

Introduzione al C

Corso di Fondamenti di Informatica
Ingegneria Clinica

Esercitazione 10

DIPARTIMENTO DI INFORMATICA
E SISTEMISTICA ANTONIO RUBERTI



SAPIENZA
UNIVERSITÀ DI ROMA

Raffaele Nicolussi

Clienti

- Progettare la struttura CLIENTI con i seguenti campi
 - Nome (stringa)
 - Cognome (stringa)
 - Via (stringa)
 - Civico (intero)
- Realizzare un array di DIM elementi fatto di elementi di tipo CLIENTI

Clienti

- Scrivere la funzione CaricaDatiClienti che
 - Riceve in ingresso l'array di strutture CLIENTE
 - Chiede all'utente quanti clienti si vuole caricare (num_clienti)
 - Carica nell'array di strutture num_clienti elementi facendoli inserire all'utente da tastiera
 - Restituisca il valore della variabile num_clienti
- Scrivere la funzione CercaCliente che
 - Riceve in ingresso un COGNOME da cercare e l'array di strutture CLIENTE
 - Stampi un messaggio che il cognome è presente nell'array

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

```
#define MAX_ARTICOLI 100
#define MAX_CLIENTI 100
```

```
typedef struct
{
    char nome[100];
    char cognome[100];
    char via[100];
    int civico;

} Cliente;
```

```
int CaricaDatiClienti(Cliente ElencoClienti[])  
{
```

```
    int num_clienti, i;
```

```
    printf("Quanti clienti vuoi inserire ");  
    scanf("%d",&num_clienti);
```

```
    printf("\n");
```

```
    for (i=0; i<num_clienti; i++)  
    {
```

```
        printf("\nUtente %d\n",i);
```

```
        printf("NOME ");  
        scanf("%s", ElencoClienti[i].nome);
```

```
        printf("\n");
```

```
printf("COGNOME ");  
    scanf("%s", ElencoClienti[i].cognome);  
    printf("\n");
```

```
printf("VIA ");  
scanf("%s", ElencoClienti[i].via);  
printf("\n");
```

```
printf("CIVICO ");  
scanf("%d", &ElencoClienti[i].civico);  
printf("\n");
```

```
    }  
    return num_clienti;  
}
```

```

void CercaCliente(Cliente ElencoClienti[], char cognome[], int size)
{
    int i;

    printf("\nRicerca del cliente %s\n",
           cognome);

    printf("=====\n");

    for(i = 0; i < size; i++)
    {
        if( (strcmp(ElencoClienti[i].cognome, cognome) == 0) )
        {
            printf("Trovato\n%s %s %s %d\n",
                   ElencoClienti[i].nome,
                   ElencoClienti[i].cognome,
                   ElencoClienti[i].via,
                   ElencoClienti[i].civico);

            break;
        }
    }

    printf("\n");
}

```

```
int main(){  
  
    Cliente ElencoClienti[MAX_CLIENTI];  
    char cog[30];  
  
    int totale_clienti = CaricaDatiClienti(ElencoClienti);  
  
    if(totale_clienti == 0)  
    {  
        printf("nessun cliente caricato\n");  
        exit(-1);  
    }  
}
```



```
else
{
    printf("\n caricati %d clienti\n", totale_clienti);
    int j;
    for(j = 0; j < totale_clienti; j++)
    {
        printf("%s %s %s %d\n",
            ElencoClienti[j].nome,
            ElencoClienti[j].cognome,
            ElencoClienti[j].via,
            ElencoClienti[j].civico);

    }
}

printf("Cognome cliente da cercare ");
scanf("%s", cog);

CercaCliente(ElencoClienti, cog, totale_clienti);

system("PAUSE");
}
```

Esercizio

- Scrivere un programma che, data una stringa di N caratteri, **la inverta** (es: "Informatica" diventa "acitamrofni")

Soluzione

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

```
#define N 31
```

```
void scambia(char *a, char *b) {  
    char aux;  
    aux = *a;  
    *a = *b;  
    *b = aux; }
```

```
/* Funzione che calcola la lunghezza della stringa */
```

```
int lung(char s[]) {  
    int i;  
    /* incremento il contatore fintanto che non incontro il carattere di terminazione */  
    for (i=0; s[i] != '\0'; i++);  
    return i;  
}
```

Soluzione

```
void inverti2(char v[])
{
    int i,l;
    l = lung(v);
    for (i=0; i<l/2; i++) scambia(&v[i],&v[l-i-1]);
}
```

```
main() {
    char s[N];
    scanf("%s",s);
    inverti2(s);
    printf("\nStringa invertita: %s",s);
}
```

Esercizio

- Scrivere la funzione "**massimoMatrice**", che calcola il massimo della matrice e la sua posizione.
- La funzione riceve come parametri:
 - un array bidimensionale di interi
 - due interi che rappresentano le due dimensioni dell'array
 - gli indirizzi di due variabili intere per memorizzare gli indici del massimo
- La funzione restituisce il valore del massimo della matrice.

Soluzione

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

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

```
#define R 3
```

```
#define C 2
```

```
void stampaMatrice(int[][C], int, int);
```

```
void leggiMatrice(int[][C], int, int);
```

```
int massimoMatrice (int[][C], int *, int *, int , int );
```

Soluzione

```
int main()
```

```
{  
    /* variabile a cui assegnare il valore restituito dalla funzione massimoMatrice */  
    int massimo;  
    /* variabili che devono essere modificate dalla funzione massimoMatrice  
    per restituire gli indici del valore massimo */  
    int    riga = -1, colonna = -1;  
    int mat[R][C];  
  
    leggiMatrice(mat, R, C);  
    stampaMatrice(mat, R, C);  
  
    massimo = massimoMatrice (mat, &riga, &colonna, R, C);  
    printf("\nMassimo della matrice: %d\n", massimo);  
    printf ("Posizione: riga %d, colonna %d\n", riga, colonna);  
  
    system("pause");  
  
    return 0;  
}
```

Soluzione

```
void leggiMatrice(int m[][C], int dim1, int dim2)
{
    int i, j;

    printf("Inserire gli elementi di una matrice di interi %dx%d\n",
           dim1, dim2);
    for (i = 0; i < dim1; i++)
        for (j = 0; j < dim2; j++) {
            printf("Elemento %d %d: ", i, j);
            scanf("%d", &m[i][j]);
        }
}
```


Soluzione

```
void stampaMatrice(int m[][C], int dim1, int dim2)
{
    int i, j;

    printf("\nStampa della matrice:\n");
    for (i = 0; i < dim1; i++) {
        for (j = 0; j < dim2; j++)
            printf("%5d", m[i][j]);
        printf("\n");
    }
}
```

Soluzione

```
int massimoMatrice (int matrice[][C], int *r, int *c, int dim1, int dim2)
{
    int i, j, max;

    max = matrice[0][0];
    *r = 0;
    *c = 0;

    for (i = 0; i < dim1; i++)
        for (j = 0; j < dim2; j++)
            if ( matrice[i][j] >= max) {
                max = matrice[i][j];
                *r = i;
                *c = j;
            }

    return max;
}
```

Esercizio

- Completare il programma scrivendo due funzioni "**simmetrica**" e "**diagonale**", che verificano rispettivamente se una data matrice quadrata e' simmetrica e diagonale.
- Entrambe le funzioni ricevono come parametri:
 - un array bidimensionale di interi
 - la dimensione dell'array
- La funzione "simmetrica" restituisce il valore 1 se l'array bidimensionale corrisponde a una matrice simmetrica, 0 altrimenti.
 - Una matrice quadrata si dice simmetrica se gli elementi della matrice sono simmetrici rispetto alla diagonale principale.
- La funzione "diagonale" restituisce il valore 1 se l'array bidimensionale corrisponde a matrice diagonale, 0 altrimenti.
 - Una matrice quadrata si dice diagonale se tutti gli elementi all'esterno della diagonale principale sono nulli.
- Una matrice diagonale e' anche simmetrica.

Soluzione

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

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

```
#define N 3
```

```
void stampaMatrice(int [][][N], int);
```

```
void leggiMatrice(int [][][N], int);
```

```
int simmetrica (int [][][N], int );
```

```
int diagonale (int [][][N], int );
```

Soluzione

```
int main()
{
    int mat[N][N];

    /* variabile a cui assegnare il valore restituito dalla funzione simmetrica */
    int simm_ok = -1;
    /* variabile a cui assegnare il valore restituito dalla funzione diagonale */
    int diag_ok = -1;

    leggiMatrice(mat, N);
    stampaMatrice(mat, N);

    simm_ok = simmetrica (mat, N);
```

Soluzione

```
if ( simm_ok == 1 )
    printf("\nLa matrice e' simmetrica\n");
if (simm_ok == 0 )
    printf("\nLa matrice non e' simmetrica\n");

diag_ok = diagonale (mat, N);

if ( diag_ok == 1 )
    printf("\nLa matrice e' diagonale\n");
if (diag_ok == 0 )
    printf("\nLa matrice non e' diagonale\n");

system("pause");

return 0;
}
```

Soluzione

```
void leggiMatrice(int m[][N], int dim)
{
    int i, j;

    printf("Inserire gli elementi di una matrice quadrata di interi %dx%d\n", dim, dim);
    for (i = 0; i < dim; i++)
        for (j = 0; j < dim; j++) {
            printf("Elemento %d %d: ", i, j);
            scanf("%d", &m[i][j]);
        }
}
```

Soluzione

```
void stampaMatrice(int m[][N], int dim)
{
    int i, j;

    printf("\nStampa della matrice:\n");
    for (i = 0; i < dim; i++) {
        for (j = 0; j < dim; j++)
            printf("%5d", m[i][j]);
        printf("\n");
    }
}
```


Soluzione

```
int simmetrica (int matrice[][N], int dim)
{
    int i, j;

    for (i = 0; i < dim; i++)
        for (j = i+1; j < dim; j++)
            if ( matrice[i][j] != matrice[j][i])
                return 0;
    return 1;
}
```

Soluzione

```
int diagonale (int matrice[][N], int dim)
{
    int i, j;

    for (i = 0; i < dim; i++)
        for (j = 0; j < dim; j++)
            if ( i != j && matrice[i][j] != 0)
                return 0;
    return 1;
}
```

Esercizio

- Scrivere la funzione "**filtro**", che dato un array di elementi binari, azzeri tutti gli elementi pari a 1 che hanno entrambi gli elementi adiacenti pari a 0. Gli elementi estremi dell'array non devono essere azzerati.
- Esempio:
 - se l'array contiene i valori [1,1,0,1,0,1,1,1,0,1,0,1]
 - viene trasformato in [1,1,0,0,0,1,1,1,0,0,0,1]
- Oltre alla funzione filtro si devono scrivere le definizioni delle due funzioni **leggiArray** e **stampaArray** i cui prototipi e attivazioni sono già contenute nel programma

Soluzione

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

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

```
#define N 12
```

```
void leggiArray (unsigned short int [], int);
```

```
void stampaArray (unsigned short int [], int);
```

```
void filtro (unsigned short int [], int);
```

Soluzione

```
int main()
{
    unsigned short int vett_bin [N];

    leggiArray(vett_bin, N);

    printf ("Stampa dell'array prima dell'attivazione della funzione filtro\n");
    stampaArray(vett_bin, N);

    filtro(vett_bin, N);

    printf ("Stampa dell'array dopo l'attivazione della funzione filtro\n");

    stampaArray(vett_bin, N);

    system("pause");

    return 0;
}
```

Soluzione

```
void leggiArray (unsigned short int vb[], int dim)
{
    int i=0, bin;

    do {
        printf ("Inserire una cifra binaria (elemento %d): ", i);
        scanf ("%us", &bin);
        if (bin == 0 || bin == 1) {
            vb[i] = bin;
            i++;
        }
        else
            printf("Valore non valido\n");
    }
    while (i < dim );
}
```

Soluzione

```
void stampaArray (unsigned short int vb[], int dim)
{
    int i;

    for ( i = 0; i < dim; i++)
        printf ("%u ", vb[i]);
    printf ("\n");
}
```

Soluzione

```
void filtro (unsigned short int vb[], int dim)
{
    int i;

    for ( i = 1; i < dim - 1; i++)
        if ( vb[i] == 1 && vb[i-1] == 0 && vb[i+1] == 0)
            vb[i] = 0;
}
```