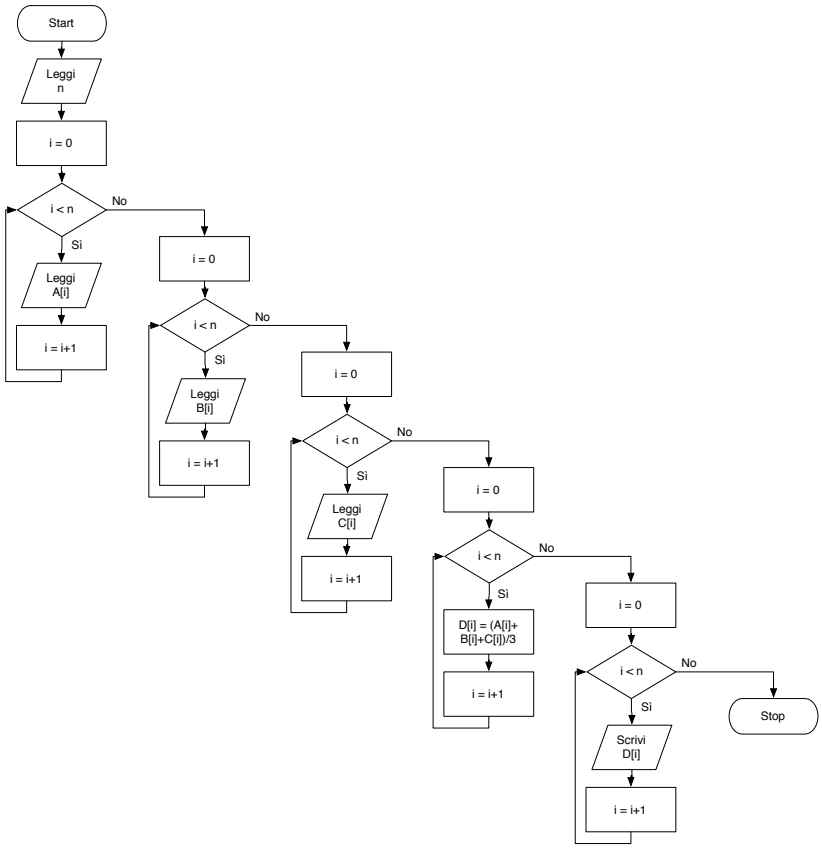
Corso di Algoritmi e Strutture Dati (IN110)
Tutorato n. 3

Marco Liverani*

Esercizio n. 1

Letti in input tre array di interi con lo stesso numero n di elementi, costruisce un quarto array di numeri floating point con la media aritmetica degli elementi corrispondenti nei primi tre (es.: $D_i = (A_i + B_i + C_i)/3$, per $i = 0, 1, \ldots, n-1$). Stampa l'array D.

Diagramma di flusso



*Università degli Studi Roma Tre, Corso di Laurea in Matematica, Corso di Algoritmi e Strutture Dati (IN110); e-mail liverani@mat.uniroma3.it – sito web del corso http://www.mat.uniroma3.it/users/liverani/IN110/

Pseudo-codifica dell'algoritmo

1: leggi l'array A di n elementi
2: leggi l'array B di n elementi
3: leggi l'array C di n elementi
4: **per** $i = 0, 1, 2, \ldots, n-1$ **ripeti**
5: $\quad D_i = (A_i + B_i + C_i)/3$
6: **fine-ciclo**
7: scrivi l'array D di n elementi
8: stop

Codifica in linguaggio C

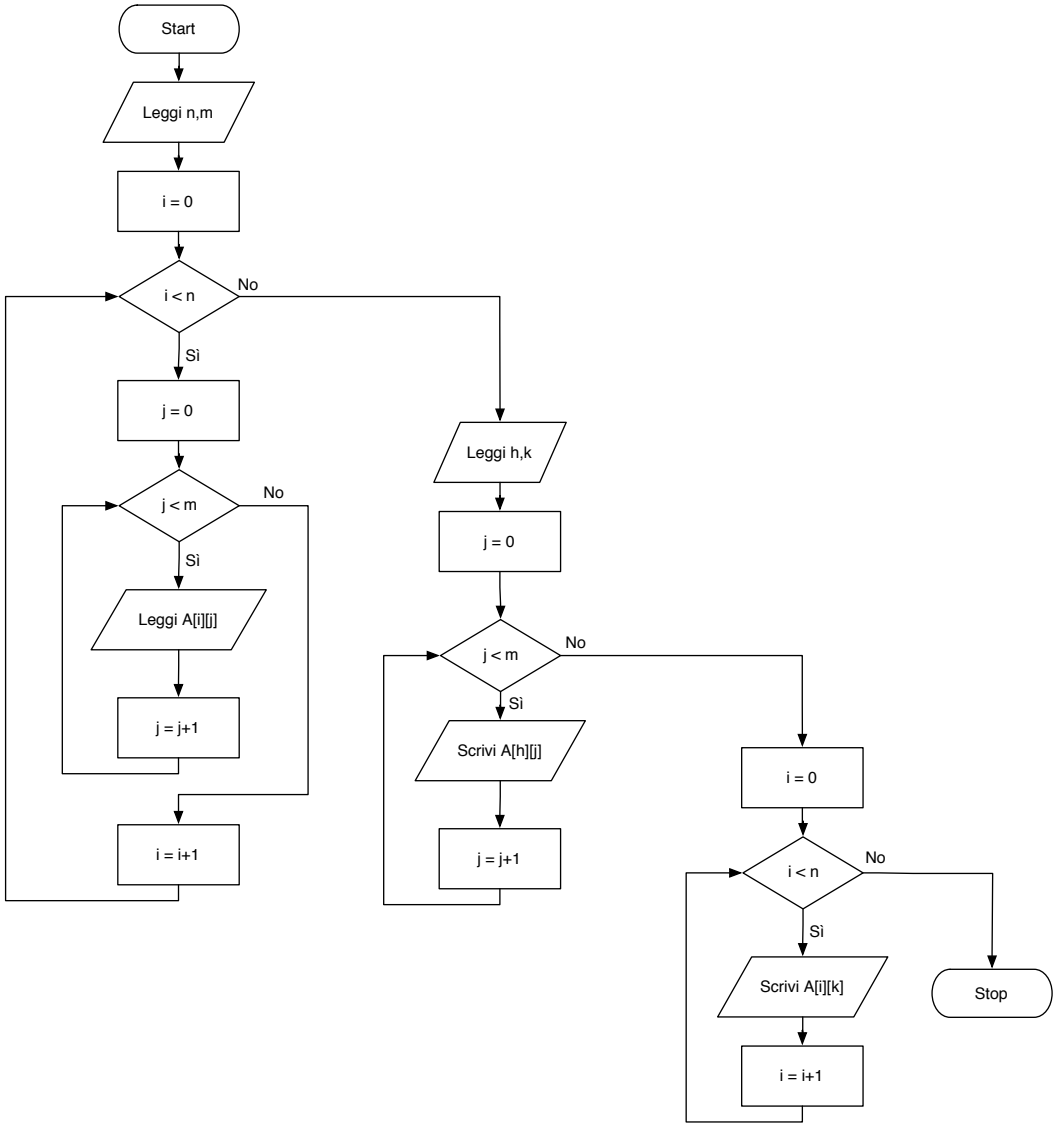
```c
#include <stdlib.h>
#include <stdio.h>
#define MAX 100

int main(void) {
  int i, n, A[MAX], B[MAX], C[MAX];
  float D[MAX];
  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("inserisci %d elementi del vettore A: ", n);
  for (i=0; i<n; i++) {
    scanf("%d", &A[i]);
  }
  printf("inserisci %d elementi del vettore B: ", n);
  for (i=0; i<n; i++) {
    scanf("%d", &B[i]);
  }
  printf("inserisci %d elementi del vettore C: ", n);
  for (i=0; i<n; i++) {
    scanf("%d", &C[i]);
  }
  for (i=0; i<n; i++) {
    D[i] = (float)(A[i]+B[i]+C[i])/3.0;
  }
  for (i=0; i<n; i++) {
    printf("%f ", D[i]);
  }
  printf("\n"); return(0);
}
```

Esercizio n. 2

Leggere una matrice $n \times m$ di numeri interi; letti in input due interi h e k ($0 \leq h < n$ e $0 \leq k < m$) stampare gli elementi della riga h e della colonna k.

Diagramma di flusso



Pseudo-codifica dell'algoritmo

1: leggi n, m
2: **per** $i = 0, 1, 2 \ldots, n-1$ **ripeti**
3: **per** $j = 0, 1, 2, \ldots, m-1$ **ripeti**
4: leggi $A_{i,j}$
5: **fine-ciclo**
6: **fine-ciclo**
7: leggi h, k
8: **per** $j = 0, 1, 2, \ldots, m-1$ **ripeti**
9: scrivi $A_{h,j}$
10: **fine-ciclo**
11: **per** $i = 0, 1, 2, \ldots, n-1$ **ripeti**
12: scrivi $A_{i,k}$
13: **fine-ciclo**
14: stop

Codifica in linguaggio C

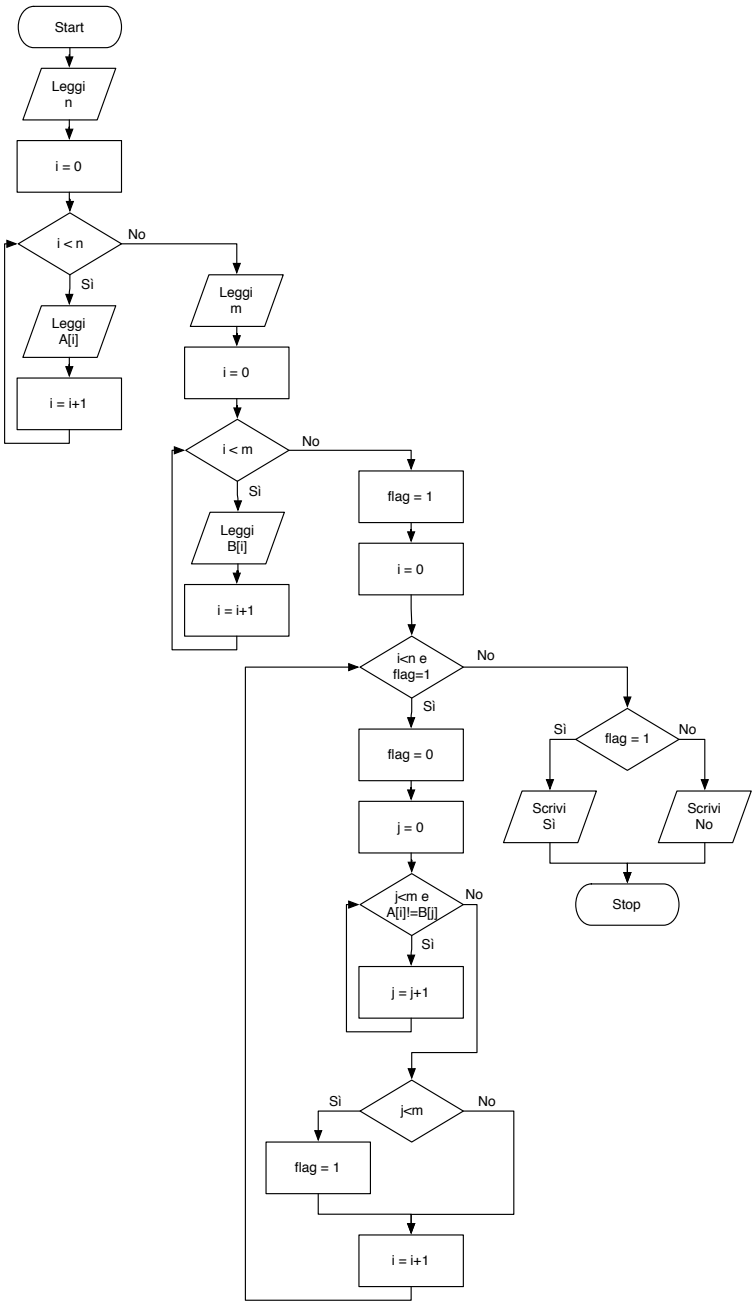
```c
#include <stdlib.h>
#include <stdio.h>
#define MAX 20

int main(void) {
  int A[MAX][MAX], i, j, n, m, h, k;
  printf("Numero di righe e di colonne: ");
  scanf("%d %d", &n, &m);
  for (i=0; i<n; i++) {
    for (j=0; j<m; j++) {
      scanf("%d", &A[i][j]);
    }
  }
  printf("Inserisci h e k: ");
  scanf("%d %d", &h, &k);
  printf("Riga %d: ", h);
  for (j=0; j<m; j++) {
    printf("%d ", A[h][j]);
  }
  printf("\nColonna %d: ", k);
  for (i=0; i<n; i++) {
    printf("%d ", A[i][k]);
  }
  printf("\n");
  return(0);
}
```

Esercizio n. 3

Letti in input due array di numeri interi (rispettivamente di n e m elementi) verifica se tutti gli elementi del primo sono contenuti nel secondo (gli elementi di $A = (1,2,3)$ sono contenuti in $B = (8,1,3,7,5,2,6)$).

Diagramma di flusso



Pseudo-codifica dell'algoritmo

1: leggi n
2: **per** $i = 0, 1, 2, \ldots, n-1$ **ripeti**
3: leggi A_i
4: **fine-ciclo**
5: leggi m
6: **per** $i = 0, 1, 2, \ldots, m-1$ **ripeti**
7: leggi B_i
8: **fine-ciclo**
9: $flag = 1$
10: $i = 0$
11: **fintanto che** $i < n$ e $flag = 1$ **ripeti**
12: $flag = 0$
13: $j = 0$
14: **fintanto che** $j < m$ e $A_i \neq B_j$ **ripeti**
15: $j = j + 1$
16: **fine-ciclo**
17: **se** $j < m$ **allora**
18: $flag = 1$
19: **fine-condizione**
20: $i = i + 1$
21: **fine-ciclo**
22: **se** $flag = 1$ **allora**
23: scrivi "l'array A è contenuto in B"
24: **altrimenti**
25: scrivi "l'array A **non** è contenuto in B"
26: **fine-condizione**
27: stop

Codifica in linguaggio C

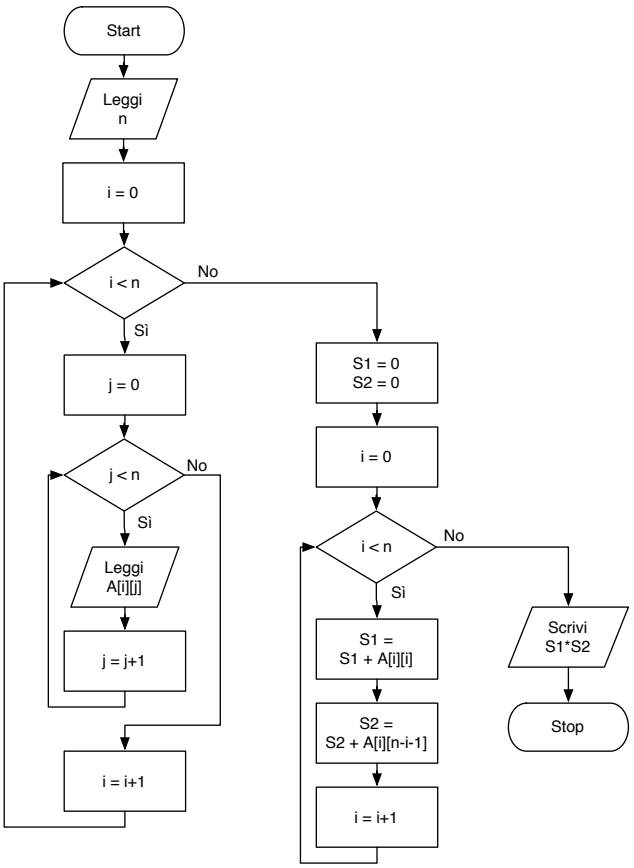
```c
#include <stdlib.h>
#include <stdio.h>
#define MAX 30

int main(void) {
  int A[MAX], B[MAX], i, j, n, m, flag;
  printf("Numero di elementi di A: ");
  scanf("%d", &n);
  printf("inserisci %d elementi: ");
  for (i=0; i<n; i++) {
    scanf("%d", &A[i]);
  }
  printf("Numero di elementi di B: ");
  scanf("%d", &m);
  printf("inserisci %d elementi: ");
  for (i=0; i<m; i++) {
    scanf("%d", &B[i]);
  }
  flag = 1;
  for (i=0; i<n && flag==1; i++) {
    flag = 0;
    j = 0;
    while (j<m && A[i]!=B[j]) {
      j = j+1;
    }
    if (j<m)
      flag = 1;
  }
  if (flag == 1)
    printf("L'array A e' contenuto in B.\n");
  else
    printf("L'array A non e' contenuto in B.\n");
  return(0);
}
```

Esercizio n. 4

Letta in input una matrice quadrata A di $n \times n$ numeri interi, stampa il prodotto della somma degli elementi della diagonale principale per la somma degli elementi della diagonale secondaria.

Diagramma di flusso



Pseudo-codifica dell'algoritmo

1: leggi n
2: **per** $i = 0, 1, 2, \ldots, n-1$ **ripeti**
3: **per** $j = 0, 1, 2, \ldots, n-1$ **ripeti**
4: leggi $A_{i,j}$
5: **fine-ciclo**
6: **fine-ciclo**
7: $S_1 = 0, S_2 = 0$
8: **per** $i = 0, 1, 2, \ldots, n-1$ **ripeti**
9: $S_1 = S_1 + A_{i,i}$
10: $S_2 = S_2 + A_{i,n-i-1}$
11: **fine-ciclo**
12: scrivi $S_1 \cdot S_2$
13: stop

Codifica in linguaggio C

```c
#include <stdlib.h>
#include <stdio.h>
#define MAX 20

int main(void) {
  int A[MAX][MAX], i, j, n, s1, s2;
  printf("Ordine della matrice quadrata (numero di righe e di colonne): ");
  scanf("%d", &n);
  for (i=0; i<n; i++) {
    printf("Inserisci gli elementi della riga %d: ", i);
    for (j=0; j<n; j++) {
      scanf("%d", &X[i][j]);
    }
  }
  s1 = 0;
  s2 = 0;
  for (i=0; i<n; i++) {
    s1 = s1 + A[i][i];
    s2 = s2 + A[i][n-i-1];
  }
  printf("S1*S2 = %d\n", s1*s2);
  return(0);
}
```