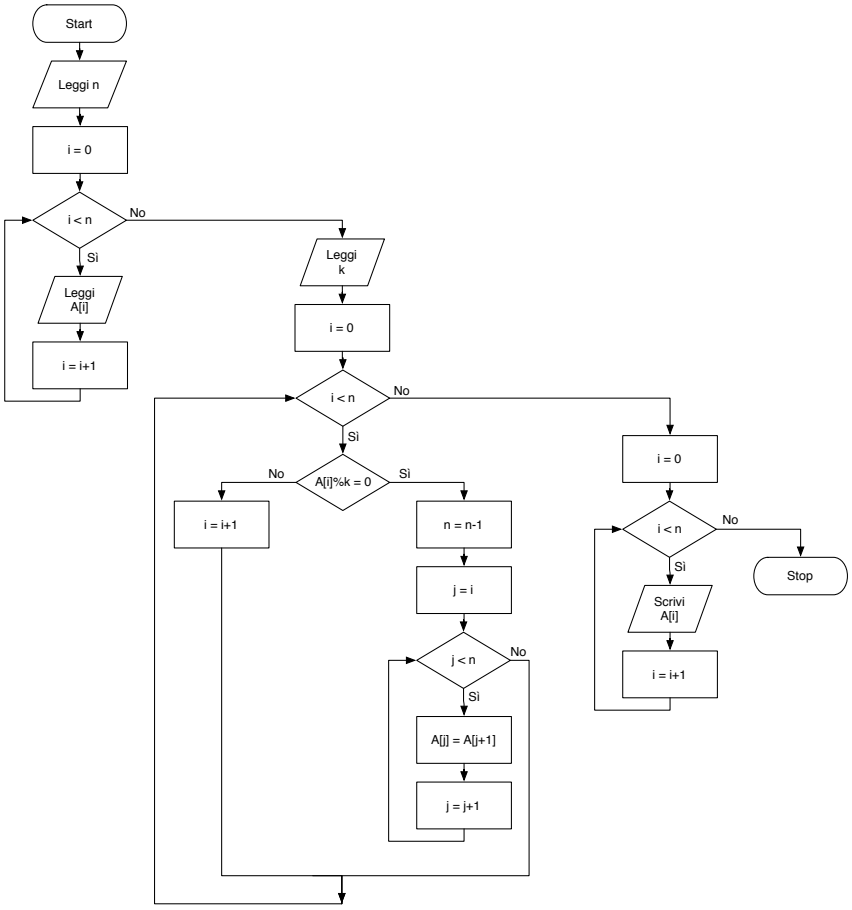
Corso di Algoritmi e Strutture Dati (IN110)
Tutorato n. 4

Marco Liverani*

Esercizio n. 1

Letto in input un array di n numeri interi ed un intero $k > 1$, eliminare dall'array tutti i multipli di k, spostando "a sinistra" gli elementi successivi.

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 $k > 1$
3: **per ogni** elemento dell'array A_i ($i = 0, \ldots, n-1$) **ripeti**
4: **se** l'elemento A_i è multiplo di k **allora**
5: $n = n - 1$
6: copia A_{j+1} in A_j per ogni $j = i, \ldots, n-1$
7: **fine-condizione**
8: **fine-ciclo**
9: stampa l'array A
10: stop

Codifica in linguaggio C

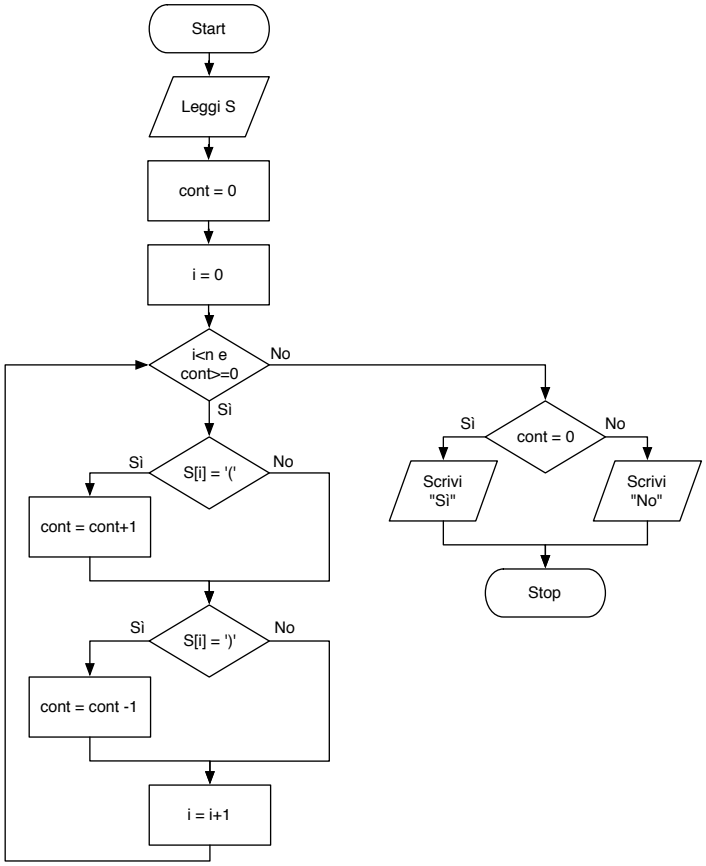
```c
#include <stdlib.h>
#include <stdio.h>
#define MAX 100

int leggi_array(int A[]) {
  int i, n;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("Inserisci %d elementi: ", n);
  for (i=0; i<n; i++)
    scanf("%d", &A[i]);
  return(n);
}

int elimina_multipli(int A[], int n, int k) {
  int i, j;

  i = 0;
  while (i<n) {
    if (A[i] % k == 0) {
      n = n-1;
      for (j=i; j<n; j++) {
        A[j] = A[j+1];
      }
    } else {
      i = i+1;
    }
  }
  return(n);
}

void stampa_array(int A[], int n) {
  int i;
  for(i=0; i<n; i++)
    printf("%d ", A[i]);
  printf("\n");
  return;
}

```

```
40 int main(void) {
41   int n, A[MAX], k;
42   n = leggi_array(A);
43   printf("Inserisci k: ");
44   scanf("%d", &k);
45   n = elimina_multipli(A, n, k);
46   stampa_array(A, n);
47   return(0);
48 }
```

Esercizio n. 2

Letta in input una stringa che rappresenta un'espressione aritmetica, verificare che le parentesi (tutte parentesi "tonde") siano collocate correttamente. Ad esempio "`a+b*(c-d)/(a+(b/2)+3/2)`" è corretta, mentre "`a+b)/(c+e)/(f+g))`" non è corretta.

Diagramma di flusso



Pseudo-codifica dell'algoritmo

1: leggi la stringa s di n caratteri
2: $c = 0, i = 0$
3: **fintanto che** $i < n$ e $c \geq 0$ **ripeti**
4: **se** s_i è una parentesi tonda aperta **allora**
5: incrementa il contatore c
6: **fine-condizione**
7: **se** s_i è una parentesi tonda chiusa **allora**
8: decrementa il contatore c
9: **fine-condizione**
10: $i = i + 1$
11: **fine-ciclo**
12: **se** $c = 0$ **allora**
13: le parentesi sono corrette
14: **altrimenti**
15: le parentesi non sono disposte correttamente
16: **fine-condizione**
17: stop

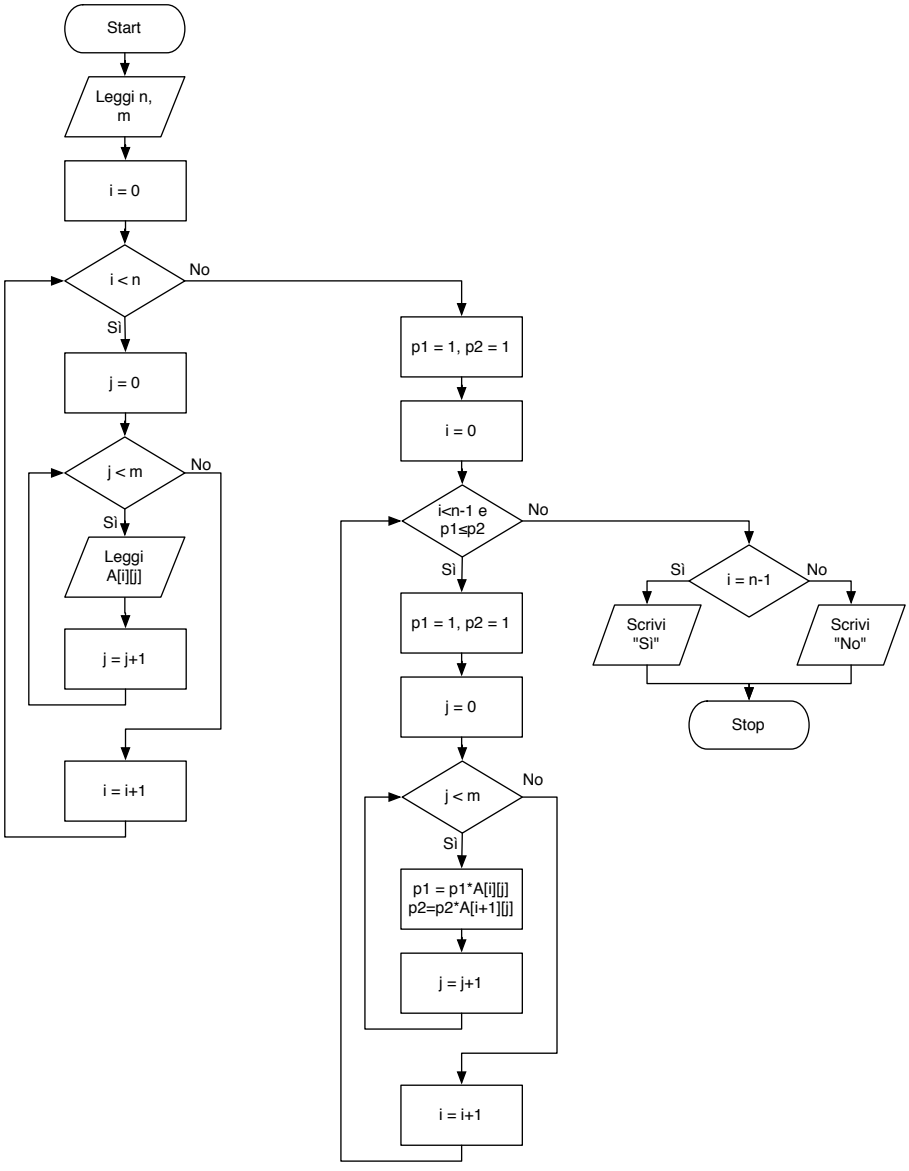
Codifica in linguaggio C

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

int verifica(char s[]) {
  int i=0, n, cont=0;
  n = strlen(s);
  while (i<n && cont>=0) {
    if (s[i] == '(')
      cont++;
    if (s[i] == ')')
      cont--;
    i++;
  }
  if (cont == 0)
    return(1);
  else
    return(0);
}

int main(void) {
  char s[MAX];
  printf("Inserisci un'espressione: ");
  scanf("%s", s);
  if (verifica(s) == 1)
    printf("Le parentesi sono corrette.\n");
  else
    printf("Le parentesi NON sono corrette.\n");
  return(0);
}
```

Esercizio n. 3

Letti in input due interi positivi n e m e gli elementi di una matrice A di ordine $n \times m$ di numeri interi $a_{i,j}$. Verificare se il prodotto di ogni riga è minore o uguale a quello della riga successiva. Stampare “sì” (una sola volta) se la condizione è verificata, stampare “no” (una sola volta) altrimenti.

Diagramma di flusso



Pseudo-codifica dell'algoritmo

1: leggi n e m
2: leggi gli elementi della matrice A con n righe e m colonne: $\forall i = 0,\ldots,n-1$ e $\forall j = 0,\ldots,m-1$ leggi $a_{i,j}$
3: considera le righe della matrice dalla prima alla penultima ($i = 0,1,\ldots,n-2$) ed esegui i passi 4-6:
4: sia p_1 il prodotto degli elementi della riga i della matrice A
5: sia p_2 il prodotto degli elementi della riga $i+1$ della matrice A
6: se $p_1 < p_2$ allora vai al passo 3 altrimenti vai al passo 7
7: se $i = n-1$ allora scrivi "Sì" altrimenti scrivi "No"
8: stop

Codifica in linguaggio C

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

void leggiMatrice(int A[MAX][MAX], int n, int m) {
  int i, j;
  for (i=0; i<n; i++) {
    printf("inserisci gli elementi della riga %d:\n", i);
    for (j=0; j<m; j++)
      scanf("%d", &A[i][j]);
  }
  return;
}

void stampaMatrice(int A[MAX][MAX], int n, int m) {
  int i, j;
  for (i=0; i<n; i++) {
    for (j=0; j<m; j++)
      printf("%4d ", A[i][j]);
    printf("\n");
  }
  return;
}

int verifica(int A[MAX][MAX], int n, int m) {
  int p1=1, p2=1, i, j;
  for (i=0; i<n-1 && p1<=p2; i++) {
    p1 = 1;
    p2 = 1;
    for (j=0; j<m; j++) {
      p1 = p1*A[i][j];
      p2 = p2*A[i+1][j];
    }
  }
  if (i == n-1)
    return(1);
  else
    return(0);
}

```

```
41 int main(void) {
42   int n, m, A[MAX][MAX];
43   printf("Numero di righe e di colonne della matrice: ");
44   scanf("%d %d", &n, &m);
45   leggiMatrice(A, n, m);
46   stampaMatrice(A, n, m);
47   if (verifica(A, n, m) == 1)
48     printf("Si'.\n");
49   else
50     printf("No.\n");
51   return(0);
52 }
```