Esercizio 1

- Realizzare un programma che provveda a leggere da input delle parole separate da spazi (stringhe di al più 63 caratteri) e le ripeta su standard output (servizio di “echo”).
- Il programma deve terminare quando l’utente inserisce la parola “fine”.

Esercizio 1 - Soluzione

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

int main(void) {
    char s[64];

    do {
      scanf("%63s", s);
      if (strcmp("fine", s) != 0)
            printf("%s ", s);
    } while (strcmp("fine", s) != 0);

    system("PAUSE");
    return (0);
}
```

Esercizio 2

Un utente specifica tramite input dei dati relativi alla programmazione nei cinema della città. In particolare, specifica prima il programma settimanale dei film in proiezione (al più 30 film), e poi le descrizioni delle sale in città (al più 10 sale). Più precisamente, specifica il programma nel modo seguente:

- titolo del film (non più di 30 caratteri senza spazi), uno e un solo spazio di separazione
- nome della sala (non più di 20 caratteri senza spazi), uno e un solo spazio di separazione
- 3 orari di inizio proiezione (3 numeri interi separati da spazi), e poi «a capo»

Poi specifica le informazioni relative alle sale cinematografiche:

- nome della sala (non più di 20 caratteri senza spazi), uno e un solo spazio di separazione
- costo del biglietto (numero reale), e poi a capo.

Esercizio 2

Programma:

```
TheKingdom Nosadella 18 20 22
Dogville Fellini 17 20 22
OttoEMezzo Capitol 17 20 23
BreakingWaves Odeon 15 19 23
```

Sale:

```
Capitol 6.00
Fellini 5.50
Modernissimo 6.00
Nosadella 6.50
```

Esercizio 2

1) Si scriva una procedura load() che riceva come parametri di ingresso **un vettore y di strutture film** (titolo film, costo biglietto), la sua dimensione massima, e che restituisca come parametro di uscita **il numero degli elementi N inseriti in y**

La procedura deve chiedere all'utente di specificare tutti i dati, memorizzandoli in opportune strutture dati, e poi incrociare i dati al fine di restituire un vettore di strutture «film».

Esercizio 2

2) Si scriva un programma C che, utilizzando la procedura load() precedentemente definita, **inserisca in un vettore prezzi** (supposto di dimensione massima pari al massimo numero di film in programma) **le strutture film di cui sopra**, derivanti dalla lettura da standard input dei programmi e delle sale.

Il programma deve inoltre stampare a terminale tutti gli elementi di prezzi il cui costo del biglietto è **inferiore alla media di tutti i costi caricati nel vettore**

Esercizio 2 - Soluzione

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

typedef struct {
    char titolo[31];
    char nomeSala[21];
    int h1, h2, h3;
} Prog;

typedef struct {
    char nomeSala[21];
    float prezzo;
} Sala;

typedef struct {
    char titolo[31];
    float prezzo;
} Film;
```

Esercizio 2 - Soluzione

```
int load(Film * y, int dim) {
    Prog p[30];
    int dim_p = 0;
    Sala s[10];
    int dim_s = 0;
    int dim_y = 0;
    int i,j;

    printf("Inserire la programmazione:\n");
    while (scanf("%s %s %d %d %d", p[dim_p].titolo, p[dim_p].nomeSala,
           &(p[dim_p].h1), &(p[dim_p].h2), &(p[dim_p].h3)) == 5  &&
           strcmp(p[dim_p].titolo, "fine") != 0)
      dim_p++;

    printf("Inserire le sale:\n");
    while (scanf("%s %f", s[dim_s].nomeSala, &(s[dim_s].prezzo)) == 2 &&
      strcmp(s[dim_s].nomeSala, "fine")!=0)
      dim_s++;

    …
```

Esercizio 2 - Soluzione

```
    …

    for (i=0; i<dim_p && dim_y<dim; i++) {
      for (j=0; j<dim_s && dim_y < dim; j++) {
      if (strcmp(p[i].nomeSala, s[j].nomeSala)==0) {
            strcpy(y[dim_y].titolo, p[i].titolo);
            y[dim_y].prezzo = s[j].prezzo;
            dim_y++;
      }
      }
    }
    return dim_y;
}
```

Esercizio 3

Conta parole

- Si scriva un programma C che conti il numero dei ***caratteri***, delle ***parole*** e delle ***linee*** che l'utente digita da standard input

Esercizio 3 - Soluzione

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

int main (void) {
   int caratteri = 0;
   int linee = 0;
   int parole = 0;
   char ch, prec = ' ';

   while (scanf("%c", &ch) == 1) {
        caratteri++;
        if (ch == '\n')
                linee++;
        if (isspace(prec) && !isspace(ch))
                parole++;
        prec = ch;
   }

   printf("Il numero di caratteri e' %d.\n", caratteri);
   printf("Il numero di parole e' %d.\n", parole);
   printf("Il numero di linee e' %d.\n", linee);

   return 0;
}
```

Esercizio 4
(allocazione dinamica)

Un utente specifica tramite standard input una sequenza di interi. Per prima cosa specifica quanti interi vuole inserire, e poi procede con l'inserimento. I valori sono inseriti in ordine casuale.

Si realizzi un programma che, letti da input tali valori interi, li stampi a video ponendo prima i numeri pari e poi i numeri dispari

Esercizio 4 - Soluzione
(allocazione dinamica)

```
int main() {
     int odd, even, lung;
     int * store, * temp;
     int i=0;
     int o, e;

     printf("Quanti interi vuoi inserire? ");
     scanf("%d", &lung);
     temp = (int *) malloc(sizeof(int) * (lung));
     store = (int *) malloc(sizeof(int) * (lung));

     odd = 0; even = 0;
     for (i=0; i<lung; i++) {
         scanf("%d", &(temp[i]));
         if (temp[i]%2==0)
                 even++;
         else
                 odd++;
     }   ...
```

Esercizio 4 - Soluzione
(allocazione dinamica)

```
    …

    o=0; e=0;
    for (i=0; i<lung; i++)
        if (temp[i]%2==0) {
                store[e] = temp[i];
                e++;
        }
        else {
                store[even+o] = temp[i];
                o++;
        }

    for (i=0; i<lung; i++)
        printf("%d ", store[i]);
    free(temp); free(store);
    return 0;
}
```

Esercizio 5
(allocazione dinamica)

Un utente specifica tramite standard input una sequenza di interi. Per prima cosa specifica quanti interi vuole inserire, e poi procede con l'inserimento. I valori sono inseriti in ordine casuale. Gli interi possono essere positivi e/o negativi.

Si realizzi un programma che copi in un secondo vettore di dimensione minima possibile, i soli interi positivi, e poi ne determini e stampi a video il massimo.

Esercizio 5 - Soluzione
(allocazione dinamica)

```
int main() {
     int dim_temp;
     int dim_store;
     int * temp;
     int * store;
     int i;
     int pos;
     int max;

     printf("Quanti interi vuoi inserire? ");
     scanf("%d", &dim_temp);

     temp = (int *) malloc(sizeof(int) * (dim_temp));
     for (i=0; i<dim_temp; i++)
         scanf("%d", &(temp[i]));

     ...
```

Esercizio 5 - Soluzione
(allocazione dinamica)

```
    ...
    pos=0;
    for (i=0; i<dim_temp; i++)
        if (temp[i]>=0)
                pos++;

    store = (int *) malloc(sizeof(int) * (pos));
    pos = 0;
    for (i=0; i<dim_temp; i++)
        if (temp[i]>=0) {
                store[pos] = temp[i];
                pos++;
        }

    max = -1;
    for (i=0; i<pos; i++)
        if (store[i] >max)
                max = store[i];
    printf("Il massimo e' %d\n", max);
    free(temp);  free(store);
    return 0;
}
```

Esercizio - RAPPRESENTAZIONE

Un elaboratore rappresenta i numeri interi su 8 bit in complemento a 2. Indicare come viene svolta la seguente operazione aritmetica e determinarne il risultato traslandolo poi in decimale per la verifica:

39 + (- 91)

Esercizio - RAPPRESENTAZIONE

+39 -> 00100111

91 -> 01011011

10100100

10100101 -> -91

Conversione del valore assoluto in base 2

Inversione dei bit

Aggiungo 1

Si esegue la somma:

```
00100111 +
10100101
---------
11001100
```

11001100

00110011

00110100 -> 52 (base 10)

Complemento a 2 – quando c'è overflow?

Nella rappresentazione a complemento a 2, l'overflow può accadere solo in conseguenza di:

- Somma di due numeri positivi
- Somma di due numeri negativi

L'overflow si verifica quando uno dei due ultimi bit di riporto è a uno, ma non lo sono entrambi (xor degli ultimi due bit di riporto)

Complemento a 2 – quando c'è overflow?

Consideriamo il caso di due numeri positivi senza overflow (caso su 4 bit):

(riporto) 0 0 0 0

3 -> 0011

4 -> 0100

0111

xor(0, 0) = 0, non c'è overflow!

Complemento a 2 – quando c'è overflow?

Consideriamo il caso di due numeri positivi *con* overflow (caso su 4 bit):

```
(riporto)      0 1 1 1
3        ->    0011
5        ->    0101
               ---------
               1000
```

xor(0, 1) = 1, OVERFLOW!!!

Sommando due numeri positivi, è stato modificato il bit del segno…
… deve essere successo qualcosa!!!

Complemento a 2 – quando c'è overflow?

Consideriamo il caso di due numeri negativi *senza* overflow (caso su 4 bit):

(riporto) 1 1 0 0

-3 -> 1101

-4 -> 1100

1001

xor(1, 1) = 0, no overflow

Il bit del segno è rimasto invariato…
… ed entrambi i riporti sono ad 1…

Complemento a 2 – quando c'è overflow?

Consideriamo il caso di due numeri negativi *con* overflow (caso su 4 bit):

```
(riporto)       1 0 0 0
-3      ->      1101
-6      ->      1010
                ---------
                0111
```

xor(1, 0) = 1, OVERFLOW!!!

Il bit del segno è stato modificato…
…cosa è successo ?

Esercizio - RAPPRESENTAZIONE

Un elaboratore rappresenta i numeri interi su 8 bit in complemento a 2. Indicare come viene svolta la seguente operazione aritmetica e determinarne il risultato traslandolo poi in decimale per la verifica:

(-13) - (-91)

Esercizio - RAPPRESENTAZIONE

13 -> 00001101
11110010
11110011 -> -13

-(-91) -> 01011011

(riporti) -> 1 1 1 1 1 1

-13	->	1	1	1	1	0	0	1	1	+
-91	->	0	1	0	1	1	0	1	1	
		---	---	---	---	---	---	---	---	
		0	1	0	0	1	1	1	0	

01001110 -> 78 in base 10

Esercizio analisi

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori di v e di N al termine dell'esecuzione? (si motivi opportunamente la risposta data)

```
int main(void) {
     int N=4, i, v;

     v = 0;
     {  ++N; }

     do {
        if ((N%2) == 1)
                v = v + N;
        else
                v = v - N;
        N--;
     } while (v > 4);

     for (i=0; i<N; i++)
        v = v - 1;
     for (i=0; i < (i?N:0); )
        v = 10;
     return (0);
}
```

Esercizio analisi

Il programma compila correttamente, e al termine dell'esecuzione le variabili `v` *ed* `N` *valgono rispettivamente -2 e 3*

- *Le prime due istruzioni pongono v a 0, e poi incrementano il valore di N, che da 4 viene a valere 5.*
- *Il primo ciclo che poi si incontra nel codice è di tipo do..while, perciò almeno una volta viene eseguito. In particolare, l'istruzione di if controlla: se N è dispari allora v viene incrementato di N, altrimenti v viene decrementato di N. Il ciclo termina quando la condizione (v>4) non vale più. Alla prima iterazione dunque v viene a valere 5, ed N è decrementato a 4; alla seconda iterazione v viene a valere 1 ed N decrementato a 3: Siccome la condizione di continuazione del ciclo non è più verificata, il ciclo si interrompe.*
- *Il secondo ciclo che si incontra decrementa di 1 il valore di v, per N volte. Quindi al termine v viene a valere -2.*
- *Il terzo ciclo presente nel programma non viene eseguito, poichè la condizione è immediatamente non verificata.*

Esercizio Record di Attivazione

Si consideri la seguente funzione:

```
int funzione(float a, float b) {
    a = a+b;
    b--;
    if (b > 0)
       return funzione(a, b);
    else
       return a;
}
```

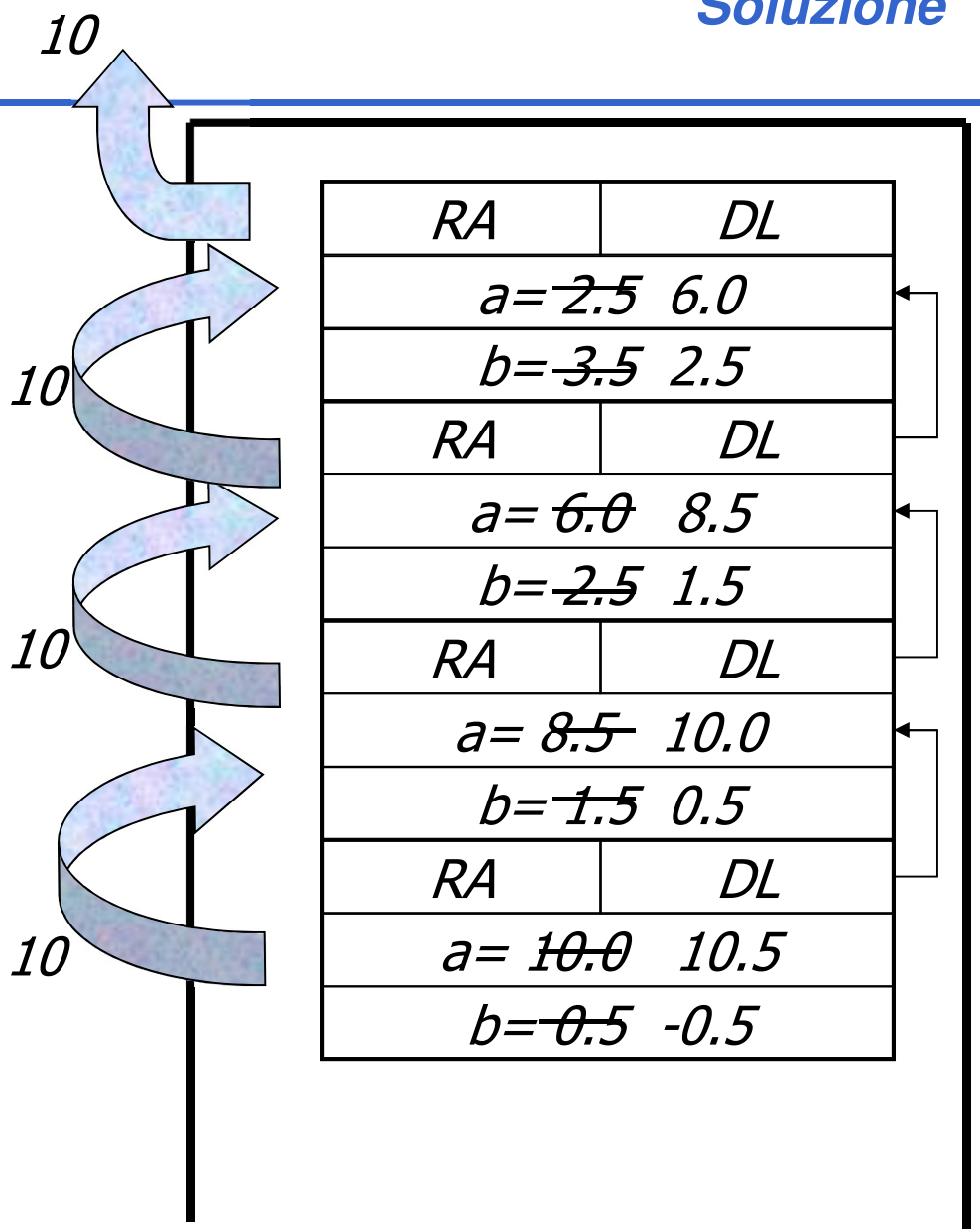
Si scriva il risultato della funzione quando invocata come:

```
funzione(2.5, 3.5)
```

e si disegnino i corrispondenti record di attivazione (si faccia particolare attenzione al fatto che la funzione restituisce un valore intero).

Esercizio Record di Attivazione

Soluzione



Esercizio Record di Attivazione

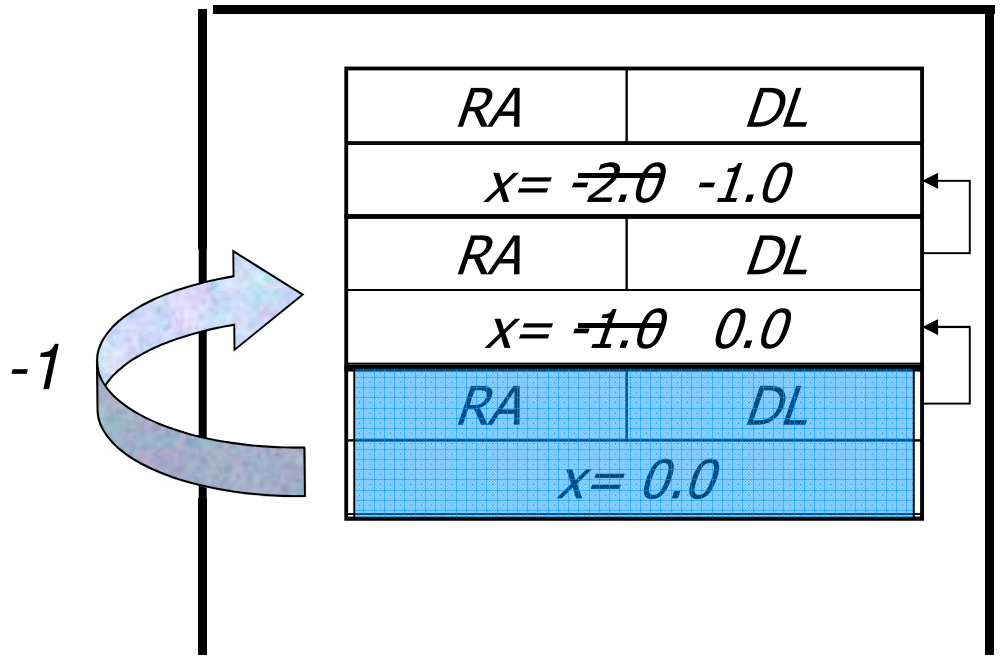
*Si consideri la seguente funzione **W**:*

```
double W(int x){
   if (x<0) {
          x++;
          return W(x)+W(x/2);
          x++;
   }
   else
          return -1;
}
```

*Si scriva il risultato della funzione quando invocata come **W(-2)** e si disegnino i corrispondenti record di attivazione.*

Esercizio Record di Attivazione
Soluzione

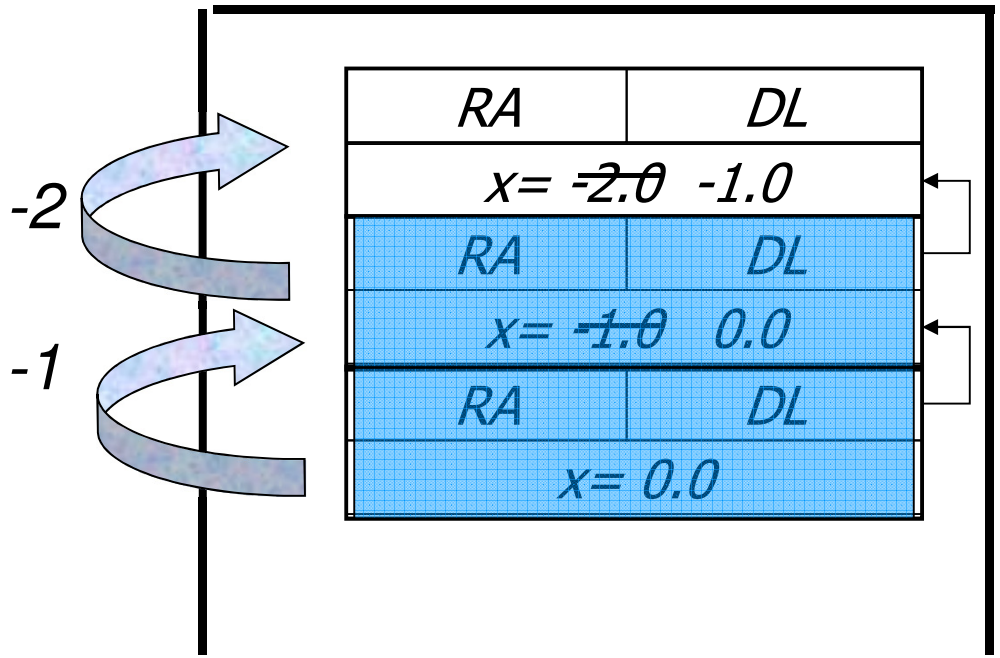
La funzione restituisce il valore –3.00. Supponendo che la valutazione degli addendi nella somma venga fatta a partire da sinistra, si ottiene prima:



poi l'eliminazione dell'ultimo record, e …

Esercizio Record di Attivazione
Soluzione

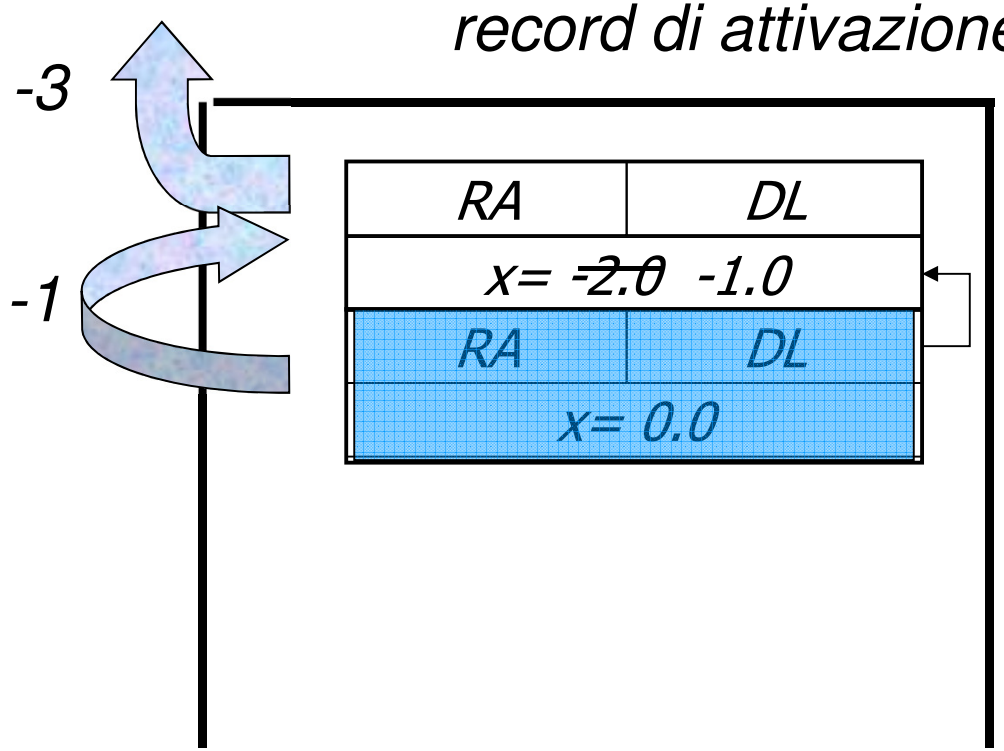
... dove viene fatta l'ultima invocazione di W(0), con somma finale restituita alla prima invocazione di W.



Quindi viene fatta la prima somma con risultato -2.00, restituita indietro

Esercizio Record di Attivazione
Soluzione

…poi l'eliminazione dell'ultimo record, e un nuovo ulteriore record di attivazione per W(0):



La somma finale è restituita alla prima invocazione di W.

Esercizio sintesi

Uno dei più antichi sistemi di codificazione di messaggi segreti si basa sulla sostituzione, secondo un certo ordine, dei caratteri componenti il messaggio.

Ad esempio, dato un messaggio composto dalle lettere:

{a, b, c}

E data una chiave di sostituzione che, per ogni lettera ne associa un'altra:

'a' → 'x'

'b' → 'y'

'c' → 'z'

Il messaggio originale può essere così riscritto:

{x, y, z}

Esercizio sintesi

Si vuole costruire un sistema di codifica/decodifica di questo tipo, facendo le seguenti assunzioni:

1. *Le lettere componenti il messaggio sono tutte minuscole, ed i messaggi non possono contenere altri caratteri che lettere (no spazi, no numeri)*
2. *Il codice di sostituzione è dato da un array di 26 caratteri, che viene interpretato nel seguente modo: nella posizione ad indice 0 vi è il carattere che deve sostituire la lettera 'a', in posizione con indice 1 vi è il carattere che deve sostituire la lettera 'b', etc.*

Esercizio sintesi

Si strutturi la soluzione implementando due funzioni:

```
void crypt( char source[],
            int length,
            char code[DIM_ALPHA],
            char dest[]);
```

```
void decrypt( char source[],
              int length,
              char code[DIM_ALPHA],
              char dest[]);
```

Ed infine si scriva un semplice main di prova.

Esercizio sintesi
Soluzione

```
void crypt(char source[], int length, char
    code[DIM_ALPHA], char dest[]) {
       int i;

       for (i=0; i<length; i++) {
             dest[i] = code[source[i] - 'a'];
       }
}
```

Esercizio sintesi
Soluzione

```
void decrypt(char source[], int length, char
    code[DIM_ALPHA], char dest[])
{
    int i;
    int j;
    int pos= -1;

    for (i=0; i<length; i++) {
       for (j=0; j<DIM_ALPHA && pos<0; j++) {
             if(source[i] == code[j])
                   pos = j;
       }
       dest[i] = 'a' + pos;
       pos = -1;
    }
}
```

Esercizio sintesi
Soluzione

```
#define DIM 256
#define DIM_ALPHA 26

int main()
{
    char source[DIM] = "abc";
    char dest1[DIM] = {'\0',...};
    char dest2[DIM] = {'\0',...};
    char codice[DIM_ALPHA] = "cab";

    printf("ORIGINALE: %s\n", source);
    crypt(source, 3, codice, dest1);
    printf("CRIPTATO: %s\n", dest1);
    decrypt(dest1, 3, codice, dest2);
    printf("DE-CRIPTATO: %s\n", dest2);

    return 0; }
```