1 Esercizio Programmazione

Completare le parti mancati del seguente codice C con tutte le istruzioni necessarie per il corretto funzionamento del programma. Inserire dei commenti nel codice per descrivere le operazioni effettuate.

```c
/*
 * Soluzione di un sistema lineare con il metodo di Jacobi
 *
 * AX = B
 *
 * Input:
 * - n: dimensioni del sistema lineare
 * - A(n,n): matrice dei coefficienti viene fornita tramite un file "matriceA.dat"
 * - B(n): vettore dei termini noti viene fornito tramite un file "vettoreB.dat"
 * - e: accuratezza richiesta approssimazione
 * Output:
 * - X(n,n): soluzione approssimata
 * - num_iter: numero di iterazioni
 * - err_k: differenza tra le ultime due approssimazioni in norma 1
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

// Funzioni per la gestione delle matrice (da considerare implementate)
void LETTURA_MATRICE(int righe,int colonne, double A[righe][colonne],FILE *input);
void STAMPA_MATRICE(int righe, int colonne, double A[righe][colonne]);
void ZEROS_MATRICE(int righe, int colonne, double A[righe][colonne]);
void ZEROS_VETTORE(int num, double V[num]);
// Funzione per il calcolo della norma n di un vettore (da considerare implementata)
double NORMA_N_VETTORE(int num, double a[num], double ordine_norma);

int main()
{
  int i=0,j=0;
  int N=0;     /* Numero di equazioni */
  double e=0;

  /* Lettura parametri di input */
  printf("Inserire il numero di equazioni N: ");
  scanf("%d", &N);
  printf("Inserire la soglia epsilon e: ");
  scanf("%lf", &e);

  /* Allocazione della matrice A e dei vettore B e X del sistema lineare */
    double A[N][N];
    double B[N][1];
    double X0[N];
    double X[N];
    double errore[N];
    /* Inizializzazione di A e B */
    ZEROS_MATRICE(N,N,A);
    ZEROS_MATRICE(N,1,B);
    ZEROS_VETTORE(N,X0);
    ZEROS_VETTORE(N,X);
```

```
53      ZEROS_VETTORE(N, errore);
54
55      /* Lettura da file dei valori della matrice A e del vettore B */
56    FILE *inputA=NULL;
57    FILE *inputB=NULL;
58
59    inputA = fopen("matriceA.dat", "r");
60    if(inputA == NULL)
61    {
62      printf("Errore nell'apertura del file matriceA.dat\n");
63      printf("Riprovare\n");
64      return(1);
65    } /* if */
66
67    inputB = fopen("vettoreB.dat", "r");
68      if(inputB == NULL)
69    {
70      printf("Errore nell'apertura del file vettoreB.dat\n");
71      printf("Riprovare\n");
72      return(1);
73    } /* if */
74
75    LETTURA_MATRICE(N,N,A,inputA);
76    LETTURA_MATRICE(N,1,B,inputB);
77
78    printf("La matrice A:\n");
79    STAMPA_MATRICE(N,N,A);
80      printf("Il vettore B:\n");
81    STAMPA_MATRICE(N,1,B);
82
83    double err_k = 10000.00;
84    int num_iter = 0;
85
86    while (err_k > e && num_iter < 100)
87    {
88      num_iter ++;
89
90  /*  ----------------PARTE MANCANTE-------------------------
91   *
92   * inserire tutte le operazioni necessarie per completare il programma
93   *
94   */
95
96      err_k = NORMA_N_VETTORE(N, errore, 1.0);
97
98    }
99
100   printf("La soluzione:\n");
101   for (i=0;i<N;i++) printf("x%d = %12.8lf\n",i,X0[i]);
102   printf("Il numero di iterazione effettuate = %d\n",num_iter);
103   printf("La norma uno del vettore errore = %lf\n:", err_k);
104
105   return 0;
106 }
```

2 Esercizio Programmazione

Completare le parti mancati del seguente codice C con tutte le operazioni necessarie per il corretto funzionamento del programma. Inserire dei commenti nel codice per descrivere le operazioni effettuate.

```c
/* Programma di Runge-Kutta:
 *
 * Calcola la soluzione numerica di equazioni differenziali del primo ordine
 * con il metodo di Runge-Kutta esplicito
 *
 * Funzioni:
 * - f(t,y): termine noto equazione differenziale
 * - g(t): soluzione analitica del problema di Cauchy
 *
 * Input:
 * - t0, y0: condizione iniziale
 * - h: passo di discretizzazione
 * - n: numero di passi
 *
 * Output:
 * stampa su un file "Runge-Kutta_output.txt" dei valori di
 * - ti: nodo i-esimo
 * - yi: approssimazione al nodo ti
 * - ei: errore al nodo i-esimo
 */

#include <stdio.h>
#include <math.h>

// Funzione f(t,y) (da considerare implementata)
double f(double t,double y);
// Funzione g(t): soluzione esatta del problema di Cauchy (da considerare
    implementata)
double g(double t);

 int main()
 {

  // Allocazione e inizializzazione delle variabili

  int n=0, k=0;
  double t0=0.,y0=0.,h=0.;
  double ti=0.,yi=0.,err;

  FILE *OUT = NULL;

  // Recupero dei dati dati di input

  printf("Inserire il numero n di passi = \n");
  scanf("%d", &n);
  printf("Inserire il valore di h = \n");
  scanf("%lf", &h);
  printf("Inserire il valore di t0 = \n");
  scanf("%lf", &t0);
  printf("Inserire il valore di y0 = \n");
  scanf("%lf", &y0);

```

```
52   printf("n = %d \t h = %14.8lf \t t0 = %14.8lf \t y0 = %14.8lf\n\n",n,h,t0,y0);
53
54   /* Apertura file di output */
55   OUT = fopen("Runge-Kutta_output.txt", "w");
56   // Controllo apertura file
57   if(OUT == NULL)
58   {
59     printf("Errore nell'apertura del file\n");
60     printf("Riprovare\n");
61     return(1);
62   } /* if */
63
64   ti = t0;
65   yi = y0;
66
67 /*  ----------------PARTE MANCANTE----------------------------
68  *
69  * inserire tutte le operazioni necessarie per completare il programma
70  *
71  */
72
73   fclose(OUT);
74
75   return 0;
76  }
```