ESERCIZIO 1

Si realizzi un programma C che:

1. allochi staticamente un vettore V1 di 10 float e allochi dinamicamente un vettore V2 di 10 double;
2. chieda all'utente di inserire gli elementi dei due vettori V1 e V2, controllando che tutti gli elementi siano numeri positivi (eventualmente chiedendo il re-inserimento degli elementi negativi); il programma inoltre deve controllare che non vengano re-inseriti elementi già presenti nell'array in uso all'istante;
3. invochi una funzione **check()**, con interfaccia da definire opportunamente, che vada a verificare se V1 e V2 contengono esattamente gli stessi elementi, anche in ordine differente.

ESERCIZIO 1 – Soluzione

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

#define DIM 10

int giaInseritoFloat(float v[], int dim, float value) {
          int trovato = 0;
          int i;
          for (i=0; i<dim && !trovato; i++)
                    if (v[i] == value)
                              trovato = 1;
          return trovato;
}

int giaInseritoDouble(double v[], int dim, double value) {
          int trovato = 0;
          int i;
          for (i=0; i<dim && !trovato; i++)
                    if (v[i] == value)
                              trovato = 1;
          return trovato;
}
```

ESERCIZIO 1 – Soluzione

```
int check(float v1[], int dim1, double v2[], int dim2) {
            int trovatiTutti =  1;
            int i;
            for (i=0; i<dim1 && trovatiTutti; i++) {
                        if (!giaInseritoDouble(v2, dim2, v1[i])) //ATTENZIONE… USO PROMOTION… E SE USASSI
                                    trovatiTutti=0;           // TRONCAMENTO???
            }
            return trovatiTutti;
}

int main() {
            float V1[DIM];
            double * V2;
            int i;
            V2 = (double *) malloc(DIM * sizeof(double));
            // ciclo inserimento V1
            i = 0;
            while (i<DIM) {
            printf("Inserisci un numero per V1:");
            scanf("%f", &(V1[i]));
            if (V1[i]>=0 && ! giaInseritoFloat(V1, i, V1[i]))
                        i++;
            else {
                        if (V1[i]<0)
                                    printf("Numero negativo!\n");
                        else
                                    printf("Numero gia' inserito!\n");
                        }
            }
```

ESERCIZIO 1 – Soluzione

```
        // ciclo inserimento V2
        i = 0;
        while (i<DIM) {
                printf("Inserisci un numero per V2:");
                scanf("%lf", &(V2[i]));
                if (V2[i]>=0 && ! giaInseritoDouble(V2, i, V2[i]))
                        i++;
                else {
                        if (V2[i]<0)
                                printf("Numero negativo!\n");
                        else
                                printf("Numero gia' inserito!\n");
                }
        }

        if (check(V1, DIM, V2, DIM))
                printf("Gli elementi ci sono tutti!\n");
        else
                printf("Manca almeno un elemento!\n");
        free(V2);
        system("pause");
        return 0;
}
```

ESERCIZIO 2

Si realizzi un programma C che:

1. chieda l'inserimento da parte dell'utente di una sequenza arbitraria di caratteri alfabetici (lunghezza max 1000 car, senza spazi); l'utente segnala l'eventuale terminazione della fase di inserimento inserendo il carattere '0' (eventualmente anche prima di inserire 1000 caratteri…)
2. si realizzi una funzione **isMinuscolo(…)**, con interfaccia da definire opportunamente, che restituisca un valore interpretabile come true se un carattere passato come parametro è effettivamente un carattere alfabetico minuscolo (tale funzione deve essere definita in un file sorgente separato di nome conta.c);

ESERCIZIO 2

3. invochi una funzione **conta()**, con interfaccia da definire opportunamente, che vada a contare quanti caratteri alfabetici minuscoli sono contenuti nella sequenza (tale funzione deve essere definita in un file sorgente separato di nome conta.c);
4. se il conteggio precedente supera la metà del numero totale di caratteri della sequenza, allora tutte le occorrenze di caratteri minuscoli devono essere sostituite con i corrispettivi maiuscoli, ad esempio ‘a’ con ‘A’.

ESERCIZIO 2 – Soluzione

«conta.h»:

```
int isAlfabetic(char c);
int isMinuscolo(char c);
int conta(char v[], int dim);
```

«conta.c»:

```
int isAlfabetic(char c) {
        return ((c>='a' && c <='z') || (c>='A' && c <='Z'));
}

int isMinuscolo(char c) {
        return (c>='a' && c <='z');
}

int conta(char v[], int dim) {
        int result = 0;
        int i;
        for (i=0; i<dim; i++)
                if (isMinuscolo(v[i]))
                        result++;
        return result;
}
```

ESERCIZIO 2 – Soluzione

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

#include "conta.h"
#define DIM 1000

int main() {
          char v[DIM];
          int i, j;
          char read;
          int minuscoli;
          // lettura
          read = 'a';
          i=0;
          while (i<DIM && read!='0') {
                    read = getchar();
                    if (isAlfabetic(read)) { // butto via i caratteri speciali...
                              v[i] = read;
                              i++;
                    }
          }
          …
```

ESERCIZIO 2 – Soluzione

```
        …

        minuscoli = conta(v, i);

        if (minuscoli > i/2)
                for(j=0; j<i; j++)
                        if(isMinuscolo(v[j]))
                                v[j] = v[j] -'a' + 'A';

        printf("Sequenza modificata: ");
        for (j=0; j<i; j++)
                printf("%c", v[j]);
        printf("\n");

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

ESERCIZIO 3

Si realizzi un programma C che:

1. allochi staticamente un vettore s1 di 20 char e allochi dinamicamente un vettore s2 di 20 char;
2. chieda all'utente di inserire gli elementi dei due vettori s1 e s2, controllando che tutti gli elementi siano caratteri alfabetici, maiuscoli o minuscoli (eventualmente chiedendo il re-inserimento dei caratteri che non superano il controllo);
3. invochi una funzione **anagramma()**, con interfaccia da definire opportunamente e inclusa in un file sorgente separato, che vada a verificare se s1 e s2 contengono esattamente gli stessi caratteri (con lo stesso numero di occorrenze), anche in ordine differente.

ESERCIZIO 3 – Soluzione

«anagramma.h»:

```
#include <stdlib.h>

int isAlfabetic(char c);
int leggi(char v[], int dim);
int anagramma(char s1[], char s2[], int dim);
```

«anagramma.c»:

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

int isAlfabetic(char c) {
          return ((c>='a' && c <='z') || (c>='A' && c <='Z'));
}

int leggi(char v[], int dim) {
          int i = 0;
          while (i<dim) {
                    v[i] = getchar();
                    if (isAlfabetic(v[i]))
                              i++;
          }
          return i;
}
```

ESERCIZIO 3 – Soluzione

```
int anagramma(char s1[], char s2[], int dim) {
        int * used;
        int i, j;
        int trovato;

        used = (int *) malloc(sizeof(int) * dim);
        for (i=0; i<dim; i++)
                used[i] = 0;

        for (i=0; i<dim; i++) {
                trovato = 0;
                for (j=0; j<dim && !trovato; j++)
                        if (s1[i] == s2[j] && !used[j]) {
                                trovato = 1;
                                used[j] = 1;
                        }
                if (!trovato) {
                        free(used);
                        return 0;
                }
        }
        free(used);
        return 1;
}
```

ESERCIZIO 3 – Soluzione

«main.c»:

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

#define DIM 5

int main() {
          char s1[DIM];
          char s2[DIM];

          leggi(s1, DIM);
          leggi(s2, DIM);

          if (anagramma(s1, s2, DIM))
                    printf("s1 e' un anagramma di s2\n");
          else
                    printf("Niente anagramma!\n");

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

ESERCIZIO 4

Si realizzi un programma C per il calcolo della trasposta di una matrice allocata dinamicamente. Il programma deve:

1. chiedere all'utente di inserire il numero di righe e colonne della matrice;
2. allocare di conseguenza una matrice di double;
3. chiedere all'utente di inserire gli elementi della matrice per colonna (prima tutta la prima colonna, poi la seconda colonna, …);
4. calcolare la trasposta della matrice inserita.

ESERCIZIO 4 – Soluzione

```
#include <stdio.h>

int main() {
            int row, col;
            int i, j;
            double * m1;
            double * m2;

            printf("Inserire numero righe: ");
            scanf("%d", &row);
            printf("Inserire numero colonne: ");
            scanf("%d", &col);
            m1 = (double *) malloc(sizeof(double) * col * row);
            m2 = (double *) malloc(sizeof(double) * col * row);

            printf("Inserimento per colonna:\n");
            for (j=0; j<col; j++) {
                        printf("Colonna %d-esima:", j);
                        for (i=0; i<row; i++)
                                    scanf("%lf", m1+col*i+j );
            }
            for (i=0; i<row; i++) {          // stampo cosa ho letto...
                        for (j=0; j<col; j++)
                                    printf("%6.2lf", *(m1+col*i+j));
                        printf("\n");
            }
```

ESERCIZIO 4 – Soluzione

```
        …

        // calcolo la trasposta
        for (i=0; i<row; i++) {
                for (j=0; j<col; j++)
                        *(m2+j*row+i) = *(m1+col*i+j);
        }
        // stampo cosa ho calcolato...
        for (i=0; i<col; i++) {
                for (j=0; j<row; j++)
                        printf("%6.2lf", *(m2+row*i+j));
                printf("\n");
        }
        free(m1);
        free(m2);
        system("pause");
        return 0;

}
```

ESERCIZIO 5

Si realizzi un programma C per il calcolo del palindromo di una stringa inserita dall'utente e allocata dinamicamente. Il programma deve:

1. chiedere all'utente di inserire il numero di caratteri MAX da allocare per la stringa s1;
2. chiedere all'utente di inserire i caratteri, uno a uno, da memorizzare nella stringa s1, controllando che si tratti di soli caratteri alfabetici (sia minuscoli che maiuscoli); si abbia cura di gestire la stringa s1 in modo che essa risulti essere una stringa «ben formata»;
3. determinare una nuova stringa s2 ottenuta come palindromo di s1, ovvero invertendo la posizione di tutti i caratteri (primo carattere di s1 diventa ultimo carattere di s2, secondo carattere di s1 diventa penultimo carattere di s2, …). Ad esempio, se s1 = "PIPPOpluto", s2 dovrà risultare "otulpOPPIP".

ESERCIZIO 5 – Soluzione

```
#include <stdio.h>

int isAlfabetic(char c) { return ((c>='a' && c <='z') || (c>='A' && c <='Z')); }

int main() {
          int dim;
          char * s1;
          char * s2;
          int i;
          char read;

          printf("Numero massimo di caratteri: ");
          scanf("%d", &dim);
          s1 = (char*) malloc(sizeof(char)* (dim+1) );
          s2 = (char*) malloc(sizeof(char)* (dim+1) );
          i=0;
          while (i<dim) {
                    read=getchar();
                    if (isAlfabetic(read)) {
                              s1[i] = read;
                              i++;
                    }
          }
          s1[i] = '\0';
```

ESERCIZIO 5 – Soluzione

```
        …

        for (i=0; i<dim; i++)
                  s2[dim-i-1] = s1[i];
        s2[dim] = '\0';

        printf("Letto: %s\n", s1);
        printf("Palindromo: %s\n", s2);

        free(s1);
        free(s2);
        system("pause");
        return 0;
}
```

ESERCIZIO 6

Si realizzi un programma C che:

1. allochi staticamente un vettore stringa1 e dinamicamente un vettore stringa2, entrambi di 15 caratteri utili (escluso terminatore);
2. chieda all'utente di inserire gli elementi dei due vettori stringa1 e stringa2, controllando che tutti gli elementi siano maiuscoli, eventualmente chiedendo il re-inserimento degli elementi che non superano il controllo;
3. realizzi una funzione **ordina()**, con interfaccia da definire opportunamente e inclusa in un file sorgente separato, capace di ordinare alfabeticamente gli elementi di un vettore di caratteri, restituendo il nuovo vettore ordinato;
4. invochi **ordina()** per ordinare i due vettori stringa1 e stringa2;
5. concateni le due stringhe ordinate in un'unica stringa complessiva (tutti i caratteri di stringa1 seguiti da tutti i caratteri di stringa2), con al più 30 caratteri utili.

ESERCIZIO 6 – Soluzione

«ordina.h»:

```
int isMaiusc(char c);
int leggi(char v[], int dim);
void ordina(char s[], int dim);
```

«ordina.c»:

```
int isMaiusc(char c) { return (c>='A' && c<='Z'); }
int leggi(char v[], int dim) {
            int i = 0;
            while (i<dim) {
                        v[i] = getchar();
                        if (isMaiusc(v[i]))
                                    i++;
            }
            return i; }

void ordina(char s[]) {// VERSIONE MOLTO BRUTTA ISPIRATA A BUBBLE SORT
            int i, j;
            char temp;
            for (i=0; s[i+1]!='\0'; i++)
                        for (j=0; s[j+1]!='\0'; j++)
                                    if (s[j] > s[j+1]) {
                                                temp = s[j];
                                                s[j] = s[j+1];
                                                s[j+1] = temp;
                                    }
}
```

ESERCIZIO 6 – Soluzione

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

#define DIM 16
int main() {
            char s1[DIM];
            char * s2;
            char result[31];
            int i;

            s2 = (char*) malloc(sizeof(char) * DIM);
            leggi(s1, DIM-1);
            s1[DIM-1] = '\0';
            leggi(s2, DIM-1);
            s2[DIM-1] = '\0';

            ordina(s1);
            ordina(s2);

            for (i=0;i<DIM-1; i++) result[i] = s1[i];
            for (i=0;i<DIM-1; i++) result[i+DIM-1] = s2[i];
            result[30] = '\0';
            printf("Concatenazione: %s\n", result);
            free(s2);
            system("pause"); return 0;
}
```

ESERCIZIO 7

Si realizzi un programma C che si occupi di effettuare la somma fra due matrici quadrate di interi, di uguali dimensioni e allocate dinamicamente. Il programma deve:

Il numero di righe/colonne è specificato dall'utente; oltre a chiedere tale informazione, il programma provvede a chiedere all'utente di inserire anche gli elementi delle matrici.

Infine, il programma deve determinare la nuova matrice somma, stampandone il contenuto a video.

ESERCIZIO 7 – Soluzione

```
#include <stdio.h>
#include <stdlib.h>
int main() {
            int size, i, j;
            int * m1, * m2, * m3;
            printf("Inserire dimensione dell amtrice quadrata: ");
            scanf("%d", &size);
            m1 = (int*) malloc(sizeof(int) * size *size);
            m2 = (int*) malloc(sizeof(int) * size *size);
            m3 = (int*) malloc(sizeof(int) * size *size);
            printf("Matrice 1 (per righe):\n");
            for (i=0; i<size; i++)
                        for(j=0; j<size; j++)
                                    scanf("%d", m1+i*size+j);
            printf("Matrice 2 (per righe):\n");
            for (i=0; i<size; i++)
                        for(j=0; j<size; j++)
                                    scanf("%d", m2+i*size+j);
            for (i=0; i<size; i++)
                        for(j=0; j<size; j++)
                                    *(m3+i*size + j) = *(m1+i*size + j) + *(m2+i*size + j);
            for (i=0; i<size; i++) {
                        for(j=0; j<size; j++)
                                    printf("%4d", *(m3+i*size + j));
                        printf("\n");
            }
            free(m1); free(m2); free(m3);
            system("pause"); return 0; }
```