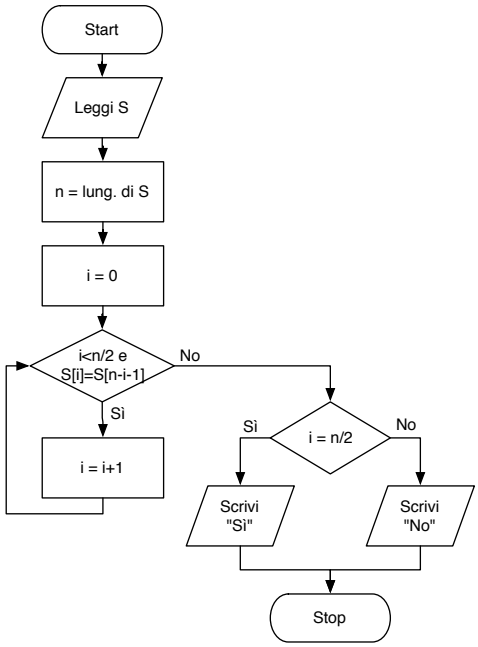
Algoritmi e Strutture Dati (IN110)
Esercitazione n. 3

Marco Liverani *

Esercizio n. 1

Letta una parola S, stabilire se S è palindroma (ossia se è simmetrica).

Diagramma di flusso



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

Pseudo-codifica dell'algoritmo

1: leggi la stringa S
2: sia n la lunghezza di S
3: $i = 0$
4: **fintanto che** $i < \frac{n}{2}$ e $S_i = S_{n-i-1}$ **ripeti**
5: $\quad i = i + 1$
6: **fine-ciclo**
7: **se** $i = \frac{n}{2}$ **allora**
8: $\quad$ scrivi "Sì"
9: **altrimenti**
10: $\quad$ scrivi "No"
11: **fine-condizione**
12: stop

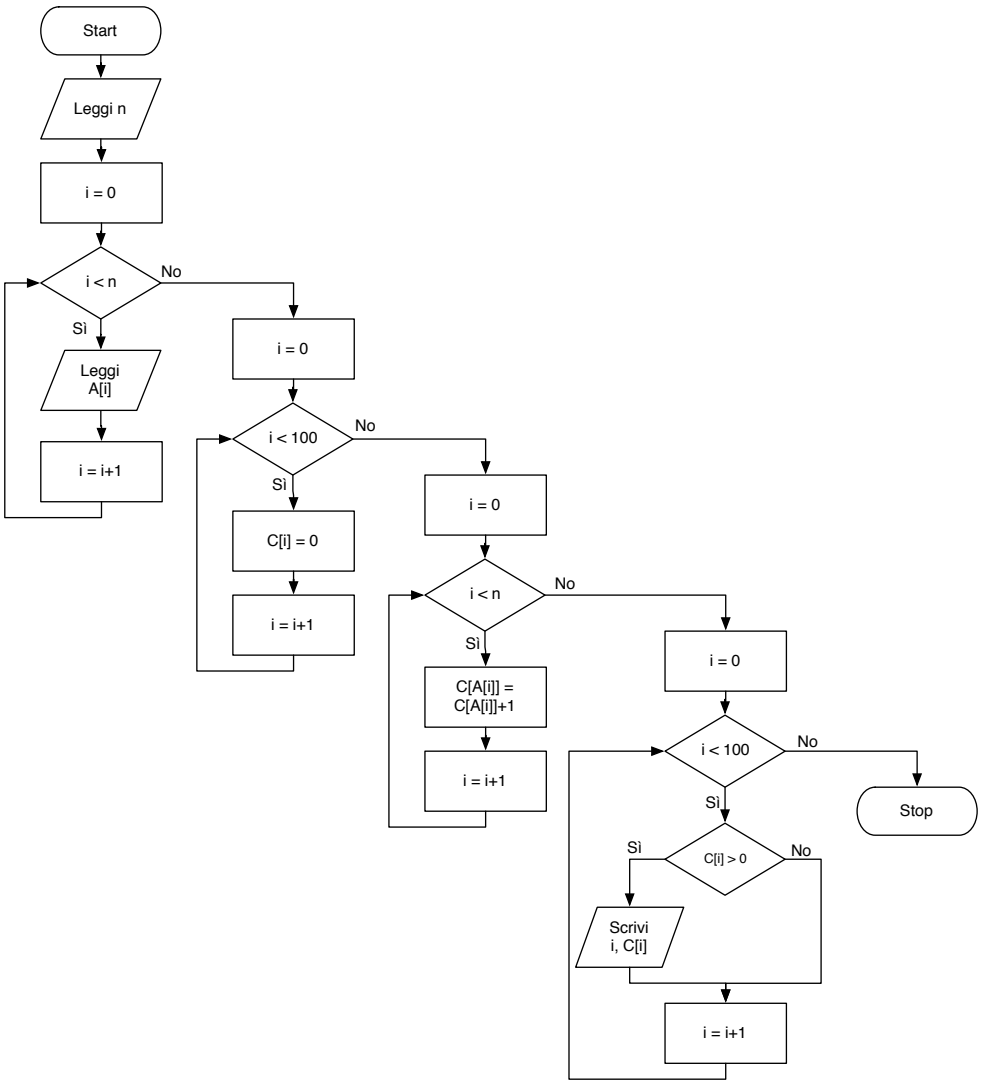
Codifica in linguaggio C

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main(void) {
  char S[50];
  int i, n;
  printf("Inserisci una parola: ");
  scanf("%s", S);
  n = strlen(S);
  for (i=0; i<n/2 && S[i]==S[n-i-1]; i++)
    ;
  if (i == n/2)
    printf("La parola e' palindroma.\n");
  else
    printf("La parola non e' palindroma.\n");
  return(0);
}
```

Esercizio n. 2

Acquisire in input un array A di n numeri interi minori di 100; per ogni elemento di A stampare il numero di volte che compare nel vettore.

Diagramma di flusso



Pseudo-codifica dell'algoritmo

1: leggi n
2: **per** $i = 0, 1, 2, \ldots, n-1$ **ripeti**
3: leggi in input un numero minore di 100 e memorizzalo in A_i
4: **fine-ciclo**
5: **per** $i = 0, 1, 2, \ldots, 99$ **ripeti**
6: $C_i = 0$
7: **fine-ciclo**
8: **per** $i = 0, 1, 2, \ldots, n-1$ **ripeti**
9: $C_{A_i} = C_{A_i} + 1$
10: **fine-ciclo**
11: **per** $i = 0, 1, 2, \ldots, 99$ **ripeti**
12: **se** $C_i > 0$ **allora**
13: scrivi "i compare C_i volte nel vettore"
14: **fine-condizione**
15: **fine-ciclo**
16: stop

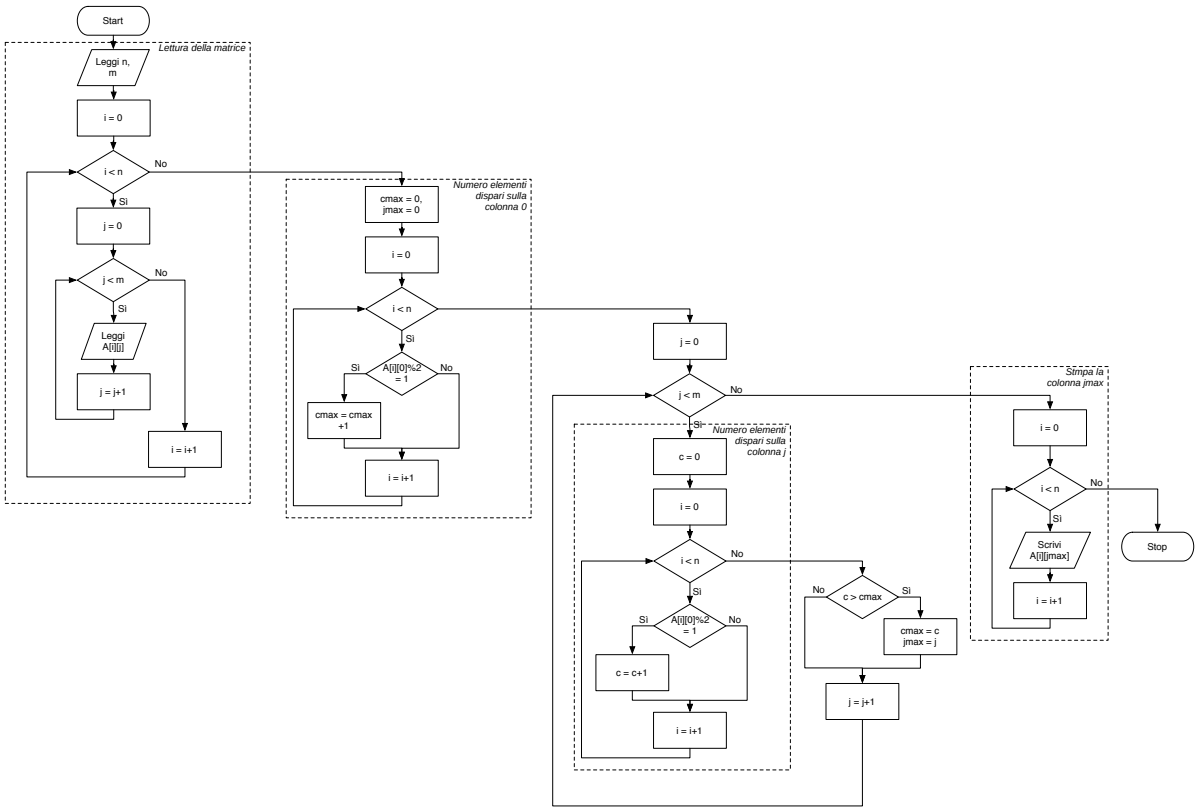
Codifica in linguaggio C

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

int leggi_array(int X[]) {
  int i, n;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("Inserisci %d numeri minori di 100: ");
  for (i=0; i<n; i++)
    scanf("%d", &X[i]);
  return;
}

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

int main(void) {
  int A[MAX], C[MAX], i, n, h, k;
  n = leggi_array(A);
  printf("L'array e' il seguente: ");
  stampa_array(A, n);
  for (i=0; i<MAX; i++)
    C[i] = 0;
  for (i=0; i<n; i++)
    C[A[i]]++;
  for (i=0; i<MAX; i++)
    if (C[i]>0)
      printf("%d compare %d volte\n", i, C[i]);
  return(0);
}
```

Esercizio n. 3

Letta in input una matrice A di $n \times m$ numeri interi, stampa la colonna con il maggior numero di elementi dispari.

Diagramma di flusso



Pseudo-codifica dell'algoritmo

1: leggi la matrice A di n righe e m colonne
2: sia $c_{\max}$ il numero di elementi dispari sulla prima colonna (quella di indice 0)
3: **per** $j = 1, 2, \ldots, m-1$ **ripeti**
4: sia c il numero di elementi dispari sulla colonna di indice j
5: **se** $c > c_{\max}$ **allora**
6: $c_{\max} = c$ e la colonna con il maggior numero di elementi dispari è la colonna $j_{\max} = j$
7: **fine-condizione**
8: **fine-ciclo**
9: stampa la colonna $j_{\max}$
10: stop

Codifica in linguaggio C

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

void leggi_matrice(int A[MAX][MAX], int *n, int *m) {
  int i, j;
  printf("Numero di righe e di colonne: ");
  scanf("%d %d", n, m);
  for (i=0; i<*n; i++) {
    printf("Elementi della riga n.%d: ", i);
    for (j=0; j<*m; j++)
      scanf("%d", &A[i][j]);
  }
  return;
}

int conta_dispari(int A[MAX][MAX], int n, int j) {
  int i, c = 0;
  for (i=0; i<n; i++)
    if (A[i][j] % 2 == 1)
      c++;
  return(c);
}

int main(void) {
  int A[MAX][MAX], n, m, j, jmax, c, cmax;
  leggi_matrice(A, &n, &m);
  jmax = 0;
  cmax = conta_dispari(A, n, 0);
  for (j=1; j<m; j++) {
    c = conta_dispari(A, n, j);
    if (c > cmax) {
      jmax = j;
      cmax = c;
    }
  }
  printf("La colonna con piu' elementi dispari e' la %d^: ", jmax+1);
  for (j=0; j<n; j++)
    printf("%d ", A[j][jmax]);
  printf("\n");
  return(1);
}
```