Informatik I

Uebung 7

Gruppe A, ETZ E91

Patrick Boenzli

Lernziele

Die Studentin/der Student lernt:

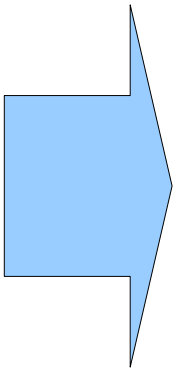
- wie man **ASCII Files einliest** (A1);
- wie man Zeilen in einem File in einzelne strings **zerlegt** (A1);
- wie man **binaere Files einliest** und verwendet (A2).

Aufgabe 1 – Zeilenumbruch

a) Problem verstehen

mit einer Zeilenbreite von 22 :

```
Wie lang' wird denn
das noch dauern? Ich
muß auf die Uhr
schauen... schickt
sich wahrscheinlich
nicht in einem so
ernsten Konzert.
```



mit einer Zeilenbreite von 10 :

```
Wie lang'
Wird denn
Das noch
dauern?
Ich muß
auf die
Uhr
schauen...
schickt
sich
wahrschein
lich nicht
```

Aufgabe 1 – Zeilenumbruch

b) Zerlegung der Aufgabe in Teilaufgaben:

1) Programm-**Argumente** Parsen (<Infile>, <Outfile>, <Zeilenlaenge>);

2) ASCII File **einlesen** (<Infile>) mit Hilfe von C++ Routinen (zeichenweise);

3) Zeile in Woerter zerteilen;

4) Woerter anordnen, Zeilenumbruch evaluieren (anhand der <Zeilenlaenge>);

5) Ergebnis in <Outfile> **schreiben**.

Aufgabe 1 – Zeilenumbruch

Program-Argumente Parsen

- Programmargumente werden im **Array** `argv[]` gespeichert
- `argc` enthaelt die **Anzahl** der Argumente

Beispiel auf Konsole:

```
 > ./test -n Fahrrad 5
```

`argc:` 4

`argv:`

test	-n	Fahrrad	5

```
// char einlesen
char* fahrZeug = argv[2];
// int einlesen (inkl. Parsing)
int   length   = atoi(argv[3]);
```

Aufgabe 1 – Zeilenumbruch

ASCII Files einlesen

- File Stream **deklarieren**
- Stream **oeffnen** (initialisieren)
- **Check**: sind wir schon am Ende des Files angelangt (EOF)?
- File kann nun Zeichen/Zeile fuer Zeichen/Zeile **eingelesen** werden
- Stream **schliessen**

Beispiel:

```
#include <iostream>

ifstream inText;                        // File Stream deklarieren
inText.open("beispielfile.txt");        // Stream oeffnen
if( !inText.eof()) {                    // eof checken:
    char zeichen = inText.get();        // ersten Character einlesen
}
inText.close();                         // File Stream schliessen
```

Aufgabe 1 – Zeilenumbruch

ASCII File schreiben

- File Stream **deklarieren**
- Stream **oeffnen** (initialisieren)
- File kann nun Zeichen/Zeile fuer Zeichen/Zeile **geschrieben** werden
- Stream **schliessen**

Beispiel:

```
#include <iostream>

ofstream outText;                       // File Stream deklarieren
outText.open("beispielfile.txt");       // Stream oeffnen

outText << "test test test\n";          // Daten schreiben

outText.close();                        // File Stream schliessen
```

Aufgabe 2 – Binary Files

a) Problem verstehen

<float>	<float>	<float>	<float>	...	<float>
Solarenergie Stunde 1	Verbrauch Stunde 1	Solarenergie Stunde 2	Verbrauch Stunde 2	...	Verbrauch Stunde 24
Bytes 0-3	Bytes 4-7	Bytes 8-11	Bytes 12-15	...	B. 188-191

<float>	<float>	...	...	...	...
Solarenergie Stunde 1	Verbrauch Stunde 1	...	...	...	...
B. 192-195	B 196-199	...	...	...	...

Aufgabe 2 – Binary Files

b) Zerlegung der Aufgabe in Teilaufgaben:

1) Binaere Files **oeffnen**

2) **Filegroesse** bestimmen

3) Speicher **allozieren**

4) File **einlesen**

Aufgabe 2 – Binary Files

Binaere Files oeffnen

Oeffnen von binaeren Files ist fast gleich wie das oeffnen von ASCII Files.

Beispiel:

```
#include <iostream>

ifstream inBinary;                          // File Stream deklarieren
inBinary.open("u.dat", ios::in|ios_base::binary);
                                           // Stream oeffnen


inBinary.close();                           // File Stream schliessen
```

Aufgabe 2 – Binary Files

Filegroesse bestimmen

Zaehlen der bytes in einem File mittels:
- read cursor bewegen: `f.seekg(0, ios::beg), seekg(0, ios::end)`
- position lessen: `f.tellg()`

Beispiel:

```
#include <iostream>

ifstream inBinary;
inBinary.open("u.dat", ios::in|ios_base::binary);

inBinary.seekg(0, ios::beg);
unsigned long startOffset = tellg();

inBinary.close();
```

Aufgabe 2 – Binary Files

File einlesen

Zum lesen wird die Funktion `read(<datapointer>, <size>)` verwendet

Beispiel:

```
#include <iostream>

ifstream inBinary;
inBinary.open( "u.dat", ios::in|ios_base::binary);

float* energy[24];
inBinary.read( (char*)&energy[0], sizeof(float));


inBinary.close();
```