

**Einführung in die Programmiersprachen C und C++**

Prof. Dr. Ulf Rehmann, Fakultät für Mathematik

Übungsblatt 5 (6 Seiten)

**Beispiel: Kalenderdatum**

Kalenderdaten sind ein Beispiel für eine mehrteilige Struktur. Sie können als Einheit aufgefasst und so bearbeitet werden, z. B., um die Anzahl der Tage zwischen zwei Daten zu ermitteln, etwa für Zinsberechnungen etc.

*Aufgabe 5.1: Erweitern Sie dies Programm so, dass die Funktion add auch negative int verarbeitet.*

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

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

struct datum {
    int t; int m; int j; int n; /* t m j = tag monat jahr          */
    /* n = Nummer des Tages im Jahr                                */
};
typedef struct datum datum;

/* Hier ist die zweite Komponente ein int-Array der Laenge 2: */
typedef struct monat { char *name; int n[2]; } monat;

/* Array aus struct monat : Die Zahlen geben die Anzahl Tage an,
   die nach Ende des Monats verfloren sind - in der zweiten Spalte fuer
   Schaltjahre */
monat mon[] = {
    {"null", { 0, 0}},
    {"Januar", { 31, 31}}, {"Februar", { 59, 60}}, {"Maerz", { 90, 91}},
    {"April", {120,121}}, {"Mai", {151,152}}, {"Juni", {181,182}},
    {"Juli", {212,213}}, {"August", {243,244}}, {"September", {273,274}},
    {"Oktober", {304,305}}, {"November", {334,335}}, {"Dezember", {365,366}}
};

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

int jahr_laenge(int j) { return 365 + schalt(j); }

int mon_laenge(datum d) { /* braucht "zulaessiges" d.m */
    return mon[d.m].n[ schalt(d.j) ] - mon[d.m-1].n[ schalt(d.j) ];
}

void printdatum(datum d) { /* braucht "zulaessiges" d.m */
    printf("%d. %s, %d\n", d.t, mon[d.m].name, d.j);
}

/* berechnet die Nummer des Tages im Jahr, braucht "zulaessiges" Datum*/
void normalize(datum *p) {
    p->n = p->t + mon[ p->m - 1 ].n[ schalt(p->j) ];
}
```

```

/* Baut ein Datum und kontrolliert Zulaessigkeit: */
datum makedatum(int tag, int monat, int jahr) {
    datum d = { tag, monat, jahr, 0 };
    if ( 1 <= d.m && d.m <= 12 && 1 <= d.t && d.t <= mon_laenge(d) ) {
        normalize(&d);
        return d;
    }
    printf("Das Datum gibt es nicht : %d.%d.%d\n", tag,monat,jahr);
    exit(1);
}

/* Liefert zum Tag n >= 1 und Jahr j
   das Datum des n-ten Tages ab und inklusive 1.1.j */
datum dat(int j, int n) {
    int i;
    for (i=1; i <= 12; i++)
        if (n <= mon[i].n[ schalt(j) ] )
            return makedatum( n - mon[i-1].n[ schalt(j) ], i, j);
    return dat( j+1, n - jahr_laenge(j) );
}

/* Berechne u - v */
int diff(datum u, datum v) {
    int n, i;
    if (u.j < v.j || (u.j == v.j && u.n < v.n ) ) return -diff(v,u);
    n = u.n;
    for(i = u.j - 1; i >= v.j; i--)
        n += jahr_laenge(i);
    n -= v.n;
    return n;
}

/* Liefert das Datum des folgenden Tages */
datum inc_tag(datum v) {
    return dat(v.j, v.n+1 );
}

datum add(datum u, int t) {
    if (t >= 0) {
        return dat(u.j, u.n + t);
    }
    printf ("Nicht implementiert (t negativ) : %d\n", t);
    exit(1);
}

char *wochentag(datum d) {
    datum e; int t;
    e = makedatum(31,12,2000);      /* ist ein Sonntag */
    t = diff(d,e) % 7;
    if ( t < 0 ) t += 7;
    return tage[t];
}

```

```

main() {
    int t, m, j;
    datum d, dd;
    while(1) {
        printf("Bitte Tag Monat Jahr (als Zahlen) eingeben : ");
        scanf("%d %d %d", &t, &m, &j);
        d = makedatum(t,m,j);
        printf("Das Datum ist : "); printdatum(d);
        printf("Es ist der %d-te Tag im Jahr %d.\n", d.n, d.j);
        printf("Es ist ein %s.\n", wochentag(d) );
        printf("Der naechste Tag ist : "); printdatum( inc_tag(d) );
        printf ("wieviele Tage dazu ? "); scanf("%d", &t);
        dd = add(d,t);
        printdatum( dd );
        printf("Unterschied nach diff : %d\n", diff(d,dd));
    }
}

```

### Zur Entspannung einige Spielprogramme:

**Turm von Hanoi, Acht-Damen-Problem (sogar für n Damen), Rösselsprung**

#### 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>
#include <ctype.h>

typedef enum { links, mitte, rechts } turm;

main() {
    int scheibenzahl;
    void bewege();
    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) {
    void bewege_scheibe();
    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 ) {
    void print_turm();
    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, dass die Anzahl der erforderlichen Bewegungen ausgegeben wird.*

## 5.2 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.

```

/* titel: 8.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 ];

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

    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.*

### 5.3 Rösselsprung:

Das folgende Programm berechnet eine Folge von Springer-Zügen im Sinne des Schachspiels, die ein ganzes Schachfeld der Größe  $ZZ \times ZZ$  bedeckt derart, dass jedes Feld genau einmal erreicht wird.

```

/* 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;
void drucke();

main() {
    z = 4;
    while ( 1 ) {
        printf("Zeilen: ");
        while ( scanf("%d",&z) != 1 ) {
            getchar();
            printf("\nFehlerhafte Eingabe, bitte Zahl < %d eingeben: ", ZZ);
        }
        if ( z <= 1 || z >= ZZ ) exit(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");
    }
}

```

Typische Ein- und Ausgabe:

Zeilen: 6  
Anfangswerte xx yy = 1 1

```

1 16  7 26 11 14
34 25 12 15  6 27
17  2 33  8 13 10
32 35 24 21 28  5
23 18  3 30  9 20
36 31 22 19  4 29

```

*Aufgabe 5.5: i) 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.*

*ii) Schreiben Sie eine Version des Programms, die alle zyklischen Lösungen findet.*