Lehrstuhl für Informatik 3
Rheinisch-Westfälische Technische Hochschule
Aachen
Univ.-Prof. Dr.-Ing. M. Nagl

Aachen, 18. 05. 2007

Nagl, Ranger, Wörzberger

Übungen zur Vorlesung
„Grundgebiete der Informatik 2:
Algorithmen und Programmiertechniken“

— Lösung zu Blatt 6 —

12. Aufgabe

```
/* Dieses Programm raet eine vom Benutzer gedachte Zahl in einem gegebenen
   Bereich, indem es den Bereich in dem die Zahl liegen kann in jedem Schritt
   halbiert. Dieses Verfahren aehnelt der binaeren Suche. */
#include <iostream>
using namespace std;

int main() {
   int untergrenze;     // Untergrenze des Bereichs
   int obergrenze;      // Obergrenze des Bereichs

   int aktUntergrenze;  // aktuelle Untergrenze waehrend des Ratens
   int aktObergrenze;   // aktuelle Obergrenze waehrend des Ratens
   int gerateneZahl;    // Zahl die geraten wird

   char eingabe;        // Eingabe des Spielers (r/k/g)
   bool eingabeOK;      // war die Eingabe OK?
   int anzahlVersuche = 0;

   /* Grenzen einlesen und ueberpruefen */
   do {
      cout << "Untergrenze: "; cin >> untergrenze;
      cout << "Obergrenze:  "; cin >> obergrenze;

      cout << "Gewählter Bereich: ["
           << untergrenze << ", " << obergrenze << "]" << endl;
      if (untergrenze > obergrenze) {
         cerr << "Fehler: leerer Bereich!" << endl;
      }
   } while (untergrenze > obergrenze);

   cout << "Denk Dir eine Zahl zwischen "
        << untergrenze << " und " << obergrenze << endl;

   aktUntergrenze = untergrenze;
   aktObergrenze = obergrenze;

   while (true) {
       /* einen Versuch machen */
      anzahlVersuche++;
```

```
      gerateneZahl = (aktUntergrenze + aktObergrenze) / 2;

      do {
         eingabeOK = true;
         cout << "Mein " << anzahlVersuche << ". Versuch ist "
              << gerateneZahl << "." << endl;
         cout << "Ist diese Zahl richtig, oder "
              << "ist Deine kleiner oder größer (r,k,g)? ";
         cin >> eingabe;

         /* Auf Eingabe reagieren: */
         switch (eingabe) {
         case 'r':
            cout << "Richtig geraten im " << anzahlVersuche << ". Versuch!" << endl;
            return 0;
         case 'k':
            aktObergrenze = gerateneZahl-1;
            break;
         case 'g':
            aktUntergrenze = gerateneZahl+1;
            break;
         default:
            eingabeOK = false;
            cerr << "Fehler: unbekannte Eingabe!" << endl;
            continue;
         }

         /* Nicht gefunden weil der Benutzer faelschlicherweise nicht 'r'
            eingegeben hat? */
         if (aktUntergrenze > aktObergrenze) {
            eingabeOK = false;
            cerr << "Das hätte aber stimmen müssen!" << endl;
         }
      } while (!eingabeOK);
   }
}
```

13. Aufgabe

(a) Eine weitere Variante ist die, den Zustand (`gef`/`nicht_gef`) als Rückgabe der Funktion zu verwenden.

```
    /* Typdefinitionen, ... */

    /* gibt gef zurueck, falls ges_Wert gefunden wurde, sonst nicht_gef. */
    zust linSuche(it_FT &f, item ges_Wert, index ia, index ie,
                  index &ind) {                  // Index falls gefunden
        /* Suchschleife, ... */
      if (indx < og) {
        ind = indx;                              // Rueckgabeparameter setzen
        return gef;                              // gef direkt zurueckgeben
      } else {
        return nicht_gef;
      }
    }
```

(b) Vor- und Nachteile der drei Varianten:

1. **Vorteil**: Wenn eine Funktion mehrere verschiedene Werte zurückgeben soll ist es oft am besten, *alle* einheitlich als Referenzparameter zurückzugeben und nicht einen als Returnparameter und die anderen als Referenzparameter.

   Nachteil: Da `linSuche` hier eine Prozedur ist (keinen Rückgabewert hat), kann sie nicht innerhalb eines Ausdrucks aufgerufen werden (z.B. `if (gef==linSuche(...)) ...`.

2. **Vorteil**: bei dieser Lösung ist der Index des gefundenen Elements mit dem zurückgegebenen Zustand zusammengefaßt, der angibt, ob der Index überhaupt einen definierten Wert hat.

   Nachteil: für den Rückgabewert muß ein neuer Typ `Suchergebnis` eingeführt werden.

3. **Vorteil**: diese Version läßt sich gut verwenden, wenn nur überprüft werden soll, ob ein Element im Feld enthalten ist.

   Nachteil: auch, wenn nur überprüft werden soll, ob ein Element im Feld enthalten ist kann sich `ind` ändern, was evtl. nicht beabsichtigt ist.

(c) In der Funktion wird strikt zwischen der Suche nach dem Wert und der Entscheidung, ob ein Wert gefunden wurde, unterschieden. Dies ist jedoch sehr umständlich und nicht leicht einzusehen. Eleganter ist die folgende Lösung. Hier wird das Ergebnis an der Stelle zurückgegeben, wo der Wert auch gefunden wurde:

```
/* Typdefinitionen, ... */

/* gibt gef zurueck, falls ges_Wert gefunden wurde, sonst nicht_gef. */
zust linSuche(it_FT &f, item ges_Wert, index ia, index ie,
              index &ind) {                  // Index falls gefunden
    for (index i = ia; i <= ie; i++) {
if (f[i] == ges_Wert) {             // Wert gefunden?
    ind = i;                        // Ja! Also setze Index
    return gef;                     // und verlasse die Funktion
}
    }

    return nicht_gef;                       // Nicht gefunden!
}
```