUALI (ENTRAMBE 1 OPPURE 0)
90              LA SOMMA E' 1 */
                bit_risultato = 1 ;

                /* SE LE DUE CIFRE BINARIE SONO UGUALI 1 IL RIPORTO E' 1 */
                if ( bit_numero1 == 1 ) /* oppure bit_numero2 == 1 */
95                  riporto = 1 ;
                else
                    /* SE LE DUE CIFRE BINARIE SONO UGUALI A 0 IL RIPORTO E' 0 */
                    riporto = 0 ;
            }
100     }

        /* STAMPA IL RISULTATO */
        printf("Il risultato per la cifra %d di peso %d e' %d e il riporto e' %d\n",
               num_bits+1, num_bits, bit_risultato, riporto) ;
105
        /* INCREMENTA IL CONTATORE "num_bits" */
        num_bits = num_bits + 1 ;
    }

110 /* STAMPA L'INFORMAZIONE SULLA CONDIZIONE DI OVERFLOW */
    printf("\n") ;
    if ( riporto == 1 )
        printf("La somma ha generato overflow\n") ;
    else
115     printf("La somma non ha generato overflow\n") ;
    exit(0) ;
}
```

## 4.15 Conversione Decimale-Binario su un numero fisso di bit

Scrivere un programma in linguaggio C che converta un numero decimale D in un numero binario rappresentato su N bit. L'utente inserisce un numero decimale intero positivo D e il numero N di bit su cui il numero decimale deve essere rappresentata. Il programma visualizzerà i bit che compongono il numero binario partendo dal bit meno significativo. Il programma segnalerà un errore se il numero N di bit inserito dall'utente non è sufficiente per rappresentare il numero decimale.

*Suggerimento.* Per effettuare la conversione usare il metodo delle divisioni successive. Ad esempio, per convertire il numero decimale D=19 su N=7 bit, si avrà:

| Numero | Resto | Cifra binaria | Peso |
|---|---|---|---|
| 19 | 1 (19%2) | 1 | 0 |
| 9 (19/2) | 1 (9%2) | 1 | 1 |
| 4 (9/2) | 0 (4%2) | 0 | 2 |
| 2 (4/2) | 0 (2%2) | 0 | 3 |
| 1 (2/2) | 1 (1%2) | 1 | 4 |
| 0 (1/2) | 0 (0%2) | 0 | 5 |
| 0 (0/2) | 0 (0%2) | 0 | 6 |

Nota: nell'applicazione del metodo delle divisioni successive, l'iterazione termina quando è stato assegnato un valore a ciascuno degli N bit.

## Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: Decimale_Binario_FixedBits.c */
/* Soluzione proposta esercizio "Conversione Decimale-Binario su un numero fisso di bit" */

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

#define BASE 2

int main(void)
{
    int numero_decimale ;   /* numero decimale */
    int N ;                 /* numero di cifre binarie */
    int bit ;               /* cifra binaria */
    int num_bits ;          /* contatore per scandire le cifre binarie */

    /* LEGGI IL NUMERO DECIMALE */
    printf("Inserire␣il␣numero␣decimale␣da␣convertire:␣") ;
    scanf("%d", &numero_decimale) ;

    /* LEGGI IL NUMERO DI BIT */
    printf("Inserisci␣il␣numero␣di␣bit:␣") ;
    scanf("%d", &N) ;

    /* VERIFICA CHE IL NUMERO DI BIT SIA SUFFICIENTE PER RAPPRESENTARE
    IL NUMERO DECIMALE */
    if ( pow(BASE,N) - 1 < numero_decimale )
        printf("Errore:␣il␣numero␣di␣bit␣e'␣insufficiente\n");
    else
    {
        /* INIZIALIZZA "num_bits" A 0 */
        num_bits = 0 ;

        /* IL CICLO CALCOLA LE CIFRE BINARIE PER RAPPRESENTARE IL NUMERO
        DECIMALE, PARTENDO DALLA CIFRA BINARIA MENO SIGNIFICATIVA (LSB) */
        while ( num_bits < N )
        {
            /* CALCOLA LA CIFRA BINARIA DI PESO "num_bits" */
            bit = numero_decimale % BASE ;

            /* CALCOLA IL NUMERO DECIMALE DA DIVIDERE PER "dividendo"
            ALLA PROSSIMA ESECUZIONE DEL CICLO */
            numero_decimale = numero_decimale/BASE ;

            /* STAMPA IL RISULTATO */
            printf("Cifra␣binaria␣di␣peso␣2^%d:␣%d\n", num_bits, bit) ;

            /* INCREMENTA IL CONTATORE "num_bits" */
            num_bits = num_bits + 1 ;
        }
    }
    exit(0) ;
```

```
}
```

## 4.16 Numeri di Fibonacci

Scrivere un programma in linguaggio C che calcoli e stampi i primi N numeri della serie di Fibonacci, con N inserito da tastiera. La serie di Fibonacci inizia con 1, 1 ed ogni numero successivo è dato dalla somma dei due precedenti: 1, 1, 2, 3, 5, 8, 13, 21 ...

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: fibonacci.c */
/* Soluzione proposta esercizio "Numeri di Fibonacci" */

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

int main(void)
{
    int N ;              /* numero di termini della serie */
    int nuovo_termine;   /* nuovo termine della serie */
    int prec_1, prec_2 ; /* due termini precedenti nella serie */
    int num_termini;     /* contatore per scandire i termini della serie */

    /* LEGGI IL NUMERO TERMINI DELLA SEQUENZA */
    printf("Inserisci␣il␣numero␣di␣termini␣della␣serie␣di␣Fibonacci:␣") ;
    scanf("%d", &N) ;

    /* INIZIALIZZA A 1 I PRIMI DUE TERMINI DELLA SERIE */
    prec_1 = 1 ;
    prec_2 = 1 ;

    /* INIZIALIZZA A 1 IL PRIMO VALORE DELLA SERIE */
    nuovo_termine = 1 ;

    /* INIZIALIZZA A 0 IL CONTATORE CHE SCANDISCE I TERMINI DELLA SERIE */
    num_termini = 0 ;

    while ( num_termini < N )
    {
        /* I PRIMI DUE TERMINI DELLA SERIE SONO UGUALI A 1.
        I TERMINI SUCCESSIVI SONO CALCOLATI COME SOMMA DEI DUE TERMINI PRECEDENTI */
        if ( num_termini >= 2 )
        {
            /* CALCOLA IL NUOVO TERMINE DELLA SERIE */
            nuovo_termine = prec_1 + prec_2 ;

            /* AGGIORNA IL VALORE DEI DUE TERMINI PRECEDENTI NELLA SERIE */
            prec_2 = prec_1 ;
            prec_1 = nuovo_termine ;
        }

        /* STAMPA UN NUOVO TERMINE DELLA SERIE */
        printf("%d␣", nuovo_termine) ;

        /* INCREMENTA IL CONTATORE "num_termini" */
        num_termini = num_termini + 1 ;
    }

    /* RIPORTA A CAPO IL CURSORE AL TERMINE DELLA STAMPA DELLA SERIE */
    printf("\n");
    exit(0) ;
}
```

Capitolo 5

# Vettori

## 5.1 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

```
/* 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") ;

    /* 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") ;

    /* LEGGI IL NUMERO DA RICERCARE NELLA SEQUENZA */
    printf("Inserisci il numero da cercare nella sequenza: ") ;
    scanf("%d",&numero) ;

    /* VERIFICA SE LA SEQUENZA DI NUMERI CONTIENE IL NUMERO INSERITO */

    /* INIZIALIZZA IL FLAG "trovato". IL FLAG ASSUME I VALORI
    -- "trovato" E' UGUALE A 0 SE IL VETTORE "vet" NON CONTIENE IL VALORE "numero"
    -- "trovato" E' UGUALE A 1 SE IL VETTORE "vet" CONTIENE IL VALORE "numero" */
    trovato = 0 ;

    /* IL CICLO FOR SCANDISCE IL VETTORE "vet" E VERIFICA SE CONTIENE
    IL VALORE "numero".

    LA RICERCA TERMINA QUANDO SI TROVA UNA CELLA "vet[i]"
    UGUALE A "numero" O QUANDO SONO STATE CONSIDERATE TUTTE LE CELLE DEL VETTORE */

    for ( i=0; i<N && trovato==0; i++ )
    {
        if ( vet[i] == numero )
            /* SE "vet" CONTIENE IL VALORE IN "numero", AGGIORNA IL FLAG "trovato" */
            trovato = 1 ;
    }

    /* STAMPA IL RISULTATO */
    if ( trovato == 0 )
        printf("Il numero %d non e' contenuto nella sequenza inserita\n", numero) ;
    else
        printf("Il numero %d e' contenuto nella sequenza inserita\n", numero) ;

    exit(0) ;
}
```

## 5.2 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

```
/* PROGRAMMAZIONE IN C */

/* File: tutti_uguali.c */
/* Soluzione proposta esercizio "Verificare se un vettore contiene tutti
elementi tra loro uguali" */

#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 */

    /* NEL CICLO FOR SI CONFRONTA OGNI CELLA DEL VETTORE CON LA CELLA PRECEDENTE.
    SI OSSERVA CHE LA CELLA CON INDICE 0 (VET[0]) NON PUO' ESSERE CONFRONTATA
    CON LA CELLA PRECEDENTE (CON INDICE -1). PERTANTO L'INDICE "i" DEL CICLO
    ASSUME I VALORI TRA 1 E N-1 */
    for ( i=1; i < N && uguali==1; i++ )
    {
        if ( vet[i] != vet[i-1] )
            /* SE LE DUE CELLE NON CONTENGONO LO STESSO VALORE, AGGIORNA IL
            FLAG "uguali" */
            uguali = 0 ;
    }

    /* STAMPA IL RISULTATO */
    if ( uguali == 0 )
        printf("La sequenza non contiene numeri tutti uguali\n") ;
    else
        printf("La sequenza contiene numeri tutti uguali\n") ;

    exit(0) ;
}
```

## 5.3 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

```
/* PROGRAMMAZIONE IN C */

/* File: vettore_ordinato.c */
/* Soluzione proposta esercizio "Verificare se un vettore di interi e' ordinato" */

#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 crescente ;         /* flag per indicare se la sequenza e' crescente */

    /* 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 LA SEQUENZA DI NUMERI E' ORDINATA IN MODO CRESCENTE */

    /* INIZIALIZZA IL FLAG "crescente". IL FLAG ASSUME I VALORI
    -- "crescente"  E' UGUALE A 1 SE LA SEQUENZA E' CRESCENTE
    -- "crescente"  E' UGUALE A 0 SE LA SEQUENZA NON E' CRESCENTE */
    crescente = 1 ;

    /* IL CICLO FOR SCANDISCE IL VETTORE "vet" E CONTROLLA SE LA SEQUENZA
    MEMORIZZATA NEL VETTORE E' CRESCENTE. LA RICERCA TERMINA QUANDO SI VERIFICA
    CHE LA SEQUENZA NON E' CRESCENTE O QUANDO SONO STATE CONSIDERATE TUTTE
    LE CELLE DEL VETTORE */
```

```
    /* 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
        CON LA CELLA PRECEDENTE (CON INDICE -1). PERTANTO L'INDICE "i" DEL CICLO
        ASSUME I VALORI TRA 1 E N-1 */
        for ( i=1; i < N && crescente==1; i++ )
        {
65          if ( vet[i] <= vet[i-1] )
                /* SEQUENZA NON CRESCENTE, AGGIORNA IL FLAG "crescente" */
                crescente = 0 ;
        }

70      /* STAMPA IL RISULTATO */
        if ( crescente == 0 )
            printf("La␣sequenza␣non␣e'␣crescente\n") ;
        else
            printf("La␣sequenza␣e'␣crescente\n") ;
75
        exit(0) ;
    }
```

## 5.4 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" */
5
#include <stdio.h>
#include <stdlib.h>

int main(void)
10 {
    const int MAXN = 200 ;  /* dimensione massima del vettore */

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

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

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

30  /* 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") ;
        for ( i=0; i<N; i++ )
        {
            /* STAMPA IL NUMERO IN POSIZIONE "i" NEL VETTORE "vet" (OSSIA vet[i]) */
            printf("Elemento␣%d:␣%d␣", i+1, vet[i]) ;

            /* STAMPA L'ISTOGRAMMA PER IL NUMERO "vet[i]", OSSIA STAMPA UN
            NUMERO DI "*" UGUALE A vet[i] */
            for ( j=0; j < vet[i]; j++ )
                printf("*") ;
            printf("\n") ;
        }
        exit(0) ;
    }
```

## 5.5 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 4.13.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: opposto_ca2_vettori_v1.c */
/* Soluzione proposta esercizio "Calcolo dell'opposto di un
numero binario rappresentato in complemento a 2 su N bit" */

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

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

    int N ;                 /* numero di cifre del numero binario */
```

```
15      int bit[MAXN] ;          /* numero binario */
        int opposto[MAXN] ;      /* opposto del numero binario */

        int inverti ;            /* flag per indicare se le cifre binarie devono
                                    essere invertite */
20      int i ;                  /* indice dei cicli */

        /* LEGGI IL NUMERO DI CIFRE BINARIE */
        do
        {
25          printf("Quanti␣bit␣saranno␣inseriti?␣") ;
            scanf("%d", &N) ;

            if ( N > MAXN || N <=0 )
                printf("Errore:␣il␣numero␣deve␣essere␣compreso␣tra␣%d␣e␣0\n",
30                     MAXN) ;
        }
        while ( N > MAXN || N <=0 ) ;

        /* 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
        MENO SIGNIFICATIVO */

        printf("Inserisci␣le␣cifre␣binarie␣partendo␣dalla␣piu'␣significativa\n") ;
        for ( i=0; i<N; i++ )
40      {
            printf("Inserisci␣il␣bit␣di␣peso␣%d:␣", N-1-i) ;
            scanf("%d",&bit[i]) ;
        }

45      /* STAMPA IL NUMERO BINARIO INSERITO */
        printf("Il␣numero␣binario␣inserito␣e'␣il␣seguente:\n") ;
        for ( i=0; i<N; i++ )
            printf("Bit␣di␣peso␣%d:␣%d\n", N-1-i, bit[i]) ;
        printf("\n") ;
50
        /* LEGGI LE CIFRE DEL NUMERO BINARIO A PARTIRE DALLA CIFRA MENO SIGNIFICATIVA
        ("bit[N-1]") A QUELLA PIU' SIGNIFICATIVA ("bit[0]") ED ESEGUI
        LA CONVERSIONE */

55      /* INIZIALIZZA IL FLAG "inverti":
        -- SE "inverti" E' UGUALE a 1: si invertono tutte le cifre binarie successive
        -- SE "inverti" E' UGUALE A 0: si ricopiano in uscita i bit successivi
        "inverti" E' INIZIALIZZATO A 0 ED ASSEGNATO A 1 QUANDO SI TROVA IL
        PRIMO BIT UGUALE A 1 */
60      inverti = 0 ;

        for ( i=N-1; i>=0; i-- )
        {
            /* CALCOLA IL VALORE OPPOSTO */
65          if ( inverti == 0 )
            {
                /* RICOPIA IN USCITA LA CIFRA BINARIA INSERITA */
                opposto[i] = bit[i] ;

70              /* SE HAI TROVATO LA PRIMA CIFRA BINARIA AD 1, AGGIORNA "inverti" */
                if ( bit[i] == 1 )
                    inverti = 1 ;
            }
            else
75          {
                /* RICOPIA IN USCITA L'INVERSO DELLA CIFRA BINARIA INSERITA */
                if ( bit[i] == 1 )
                    opposto[i] = 0 ;
                else
80                  opposto[i] = 1 ;
            }
        }

        /* STAMPA IL RISULTATO A PARTIRE DALLA CIFRA PIU' SIGNIFICATIVA */
```

```
85      printf("Il␣numero␣binario␣risultante␣e'␣il␣seguente:\n");
        for ( i=0; i<N; i++ )
            printf("bit␣di␣peso␣%d:␣%d\n", N-1-i, opposto[i]) ;
        printf("\n") ;

90      exit(0) ;
    }
```

## 5.6 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

```
/* PROGRAMMAZIONE IN C */

/* File: shift_vettore.c */
/* Soluzione proposta esercizio "Operazione di shift di un vettore" */
5
#include <stdio.h>
#include <stdlib.h>

int main(void)
10 {
    const int MAXN = 200 ;  /* dimensione massima del vettore */

    int N ;                 /* dimensione del vettore */
    int vet[MAXN] ;         /* sequenza di numeri interi */
15  int i ;                 /* indice dei cicli */

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

        /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
        if ( N > MAXN || N <=0 )
25          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") ;

    /* ESEGUI UNO SPOSTAMENTO (SHIFT) A SINISTRA DI UNA POSIZIONE DEL CONTENUTO
    DEL VETTORE. ASSEGNA IL VALORE 0 ALLA CELLA vet[N-1] */
    for ( i=0; i<N-1; i++ )
        /* COPIA NELLA CELLA vet[i] IL CONTENUTO DELLA CELLA SUCCESSIVA vet[i+1] */
        vet[i] = vet[i+1] ;

    /* ASSEGNA IL VALORE 0 ALLA CELLA vet[N-1]. NOTA: QUESTA ASSEGNAZIONE DEVE
    ESSERE FATTA AL TERMINE DEL CICLO FOR. INFATTI SE VIENE FATTA PRIMA DEL CICLO
    FOR SI PERDEREBBE IL VALORE INIZIALMENTE CONTENUTO NELLA CELLA vet[N-1].
    QUESTO VALORE DEVE INVECE ESSERE ASSEGNATO ALLA CELLA vet[N-2] */
    vet[N-1] = 0 ;

    /* STAMPA IL VETTORE DI INTERI */
    printf("Stampa del vettore dopo l'operazione di shift a sinistra\n");
    for ( i=0; i<N; i++ )
        printf("Elemento %d: %d\n", i+1, vet[i]) ;
    printf("\n") ;

    /* ESEGUI UNO SPOSTAMENTO (SHIFT) A DESTRA DI UNA POSIZIONE DEL CONTENUTO
    DEL VETTORE. ASSEGNA IL VALORE 0 ALLA CELLA vet[0] */
    for ( i=N-1; i>0; i-- )
        /* COPIA NELLA CELLA vet[i] IL CONTENUTO DELLA CELLA PRECEDENTE vet[i-1] */
        vet[i] = vet[i-1] ;

    /* ASSEGNA IL VALORE 0 ALLA CELLA vet[0]. NOTA: QUESTA ASSEGNAZIONE DEVE
    ESSERE FATTA AL TERMINE DEL CICLO FOR. INFATTI SE VENISSE FATTA PRIMA DEL
    CICLO FOR SI PERDE IL VALORE INIZIALMENTE CONTENUTO NELLA CELLA vet[0].
    QUESTO VALORE DEVE INVECE ESSERE ASSEGNATO ALLA CELLA vet[1] */
    vet[0] = 0 ;

    /* STAMPA IL VETTORE DI INTERI */
    printf("Stampa del vettore dopo l'operazione di shift a destra\n");
    for ( i=0; i<N; i++ )
        printf("Elemento %d: %d\n", i+1, vet[i]) ;
    printf("\n") ;

    exit(0) ;
}
```

## 5.7 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

```
/* PROGRAMMAZIONE IN C */

/* File: compattazione.c */
/* Soluzione proposta esercizio "Compattazione di un vettore" */

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

int main(void)
{

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

    int vet[MAXN] ;                      /* sequenza di numeri interi */
    int compatto[MAXN] ;                 /* sequenza compatta di numeri interi */
    int N ;                              /* dimensione del vettore "vet" */
    int N_compatto ;                     /* dimensione del vettore "compatto" */
    int i, j ;                           /* indici dei cicli */
    int trovato ;                        /* flag per la ricerca */

    /* 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") ;

    /* AGGIORNA IL VETTORE "compatto" */

    /* INIZIALMENTE IL VETTORE "compatto" NON CONTIENE NESSUN NUMERO */
    N_compatto = 0 ;

    /* IL CICLO FOR SCANDISCE IL VETTORE "vet" */
    for ( i=0; i< N; i++ )
    {
        /* CONSIDERA SOLO LE CELLE IN "vet" CON VALORE DIVERSO DA 0 */
        if ( vet[i] != 0 )
        {
            /* INIZIALIZZA IL FLAG "trovato". IL FLAG ASSUME I VALORI
            -- "trovato" E' UGUALE A 0 SE IL VETTORE "compatto" NON CONTIENE
               IL VALORE IN "vet[i]"
            -- "trovato" E' UGUALE A 1 SE IL VETTORE "compatto" CONTIENE
```

```
                IL VALORE IN "vet[i]" */
            trovato=0;

            /* IL CICLO FOR SCANDISCE IL VETTORE "compatto"  E VERIFICA SE
            IL VALORE IN "vet[i]" E' PRESENTE NEL VETTORE "compatto".

            LA RICERCA TERMINA QUANDO SI TROVA ALMENO UNA CELLA "compatto[j]"
            CHE HA LO STESSO VALORE DI "vet[i]" O QUANDO SONO STATE CONSIDERATE
            TUTTE LE CELLE DEL VETTORE "compatto" */

            for ( j=0; j < N_compatto && trovato == 0; j++ )
            {
                /* SE "compatto" CONTIENE "vet[i]", AGGIORNA IL FLAG "trovato" */
                if ( compatto[j] == vet[i] )
                    trovato=1 ;
            }

            if ( trovato == 0 )
            {
                /* SE "trovato" E' UGUALE A 0, IL VETTORE "compatto" NON CONTIENE
                IL VALORE IN "vet[i]". ACCODA NEL VETTORE "compatto" IL VALORE IN
                "vet[i]" E INCREMENTA LE DIMENSIONI DEL VETTORE "compatto" */
                compatto[N_compatto] = vet[i] ;
                N_compatto = N_compatto + 1 ;
            }
        }
    }

    /* STAMPA DEL VETTORE RISULTANTE (VETTORE "compatto") */
    printf("Stampa del vettore risultante\n");
    if (N_compatto == 0)
        printf("Il vettore risultante non contiene nessun elemento \n") ;
    else
    {
        printf("Il vettore risultante contiene %d elementi \n", N_compatto) ;
        for ( i=0; i< N_compatto; i++ )
            printf("Elemento %d: %d\n", i+1, compatto[i]) ;
        printf("\n") ;
    }
    exit(0) ;
}
```

## 5.8 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 lunghezze 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

```
/* PROGRAMMAZIONE IN C */

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

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

int main(void)
{
```

```
    const int MAXN = 30 ;        /* dimensione massima dei vettori */

    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 ;                /* 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 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]) ;
    }
    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
    {
        printf("Quanti numeri saranno inseriti nel secondo vettore? ") ;
        scanf("%d", &N2) ;

        /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
        if ( N2 > MAXN || N2 <=0 )
            printf("Errore: il numero deve essere compreso tra 0 e %d\n", MAXN) ;
    }
    while ( N2 > MAXN || N2 <= 0 ) ;

    /* LEGGI IL SECONDO VETTORE */
    printf("Inserisci il secondo vettore di %d elementi\n", N2) ;
    for ( i=0; i< N2; i++ )
    {
        printf("Elemento %d: ", i+1) ;
        scanf("%d", &vet2[i]) ;
    }
    printf("\n") ;

    /* STAMPA DEL SECONDO VETTORE */
    printf("Stampa il secondo vettore\n");
    for ( i=0; i< N2; i++ )
        printf("Elemento %d: %d\n",i+1, vet2[i]) ;
    printf("\n") ;

    /* AGGIORNA IL VETTORE "intersezione" */

    /* INIZIALMENTE IL VETTORE "intersezione" NON CONTIENE NESSUN NUMERO */
    N_intersezione = 0 ;
```

```
    /* IL CICLO FOR SCANDISCE IL VETTORE "vet1" */
    for ( i=0; i<N1; i++ )
    {
        /* INIZIALIZZA IL FLAG "trovato". IL FLAG ASSUME I VALORI
          -- "trovato" E' UGUALE A 0 SE IL VETTORE "vet2" NON CONTIENE
             IL VALORE IN "vet1[i]"
          -- "trovato" E' UGUALE A 1 SE IL VETTORE "vet2" CONTIENE IL
             VALORE IN "vet1[i]" */
        trovato = 0;

        /* PER OGNI ELEMENTO "vet1[i]" DI "vet1", IL CICLO FOR SCANDISCE IL
        VETTORE "vet2" E VERIFICA SE "vet2" CONTIENE IL VALORE IN "vet1[i]"

        LA RICERCA TERMINA QUANDO SI TROVA UNA CELLA "vet2[j]" UGUALE A "vet1[i]"
        O QUANDO SONO STATE CONSIDERATE TUTTE LE CELLE DEL VETTORE "vet2" */

        for ( j=0; j<N2 && trovato==0; j++ )
        {
            if ( vet2[j] == vet1[i] )
            {
                /* SE "vet2" CONTIENE IL VALORE IN "vet1[i]", QUESTO
                VALORE E' INSERITO NEL VETTORE "intersezione" */
                intersezione[N_intersezione] = vet1[i] ;

                /* INCREMENTA LA DIMENSIONE DEL VETTORE "intersezione" */
                N_intersezione = N_intersezione + 1 ;

                /* AGGIORNA IL FLAG "trovato" */
                trovato = 1 ;
            }
        }
    }

    /* STAMPA DEL VETTORE "intersezione" */
    printf("Stampa del vettore intersezione\n");
    if (N_intersezione == 0)
        printf("Il vettore intersezione non contiene nessun elemento \n") ;
    else
    {
        printf("Il vettore intersezione contiene %d elementi \n",
               N_intersezione) ;
        for ( i=0; i< N_intersezione; i++ )
            printf("Elemento %d: %d\n", i+1, intersezione[i]) ;
        printf("\n") ;
    }
}
```

## 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
    {
        printf("Quanti numeri saranno inseriti nel secondo vettore? ") ;
        scanf("%d", &N2) ;

        /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
        if ( N2 > MAXN || N2 <=0 )
            printf("Errore: il numero deve essere compreso tra %d e 0\n", MAXN) ;
    }
    while ( N2 > MAXN || N2 <= 0 ) ;

    /* LEGGI IL SECONDO VETTORE */
    printf("Inserisci il secondo vettore di %d elementi\n", N2) ;
    for ( i=0; i< N2; i++ )
    {
        printf("Elemento %d: ", i+1) ;
        scanf("%d", &vet2[i]) ;
    }
    printf("\n") ;

    /* STAMPA DEL SECONDO VETTORE */
    printf("Stampa il secondo vettore\n");
    for ( i=0; i< N2; i++ )
        printf("Elemento %d: %d\n",i+1, vet2[i]) ;
    printf("\n") ;

    /* AGGIORNAMENTO DEL VETTORE "intersezione" */

    /* INIZIALMENTE IL VETTORE "intersezione" NON CONTIENE NESSUN NUMERO */
    N_intersezione = 0 ;

    /* IL CICLO FOR SCANDISCE IL VETTORE "vet1" */
    for ( i=0; i<N1; i++ )
    {
        /* INIZIALIZZA IL FLAG "presente". IL FLAG ASSUME I VALORI
           -- "presente" E' UGUALE A 0 SE IL VETTORE "intersezione" NON C
```

```
                CONTIENE IL VALORE IN "vet1[i]"
            -- "presente" E' UGUALE A 1 SE IL VETTORE "intersezione"
                CONTIENE IL VALORE IN "vet1[i]" */
        presente = 0 ;

        /* IL CICLO FOR SCANDISCE IL VETTORE "intersezione" E VERIFICA SE IL
        VALORE IN "vet1[i]" E' GIA' PRESENTE NEL VETTORE "intersezione"

        LA RICERCA TERMINA QUANDO SI TROVA UNA CELLA "intersezione[j]"
        UGUALE A "vet1[i]" O QUANDO SONO STATE CONSIDERATE TUTTE LE CELLE
        DEL VETTORE "intersezione" */

        for ( j=0; j<N_intersezione && presente==0; j++ )
        {
            /* SE "intersezione" CONTIENE "vet1[i]", AGGIORNA IL FLAG
            "presente" */
            if ( intersezione[j] == vet1[i] )
                presente=1 ;
        }

        /* SE IL VETTORE "intersezione" NON CONTIENE IL VALORE IN "vet1[i]",
        VERIFICA SE VETTORE "vet2" CONTIENE IL VALORE IN "vet1[i]" */
        if ( presente == 0 )
        {
            /* INIZIALIZZA IL FLAG "trovato". IL FLAG ASSUME I VALORI
            -- "trovato" E' UGUALE A 0 SE IL VETTORE "vet2" NON CONTIENE
            IL VALORE IN "vet1[i]"
            -- "trovato" E' UGUALE A 1 SE IL VETTORE "vet2" CONTIENE
            IL VALORE IN "vet1[i]" */
            trovato = 0 ;

            /* PER OGNI ELEMENTO vet1[i] DI vet1, IL CICLO FOR SCANDISCE IL
            VETTORE "vet2" E VERIFICA SE "vet2" CONTIENE IL VALORE IN "vet1[i]"

            LA RICERCA TERMINA QUANDO SI TROVA UNA CELLA "vet2[j]" UGUALE
            A "vet1[i]" O QUANDO SONO STATE CONSIDERATE TUTTE LE CELLE DEL
            VETTORE "vet2" */

            for ( j=0; j<N2 && trovato==0; j++ )
            {
                if ( vet2[j] == vet1[i] )
                {
                    /* SE "vet2" CONTIENE IL VALORE IN "vet1[i]", QUESTO
                    VALORE E' INSERITO NEL VETTORE "intersezione" */
                    intersezione[N_intersezione] = vet1[i] ;

                    /* INCREMENTA LA DIMENSIONE DEL VETTORE "intersezione" */
                    N_intersezione = N_intersezione + 1 ;

                    /* AGGIORNA IL FLAG "trovato" */
                    trovato = 1 ;
                }
            }
        }
    }

    /* STAMPA DEL VETTORE "intersezione" */
    printf("Stampa del vettore intersezione\n");
    if (N_intersezione == 0)
        printf("Il vettore intersezione non contiene nessun elemento \n") ;
    else
    {
        printf("Il vettore intersezione contiene %d elementi \n", N_intersezione) ;
        for ( i=0; i< N_intersezione; i++ )
            printf("Elemento %d: %d\n", i+1, intersezione[i]) ;
        printf("\n") ;
    }
    exit(0) ;
}
```

## 5.9 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 5.7 "Compattazione di un vettore".

### Soluzione

```
/* 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++ )
    {
        printf("Elemento␣%d:␣", i+1) ;
        scanf("%d", &vet[i]) ;
    }
    printf("\n") ;

```

```
    /* STAMPA IL VETTORE DI INTERI */
    printf("Stampa del vettore inserito\n") ;
    for ( i=0; i<N; i++ )
        printf("Elemento %d: %d\n", i+1, vet[i]) ;
    printf("\n") ;

    /* AGGIORNA IL VETTORE "compatto" E CALCOLA IL NUMERO DI OCCORRENZE */

    /* INIZIALMENTE IL VETTORE "compatto" NON CONTIENE NESSUN NUMERO */
    N_compatto = 0 ;

    /* IL CICLO FOR SCANDISCE IL VETTORE "vet1" */
    for ( i=0; i< N; i++ )
    {
        /* INIZIALIZZA IL FLAG "trovato". IL FLAG ASSUME I VALORI
        -- "trovato" E' UGUALE A 0 SE IL VETTORE "compatto" NON CONTIENE
           IL VALORE IN "vet[i]"
        -- "trovato" E' UGUALE A 1 SE IL VETTORE "compatto" CONTIENE
           IL VALORE IN "vet[i]" */
        trovato=0 ;

        /* PER OGNI ELEMENTO vet1[i] DI vet1, IL CICLO FOR SCANDISCE IL VETTORE
        "compatto" E VERIFICA SE "compatto" CONTIENE IL VALORE IN "vet1[i]"

        LA RICERCA TERMINA QUANDO SI TROVA UNA CELLA "compatto[j]"
        UGUALE A "vet1[i]" O QUANDO SONO STATE CONSIDERATE TUTTE LE CELLE
        DEL VETTORE "compatto" */

        for ( j=0; j< N_compatto && trovato==0; j++ )
        {
            /* SE "compatto" CONTIENE "vet1[i]", AGGIORNA IL FLAG "trovato" */
            if ( compatto[j] == vet[i] )
                trovato = 1 ;
        }

        if ( trovato == 0 )
        {
            /* SE "trovato" E' UGUALE A 0, COPIA NEL VETTORE "compatto" IL
            VALORE IN "vet[i]" */
            compatto[N_compatto] = vet[i] ;
            N_compatto = N_compatto + 1 ;

            /* CALCOLA IL NUMERO DI OCCORRENZE DI "vet[i]" NEL VETTORE "vet".
            IL CICLO FOR SCANDISCE IL VETTORE "vet" E CONTA QUANTE VOLTE
            IL VALORE IN "vet[i]" E' PRESENTE NEL VETTORE "vet" */
            occorrenze = 0 ;
            for ( t=0; t< N; t++ )
            {
                if ( vet[t] == vet[i] )
                    occorrenze = occorrenze + 1 ;
            }

            /* STAMPA DELLE OCCORRENZE */
            printf("Elemento %d: %d, occorrenze %d\n", i+1, vet[i], occorrenze) ;
        }
    }
    exit(0) ;
}
```

## 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.

```
/* PROGRAMMAZIONE IN C */

/* File: num_occorrenze_v2.c */
/* Soluzione proposta esercizio "Calcolo di occorrenze" */

/* 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*/

#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 N ;                            /* dimensione del vettore "vet" */
    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++ )
    {
        printf("Elemento %d: ", i+1) ;
        scanf("%d", &vet[i]) ;
    }
    printf("\n") ;

    /* STAMPA IL VETTORE DI INTERI */
    printf("Stampa del vettore inserito\n") ;
    for ( i=0; i<N; i++ )
        printf("Elemento %d: %d\n", i+1, vet[i]) ;
    printf("\n") ;

    /* CALCOLA IL NUMERO DI OCCORRENZE */

    /* IL CICLO FOR SCANDISCE IL VETTORE "vet1".
    PER OGNI CELLA "vet[i]", VERIFICA SE ESISTE UNA CELLA IN UNA DELLE POSIZIONI
    PRECEDENTI, CHE CONTIENE UN VALORE UGUALE A "vet[i]" */
    for ( i=0; i< N; i++ )
    {
        /* INIZIALIZZA IL FLAG "trovato". IL FLAG ASSUME I VALORI
        -- "trovato" E' UGUALE A 0 SE IL VETTORE "vet" NON CONTIENE
        UN'ALTRA CELLA CON LO STESSO VALORE DI "vet[i]"
        -- "trovato"  E' UGUALE A 1 SE IL VETTORE "vet" CONTIENE
        UN'ALTRA CELLA CON LO STESSO VALORE DI "vet[i]" */
        trovato=0 ;

        /* IL CICLO FOR SCANDISCE TUTTE LE CELLE DEL VETTORE "vet"
        CHE PRECEDONO "vet[i]" */
        for ( j = 0; j < i && trovato==0; j++ )
        {
            /* SE SE ESISTE UNA CELLA IN UNA DELLE POSIZIONI PRECEDENTI,
            CHE CONTIENE UN VALORE UGUALE A "vet[i]", AGGIORNA "trovato" */
            if ( vet[j] == vet[i] )
                trovato = 1 ;
        }

        if ( trovato==0 )
        {
```

```
            /* SE "trovato" E' UGUALE A 0, IL VALORE IN "vet[i]" E' CONSIDERATO
            PER LA PRIMA VOLTA. SI CALCOLANO LE OCCORRENZE DI "vet[i]" */

            /* IL CICLO FOR SCANDISCE IL VETTORE "vet" E CONTA QUANTE VOLTE
            IL VALORE IN "vet[i]" E' PRESENTE NEL VETTORE "vet" */

            occorrenze = 0 ;
            for ( t=0; t<N; t++ )
            {
                if ( vet[t] == vet[i] )
                    occorrenze = occorrenze + 1 ;
            }

            /* STAMPA DELLE OCCORRENZE */
            printf("Valore␣%d,␣occorrenze␣%d\n", vet[i], occorrenze) ;
        }
    }
    exit(0) ;
}
```

## 5.10 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

```
/* 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
```

```
    {
25      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 )
30          printf("Errore: il numero deve essere compreso tra 0 e %d\n",
                    MAXN) ;
    }
    while ( N1 > MAXN || N1 <= 0 ) ;

35  /* 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) ;
40      scanf("%d", &vet1[i]) ;
    }
    printf("\n") ;

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

50  /* LEGGI LE DIMENSIONI DEL SECONDO VETTORE */
    do
    {
        printf("Quanti numeri saranno inseriti nel secondo vettore? ") ;
        scanf("%d", &N2) ;
55
        /* LA DIMENSIONE MASSIMA DEL VETTORE E' COMPRESA TRA 1 E MAXN */
        if ( N2 > MAXN || N2 <=0 )
            printf("Errore: il numero deve essere compreso tra %d e 0\n", MAXN) ;
    }
60  while ( N2 > MAXN || N2 <= 0 ) ;

    /* LEGGI IL SECONDO VETTORE */
    printf("Inserisci il secondo vettore di %d elementi\n", N2) ;
    for ( i=0; i< N2; i++ )
65  {
        printf("Elemento %d: ", i+1) ;
        scanf("%d", &vet2[i]) ;
    }
    printf("\n") ;
70
    /* STAMPA DEL SECONDO VETTORE */
    printf("Stampa il secondo vettore\n");
    for ( i=0; i< N2; i++ )
        printf("Elemento %d: %d\n",i+1, vet2[i]) ;
75  printf("\n") ;

    /* AGGIORNA IL VETTORE "fusione" */

    /* IL VETTORE "fusione" HA DIMENSIONE PARI ALLA SOMMA DELLE
80  DIMENSIONI DI "vet1" E "vet2" */
    N_fusione = N1 + N2 ;

    /* I VETTORI "vet1", "vet2" E "fusione" SONO VISITATI RISPETTIVAMENTE
    CON GLI INDICI "j", "t", E "i" */
85  for ( i=0, j=0, t=0; i< N_fusione && j<N1 && t< N2; i++ )
    {
        if ( vet1[j] <= vet2[t] )
        {
            /* GLI ELEMENTI DI "vet1" SONO ACCODATI NEL VETTORE "fusione" */
90          /* SE "vet1[j]" E' MINORE O UGUALE DI "vet2[t]", ALLORA "vet1[j]"
            E' COPIATO IN "fusione[i]" PER PRIMO. VIENE INCREMENTATO "j",
            MENTRE "i" E' INCREMENTATO DAL CICLO FOR */
            fusione[i] = vet1[j] ;
```

```
            j = j + 1 ;
        }
        else /* vet1[j] > vet2[t] */
        {
            /* GLI ELEMENTI DI "vet2" SONO ACCODATI NEL VETTORE "fusione" */
            /* SE "vet1[t]" E' MAGGIORE DI "vet2[j]", ALLORA "vet2[t]"
            E' COPIATO IN "fusione[i]" PER PRIMO. VIENE INCREMENTATO "t", MENTRE
            "i" E' INCREMENTATO DAL CICLO FOR */
            fusione[i] = vet2[t] ;
            t = t + 1 ;
        }
    }

    if ( i < N_fusione )
    {
        /* IL VETTORE "fusione" DEVE ESSERE ANCORA COMPLETATO INSERENDO
        GLI ELEMENTI FINALI DI "vet1" O "vet2" */

        if ( j == N1 )
        {
            /* TUTTI GLI ELEMENTI DI "vet1" SONO STATI COPIATI IN "fusione".
            "fusione" VIENE ORA COMPLETATO CON GLI ELEMENTI DI "vet2" NON ANCORA
            CONSIDERATI */

            for (  ; i< N_fusione; i++, t++ )
                fusione[i] = vet2[t] ;
        }
        else
        {
            /* TUTTI GLI ELEMENTI DI "vet2" SONO STATI COPIATI IN "fusione".
            "fusione" VIENE ORA CON GLI ELEMENTI DI "vet1" NON ANCORA
            CONSIDERATI */
            for (  ; i< N_fusione; i++, j++ )
                fusione[i] = vet1[j] ;
        }
    }

    /* STAMPA DEL VETTORE "fusione"*/
    printf("Il␣vettore␣risultante␣contiene␣contiene␣%d␣elementi\n", N_fusione);
    for ( i=0; i< N_fusione; i++ )
        printf("Elemento␣%d:␣%d\n",i+1,fusione[i]);
    printf("\n");
    exit(0) ;
}
```

Capitolo 6

# Caratteri e stringhe

## 6.1 Conta vocali e consonanti

Scrivere un programma in linguaggio C che legga una frase introdotta da tastiera. La frase è terminata dall'introduzione del carattere di invio. La frase contiene sia caratteri maiuscoli che caratteri minuscoli, e complessivamente al più 100 caratteri. Il programma dovrà stampare su schermo le seguenti informazioni:

- per ognuna delle lettere dell'alfabeto, il numero di volte che la lettera compare nella stringa
- il numero di consonanti presenti nella stringa
- il numero di vocali presenti nella stringa.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: contavocaliconsonanti.c */
/* Soluzione proposta esercizio "Conta vocali e consonanti" */

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

int main(void)
{
    const int MAXDIM = 100 ;      /* dimensione massima stringa di caratteri */
    const int NUMLETTERE = 26 ;   /* numero di lettere dell'alfabeto */

    char frase[MAXDIM +1] ;       /* stringa di caratteri inserita */
    int lung_stringa ;            /* lunghezza della stringa inserita */
    int vocali, consonanti ;      /* contatori numero di vocali e di consonanti */
    int contatori[NUMLETTERE];    /* memorizza il numero di occorrenze per
                                     ogni lettera */
    int posizione_alfabeto ;      /* posizione nell'alfabeto di una lettera */
    int i ;                       /* indice dei cicli */


    /* LEGGI LA FRASE INSERITA DA TASTIERA */
    printf ("Inserisci una frase di al massimo %d caratteri: ", MAXDIM) ;
    gets(frase) ;

    /* CALCOLA LA LUNGHEZZA DELLA FRASE */
    lung_stringa = strlen(frase) ;

    /* STAMPA LA FRASE INSERITA */
    printf("La frase inserita e': ") ;
```

```
    puts(frase) ;
    printf("La␣frase␣contiene␣%d␣caratteri␣(inclusi␣gli␣spazi)\n", lung_stringa) ;

    /* AZZERA IL VETTORE DEI CONTATORI. OGNI CELLA DI QUESTO VETTORE E'
    ASSOCIATA A UNA LETTERA DELL'ALFABETO. LA CELLA 0 ALLA LETTERA A,
    LA CELLA 1 ALLA B E COSI' VIA */

    for ( i=0; i<NUMLETTERE; i++ )
        contatori[i] = 0 ;

    /* ANALIZZA LA FRASE LETTERA PER LETTERA E AGGIORNA IL VETTORE DEI CONTATORI */
    for ( i=0; i<lung_stringa; i++ )
    {
        if ( frase[i] >= 'A' && frase[i] <= 'Z' )
        {
            /* IL CARATTERE ESAMINATO E' UNA LETTERA MAIUSCOLA POICHE'
            IL SUO CODICE ASCII E' COMPRESO TRA QUELLI DELLE LETTERE A E Z.
            PER RICAVARE LA CELLA DEL VETTORE "contatori" DA INCREMENTARE
            DEVI IDENTIFICARE LA POSIZIONE DELLA LETTERA NELL'ALFABETO.
            POICHE' I CODICI ASCII DELLE LETTERE MAIUSCOLE SONO CONSECUTIVI,
            BASTERA' SOTTRARRE AL CARATTERE ESAMINATO IL CODICE ASCII DELLA
            PRIMA LETTERA DELL'ALFABETO ('A') */

            posizione_alfabeto = frase[i] - 'A' ;
            contatori[posizione_alfabeto] ++ ;
        }
        else
        {
            if ( frase[i] >= 'a' && frase[i] <= 'z' )
            {
                /* IL CARATTERE ESAMINATO E' UNA LETTERA MINUSCOLA POICHE'
                IL SUO CODICE ASCII E' COMPRESO TRA QUELLI DELLE LETTERE a E z.
                PER RICAVARE LA CELLA DEL VETTORE "contatori" DA INCREMENTARE
                DEVI IDENTIFICARE LA POSIZIONE DELLA LETTERA NELL'ALFABETO.
                POICHE' I CODICI ASCII DELLE LETTERE MINUSCOLE SONO CONSECUTIVI,
                 BASTERA' SOTTRARRE AL CARATTERE ESAMINATO IL CODICE ASCII DELLA
                PRIMA LETTERA DELL'ALFABETO ('a') */

                posizione_alfabeto = frase[i] - 'a' ;
                contatori[posizione_alfabeto] ++ ;
            }
        }
    }

    /* STAMPA I CONTATORI DELLE VARIE LETTERE */
    for ( i=0; i<NUMLETTERE; i=i+1 )
        printf ("La␣lettera␣%c␣compare␣%d␣volte␣\n",
                'A'+i , contatori[i]) ;

    /* CALCOLA IL NUMERO DI VOCALI */
    /* SOMMA IL NUMERO DI OCCORRENZE PRESENTI NEL VETTORE "contatori"
    NELLE CELLE ASSOCIATE ALLE LETTERE A, E, I, O, U, Y */
    vocali = contatori['A'-'A'] + contatori['E'-'A'] + contatori['I'-'A'] +
             contatori['O'-'A'] + contatori['U'-'A'] + contatori['Y'-'A'] ;

    /* CALCOLA IL NUMERO DI CONSONANTI */
    /* IL NUMERO DI CONSONANTI SI OTTIENE SOTTRAENDO DAL NUMERO COMPLESSIVO
    DI OCCORRENZE DI TUTTE LE LETTERE, IL NUMERO COMPLESSIVO DI VOCALI */

    consonanti = 0 ;
    for ( i=0; i<NUMLETTERE; i=i+1 )
        consonanti = consonanti + contatori[i] ;

    consonanti = consonanti - vocali ;

    /* STAMPA IL NUMERO DI VOCALI E CONSONANTI */
    printf ("Il␣numero␣di␣vocali␣e':␣%d\n", vocali) ;
    printf ("Il␣numero␣di␣consonanti␣e':␣%d\n", consonanti) ;
    exit(0) ;
}
```

## 6.2 Sostituisci carattere

Scrivere un programma in linguaggio C che legga una frase introdotta da tastiera. La frase è terminata dall'introduzione del carattere di invio e contiene complessivamente al più 100 caratteri. Il programma deve svolgere le seguenti operazioni:

- visualizzare la frase inserita
- costruire una nuova frase in cui tutte le occorrenze del carattere `'.'` sono sostituite con il carattere di ritorno di linea `'\n'`. Il programma deve memorizzare la nuova frase in una opportuna variabile
- visualizzare la nuova frase.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: sostituiscicarattere.c */
/* Soluzione proposta esercizio "Sostituisci carattere" */

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

int main(void)
{
    const int MAXDIM = 100 ;           /* dimensione max stringa di caratteri */

    char frase[MAXDIM + 1] ;           /* stringa di caratteri inserita */
    char frasemodificata[MAXDIM + 1] ; /* nuova stringa modificata */
    int lung_stringa ;                 /* lunghezza della stringa inserita */
    int i ;                            /* indice dei cicli */

    /* LEGGI LA FRASE INSERITA DA TASTIERA */
    printf ("Inserisci una frase di al massimo %d caratteri: ", MAXDIM) ;
    gets(frase) ;

    /* CALCOLA LA LUNGHEZZA DELLA FRASE */
    lung_stringa = strlen(frase) ;

    /* STAMPA LA FRASE INSERITA */
    printf("La frase inserita e': ") ;
    puts(frase) ;
    printf("La frase contiene %d caratteri (inclusi gli spazi)\n", lung_stringa) ;

    /* ANALIZZA LA FRASE INSERITA CARATTERE PER CARATTERE. RICOPIA LA FRASE
    NELLA STRINGA "frase modificata". SE LA STRINGA INSERITA CONTIENE IL
    CARATTERE ".", SOSTITUISCILO CON IL CARATTERE DI RITORNO DI LINEA "\n" */
    for ( i=0; i<lung_stringa; i=i+1 )
    {
        if ( frase[i] == '.' )
            frasemodificata[i] = '\n' ;
        else
            frasemodificata[i] = frase[i] ;
    }
    frasemodificata[lung_stringa] = '\0' ;

    /* STAMPA LA FRASE MODIFICATA */
    printf("La frase modificata e': \n") ;
    puts(frasemodificata) ;
    exit(0) ;
}
```

## 6.3 Codifica di una parola

Scrivere un programma in linguaggio C che legga una frase introdotta da tastiera. La frase è terminata dall'introduzione del carattere di invio. La frase contiene sia caratteri

maiuscoli che caratteri minuscoli, e complessivamente al più 100 caratteri. Il programma deve svolgere le seguenti operazioni:

- visualizzare la frase inserita;
- costruire una nuova frase tale che ogni lettera vocale presente nella frase di partenza sia seguita dalla lettera `'f'` (se la vocale è minuscola) o dalla lettera `'F'` (se la vocale è maiuscola) nella nuova frase. Il programma deve memorizzare la nuova frase in una opportuna variabile.
- visualizzare la nuova frase.

Ad esempio, la frase `VacAnze di NaTAle` diviene `VafcAFnzef dif NafTAFlef`.

## Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: codificadiunaparola.c */
/* Soluzione proposta esercizio "Codifica di una parola" */

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

int main(void)
{
    const int MAXDIM = 100 ;           /* dimensione max stringa di caratteri */

    char frase[MAXDIM + 1] ;             /* stringa di caratteri inserita */
    char frasemodificata[2*MAXDIM + 1] ; /* nuova stringa modificata */
    int lung_stringa ;                   /* lunghezza della stringa inserita */
    int i, j ;                           /* indici dei cicli */

    /* LEGGI LA FRASE INSERITA DA TASTIERA */
    printf ("Inserisci una frase di al massimo %d caratteri: ", MAXDIM) ;
    gets(frase) ;

    /* CALCOLA LA LUNGHEZZA DELLA FRASE */
    lung_stringa = strlen(frase) ;

    /* STAMPA LA FRASE INSERITA */
    printf("La frase inserita e': ") ;
    puts(frase) ;
    printf("La frase contiene %d caratteri (inclusi gli spazi)\n", lung_stringa) ;

    /* COSTRUISCI LA NUOVA FRASE */
    /* L'INDICE "i" E' USATO PER SCORRERE LA STRINGA "frase". L'INDICE "j" E'
    USATO PER SCORRERE LA STRINGA "frasemodificata" */
    for ( i=0, j=0; i<lung_stringa; i++ )
    {
        /* RICOPIA IL CARATTERE IN "frase[i]" nella cella "frasemodificata[j]" */
        /* INCREMENTA IL CONTATORE "j" PER ACCEDERE ALLA CELLA SUCCESSIVA
        NELLA STRINGA "frasemodificata" */
        frasemodificata[j] = frase[i] ;
        j = j + 1 ;

        /* SE "frase[i]" CONTIENE UNA VOCALE MINUSCOLA,
        INSERISCI IL CARATTERE "f" NELLA CELLA "frasemodificata[j]" */
        /* INCREMENTA IL CONTATORE "j" PER ACCEDERE ALLA CELLA SUCCESSIVA
        NELLA STRINGA "frasemodificata" */
        if ( frase[i] == 'a' || frase[i] == 'e' || frase[i] == 'i'
                || frase[i] == 'o' || frase[i] == 'u')
        {
            frasemodificata[j] = 'f' ;
            j = j + 1 ;
        }
```

```
        else
        {
            /* SE "frase[i]" CONTIENE UNA LETTERA VOCALE IN CARATTERE MAIUSCOLO,
            INSERISCI IL CARATTERE "F" NELLA CELLA "frasemodificata[j]" */
            /* INCREMENTA IL CONTATORE "j" PER ACCEDERE ALLA CELLA SUCCESSIVA
            NELLA STRINGA "frasemodificata" */
            if ( frase[i] == 'A' || frase[i] == 'E' || frase[i] == 'I'
                    || frase[i] == 'O' || frase[i] == 'U' )
            {
                frasemodificata[j] = 'F' ;
                j = j + 1 ;
            }
        }
    }
    frasemodificata[j] = '\0' ;

    /* STAMPA LA FRASE MODIFICATA */
    printf("La frase modificata e': ") ;
    puts(frasemodificata) ;
    exit(0) ;
}
```

## 6.4 Primo carattere maiuscolo

Scrivere un programma in linguaggio C che legga una frase introdotta da tastiera. La frase è terminata dall'introduzione del carattere di invio. La frase contiene sia caratteri maiuscoli che caratteri minuscoli, e complessivamente al più 100 caratteri. Il programma deve svolgere le seguenti operazioni:

- visualizzare la frase inserita
- costruire una nuova frase in cui il primo carattere di ciascuna parola nella frase di partenza è stato reso maiuscolo. Tutti gli altri caratteri devono essere resi minuscoli. Il programma deve memorizzare la nuova frase in una opportuna variabile
- visualizzare la nuova frase.

Ad esempio la frase `cHe bElLA gIOrnaTa` diviene `Che Bella Giornata`.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: primocaratteremaiuscolo.c */
/* Soluzione proposta esercizio "Primo carattere maiuscolo" */

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

int main(void)
{
    const int MAXDIM = 100 ;      /* dimensione massima stringa di caratteri */

    char frase[MAXDIM +1] ;       /* stringa di caratteri inserita */
    char nuovafrase[MAXDIM +1] ;  /* stringa di caratteri modificata */
    int lung_stringa ;            /* lunghezza della stringa inserita */
    int i ;                       /* indice dei cicli */

    /* LEGGI LA FRASE INSERITA DA TASTIERA */
    printf ("Inserisci una frase di al massimo %d caratteri: ", MAXDIM) ;
    gets(frase) ;

    /* CALCOLA LA LUNGHEZZA DELLA FRASE */
    lung_stringa = strlen(frase) ;
```

```
    /* STAMPA LA FRASE INSERITA */
    printf("La␣frase␣inserita␣e':␣") ;
    puts(frase) ;
    printf("La␣frase␣contiene␣%d␣caratteri␣(inclusi␣gli␣spazi)\n", lung_stringa) ;

    /* COSTRUISCI LA NUOVA FRASE */
    for ( i=0; i<lung_stringa; i++ )
    {
        /* IL CARATTERE "frase[i]" E' LA PRIMA LETTERA DI UNA PAROLA SE IL
        CARATTERE PRECEDENTE ("frase[i-1]") ERA UNO SPAZIO OPPURE SE E' IL PRIMO
        CARATTERE DELLA FRASE (OSSIA i==0). IN QUESTO CASO IL CARATTERE "frase[i]"
        E' CONVERTITO IN CARATTERE MAIUSCOLO. IN TUTTI GLI ALTRI CASI IL CARATTERE
        "frase[i]" E' CONVERTITO IN CARATTERE MINUSCOLO */
        if ( (i==0) || isspace(frase[i-1]) )
            nuovafrase[i] = toupper(frase[i]) ;
        else
            nuovafrase[i] = tolower(frase[i]) ;
    }
    nuovafrase[lung_stringa] = '\0' ;

    /* STAMPA LA FRASE MODIFICATA */
    printf("La␣frase␣modificata␣e':␣") ;
    puts(nuovafrase) ;
    exit(0);
}
```

## 6.5 Conversione binario decimale

Scrivere un programma in linguaggio C che legga da tastiera un numero binario puro sotto forma di una stringa di caratteri (0 o 1) lunga al massimo 24 bit. Il programma deve:

- controllare che la stringa inserita sia corretta, vale a dire composta solo da caratteri 0 e 1
- convertire il numero binario inserito nell'equivalente valore decimale
- stampare sul video il valore decimale.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: conversionebindec.c */
/* Soluzione proposta esercizio "Conversione binario decimale" */

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

int main(void)
{
    const int MAXDIM = 24 ;        /* dimensione massima stringa di caratteri */

    char binario[MAXDIM + 1] ;     /* stringa contenente il numero binario */

    int num_cifre ;                /* numero di cifre nel numero binario */
    int decimale ;                 /* numero decimale risultante */
    int corretto ;                 /* flag per la ricerca */
    int i ;                        /* indice dei cicli */

    /* LEGGI IL NUMERO BINARIO */
    printf("Inserisci␣un␣numero␣binario␣puro␣di␣al␣massimo␣%d␣cifre:␣", MAXDIM) ;
    gets(binario) ;

    /* CALCOLA IL NUMERO DI CIFRE DEL NUMERO BINARIO */
    num_cifre = strlen(binario) ;

    /* VISUALIZZA IL NUMERO INSERITO */
```

```
    printf("Il numero binario inserito e' %s e contiene %d cifre\n",
30          binario, num_cifre);

    /* VERIFICA SE IL NUMERO INSERITO CONTIENE SOLO CARATTERI 0 E 1 */
    /* IL NUMERO BINARIO NON E' CORRETTO SE CONTIENE ALMENO UNA CIFRA DIVERSA
    SIA DA 0 CHE DA 1 */
35  corretto = 1 ;
    for ( i=0 ; i<num_cifre; i++ )
        if ( binario[i]!='0' && binario[i]!='1' )
            corretto = 0 ;

40  if ( corretto == 0 )
        printf("Il numero binario inserito non e' valido\n") ;
    else
    {
        /* CONVERTI IL NUMERO BINARIO NEL NUMERO DECIMALE CORRISPONDENTE */
45      decimale = 0 ;
        for ( i=0; i<num_cifre; i++)
        {
            if ( binario[i] == '1' )
                decimale = 2*decimale + 1 ;
50          else
                decimale = 2*decimale ;
        }

        /* STAMPA IL RISULTATO */
55      printf("Il valore decimale e': %d\n", decimale) ;
    }
    exit(0) ;
}
```

## 6.6 Parola palindroma

Scrivere un programma in linguaggio C che riceve in ingresso una parola inserita da tastiera. Si consideri che la parola può contenere sia caratteri maiuscoli che caratteri minuscoli, e complessivamente al massimo 30 caratteri. Il programma deve svolgere le seguenti operazioni:

- visualizzare la parola inserita
- aggiornare la parola in modo che tutti i caratteri siano minuscoli. Il programma deve visualizzare la parola ottenuta
- verificare se la parola è palindroma. Una parola è palindroma se può essere letta indifferentemente da sinistra verso destra e da destra verso sinistra. Ad esempio, le seguenti parole sono palindrome: `otto`, `madam`.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: palindroma.c */
/* Soluzione proposta esercizio "Parola palindroma" */
5
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
10
int main(void)
{
    const int MAXDIM = 30 ;     /* dimensione massima stringa di caratteri */

15  char parola[MAXDIM+1] ;     /* stringa di caratteri inserita */
    int numcaratteri ;          /* numero di caratteri della stringa inserita */
    int palindroma ;            /* flag per la ricerca */
    int i, j ;                  /* indici dei cicli */
```

```
20
    /* LEGGI LA STRINGA DI CARATTERI INSERITA DA TASTIERA */
    printf("Inserisci una parola di al massimo %d caratteri: ", MAXDIM) ;
    scanf("%s", parola) ;

25  /* VISUALIZZA LA STRINGA DI CARATTERI INSERITA */
    printf("La parola inserita e': %s \n", parola) ;

    /* LEGGI IL NUMERO DI CARATTERI DELLA STRINGA */
    numcaratteri = strlen(parola) ;
30  printf("La parola contiene %d caratteri\n", numcaratteri) ;

    /* CONVERTI TUTTI I CARATTERI DELLA STRINGA IN CARATTERI MINUSCOLI */
    for ( i=0; i < numcaratteri ; i++ )
        parola[i] = tolower(parola[i]) ;
35
    /* VISUALIZZA LA STRINGA DI CARATTERI DOPO LA CONVERSIONE */
    printf("La parola inserita scritta solo con caratteri in minuscolo e': %s\n",
           parola) ;

40  /* VERIFICA SE LA STRINGA "parola" E' PALINDROMA */

    /* INIZIALIZZA IL FLAG "palindroma". IL FLAG ASSUME I VALORI
    -- "palindroma" E' UGUALE A 1 SE "parola"  E' PALINDROMA
    -- "palindroma" E' UGUALE A 0 SE "parola"  NON E' PALINDROMA
45  */
    palindroma = 1 ;

    /* IL CICLO FOR SCANDISCE LA STRINGA DI CARATTERI "parola" E VERIFICA
    SE E' PALINDROMA L'INDICE "i" SCORRE LA PRIMA META' DI "parola". L'INDICE
50  "j" SCORRE LA SECONDA META' DI "parola" PARTENDO DALL'ULTIMO CARATTERE.
    LA RICERCA TERMINA QUANDO SI TROVA SI VERIFICA CHE LA STRINGA "parola"
    NON E' PALINDROMA O QUANDO SONO STATI CONSIDERATI TUTTI I CARATTERI
    DI "parola" */

55  for ( i=0, j=numcaratteri - 1 ;
            i< numcaratteri/2 && palindroma==1;
            i++, j-- )
    {
        if ( parola[i] != parola[j] )
60          palindroma = 0 ;
    }

    /* STAMPA DEL RISULTATO */
    if ( palindroma == 1 )
65      printf("La parola e' palindroma\n") ;
    else
        printf("La parola non e' palindroma\n") ;

    exit(0) ;
70  }
```

## 6.7 Ricerca sottostringa

Si scriva un programma in linguaggio C che riceva in ingresso due parole inserite da tastiera. Si consideri che ciascuna parola può contenere al massimo 30 caratteri. Il programma deve verificare se la seconda parola inserita è contenuta almeno una volta all'interno della prima parola (ossia se la seconda parola è una sottostringa della prima parola).

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: ricercasottostringa_v1.c */
/* Soluzione proposta esercizio "Ricerca sottostringa" */
5

#include <stdio.h>
```

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

int main(void)
{
    const int MAXDIM = 30 ;            /* dimensione max stringa di caratteri */

    char parola1[MAXDIM + 1] ;         /* prima stringa di caratteri */
    char parola2[MAXDIM + 1] ;         /* seconda stringa di caratteri */
    int lung_stringa1, lung_stringa2 ; /* lunghezza delle due stringhe */

    /* LEGGI LA PRIMA PAROLA INSERITA DA TASTIERA */
    printf ("Inserisci una parola di al massimo %d caratteri: ", MAXDIM) ;
    gets(parola1) ;

    /* CALCOLA LA LUNGHEZZA DELLA PAROLA */
    lung_stringa1 = strlen(parola1) ;

    /* STAMPA LA PAROLA INSERITA */
    printf("La parola %s contiene %d lettere\n", parola1, lung_stringa1) ;

    /* LEGGI LA SECONDA PAROLA INSERITA DA TASTIERA */
    printf ("Inserisci una parola di al massimo %d caratteri: ", MAXDIM) ;
    gets(parola2) ;

    /* CALCOLA LA LUNGHEZZA DELLA PAROLA */
    lung_stringa2 = strlen(parola2) ;

    /* STAMPA LA PAROLA INSERITA */
    printf("La parola %s contiene %d lettere\n", parola2, lung_stringa2) ;

    /* VERIFICA SE "parola2" E' CONTENUTA IN "parola1" */
    if ( lung_stringa1 < lung_stringa2 )
        printf("La seconda parola e' piu' lunga della prima parola \n") ;
    else
    {
        if ( strstr(parola1, parola2) != NULL )
            printf("La seconda parola e' contenuta nella prima \n") ;
        else
            printf("La seconda parola non e' contenuta nella prima \n") ;
    }
    exit(0) ;
}
```

## Soluzione alternativa

```c
/* PROGRAMMAZIONE IN C */

/* File: ricercasottostringa_v2.c */
/* Soluzione proposta esercizio "Ricerca sottostringa" */

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

int main(void)
{
    const int MAXDIM = 30 ;            /* dimensione massima stringa di caratteri */

    char parola1[MAXDIM + 1] ;         /* prima stringa di caratteri inserita */
    char parola2[MAXDIM + 1] ;         /* seconda stringa di caratteri inserita */
    int lung_stringa1, lung_stringa2 ; /* lunghezza delle due stringhe inserite */

    int contenuto, finito ;            /* flag per la ricerca */
    int i, j ;                         /* indici dei cicli */

    /* LEGGI LA PRIMA PAROLA INSERITA DA TASTIERA */
    printf ("Inserisci una parola di al massimo %d caratteri: ", MAXDIM) ;
    gets(parola1) ;
```

```
25      /* CALCOLA LA LUNGHEZZA DELLA PAROLA */
        lung_stringa1 = strlen(parola1) ;

        /* STAMPA LA PAROLA INSERITA */
        printf("La␣parola␣%s␣contiene␣%d␣lettere\n", parola1, lung_stringa1) ;
30
        /* LEGGI LA SECONDA PAROLA INSERITA DA TASTIERA */
        printf ("Inserisci␣una␣parola␣di␣al␣massimo␣%d␣caratteri:␣", MAXDIM) ;
        gets(parola2) ;

35      /* CALCOLA LA LUNGHEZZA DELLA PAROLA */
        lung_stringa2 = strlen(parola2) ;

        /* STAMPA LA PAROLA INSERITA */
        printf("La␣parola␣%s␣contiene␣%d␣lettere\n", parola2, lung_stringa2) ;
40
        /* VERIFICA SE "parola2" E' CONTENUTA IN "parola1" */
        if ( lung_stringa1 < lung_stringa2 )
            printf("La␣seconda␣parola␣e'␣piu'␣lunga␣della␣prima␣parola␣\n") ;
        else
45      {
            /* IL CICLO FOR ESTERNO SCORRE LA STRINGA "parola1".
            PER OGNI CARATTERE "parola1[i]" IL CICLO FOR INTERNO ANALIZZA LA
            LA SOTTOSTRINGA CONTENENTE I CARATTERI COMPRESI TRA "parola1[i]"
            E "parola1[i+lung_stringa2-1]", E VERIFICA SE TALE SOTTOSTRINGA
50          E' UGUALE A "parola2" */

            /* IL FLAG "finito==1" INDICA LA CONDIZIONE DI FINE RICERCA.
            IL FLAG E' INIZIALIZZATO A 0 E VIENE ASSEGNATO A 1 SE "parola2" E'
            CONTENUTA IN "parola1" */
55          finito = 0 ;
            for ( i=0; i+(lung_stringa2-1)<lung_stringa1 && finito==0; i++ )
            {
                /* "j" E' L'INDICE DEL CICLO FOR INTERNO. VIENE UTILIZZATO PER
                SCORRERE I CARATTERI DELLA SOTTOSTRINGA "parola2" E DELLA
60              SOTTOSTRINGA CONTENENTE I CARATTERI COMPRESI TRA "parola1[i]"
                E "parola1[i+lung_stringa2-1]" */

                /* IL FLAG "contenuto==1" INDICA CHE LE DUE SOTTOSTRINGHE SONO
                UGUALI. IL FLAG E' INIZIALIZZATO A 1 E VIENE ASSEGNATO A 0 SE
65              ALMENO UN CARATTERE "parola1[i+j]" NELLA SOTTOSTRINGA E' DIVERSO
                DAL CORRISPONDENTE CARATTERE "parola2[j]" */
                contenuto = 1 ;
                for ( j=0; j<lung_stringa2 && contenuto==1; j++ )
                {
70                  if ( parola1[i+j] != parola2[j] )
                        contenuto = 0 ;
                }

                /* SE AL TERMINE DEL CONFRONTO TRA LE DUE STRINGHE "contenuto" E'
75              ANCORA UGUALE A 1, ALLORA "parola2" E' CONTENUTA IN "parola1".
                IL FLAG "finito" VIENE AGGIORNATO, E SI CONCLUDE LA RICERCA */
                if ( contenuto==1 )
                    finito = 1 ;
            }
80      }

        /* STAMPA IL RISULTATO */
        if ( contenuto == 1 )
            printf("La␣seconda␣parola␣e'␣contenuta␣nella␣prima␣\n") ;
85      else
            printf("La␣seconda␣parola␣non␣e'␣contenuta␣nella␣prima␣\n") ;

        exit(0) ;
    }
```

## 6.8 Sostituisci sottostringa

Si scriva un programma in linguaggio C che riceva in ingresso due parole inserite da tastiera. Si consideri che ciascuna parola può contenere al massimo 30 caratteri. Il programma deve sostituire ogni occorrenza della seconda parola nella prima parola con una sequenza di caratteri `'*'`.

Ad esempio, inserite le parole `abchdfffchdtlchd` e `chd`, il programma deve visualizzare la parola `ab***fff***tl***`.

### Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: sostituiscisottostringa.c */
/* Soluzione proposta esercizio "Sostituisci sottostringa" */

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

int main(void)
{
    const int MAXDIM = 30 ;            /* dimensione max stringa di caratteri */

    char parola1[MAXDIM + 1] ;         /* prima stringa di caratteri inserita */
    char parola2[MAXDIM + 1] ;         /* seconda stringa di caratteri inserita */
    int lung_stringa1, lung_stringa2 ; /* lunghezza delle due stringhe inserite */

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


    /* LEGGI LA PRIMA PAROLA INSERITA DA TASTIERA */
    printf ("Inserisci una parola di al massimo %d caratteri: ", MAXDIM) ;
    gets(parola1) ;

    /* CALCOLA LA LUNGHEZZA DELLA PAROLA */
    lung_stringa1 = strlen(parola1) ;

    /* STAMPA LA PAROLA INSERITA */
    printf("La parola %s contiene %d lettere\n", parola1, lung_stringa1) ;


    /* LEGGI LA SECONDA PAROLA INSERITA DA TASTIERA */
    printf ("Inserisci una parola di al massimo %d caratteri: ", MAXDIM) ;
    gets(parola2) ;

    /* CALCOLA LA LUNGHEZZA DELLA PAROLA */
    lung_stringa2 = strlen(parola2) ;

    /* STAMPA LA PAROLA INSERITA */
    printf("La parola %s contiene %d lettere\n", parola2, lung_stringa2) ;


    /* VERIFICA SE "parola2" E' CONTENUTA IN "parola1" */
    if ( lung_stringa1 < lung_stringa2 )
        printf("La seconda parola e' piu' lunga della prima parola \n") ;
    else
    {
        /* IL CICLO FOR ESTERNO SCORRE LA STRINGA "parola1".
        PER OGNI CARATTERE "parola1[i]" IL CICLO FOR INTERNO ANALIZZA LA
        LA SOTTOSTRINGA CONTENENTE I CARATTERI COMPRESI TRA "parola1[i]"
        E "parola1[i+lung_stringa2-1]", E VERIFICA SE TALE SOTTOSTRINGA
        E' UGUALE A "parola2" */

        for ( i=0; i+(lung_stringa2-1)<lung_stringa1; i++ )
        {
```

```
            /* "j" E' L'INDICE DEL CICLO FOR INTERNO. VIENE UTILIZZATO PER
            SCORRERE I CARATTERI DELLA SOTTOSTRINGA "parola2" E DELLA
            SOTTOSTRINGA CONTENENTE I CARATTERI COMPRESI TRA "parola1[i]" E
            "parola1[i+lung_stringa2-1]" */

            /* IL FLAG "contenuto==1" INDICA CHE LE DUE SOTTOSTRINGHE SONO
            UGUALI.
            IL FLAG E' INIZIALIZZATO A 1 E VIENE ASSEGNATO A 0 SE ALMENO UN
            CARATTERE "parola1[i+j]" NELLA SOTTOSTRINGA E' DIVERSO DAL
            CORRISPONDENTE CARATTERE "parola2[j]" */
            contenuto = 1 ;
            for ( j=0; j<lung_stringa2 && contenuto==1; j++ )
            {
                if ( parola1[i+j] != parola2[j] )
                    contenuto = 0 ;
            }

            /* SE AL TERMINE DEL CONFRONTO TRA LE DUE STRINGHE "contenuto" E'
            ANCORA UGUALE A 1, ALLORA "parola2" E' CONTENUTA IN "parola1".
            SOSTITUISCI ALLORA TUTTI I CARATTERI COMPRESI TRA "parola1[i]"
            E "parola1[i+lung_stringa2-1]" CON IL CARATTERE '*' */
            if ( contenuto==1 )
            {
                for ( j=0; j<lung_stringa2; j++ )
                    parola1[i+j] = '*' ;

                /*PER OTTIMIZZARE LA RICERCA SALTA NELLA STRINGA "parola1"
                LA SOTTOSEQUENZA DI ASTERISCHI APPENA INSERITA */
                i = i + lung_stringa2 - 1 ;
            }
        }
    }

    /* STAMPA IL RISULTATO */
    printf("La␣parola␣risultante␣e'␣%s␣\n", parola1) ;

    exit(0) ;
}
```