Einführung in die Programmiersprachen C und C++

Markus Kirschmer, Fakultät für Mathematik

Übungsblatt 5

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

5.1 Bestimmung von Ostern zu einem (gregorianischen) Datum:

Ostern findet jedes Jahr am ersten Sonntag nach dem ersten Frühlings-Vollmond statt, frühestens am 22.3. und spätestens am 25.4.

Die Berechnung des Ostersonntags ist recht aufwendig. Viele Autoren haben Algorithmen zum Bestimmen des Ostersonntags gegeben, unter anderem C.F. Gauß, vgl.

`de.wikipedia.org/wiki/Gaußsche_Osterformel`

oder auch von D. Knuth.

`doi.org/10.1145/366920.366980`

Aufgabe 5.1. *Ändern Sie das Programm* `date.c` *von Blatt 4 so ab, daß das Datum des Ostersonntags des eingegebenen Jahres mit angegeben wird.*

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;

void print_turm ( turm welcher ) {
  switch (welcher) {
    case links  : printf ("LINKS "); break;
    case mitte  : printf ("MITTE "); break;
    case rechts : printf ("RECHTS"); break;
  }
}

void bewege_scheibe ( turm von, turm nach ) {
  printf ("Scheibe von "); print_turm (von);
  printf (" nach ");       print_turm (nach);
  printf ("\n");
}

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);
  }
}

int main() {
  int scheibenzahl;
  printf("Scheibenzahl: ");
  while ( scanf("%d", &scheibenzahl) == 0 ) {
    getchar(); printf("\nFalsche Eingabe, bitte Zahl >= 0 eingeben: ");
  }
  if (scheibenzahl <= 0)
    return 0;
  printf ("%d %s\n",scheibenzahl,"Scheiben erfordern folgende Bewegungen:");
  bewege (scheibenzahl, links, mitte, rechts);
}
```

Aufgabe 5.2. *Schreiben Sie das Programm so um, daß 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 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;
}

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");
  }
}
```


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 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 z, xx, yy;

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");
  }
}

int main() {
  z = 4;

  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");
}
```

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

```
/* 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();
}
```

Aufgabe 5.7. *1. Erweitern Sie das Programm, so daß alle Lösungen gefunden werden.*

2. 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.*