



Corso di Informatica Medica

Esercitazione II

2013-2014

Alessandro A. Nacci
nacci@elet.polimi.it - alessandronacci.com





Esercizio I

- Scrivere un programma che
 - legga due array di interi da tastiera
 - dica quale dei due array ha valor medio più alto
 - calcoli l'array concatenato tra i due array in ingresso
 - trovi tutti i numeri primi inseriti
 - trovi il massimo ed il minimo tra tutti i valori inseriti
 - dica in quale dei due array sono presenti i due valori di massimo e minimo
 - calcoli l'array riversato

*In questa soluzione,
non usare le funzioni!*



Esercizio I: dichiarazione variabili

LEZIONE PRECEDENTE

```
int i, j;
```

```
int a[DIM];
```

```
int b[DIM];
```

```
int ab[DIM2];
```

```
int ab_r[DIM2];
```

```
float medio_a, medio_b;
```

```
float somma = 0;
```

```
bool primo = 1; //booleana
```



Esercizio 1: lettura array

LEZIONE PRECEDENTE

```
// LETTURA VALORI
// -----

// Leggo l'array A
for (i = 0; i < DIM; i++){
    printf("\nInserisci i valore %d per l'array a: ", i);
    scanf("%d",&a[i]);
}

// Stampo l'array A
printf("\narray a = ");
for (i = 0; i < DIM; i++)
    printf("%d ", a[i]);

// Leggo l'array B
for (i = 0; i < DIM; i++){
    printf("\nInserisci i valore %d per l'array b: ", i);
    scanf("%d",&b[i]);
}

// Stampo l'array B
printf("\narray b = ");
for (i = 0; i < DIM; i++)
    printf("%d ", b[i]);
```




Esercizio I: calcolo del valor medio

LEZIONE PRECEDENTE

```
// VALOR MEDIO
// -----

// Calcolo valor medio si A

for (i = 0; i < DIM; i++){
    somma += a[i];
}
medio_a = somma / DIM;

// Calcolo valor medio si B
somma = 0;
for (i = 0; i < DIM; i++){
    somma += b[i];
}
medio_b = somma / DIM;

if(medio_a == medio_b)
    printf("L'array a e l'array b hanno lo stesso valor medio.\n");
else if (medio_a > medio_b)
    printf("L'array a ha valor medio piu' alto dell'array b.\n");
else if (medio_a < medio_b)
    printf("L'array b ha valor medio piu' alto dell'array a.\n");
```



Esercizio I: concatenazione

LEZIONE PRECEDENTE

```
// CONCATENAZIONE
// -----

for (i=0; i < DIM; i++)
    ab[i] = a[i];

for (i=0; i < DIM; i++)
    ab[i+DIM] = b[i];

// Stampo l'array AB
printf("\narray a.b = ");
for (i = 0; i < DIM2; i++) {
    printf("%d ", ab[i]);
}

printf("\n");
```



Esercizio I: numeri primi

```
// NUMERI PRIMI
// -----

printf("I numeri primi inseriti sono: ");

for (i = 0; i < DIM2; i++){
    primo = 1; // Inizializzo la variabile
    for (j = 2; j < ab[i]; j++){
        if ((ab[i] % j) == 0 ){
            primo = 0;
            break;
        }

        if (primo)
            printf("%d ", ab[i]);
    }

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




Esercizio I: massimo e minimo

```
// MAX e MIN
// -----

int max = 0;
int min = 2147483647;    //  $2^{(n-1)} - 1$  :::: CPL2 max value
                        // n = 32 poiché int è rappresentato
                        // con 32 bit.

for (i = 0; i < DIM2; i++){

    if (ab[i] > max)
        max = ab[i];

    if (ab[i] < min)
        min = ab[i];

}

printf("MIN = %d\n", min);
printf("MAX = %d\n", max);
```



Esercizio I: massimo e minimo

```
// TROVARE DOVE SONO MAX e MIN
// -----

bool max_in_array_a = 0;
bool min_in_array_a = 0;

for (i = 0; i < DIM; i++){
    if (a[i] == max)
        max_in_array_a = 1;

    if(a[i] == min)
        min_in_array_a = 1;
}

printf("\n");

if (max_in_array_a)
    printf("Il valore massimo %d e' nell'array a", max);
else
    printf("Il valore massimo %d e' nell'array b", max);

printf("\n");

if (min_in_array_a)
    printf("Il valore minimo %d e' nell'array a", min);
else
    printf("Il valore minimo %d e' nell'array b", min);

printf("\n");
```



Esercizio I: array riversato

```
// ARRAY RIVERSATO
// -----

for (i = 0; i < DIM2; i++)
    ab_r[DIM2 - i - 1] = ab[i];

printf("\narray (a.b)_r = ");
for (i = 0; i < DIM2; i++){
    printf("%d ", ab_r[i]);
}

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



WARNING

MATRICI E FUNZIONI

Il passaggio di una matrice ad una funzione
e' sempre per indirizzo e mai per copia.



Esercizio 2: Matrici e Funzioni

```
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
    int mat[R][C];
    foo1(mat);
    foo2(mat);
    foo3(mat);
}
```

```
void foo1(int *mat)
```

```
{
```

```
    int i, j;
```

```
    for(i=0; i<R; i++)
```

```
        for(j=0; j<C; j++)
```

```
            *(mat+i*C+j) = (i+j) * 2;
```

```
}
```

Puntatore ad intero!

Calcolo manuale dello
spiazzamento

Valore contenuto all'indirizzo tra parentesi

*(mat+i*C+j)

INDIRIZZO



Esercizio 2: Matrici e Funzioni

```
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
    int mat[R][C];
    foo1(mat);
    foo2(mat);
    foo3(mat);
}
```

*Esplicito che il puntatore
punta al primo elemento
di una matrice $R \times C$*

```
void foo2(int mat[R][C])
{
    int i, j;
    for(i=0; i<R; i++)
        for(j=0; j<C; j++)
            mat[i][j] = (i+j) * 3;
}
```

Accesso comodo alla matrice!



Esercizio 2: Matrici e Funzioni

Esplicito che il puntatore
punta al primo elemento
di una matrice con C colonne

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

```
#define R 3
```

```
#define C 3
```

```
void foo1(int*);
```

```
void foo2(int mat[R][C]);
```

```
void foo3(int mat[][C]);
```

```
int main()
```

```
{
```

```
    int mat[R][C];
```

```
    foo1(mat);
```

```
    foo2(mat);
```

```
    foo3(mat);
```

```
}
```

Il valore di mat cambia

Il valore di mat cambia

Il valore di mat cambia

```
void foo3(int mat[][C])
```

```
{
```

```
    int i, j;
```

```
    for(i=0; i<R; i++)
```

```
        for(j=0; j<C; j++)
```

```
            mat[i][j] = (i+j) * 4;
```

```
}
```

Accesso comodo alla matrice!

Blue	Green	Yellow
Red	Red	Purple
Green	Brown	Dark Brown
X	X	X

NOTA BENE

Ai fini del calcolo dello
spiazzamento il numero di
righe non è essenziale!

Blue	Green	Yellow	Red	Red	Purple	Green	Brown	Dark Brown	X	X	X
------	-------	--------	-----	-----	--------	-------	-------	------------	---	---	---



Esercizio 3 (a)

- Quale è l'output del seguente codice?

```
#include <stdio.h>

typedef enum{bianco,azzurro_chiario,giallo,rosso,verde_scuero,rosa,
    azzurro_scuero,verde_chiario,nero,marrone} colore;

int main(){

    int auto_codice[3];           //ATTENZIONE QUI!
    colore auto_colore[3];       // tipo_di_dato nome_varibile[]
    int i;

    auto_codice[0] = 1000;
    auto_colore[0] = giallo;

    auto_codice[1] = 4000;
    auto_colore[1] = rosa;

    auto_codice[2] = 8000;
    auto_colore[2] = nero;

    for (i = 0; i < 3; i++ )
        printf("L'auto con codice %d e' di colore %d\n",
            auto_codice[i], auto_colore[i]);

    return 0;
}
```

```
Terminal — bash — 80x24
> ./es3
> L'auto con codice 1000 e' di colore 2
> L'auto con codice 4000 e' di colore 5
> L'auto con codice 8000 e' di colore 8
```



Esercizio alla lavagna: puntatori e memoria

Handwritten notes on a chalkboard illustrating memory and pointers.

Memory Layout:

Index	Value	Label
0	3	→ a
1	0	→ b
2	3	→ m1
3	~	
4	13	
5	~	
6	1	
7	5	→ c
8	3	→ d
9	~	→ e

Code Snippets:

```
int main()
{
    int a = 3;
    printf("%d", &a);
    int * b;
    b = &a;

    int m1[2][2];
    int * c;
    m1[0][1] = 13;
    c = &(m1[1][0]);
    int d = *b;
    int e;
```



**Tutte il materiale sarà
disponibile sul mio sito
internet!**

alessandronacci.com

**Potete lasciare il vostro giudizio
qui:**

<http://tinyurl.com/IEIMExe2014>

See You Next Time!

