

AA 2019-20

FONDAMENTI DI INFORMATICA

LABORATORIO

[prof.ssa RAFFAELA MIRANDOLA]



POLITECNICO
MILANO 1863

Esercizio 0: Numeri primi

Scrivere un programma che, dato un numero intero positivo inserito dall'utente, stampi a video "primo" se tale numero è primo.

Esempio:

```
Inserisci: 3    -> "primo"  
Inserisci: 4    -> "non è primo"  
Inserisci: 37   -> "è primo"
```

Suggerimenti:

- un numero è primo se è divisibile solo per sé stesso e per 1
- usare l'operatore % per calcolare il resto della divisione intera, fra il numero e i precedenti.

Esercizio 0: Numeri primi / Soluzione

```
#include <stdio.h>

int main(int argc, const char * argv[]) {
    int i,n=0;
    int divisibile = 0; // flag di fine ciclo se trovo un divisore.
    printf("gimme the number ");
    scanf("%d", &n);

    // inutile ciclare su 1 e n
    for (i=2; i<n && !divisibile; i++) {

        if(n%i==0)
            divisibile = 1;
    }

    if (divisibile)
        printf("divisibile per %i \n", i-1);
    else
        printf("primo\n");
    return 0;
}
```

Esercizio 1: Array

Si scriva un programma che legga da input due array A di 10 elementi e B di 5 elementi.

Il programma stampi: "CONTIENE"

se A contiene la sequenza contigua dei numeri di B.

ES:

A= [3,4,66,77,88,9,33,11, 66,100]

B=[77,88,9,33,11]

output: "CONTIENE"

ma non stampa nulla per B = [77,88,9,34,11]

Esercizio 1: Array

```
#include <stdio.h>

#define MAX 10
int main(int argc, const char * argv[]) {

    int A[MAX] = {3,4,66,77,88,9,33,11, 66,100};
    int B[MAX/2] = {77,88,9,33,11};
    // per semplicità mettiamo il calcolo vero e proprio senza lettura / scanf...
    int i, j=0;
    for (i=0; i<MAX && j<MAX/2; i++) {
        if (A[i] == B[j]){
            j++;
        }else{
            j=0;
        }
    }

    if (j == MAX/2)
        printf("CONTIENE\n");
    else
        printf("non CONTIENE\n");
    return 0;
}
```

Esercizio 2: Matrici

Scrivere un programma in linguaggio C che legga in input dei numeri **float** e riempia una matrice 4x4.
(usare #define)

il programma deve stampare tutti gli elementi delle celle della diagonale principale, ossia con indice riga == indice colonna.

es 1 2 3 4
 5 6 7 8
 9 10 11 12
 13 14 15 16

output:

1
6
11
16

Esercizio 2: Matrici / Soluzioni

```
#include <stdio.h>

#define NUM 4
typedef float numero;
typedef numero matrice[NUM][NUM];

int main(int argc, const char * argv[]) {
    int r,c;
    matrice matr;
    for (r=0; r<NUM; r++){
        for (c=0; c<NUM; c++){
            printf("Inserisci elemento [%d, %d]", r,c);
            scanf("%f", &matr[r][c]);
        }
    }

    printf("\n");
    for (r=0; r<NUM; r++){
        printf("%f ", matr[r][r]);
    }

    return 0;
}
```

Esercizio 3: Matrici

Scrivere un programma che usi due matrici NxN, dette A e AT.

L'utente inserisce i valori di A e il programma riempie AT in modo che sia la Trasposta di A (e la stampi).

https://it.wikipedia.org/wiki/Matrice_trasposta

$$A = \begin{pmatrix} 2 & 4 & 8 \\ 3 & 2 & 0 \\ 5 & 3 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad A^T = \begin{pmatrix} 2 & 3 & 5 & 0 \\ 4 & 2 & 3 & 1 \\ 8 & 0 & 1 & 0 \end{pmatrix}$$

Esercizio 3: Matrici / Soluzioni

```
#include <stdio.h>

#define R 4
#define C 3

int main(int argc, const char * argv[]) {
    int A[R][C] = {
        {2,4,8},
        {3,2,0},
        {5,3,1},
        {0,1,0}
    };
    int AT[C][R];

    // per semplicita mettiamo il calcolo vero e proprio senza lettura / scanf...
    int r, c;
    for (r=0; r<R; r++) {
        for (c=0; c<R; c++) {
            AT[c][r] = A[r][c];
        }
    }

    // stampa x conferma: (attenzione agli indici ed agli estremi del for.
    for (r=0; r<C; r++) {
        for (c=0; c<R; c++) {
            printf("%d ", AT[r][c]);
        }
        printf("\n");
    }
    return 0;
}
```

Esercizio 4: Matrici

Scrivere un programma in linguaggio C che legga in input due matrici 4x4, dette A e B

il programma deve stampare tutti gli indici riga/colonna che dove le celle di A hanno lo stesso valore delle celle di B nelle posizioni corrispondenti

Mat. A

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Mat. B

1	9	10	4
55	0	7	4
19	22	22	12
3	4	5	88

output:

(0,0)
(0,3)
(1,2)
(2,3)

Esercizio 4: Matrici

```
#include <stdio.h>

#define MAX 4

int main(int argc, const char * argv[]) {
    int A[MAX][MAX] = {
        {1,2,3,4},
        {5,6,7,8},
        {9,10,11,12},
        {13, 14, 15, 16}
    };
    int B[MAX][MAX] = {
        {1,9,10,4},
        {55, 0, 7,4},
        {19,22,22,12},
        {3,4,5,88}
    };

    // per semplicita mettiamo il calcolo vero e proprio senza lettura / scanf...
    int r, c;
    for (r=0; r<MAX; r++) {
        for (c=0; c<MAX; c++) {
            if (A[r][c] == B[r][c]){
                printf("%d in [%d %d]\n", A[r][c], r,c);
            }
        }
    }
    return 0;
}
```

Esercizio 5: Matrici

Scrivere un programma in linguaggio C che legga in input una matrice A quadrata N x N

il programma deve cercare tutti i massimi locali (ossia i valori che sono il massimo delle celle adiacenti) e sostituirli con zero.

(oss: ne caso di riga/colonna 0, si consideri la posizione N-1, sia dall'alto lato)

A

1	9	10	4
55	-5	7	4
19	23	22	12
3	4	5	88

output

1	0	0	4
0	-5	7	4
19	0	22	12
3	4	5	0

```

#include <stdio.h>

#define N 4

int main()
{
    int i, j;
    int mat[N][N];
    int massimiLocali[N][N];

    printf("Inserisci i valori della matrice %dx%d\n", N, N);           /*leggo i valori di mat*/
    for(i=0; i<N; i++) {
        for(j=0; j<N; j++) {
            scanf("%d", &mat[i][j]);
        }
    }

    for(i=0; i<N; i++)                                                /*inizializzo la matrice massimiLocali*/
    {
        for(j=0; j<N; j++)
        {
            massimiLocali[i][j] = 0;
        }
    }

    for(i=0; i<N; i++) {                                              /*trovo i massimi locali di mat e inserisco 1 alla posizione (i, j)
        for(j=0; j<N; j++) {                                          di massimiLocali se il corrispondente valore di mat è massimo locale*/
            if(mat[i][j]>=mat[(i+1)%N][j] && mat[i][j]>=mat[(i-1)%N][j] &&
               mat[i][j]>=mat[i][(j+1)%N] && mat[i][j]>=mat[i][(j-1)%N]) /*se tutti i valori adiacenti alla casella considerata non
            {                                                            sono maggiori del valore della casella, allora è un massimo
                massimiLocali[i][j]=1;                                  locale*/
            }
        }
    }

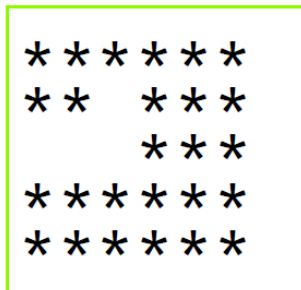
    for(i=0; i<N; i++) {
        for(j=0; j<N; j++) {
            /*se alla posizione (i, j) c'è un massimo locale*/      /*ora posso sostituire il valore dei massimi locali con 0; se lo avessi
            if(massimiLocali[i][j]==1) {                             fatto immediatamente, avrei potuto trovare più massimi, perché avrei
                /*allora lo sostituisco con 0*/                      modificato la matrice iniziale prima di terminare il controllo*/
                mat[i][j]=0;
            }
            /*e stampo a video il valore della casella*/
            printf("%3d", mat[i][j]);
        }
        printf("\n");
    }

    return 0;
}


```

Esercizio 6: Matrici

Scrivere un programma in linguaggio C che legga in input una matrice 5x5 di asterischi e spazi (labirinto)
il programma deve verificare se esiste un percorso che, data una coordinata r,c in input, dica se e' possibile uscire dalla matrice (trovando spazi contigui)



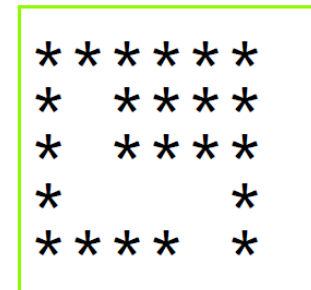
```
* * * * *  
* *   * * *  
      * * *  
* * * * *  
* * * * *
```



```
* * * * *  
* * *   * *  
* * *   * *  
*       * *  
* * * * *
```



```
* * * * *  
*       *  
* * * * *  
*       *  
* * * * *
```



```
* * * * *  
*   * * *  
*   * * *  
*       *  
* * * * *
```

```

#include <stdio.h>

#define DIM 5

#define N 0
#define W 1
#define S 2
#define E 3

int main()
{
    int i, j;
    char labirinto[DIM][DIM];
    char c;
    /*indica la direzione in cui mi sto muovendo, 0 nord, 1 ovest, 2 sud, 3 est*/
    int lastDir, dir, flag;
    int i0, j0, dir0=-1;
    int trovato = 0;

    /*leggo il labirinto e la posizione iniziale*/
    printf("Inserisci il labirinto (dimensione %dx%d), composto da ' ' e '*' \n", DIM, DIM);
    for(i=0; i<DIM; i++) {
        for(j=0; j<DIM; j++) {
            scanf("%c", &labirinto[i][j]);
        }
        scanf("%c", &c);
    }

    printf("Inserisci la posizione iniziale \n");
    scanf("%d %d", &i0, &j0);

    /*se la casella iniziale contiene un asterisco, non posso uscire*/
    if(labirinto[i0][j0]==' ') {
        /*se mi trovo sul bordo sono già fuori dal labirinto*/
        if(i0==0 || i0==DIM-1 || j0==0 || j0==DIM-1) {
            trovato=1;
        }
        /*cerco una direzione possibile, se non la trovo non posso uscire dal labirinto*/
        else if(labirinto[i0-1][j0]==' ') {
            dir0=S;
        }
        else if(labirinto[i0][j0-1]==' ') {
            dir0=E;
        }
        else if(labirinto[i0+1][j0]==' ') {
            dir0=N;
        }
        else if(labirinto[i0][j0+1]==' ') {
            dir0=W;
        }
    }
}

```

/*è noto che per uscire da un labirinto basti
svoltare sempre a destra; proviamo, quindi a scrivere
questa proprietà sotto forma di codice*/

```

        if(dir0>=0) {
            i = i0;
            j = j0;
            lastDir = dir0;
            do {
                for(dir=(lastDir+3)%4, flag=0; flag==0; dir=(dir+1)%4){
                    switch(dir) {
                        case N: if(labirinto[i-1][j]==' ') {
                            printf("N ");
                            flag=1;
                            i--;
                            lastDir=N;
                        }
                        break;
                        case W: if(labirinto[i][j-1]==' ') {
                            printf("W ");
                            flag=1;
                            j--;
                            lastDir=W;
                        }
                        break;
                        case S: if(labirinto[i+1][j]==' ') {
                            printf("S ");
                            flag=1;
                            i++;
                            lastDir=S;
                        }
                        break;
                        case E: if(labirinto[i][j+1]==' ') {
                            printf("E ");
                            flag=1;
                            j++;
                            lastDir=E;
                        }
                        break;
                    }
                }
                if(i==0 || i==DIM-1 || j==0 || j==DIM-1) {
                    trovato=1;
                }
            } while(trovato==0 && (i!=i0 || j!=j0 || lastDir!=dir0));
        }
    }

    if(trovato==1) {
        printf("\nSi puo' uscire dal labirinto\n");
    }
    else {
        printf("\nNON si puo' uscire dal labirinto\n");
    }

    return 0;
}

```