1. Pflicht-Übung

Num. Mathematik

2 SFT (SS02)

von

Roland Steffen
SFT1

Quellcode:

```
/*
  Löst ein spezielles lineares GLS (A*x=b; tridiagonale Koeffizientenmatrix A)

  Programm benötigt eine Eingabedatei in der Form:

    Matrizenrang n:
      n

    obere Nebendiagonale:
      o1 o2 o3 ...

    Hauptdiagonale:
      d1 d2 d3 d4 ...

    untere Nebendiagonale:
      u1 u2 u3 ...

    Konstantenvektor:
      b1 b2 b3 b4 ...

  17.06.2002
*/

#include <stdio.h>
#include <stdlib.h> // fuer 'calloc'
#include <string.h> // fuer 'strncmp'
#include <conio.h>  // fuer 'clrscr'


int keywrd_search(FILE *file_p, char *keyword);
int readInt(FILE *file, int *n, char *key);
int readDoubleMatrix(FILE *file, char *key, double *A, int n);
void writeDoubleMatrix(FILE *file, char *key, double *x, int rows, int cols);
void triGauss(double *u, double *d, double *o, double *b, double *x, int n);

main()
  {
  FILE *file;
  char filename[FILENAME_MAX+1];
  double *u;            // fuer untere Nebendiagonale der Koeffizientenmatrix
  double *d;            // fuer Hauptdiagonale der Koeffizientenmatrix
  double *o;            // fuer obere Nebendiagonale der Koeffizientenmatrix
  double *b;            // fuer Konstantenvektor
  double *x;            // fuer Loesungsvektor
  int n;

  clrscr();
  printf("Sie wollen ein spezielles lineares Gleichungssystem, der Form A*x=b mit
        tridiagonaler Koeffizientenmatrix A, mit dem gaussschen Algorithmus loesen?\n\n");
  printf("Dann geben Sie bitte eine Datei mit den entsprechen Daten an\n\n\t--> ");

  scanf("%S", filename);  // Dateinamen einlesen
  if((file = fopen(filename , "r")) == NULL) goto Fehler;   // Datei oeffen

  if(readInt(file, &n, "Matrizenrang") != 1) goto Fehler;  // Rang einlesen

  // Speicherplatz reservieren und Daten einlesen:

  // obere Nebendiagonale einlesen:
  if((o = (double *) malloc((n-1) * sizeof(double))) == NULL) return 0;
  if(readDoubleMatrix(file, "obere Nebendiagonale", o, n-1) != 1) goto Fehler;

  // Hauptdiagonale einlesen:
  if((d = (double *) malloc(n * sizeof(double))) == NULL) return 0;
  if(readDoubleMatrix(file, "Hauptdiagonale", d, n) != 1) goto Fehler;

  // untere Nebendiagonale einlesen:
  if((u = (double *) malloc((n-1) * sizeof(double))) == NULL) return 0;
  if(readDoubleMatrix(file, "untere Nebendiagonale", u, n-1) != 1) goto Fehler;

  // Konstantenvektor einlesen:
  if((b = (double *) malloc(n * sizeof(double))) == NULL) return 0;
  if(readDoubleMatrix(file, "Konstantenvektor", b, n) != 1) goto Fehler;
```

```
  fclose(file);  // Datei schliessen:

  // Speicherplatz fuer Loesungsvektor reservieren:
  if((x = (double *) calloc((size_t) n, sizeof(double))) == NULL) goto Fehler;

  triGauss(u, d, o, b, x, n); // Loesung bestimmen

  printf("\nLoesung wurde berechnet. Geben Sie bitte eine Ausgabedatei an\n\n\t--> ");

  scanf("%S", filename);  // Dateinamen einlesen
  if((file = fopen(filename , "w")) == NULL) goto Fehler;   // Datei oeffen

  writeDoubleMatrix(file, "Loesungsvektor", x, n, 1);  // Loesungsvektor schreiben

  fclose(file);  // Datei schliessen

  // allokierten Speicherplatz freigeben:
  free(o);
  free(d);
  free(u);
  free(b);
  free(x);

  printf("\n\nProgramm beendet");

  return 1;

  Fehler:
    printf("\nFehler beim Lesen bzw. Oeffen der Datei \"%s\"\n" , filename);
    fclose(file);
    return 0;
  }

// Funktion zur Ausgabe einer Matrix in eine Datei:
void writeDoubleMatrix(FILE *file, char *key, double *x, int rows, int cols)
  {
  int i, j;

  fprintf(file, "%s:\n", key);
  printf("\n\n%s:\n", key);
  for(j = 0; j < rows; j++)
    {
    for(i = 0; i < cols; i++)
      {
      fprintf(file, "%lf ", *(x + j*cols + i));
      printf("%lf ", *(x + j*cols + i));
      }
    fprintf(file, "\n");
    printf("\n");
    }
  }

// Funktion zum Aufloesen einer tridiagonal Matrix:
void triGauss(double *u, double *d, double *o, double *b, double *x, int n)
  {
  int i;
  double rq;

  // vereinfachter Gaussscher Algorithmus:
  for(i = 1; i < n; i++)
    {
    rq = *(u+i-1) / *(d+i-1);
    *(d+i) -= rq**(o+i-1);
    *(b+i) -= rq**(b+i-1);
    }

  // vereinfachte Rueckwaertsaufloesung:
  *(x+n-1) = *(b+n-1) / *(d+n-1);
  for(i = n-2; i >= 0; i--) *(x+i) = (*(b+i) - *(o+i) * *(x+i+1)) / *(d+i);
  }

// Funktion zum einlesen eines integer Werts:
int readInt(FILE *file, int *n, char *key)
  {
  if(keywrd_search(file, key) != 1) return 0;
  if(fscanf(file, "%d", n) != 1) return 0;
  printf("\n\n%s: %d", key, *n);
  return 1;
```

```
  }

// Funktion zum einlesen einer Matrix mit Double-Werten:
int readDoubleMatrix(FILE *file, char *key, double *A, int n)
  {
  int i;

  // Dateizeiger auf Key setzen:
  if(keywrd_search(file, key) != 1) return 0;

  // Matrix einlesen:
  printf("\n\n%s:\n", key);
  for(i = 0; i < n; i++)
    {
    if(fscanf(file, "%lf", (A+i)) != 1) return 0;
    printf("%lf\n", *(A+i));
    }
  return 1;
  }

// Positionieren des Lesekopfes in Zeile nach Schluesselwort setzten:
int keywrd_search(FILE *file_p, char *keyword)
  {
  char line[81];
  int n;
  int nchars;
  rewind(file_p);
  nchars = strlen(keyword);
  n = nchars > 80 ? 80 : nchars;
  do {
    if(fgets(line, 80, file_p) == NULL)
      {
      printf ("\n\nKeyword \"%s\" nicht gefunden\n", keyword);
      return 0;
      }
    } while(strncmp(line, keyword, n) != 0);
  return 1;
  }
```

Testbeispiel:

Matrizenrang n:
6

obere Nebendiagonale:
1.0
4.0
1.0
2.0
2.0

Hauptdiagonale:
3.0
5.0
3.0
7.0
4.0
5.0

untere Nebendiagonale:
3.0
1.0
2.0
3.0
3.0

Konstantenvektor:
4.0
12.0
5.0
11.0
9.0
8.0

Koeffizientenmatrix A:
3.0 1.0 0.0 0.0 0.0 0.0
3.0 5.0 4.0 0.0 0.0 0.0
0.0 1.0 3.0 1.0 0.0 0.0
0.0 0.0 2.0 7.0 2.0 0.0
0.0 0.0 0.0 3.0 4.0 2.0
0.0 0.0 0.0 0.0 3.0 5.0

Loesungsvektor:
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000