A.A. 08/09

Fondamenti di Programmazione

(canale E-O)

Docente: Prof.ssa Tiziana Calamoneri
calamo@di.uniroma1.it

Esercitatore: Dott. Roberto Petroccia
petroccia@di.uniroma1.it

Pagina del corso:
http://twiki.di.uniroma1.it/twiki/view/Programmazione1/EO/WebHome

---

Esercitazione del 03/12/08

ERRORI CON VETTORI

```
#include <stdio.h>
int raddoppia(int a) {
        return 2*a;
}
void raddoppiaArray (int A[], int n) {
    int i = 0;
    for (i = 0; i < n; i++) {
        A[i] = raddoppia(A[i]);
    }
}

int main() {
    leggi n;
    int A[n];
    riempi A;
    raddoppiaArray(A , n);
    stampa A;
    return 0;
}
```

ERRORI CON VETTORI

NON void raddoppiaArray(int A[n], int n)

```
#include <stdio.h>
int raddoppia(int a) {
        return 2*a;
}
void raddoppiaArray (int A[], int n) {
    int i = 0;
    for (i = 0; i < n; i++) {
        A[i] = raddoppia(A[i]);
    }
}

int main() {
    leggi n;
    int A[n];
    riempi A;
    raddoppiaArray(A , n);
    stampa A;
    return 0;
}
```

ERRORI CON VETTORI

NON void raddoppiaArray(int A[n], int n)

```
#include <stdio.h>
int raddoppia(int a) {
        return 2*a;
}
void raddoppiaArray (int A[], int n) {
    int i = 0;
    for (i = 0; i < n; i++) {
        A[i] = raddoppia(A[i]);
    }
}

int main() {
    leggi n;
    int A[n];
    riempi A;
    raddoppiaArray(A , n);
    stampa A;
    return 0;
}
```

void raddoppiaArray(int n, int A[n])

ERRORI CON VETTORI

NON void raddoppiaArray(int A[n], int n)

```
#include <stdio.h>
int raddoppia(int a) {
        return 2*a;
}
void raddoppiaArray (int A[], int n) {
    int i = 0;
    for (i = 0; i < n; i++) {
        A[i] = raddoppia(A[i]);
    }
}

int main() {
    leggi n;
    int A[n];
    riempi A;
    raddoppiaArray(A , n);
    stampa A;
    return 0;
}
```

void raddoppiaArray(int n, int A[n])

NON raddoppiaArray(A[], n)

ERRORI CON VETTORI

```
#include <stdio.h>
int somma(int A[i], int A[i+1]) {
        return A[i]+A[i+1];
}
void sommaArray (int A[], int n) {
      int i = 0;
      for (i = 0; i < n-1; i++) {
          A[i] = somma(A[i], A[i+1]);
      }
   }

int main() {
        leggi n;
        int A[n];
        riempi A;
        sommaArray(A , n);
        stampa A;
        return 0;
}
```

ERRORE: usare A[i] e A[i+1] come i due parametri della funzione è errato, la funzione deve prendere in input due interi (int a, int b) poi al momento della chiamata verranno passati i valori dell'array

ERRORI CON VETTORI

CORRETTA

```
#include <stdio.h>
int somma(int a, int b) {
        return a+b;
}
void sommaArray (int A[], int n) {
      int i = 0;
      for (i = 0; i < n-1; i++) {
         A[i] = somma(A[i], A[i+1]);
      }
   }

int main() {
        leggi n;
        int A[n];
        riempi A;
        sommaArray(A , n);
        stampa A;
        return 0;
}
```

ERRORI CON CALL BY REFERENCE

```
#include <stdio.h>
void swap(int a, int b) {
        int temp;
        temp = a;
        a = b;
        b = temp
}
```

ERRORE

ERRORI CON CALL BY REFERENCE

```
#include <stdio.h>
void swap(int a, int b) {
        int temp;
        temp = a;
        a = b;
        b = temp
}
```

ERRORE

```
#include <stdio.h>
void swap(int *a, int *b) {
        int* temp;
        temp = a;
        a = b;
        b = temp
}
```

ERRORE

ERRORI CON CALL BY REFERENCE

```
#include <stdio.h>
void swap(int a, int b) {
        int temp;
        temp = a;
        a = b;
        b = temp
}
```

ERRORE

```
#include <stdio.h>
void swap(int *a, int *b) {
        int* temp;
        temp = a;
        a = b;
        b = temp
}
```

ERRORE

```
#include <stdio.h>
void swap(int *a, int *b) {
        int temp;
        temp = *a;
        *a = *b;
        *b = temp
}
```

CORRETTA

Indice

03/12/08 - Roberto Petroccia: petroccia@di.uniroma1.it

Esercizi su vettori e matrici

1. Scrivere una funzione che presi in input due vettori A e B di stessa lunghezza, sottrae ai valori nella prima metà di B(partendo dalla testa) i valori della seconda metà di A(partendo dalla coda) e somma ai valori della seconda metà di B(partendo dalla testa) quelli della prima metà di A (partendo dalla coda). Se A={1,2,3,4,5} B={6,7,4,5,9}. A resta lo stesso e B diventa {1,3,7,7,10}. 5/2 = 2 quindi B[0] -= A[n-1] e B[1] -= A[n-2], poi B[2] += A[n-3], B[3] += A[n-4] e B[4] += A[n - 5].

2. Scrivere una funzione che presa in input una matrice di double controlli se è stocastica. Una matrice si dice stocastica se è quadrata ed ha tutti elementi non negativi tali che la somma degli elementi su ogni riga (o su ogni colonna) è uguale a 1. Se la matrice è stocastica la funzione restituisce 1 altrimenti restituisce 0. Le dimensioni della matrice vanno lette da input e la matrice può essere riempita come volete (rand() oppure a mano). Tutti i controlli su dimensioni e valori della matrice vanno nella funzione.

3. Scrivere una funzione che presa in input una matrice di double controlli se è bistocastica. Ossia le stesse condizioni dell'esercizio precedente ma tutte le righe e tutte le colonne hanno somma 1.

03/12/08 - Roberto Petroccia: petroccia@di.uniroma1.it

Esercizi ricorsione

1. Scrivere una funzione ricorsiva void hiphurra (int k) che stampa k hip seguiti da k hurra. Esempio hiphurra(3) stampa: hip hip hip hurra hurra hurra.

2. Scrivere una funzione che conti il numero delle occorrenze di un valore k in un vettore in maniera ricorsiva che abbia come tipo di ritorno void.

3. Scrivere una funzioni che preso un vettore di interi raddoppi il valore di ogni cella in maniera ricorsiva.

4. Scrivere una funzione iterativa e ricorsiva che presi in input due vettori A e B lunghi n stampi insieme il primo valore di A e l'ultimo valore di B, poi il secondo valore di A e il penultimo di B e così via.

5. Scrivere una funzione ricorsiva che calcoli l'esponenziale sfruttando le seguenti uguaglianze:

$$\begin{aligned} m^{2n} &= (m \times m)^n \\ m^{2n+1} &= m \times m^{2n} \\ m^0 &= 1 \end{aligned}$$

Input: Il metodo main leggerà due interi m ed n (il primo intero m è la base ed il secondo intero n è l'esponente) e chiamerà la funzione ricorsiva

03/12/08 - Roberto Petroccia: petroccia@di.uniroma1.it

1. Ex1 vettori e matrici

```
#include <stdio.h>
void calcolaArray (int A[], int B[], int n, int i) {
        if (i >= n) {
            return;
        }
        if (i < n/2) {
            B[i] -= A[n-1-i];
        }
        else {
            B[i] += A[n-1-i];
        }
        calcolaArray(A,B,n,i+1);
    }

int main() {
        leggi n;
        crea A;
        crea B;
        calcolaArray(A ,B, n, 0);
        stampa B;
        return 0;
}
```

1. Ex1 vettori e matrici

```
#include <stdio.h>
void calcolaArray (int A[], int B[], int n) {
        int i = 0;
        for (i = 0; i < n/2; i++) {
            B[i] -= A[n-1-i];
        }
        for (i = n/2; i < n; i++){
            B[i] += A[n-1-i];
        }
    }

int main() {
        leggi n;
        crea A;
        crea B;
        calcolaArray(A ,B, n);
        stampa B;
        return 0;
}
```

1. Ex2 vettori e matrici

```
#include <stdio.h>
int stocasticaColonne (int n, double A[][n]) {
        int i, j;
        double somma;
        for (i = 0; i < n; i++) {
            somma = 0.0;
            for (j = 0; j < n; j++) {
                if (A[i][j] > 1) {
                    return 0
                }
                somma += A[i][j];
            }
            if (somma != 1) {
                return 0;
            }
        }
        return 1;
}
```

```
int stocasticaRighe (int n, double A[][n]) {
        int i, j;
        double somma;
        for (j = 0; j < n; j++) {
            somma = 0.0;
            for (i = 0; i < n; i++) {
                if (A[i][j] > 1) {
                    return 0
                }
                somma += A[i][j];
            }
            if (somma != 1) {
                return 0;
            }
        }
        return 1;
}
```

```
int stocastica (int n, double A[][n]) {
        return (stocasticaRighe(n,A) || stocasticaColonne(n,A));
    }
```

1. Ex3 vettori e matrici

```
#include <stdio.h>
int stocasticaColonne (int n, double A[][n]) {
        int i, j;
        double somma;
        for (i = 0; i < n; i++) {
            somma = 0.0;
            for (j = 0; j < n; j++) {
                if (A[i][j] > 1) {
                    return 0
                }
                somma += A[i][j];
            }
            if (somma != 1) {
                return 0;
            }
        }
        return 1;
}
```

```
int stocasticaRighe (int n, double A[][n]) {
        int i, j;
        double somma;
        for (j = 0; j < n; j++) {
            somma = 0.0;
            for (i = 0; i < n; i++) {
                if (A[i][j] > 1) {
                    return 0
                }
                somma += A[i][j];
            }
            if (somma != 1) {
                return 0;
            }
        }
        return 1;
}
```

```
int bistocastica (int n, double A[][n]) {
        return (stocasticaRighe(n,A) && stocasticaColonne(n,A));
    }
```

1. Ex1 ricorsione

```
#include <stdio.h>

void hipHurra(int k) {
    if (k == 0) {
        return;
    }
    printf("hip ");
    hipHurra(k-1);
    printf("hurra ");
}

int main () {
    int num;
    scanf("%d", &num);
    hipHurra(num);
    return 0;
}
```

1. Ex2 ricorsione

```
#include <stdio.h>

void occorrenze(int k, int V[], int n, int i, int* occ) {
    if (i >= n) {
        return;
    }
    if (V[i] == k) {
        *occ++;
    }
    occorrenze(k, V, n, i+1, occ);
}

int main () {
    int occ = 0;
    leggi Input;
    occorrenze(k, V, n, 0, &occ);
    return 0;
}
```

1. Ex2 ricorsione

```
#include <stdio.h>

void occorrenze(int k, int V[], int n, int* occ) {
    if (n <= 0) {
        return;
    }
    if (*V == k) {
        *occ++;
    }
    occorrenze(k, V+1, n-1, occ);
}

int main () {
    int occ = 0;
    leggi Input;
    occorrenze(k, V, n, &occ);
    return 0;
}
```

1. Ex3 ricorsione

```
#include <stdio.h>

void raddoppia(int V[], int n, int i) {
    if (i >= n) {
        return;
    }
    V[i] *= 2;
    raddoppia(V, n, i+1);
}

int main () {
    leggi Input;
    raddoppia(V, n, 0);
    return 0;
}
```

1. Ex3 ricorsione

```
#include <stdio.h>

void raddoppia(int V[], int n) {
    if (n <= 0) {
        return;
    }
    *V *= 2;
    raddoppia(V+1, n-1);
}

int main () {
    leggi Input;
    raddoppia(V,n);
    return 0;
}
```

1. Ex4 ricorsione

```
#include <stdio.h>

void stampaVettori(int A[], int B[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d - %d\n", A[i], B[n-1-i]);
    }
}

int main () {
    leggi Input;
    stampaVettori(A, B, n);
    return 0;
}
```

1. Ex4 ricorsione

```
#include <stdio.h>

void stampaVettori(int A[], int B[], int n, int i) {
    if (i >= n) {
        return;
    }
    printf("%d - %d\n", A[i], B[n-1-i]);
    stampaVettori(A, B, n, i+1);
}

int main () {
    leggi Input;
    stampaVettori(A, B, n, 0);
    return 0;
}
```

1. Ex4 ricorsione

```
#include <stdio.h>

void stampaVettori(int A[], int B[], int n) {
    if (n <= 0) {
        return;
    }
    printf("%d - %d\n", *A, *(B + n - 1));
    stampaVettori(A+1, B, n-1);
}

int main () {
    leggi Input;
    stampaVettori(A, B, n);
    return 0;
}
```

1. Ex5 ricorsione

```
int fastExp(int base, int exp) {
    if (exp == 0) {
        return 1;
    }
    if (exp%2 == 0) {
        return fastExp(base*base, exp/2);
    }
    else {
        return base * fastExp(base, exp-1);
    }
}

int main () {
    int m, n;
    scanf("%d %d", &m, &n);
    printf("%d\n", fastExp(m, n));
    return 0;
}
```