Einführung in die Programmiersprachen C und C++

Dr. Franz Gähler, Fakultät für Mathematik

Übungsblatt 5 (7 Seiten)

Zur Wiederholung wichtiger Spracheigenschaften nochmals Kalenderprogramme, sowie – auch zur Entspannung – einige Spielprogramme.

5.1 Bestimmung eines Wochentages zu einem (gregorianischen) Datum:

Die Funktion `schalt(j)` gibt 1 zurück, wenn `j` ein Schaltjahr bezeichnet, sonst 0.
Ein Schaltjahr liegt vor, wenn die Jahreszahl durch 4, aber nicht durch 100 teilbar ist, oder aber durch 400 teilbar ist.
Das Array `int days[]` gibt bei Position `m-1` an, wieviele Tage eines Jahres in einem Nicht-Schaltjahr vor dem 1. des Monats `m` vergangen sind.
Daraus lässt sich leicht die Funktion `int tag_im_jahr(int t, int m, int j)` ableiten, die zu jedem Datum `t,m,j` angibt, um den wievielten Tag im Jahr es sich handelt: nämlich um den Tag mit der Nummer `t + days[m-1]`, wozu man 1 addieren muss, wenn – in einem Schaltjahr – ein Monat nach Februar gefragt ist.
Das wird ausgedrückt durch `(m <= 2 ?  0 :  schalt(j))`, also 0 für Januar oder Februar, und `schalt(j)` sonst.
Die Funktion `int w_diff(int j, int jj)` bestimmt die Zahl der Tage modulo 7 zwischen dem 1. Januar des Jahres `j` und dem 1. Januar des Jahres `jj`. Daher lässt sich der Wochentag des 1. Januar für jedes Jahr ermitteln, wenn man ihn für ein Jahr kennt (der 1. 1. 1995 war ein Sonntag).
Der Wert der Funktion wird so ermittelt: Gäbe es keine Schaltjahre, so lägen zwischen dem `1.1.j` und dem `1.1.jj` genau `(j-jj)*365` Tage, also modulo 7 genau `j-jj` modulo 7.
Bei gregorianischer Zeitrechnung hätte es seit dem 1.1. im (fiktiven) Jahr 0 bis zum `1.1.j` genau `(j-1)/4 - (j-1)/100 + (j-1)/400 + 1` Schalttage gegeben, analog für das Jahr `jj`. Die zusätzliche 1 zählt den Schalttag im Jahr 0. Die Differenz der (seit 1.1.0 verflossenen) Schalttage für die Jahre `j` und `jj` ist zu `j - jj` hinzuzuzählen und das Ergebnis modulo 7 zu nehmen.
Für den Fall eines negativen Ergebnisses `(j < jj)` hat man zu beachten, dass `-a % u = -(a % u)` ist, man also für den Divisor 7 einen Rest zwischen -6 und 0 erhält, also die Anzahl der Tage, die man von Sonntag zurückrechnen muss, um auf den richtigen Wochentag zu kommen, man muss also stattdessen die um 7 erhöhte Zahl nehmen, um die Position des Wochentags im Array `char *tage[]` zu finden.

```
#include <stdio.h>

char *tage[] = { "Sonntag", "Montag", "Dienstag",
                 "Mittwoch", "Donnerstag", "Freitag", "Samstag" };

/* Schaltjahr */

int schalt(int j) {
  return ( j % 4 == 0 && j % 100 != 0 || j % 400 == 0 );
}

/* Anzahl der am 1.1., 1.2., 1.3. usw. im  Nicht-Schaltjahr verflossenen Tage */

int days[] = {0,31,59,90,120,151,181,212,243,273,304,334,365};

/* gibt fuer ein Datum die Nummer des Tages im Jahr zurueck */

int tag_im_jahr(int t, int m, int j) {
  return t + days[m-1] +  (m <= 2 ? 0 : schalt(j) );
}
```



```
/* Berechnet die Anzahl der Tage zwischen dem 1.1 des Jahres j
   und dem 1.1. des Jahres jj modulo 7 (gregorianisch)        */

int w_diff(int j, int jj) {
  int d;
  j--; jj--;
  d = (j - jj + j/4 - j/100 + j/400 - jj/4 + jj/100 - jj/400) % 7;
  return d >= 0 ? d : 7+d;
}

/* Ausgabe des Wochentages eines gregorianischen Datums  */
int main() {
  int t,m,j, n;
  while(1) {
    printf(" t m j : ");
    scanf("%d%d%d", &t, &m, &j);
    printf("Wochentag  1. 1. %4d : %s\n",j, tage[ w_diff(j, 1995) ] );
    n = (w_diff(j, 1995) + tag_im_jahr(t,m,j) - 1 ) % 7;
    printf("Wochentag %2d.%2d. %4d : %s\n",t,m,j, tage[n] );
  }
}
```

Aufgabe 5.1: Schreiben Sie ein Programm, das die Anzahl der Tage zwischen zwei gregorianischen Daten ermittelt, indem Sie die Funktion `w_diff` *geeignet modifizieren.*

5.2 Der Turm von Hanoi

Ein Turm aus einer Anzahl von der Größe nach geordneten übereinandergelegten Scheiben (größte zuunterst) ist von einer Position ("links") in eine andere Position ("rechts")zu versetzen. Dabei darf nur eine Hilfsposition ("mitte") zur Zwischenlagerung von Scheiben verwendet werden. Bedingung: Nie darf in einer der Positionen eine Scheibe auf eine kleinere gelegt werden. Für das klassische Standardproblem ist `n = 10`.

```
/* titel: turm.c, turm von hanoi */
#include <stdio.h>

typedef enum { links, mitte, rechts } turm;
// Funkionsprototypen
void bewege(int, turm, turm, turm);
void bewege_scheibe(turm, turm);
void print_turm(turm);

int main() {
  int scheibenzahl;
  while (1) {
    printf("\nScheibenzahl (Abbruch mit 0): ");
    while ( scanf("%d", &scheibenzahl) == 0 ) {
      getchar(); printf("\nFalsche Eingabe, bitte Zahl >= 0 eingeben: ");
    }
    if (scheibenzahl <= 0)
      break;
    printf ("%d %s\n",scheibenzahl,"Scheiben erfordern folgende Bewegungen:");
    bewege (scheibenzahl, links, mitte, rechts);
  }
}
void bewege (int anzahl, turm von, turm mittels, turm nach) {
  if ( anzahl == 1 )
```

```
      bewege_scheibe (von, nach);
    else {
      bewege ( anzahl - 1, von, nach, mittels);
      bewege_scheibe ( von, nach );
      bewege ( anzahl - 1, mittels, von, nach);
    }
  }
  void bewege_scheibe ( turm von, turm nach ) {
    printf ("Scheibe von "); print_turm (von);
    printf (" nach ");       print_turm (nach);
    printf ("\n");
  }
  void print_turm ( turm welcher ) {
    switch (welcher) {
      case links  : printf ("LINKS "); break;
      case mitte  : printf ("MITTE "); break;
      case rechts : printf ("RECHTS"); break;
    }
  }
```

Aufgabe 5.2: Schreiben Sie das Programm so um, das die Anzahl der erforderlichen Bewegungen ausgegeben wird. Haben Sie eine Vermutung, was das Ergebnis allgemein ist? Beweisen Sie diese!

5.3 Das n-Damen-Problem n 'Damen' sind auf einem Schachbrett der Größe n mal n so aufzustellen, dass keine von einer anderen angegriffen ist im Sinne der Regeln der Damenzüge im Schachspiel.
Typische Ein- und Ausgabe:

```
Welche Zahl ( < 20) ?  8
1 5 8 6 3 7 2 4
```

Dies bedeutet: Die Dame in der ersten Spalte steht in Zeile 1, die in der zweiten Spalte steht in Zeile 5 usw.
Die Funktion `int versuche()` implementiert einen "Backtracking"-Algorithmus.
Dabei wird folgendermaßen vorgegangen: Die Damen werden spaltenweise gesetzt. Ist Spalte `i` konfliktfrei besetzt (auf Position `j`), so wird die erste konfliktfreie Position in Spalte `i+1` gesucht, falls keine solche exisitiert, wird die Position in Spalte `i` durch die nächstmögliche Position ersetzt und wie oben fortgefahren. Es wird entweder eine Lösung gefunden oder festgestellt, dass keine solche existiert.

```
/* titel: dame.c 8 Damen (und mehr) */

#include <stdio.h>

#define ZZ   20             /*  Zahl der Zeilen und Spalten   */
int max, DAME[ ZZ ], ZEILE[ ZZ ], DIA0[ 2*ZZ-1 ], DIA1[ 2*ZZ-1 ];
int versuche(int);

int main() {
      int i;
      printf("Welche Zahl ( < %d, Abbruch mit 0) ? ", ZZ);
      while( scanf("%d", &max)==0 ) {
        getchar();
        printf("Falsche Eingabe, bitte Zahl eingeben: ");
      }
      if (max <=  0) return 0;
      if (max >= ZZ) max = ZZ-1;

      for (i=0; i<  max; i++) DAME[i] = ZEILE[i] = 0;
      for (i=0; i<2*max; i++) DIA0[i] = DIA1 [i] = 0;
```



```
      if ( versuche(0) ) {
         for (i=0; i < max; i++)
             printf("%d ", DAME[i]);
         printf("\n");
      }
}

int versuche(int i){
      int j, q;
      for (j = q = 0; q == 0 && j < max; j++) {
          if ( ZEILE[j] == 0 && DIA0[i+j] == 0 && DIA1[i-j+max-1] == 0 ) {
             DAME[i] = j+1;
             ZEILE[j] = DIA0[i+j] = DIA1[i-j+max-1] = 1;
             if (i == max - 1)
                q = 1;
             else
                if ( (q = versuche(i+1) ) == 0)
                   ZEILE[j] = DIA0[i+j] = DIA1[i-j+max-1] = 0;
          }
       }
       return q;
}
```

Aufgabe 5.3: Modifizieren Sie das Programm, so dass alle möglichen Lösungen ausgegeben werden.

Aufgabe 5.4: Ändern Sie die Ausgabe so ab, dass ein Feld von Charakteren der Größe n dargestellt wird, wobei die Positionen der Damen durch ein '' gekennzeichnet wird.*

5.4 Rösselsprung
Das folgende Programm berechnet eine Folge von Springer-Zügen im Sinne des Schachspiels, die ein ganzes Schachfeld der Größe ZZ × ZZ bedeckt derart, dass jedes Feld genau einmal erreicht wird.
Auch hier wird in der Funktion `versuche()` ein Backtracking-Algorithmus implementiert.

```
/* titel: springer.c, Roesselsprung-Programm */

#include <stdio.h>

#define ZZ 8                                /* Feldgroesse */
int z;

int FELD[ ZZ ][ ZZ ];                       /* Feld                 */
int a[] = { 2, 1, -1, -2, -2, -1,  1,  2 };  /* Koordinaten der      */
int b[] = { 1, 2,  2,  1, -1, -2, -2, -1 };  /* Springerzuege        */

int xx, yy;
int versuche(int,int);
void drucke(void);

int main() {
    z = 4;
    while ( 1 ) {
      printf("Zeilen (<=%d): ", ZZ);
      while ( scanf("%d",&z) != 1 ) {
        getchar();
```

```
          printf("\nFehlerhafte Eingabe, bitte Zahl <= %d eingeben: ", ZZ);
        }
        if ( z <= 1 || z > ZZ) return 0;
        { int i,j;
          for (i=0;i<z;i++)
            for (j=0;j<z;j++)
              FELD[i][j] = 0;
        }
        printf("Anfangswerte xx yy <= %d: ", z);
        while (scanf("%d %d", &xx, &yy) != 2) {
          getchar();
          printf("\nFehlerhafte Eingabe, bitte zwei Zahlen <= %d eingeben: ", z);
        }
        FELD[ xx-1 ][ yy-1 ] = 1;
        if ( versuche( xx-1, yy-1) )
          drucke();
        else printf("\nKeine Loesung\n");
      }
    }

    int versuche( int x, int y) {
          int k, q, u, v;
          for (k = q = 0; q == 0 && k < 8; k++) {
              u = x+a[k]; v = y+b[k];
              if ( 0 <= u && u < z && 0 <= v && v < z && FELD[u][v] == 0 ){
                  FELD[u][v] = FELD[x][y] + 1;
                  if ( FELD[u][v] == z*z )
                      q = 1;
                  else
                      if ( (q = versuche( u, v) ) == 0)
                          FELD[u][v] = 0;
              }
          }
          return(q);
    }

    void drucke() {
         int i, j;
         printf("\n");
         for (i = 0; i < z; i++) {
             for (j = 0; j < z; j++)
                 printf("%3d", FELD[i][j]);
             printf("\n");
         }
    }
```

Aufgabe 5.5: Modifizieren Sie das Programm derart, dass ein "zyklischer" Rösselsprung gefunden wird, so dass also das erste Feld durch einen Springerzug vom letzten Feld aus erreicht wird.

Aufgabe 5.6: Schreiben Sie eine Version des Programms, die alle zyklischen Lösungen findet.

5.5 Ein Programm zur Lösung von Sudokus

Aufgabe 5.7: i) Erweitern Sie das Programm, so dass alle Lösungen gefunden werden.
ii) Schreiben Sie eine Eingabe-Funktion, die neun Zeilen aus je einem String von neun Ziffern in der Form `530070000` *oder ähnlich akzeptiert (und nichts anderes!) und die neun Ziffern der i-ten Zeile als die Einträge der i-ten Eingabe-Matrix interpretiert.*

```
/* sudoku.c */
#include <stdio.h>

int feld[9][9] = {         /* Das zu loesende Sudoku */
  {5,3,0, 0,7,0, 0,0,0},
  {6,0,0, 1,9,5, 0,0,0},
  {0,9,8, 0,0,0, 0,6,0},

  {8,0,0, 0,6,0, 0,0,3},
  {4,0,0, 8,0,3, 0,0,1},
  {7,0,0, 0,2,0, 0,0,6},

  {0,6,0, 0,0,0, 2,8,0},
  {0,0,0, 4,1,9, 0,0,5},
  {0,0,0, 0,8,0, 0,7,9},
};

void output() {
  int i,j;
  printf("-------------------\n");
  for ( i = 0; i < 9; i++ ) {
    for ( j = 0; j < 9; j++ ) {
      printf("%d ",feld[i][j]);
      if ( j % 3 == 2) printf(" ");
    }
    printf("\n");
    if ( i % 3 == 2 ) printf("\n");
  }
}

/*  Es wird u>0 vorausgesetzt.  Rueckgabe von 1, falls u bei Position
(a,b) moeglich, sonst Rueckgabe von 0. */

int check(int a, int b, int u) {
  int i,j;
  if (feld[a][b])        return 0;
  for (i = 0; i < 9; i++) {
    if (u == feld[a][i]) return 0;
    if (u == feld[i][b]) return 0;
  }
  for (i = a - a%3 ; i < a - a%3 + 3; i++)
    for (j = b - b%3 ; j < b - b%3 + 3; j++)
      if (u == feld[i][j])
        return 0;
  return 1;
}
```

```
int versuche(int k) {  /* k --> k/9 k%9 */
  int i, q;
  while ( k<=80 && feld[k/9][k%9] ) k++; /* naechstes freies Feld  */
  if ( k == 81 ) return 1;               /* alles belegt, fertig   */
  for ( q=0, i=1 ; q == 0 && i <= 9 ; i++) {
    if ( check(k/9, k%9, i) ) {  /* teste das Feld k mit allen i    */
      feld[k/9][k%9] = i;
      if ( (q = versuche(k+1) ) == 0) /* naechstes Feld  */
        feld[k/9][k%9] = 0;  /* Falls erfolglos, Ruecksetzen    */
    }
  }
  return q;
}

int main() {
  output();
  versuche(0);
  output();
}
```