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

Aachen, 27. 04. 2007

Nagl, Ranger, Wörzberger

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

— Lösung zu Blatt 3 —

6. Aufgabe

(a) Bedingte Anweisungen und Auswahlanweisungen.

1. Mit `if` statt `switch`

```
cout << "Was tun?";    cin >> c;
     if (c == 'n') { cout << "Neue Datei anlegen" << endl; }
else if (c == 'u') { cout << "Datei umbenennen"   << endl; }
else if (c == 'l') { cout << "Datei loeschen"     << endl; }
else               { cout << "Unbekannte Eingabe" << endl; }
```

2. Mit `switch` statt `if`

```
cout << "Zahl eingeben (0..15)";
cin >> i;
switch(i) {
  case 0: case 1: case 2: case 3: case 4:
  case 5: case 6: case 7: case 8: // Fallthrough
  case 9:
       cout << "Hex: " << char(i+'0') << endl;
       break;
  case 10: case 11: case 12: case 13: case 14:
  case 15:                      //  Fallthrough
       cout << "Hex: " << char(i-10+'A') << endl;
       break;
  default:
       cout << "Zahl nicht im legalen Bereich" << endl;
}
```

Es fällt auf, daß viel Schreibarbeit entsteht, da sich einzelne Fälle beim C++-`switch`-Statement nicht zu Bereichen zusammenfassen lassen.

(b) Kontrollierte Sprünge.

1. Statt `goto` kann oft `break` oder `continue` verwendet werden.

```
cout << "Zahl eingeben:";   cin >> i;
j = 2;
while( j*j <= i ) {
   if (0 == i%j) { break; }
   j++;
}
if (j*j <= i) { cout << "keine Primzahl" << endl; }
else          { cout <<       "Primzahl" << endl; }
```

2. Eine Alternative ohne Verwendung von `continue` ist unten angegeben.

```
i = 0;   c=' ';
while (c != '.' && NULL != cin.get(c)) {
   if ('\n' == c) {             // Return eingegeben?
      i++;
      cout << i << ":";         // Zeilennummer ausgeben
   }
}
```

(c) Die `for`-Schleife aus der Aufgabenstellung wurde unten durch eine `while`-Schleife ersetzt.

```
char text[] = "Wie lang ist der Text?";
i = 0;
while( 0 != text[i] ) { ++i; }
cout << "Der Text ist " << i << " Zeichen lang" << endl;
```

7. Aufgabe

Die Änderungen gegenüber dem alten Listing beschränken sich auf neue Header, die Ausgabe am Ende des Programms und die Kommentierung der Leseschleife.

In der Aufgabenstellung war nur *angedeutet*, daß nicht alle Zeichen druckbar sind. Dies wird hier vor der Ausgabe getestet. So liefert auch `./histogramm < histogramm` vernünftige Ergebnisse. Beachten Sie, daß `cout << zeichenIndex` eine ganz andere Ausgabe liefert als `cout << (char)zeichenIndex`!

Diese Lösung berechnet mit $\log_{10}$ die für die größte Zeichenzahl nötige Breite, um eine hübschere Tabelle zu produzieren. Derartiges war nicht wirklich verlangt.

```
#include <math.h>     // Fuer log10() (Logarithmus)
#include <iostream>
#include <iomanip>    // Fuer setw() (ostream-Feldbreite einstellen)
#include <ctype.h>    // Fuer isprint() etc. (character types)

using namespace std;

int main() {
   unsigned int const ANZAHL_ZEICHEN = 256;   // Die Anzahl der unterschiedlichen Zeichen.
   unsigned int  gezaehlteZeichen[ANZAHL_ZEICHEN];// Fuer jedes Zeichen ein Zaehler
   // Zusammenfassend:
   unsigned int  anzahlBuchstaben = 0;// Buchstaben
   unsigned int  anzahlWhitespace = 0;// Leerzeichen etc.
   char gelesenesZeichen; // Das zuletzt gelesene Zeichen
   unsigned int  maximaleHaeufigkeit = 0;


   for(unsigned int zeichenIndex = 0   // Initialisiere alle Zaehler mit 0.
          ; zeichenIndex < ANZAHL_ZEICHEN
          ; ++zeichenIndex) {
      gezaehlteZeichen[zeichenIndex] = 0;
   }

   cin.get(gelesenesZeichen);   // Lese die Eingabe und erhoehe die entsprechenden Zaehler.
   while (!cin.eof()) {
      ++gezaehlteZeichen[(unsigned int)gelesenesZeichen];
      cin.get(gelesenesZeichen);
   }

   /* Breite der Zahlenausgabe fuer eine huebschere Tabelle bestimmen.
      Dabei beachten, dass log10(0) undefiniert ist. */
   for (unsigned int zeichenIndex = 0
           ; zeichenIndex < ANZAHL_ZEICHEN
           ; ++zeichenIndex) {
      if (gezaehlteZeichen[zeichenIndex] > maximaleHaeufigkeit) {
         maximaleHaeufigkeit = gezaehlteZeichen[zeichenIndex];
      }
   }
   int const BREITE_ANZAHL = (int)(log10(double((maximaleHaeufigkeit>0 ? maximaleHaeufigkeit:1 ))))+1;
   int const BREITE_CODE   = (int)(log10(double(ANZAHL_ZEICHEN>0 ? ANZAHL_ZEICHEN:1 )))+1;

   cout << "Die gelesenen Zeichen und ihre Anzahlen:" << endl;
   for(unsigned int zeichenIndex = 0   // Fuer jedes Zeichen den Zaehler geeignet ausgeben.
          ; zeichenIndex < ANZAHL_ZEICHEN
          ; ++zeichenIndex) {
      if (0 != gezaehlteZeichen[zeichenIndex]) {
         cout << "`";
         if ( isprint((char)zeichenIndex) ) {
            cout << (char)zeichenIndex;
         } else {
            cout << "<undruckbares Zeichen, Code "
                 << setw(BREITE_CODE) << zeichenIndex << ">";
         }
         cout << "': " << setw(BREITE_ANZAHL)
              <<gezaehlteZeichen[zeichenIndex] << endl;

         if ( isalpha((char)zeichenIndex) ) {
            anzahlBuchstaben += gezaehlteZeichen[zeichenIndex];
         }
         if ( isspace((char)zeichenIndex) ) {
            anzahlWhitespace += gezaehlteZeichen[zeichenIndex];
         }
      }
   }
   cout << "Anzahl Buchstaben: " << anzahlBuchstaben << endl;
   cout << "Anzahl Whitespace: " << anzahlWhitespace << endl;
   return 0;
}
```

Anmerkungen

Warum so wenige Kommentare?

Sie sollen sich hier auch im Lesen von C++-Code üben. Hier werden daher nur außergewöhnliche Konstruktionen erläutert und eigenartige Tricks komplett vermieden. Diese Fähigkeit werden Sie brauchen, denn das einzige, was immer mit dem Code übereinstimmt, ist der Code.

Außerdem ist der Platz auf Übungsblättern begrenzt.