

Einführung in die Programmiersprachen C und C++

Prof. Dr. Ulf Rehmann, Fakultät für Mathematik

Übungsblatt 9 (4 Seiten)

Als Beispiele zur Verwendung von Strukturen in C und zum Gebrauch des Manuals werden zwei Programme `crypt-test.c` und `crypt.c` gegeben (siehe Dateienverzeichnis), die die Passwort-Codierung in Unix erläutern. Nur eines der Programme ist hier wiedergegeben.

```

/* crypt-test.c */
#include <stdio.h>
#include <string.h>
#define XOPEN_SOURCE
#include <unistd.h>
#include <stdlib.h>

char passwd[20]; // zu verschluesseldes Passwort
char cpasswd[20]; // verschluesselttes Passwort
char cc[]="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789./";
char salt[3];

main() {
    int c;
    c = strlen(cc);
    while(1) {
        printf(" : ");
        scanf("%s",passwd);
        srand(salt[0]); // Eingabe
        salt[0] = cc[rand()%c]; // 'seed' fuer Zufallszahlen
        salt[1] = cc[rand()%c]; // Auswahl von 'salt'
        salt[2] = '\0';
        strcpy(cpasswd, crypt(passwd,salt) );
        printf("%s\n", cpasswd);
    }
}

```

Kommandozeilen-Argumente werden in C++ wie in C behandelt. Der Zugriff auf Dateien ist einfach: Durch den Befehl `ifstream from(argv[1])`; wird ein Objekt vom Typ `ifstream` (Eingabestrom) definiert, über den die Zeichen der `argv[1]` der Reihe nach mit den üblichen Mitteln wie `cin`, `cin.get()`, ... gelesen werden können. Dabei ist `from` ein frei gewählter Name. Die Anweisung `ifstream from(argv[1])`; ist analog zu einer Deklaration mit Initialisierung wie etwa `int anzahl = 3`; zu sehen: `ifstream` ist ein Datentyp, `from` das Objekt mit frei gewähltem Namen (wie `anzahl`, das initialisiert wird auf die Datei namens `argv[1]`). Ganz analog arbeitet der Befehl `ofstream to(argv[2])`; hier ist `to` das Objekt vom Typ `ofstream`, eine Datei, in die geschrieben wird.

```

// fileio.cc    Lesen und Schreiben von Dateien

#include <iostream.h>
#include <fstream.h>

void error(int i, char* s, char* t = "") {
    cerr << s << ' ' << t << '\n'; exit(i);
}

```

```

}

int main(int argc, char* argv[] ) {
    if (argc != 3) {
        cerr << "Falscher Aufruf, korrekter Aufruf:\n";
        error (1, argv[0], "Quelldatei Zieldatei");
    }
    ifstream from(argv[1]); // Deklaration eines ifstream namens from
                           // der aus der Datei namens argv[1] liest.
    if (!from) error(2, argv[1], "kann nicht zum Lesen geöffnet werden.\n");
    ofstream to(argv[2]);
    if (!to) error(3, argv[2], "kann nicht zum Schreiben geöffnet werden.\n");
    char c;
    while (from.get(c))
        to.put(c);
    from.close(); to.close(); exit(0);
}

```

Aufgabe 9.1: Testen Sie das komplizierte Programm `fileio`, indem Sie Befehle wie

geben, wobei "quelle" und "ziel" auch weglassen werden bzw. durch Namen nicht existenter Dateien

ersetzt werden. Die \$status-Variablen zeigt dann den durch exit zurückgegebenen int-Wert.

Aufgabe 9.2: Modifizieren Sie das Programm `fileio.cc` so, dass ein Aufruf des ausführbaren Programms

fileio folgendes tut:

.fileio (ohne Parameter): Programm liest von der Tastatur (cin) und schreibt die gelesenen Daten auf den Bildschirm (nach cout).

.fileio quelle (ein Parameter): Liest aus Datei quelle, schreibt den Inhalt nach cout,

.fileio quelle ziel (zwei Parameter): Liest aus Datei quelle, schreibt den Inhalt in die Datei ziel

quelle1, ..., quellen und schreibt alles in die Datei ziel, die Dateien quelle1, ... quellen werden also aneinandergelängt.

Aufgabe 9.3: Benutzen Sie die Programme `fileio.cc` und `quicksort()`, um ein Programm zu schreiben das die Wörter aus einer Textdatei liest und alphabetisch geordnet mit Häufigkeitsangabe ausgibt. (Hinweis: Benutzen Sie z. B. das Programm `cc.cc` von Blatt 1, um Wörter zu erkennen.)

Mit einfachen Ergänzungen kann man z. B. die Wechselkurs-Berechnungen von "euro.cc" in einer Datei,

B. namens `euro.prot`, mitprotokollieren:

Es sind nur die Zeilen

```
{\t ofstream proto("euro.prot"); }
```

```
...
```

```
proto << Euro << " EURO = " << DM << " DM\n";
```

zu ergänzen, sowie vielleicht der Hinweis

```
cout << "Konvertierung protokolliert nach Datei euro.prot\n";
```

und am Schluss `proto.close()`;

```
// euro.prot.cc
```

```
// Umwandlung von Euro in DM und umgekehrt
```

```
#include <iostream.h>
```

```
// Präprozessor-Anweisung
```

```
#include <fstream.h>
```

```
main() {
    // Deklaration u. Initialisierung
    const float faktor = 1.96583;
    float x, Euro, DM;
    // Deklaration
    char ch;
    // Deklaration
    ostream proto("euro.prot");

    while(1) {
        cout << "Bitte e (EUR0) oder d (DM) gefolgt von Betrag eingeben (e0=Abbruch): ";
        cin >> ch >> x;
        // Eingabe nach ch und x
        while (cin.fail()) {
            cin.clear(); cin.ignore(20,'\n');
            cout << "Fehlerhafte Eingabe: e (EUR0) oder d (DM) gefolgt von Betrag: ";
            cin >> ch >> x;
        }
        if (x==0) break;
        if (ch == 'e') {
            Euro = x; DM = x*faktor;
        }
        else {
            Euro = x/faktor; DM = x;
        }
        cout << Euro << " EURO = " << DM << " DM\n"; // Ausgabe ...
        proto << Euro << " EURO = " << DM << " DM\n";
    }
    cout << "Konvertierung protokolliert nach Datei euro.prot\n";
    proto.close();
}
```

Aufgabe 9.4: Schreiben Sie ein Programm, das Daten, die es zur Verbrennung braucht, aus einer Datei liest (z. B. eine Liste von Beträgen, die umzurechnen sind wie oben).

Zur Entspannung ein Spielzeug:

Ein Raumschiff soll auf dem Mond landen. Es hat eine bestimmte Treibstoffmenge, die es zum Bremsen verwenden soll. Möglichst soll die Landegeschwindigkeit gleich 0 m/sec sein. Jede Sekunde können Sie als Kommandant eine bestimmte Menge Treibstoff verbrauchen, um das Schiff abzubremsen, aber die Gravitationsbeschleunigung wirkt dem natürlich entgegen.

Die Zustände des Raumschiffes werden durch die Daten v (Geschwindigkeit in m/sec), h (Höhe über dem Mondboden in m), f (Treibstoff-Vorrat), b (Beschleunigung in m/sec²) gekennzeichnet, die natürlich im Sinne der objektorientierten Programmierung durch eine Klasse repräsentiert werden. Einige Member-Funktionen übernehmen die Zustandsanzeigen und die Steuerung des Raumschiffes nach Ihren Anweisungen, z. B. sorgt die Member-Funktion decel für die Verzögerung (deceleration) bei gegebener Bremsstreibstoffmenge.

Die wirklichen Abhängigkeiten von h und v von der Zeit t lauten

$$h(t) = h_0 + t v_0 + \frac{1}{2} a t^2; \quad v(t) = v_0 + a t.$$

decel rechnet also sozusagen im Sekunden-Takt.

Steiger ist, wer bei minimalem Treibstoffverbrauch mit Geschwindigkeit 0 die Höhe 0 erreicht.

Die Anzeige liefert auch die jeweils aktuelle Bremsverzögerung in Vielfachen von g, der Erdbeschleunigung. Harte Bremsstöße, die etwa mit 20 g arbeiten, führen vermutlich zur Zerstörung des Raumschiffes und der Besatzung und sind daher tunlichst zu vermeiden.

Aufgabe 9.5: Setzen Sie in Programm den Wert B auf 0 und simulieren Sie damit ein Apollo-Manöver eines Raumschiffes (Atlantis) an ein anderes (Mir).

```
// mondlandung.cc
#include <stdio.h>
#include <iostream.h>

#define B 1.62      /* m/sec Fallbeschleunigung Mond */
#define G 9.81     /* m/sec Fallbeschleunigung Erde */
#define GM 2*g

class state {
    double v, h, f, b; // Geschwindigkeit, Hoehe, Treibstoff-Vorrat, Beschleunigung
public:
    double height() { return h; }
    double fuel() { return f; }
    state (double vv, double hh, double ff) {
        v = vv; h = hh; f = (ff >= 0) ? ff : -ff; b = 0.0;
    }
    state decel(double ff) {
        ff = (ff >= 0) ? ff : -ff;
        if ( ff >= f ) ff = f;
        f = f - ff; b = ff - B; h = h + v + b/2; v = v + b;
    }
    void print() {
        if ( h > 0.0 )
            printf("%8.2lf %8.2lf %8.2lf %8.2lfg   :: ", v, h, f, b/G);
        else {
            if ( v <= -2.0 ) printf("Crash:\n");
            else printf("Landing:\n");
            printf("%8.2lf %8.2lf %8.2lf %8.2lfg   :: \n", v, h, f, b/G);
            printf("Aufprall entspricht einem Sturz aus %2.2lf m \n
            Hoehe auf die Erde.\n", v*v/(2*g) );
        }
    }
};

main() {
    state s = state(-106.0, 533.0, 140.0); double ff;
    printf("      v      h      f      b      :: Treibstoff ?\n");
    while ( s.height() > 0.0 ) {
        if ( s.fuel() > 0.0 ) {
            s.print(); cin >> ff;
        }
        else ff = 0.0;
        s.decel(ff);
    }
    s.print();
} /* Ein/Ausgabe:      v      h      f      b      :: Treibstoff ?
-106.00    533.00    140.00    0.00g   :: 10
-97.62     431.19    130.00    0.85g   :: 20
...
-23.96     15.66     45.00     0.85g   :: 10
Crash:
-15.68     -4.11     35.00     0.85g   ::
Aufprall entspricht einem Sturz aus 12.37 m Hoehe auf die Erde.
*/
```