

Übung 06: Informationstechnik (IT)

Marc Weber

Institutsleitung

Prof. Dr.-Ing. Dr. h.c. J. Becker

Prof. Dr.-Ing. E. Sax

Prof. Dr. rer. nat. W. Stork

Institut für Technik der Informationsverarbeitung (ITIV)



Teil 1: Besprechung Aufgabenblatt 6

Aufg. 6.01: Verständnisfragen Lsg. (1)

- a) Eine Datei ist aus der Sicht eines C++ Programms eine Folge von
- Bytes → Richtig
 - Datensätzen → Falsch
 - int-Werten → Falsch
- b) Die Methoden, Operatoren und Manipulatoren, die Sie im Zusammenhang mit `cin` und `cout` bereits verwendet haben, stehen auch für File-Streams zur Verfügung. → Richtig
- c) Wenn eine Datei im Default-Modus, d.h. ohne explizite Angabe eines Eröffnungsmodus eröffnet wird, ist die aktuelle Dateiposition
- nicht festgelegt → Falsch
 - der Dateianfang → Richtig
 - das Dateieinde → Falsch

Aufg. 6.01: Verständnisfragen Lsg. (2)

- d) Zur Abfrage, ob beim Lesen das Dateiende bereits erreicht wurde, kann die Methode eof () aufgerufen werden.
- e) Nennen Sie einen Vorteil und einen Nachteil einer verketteten Liste gegenüber einem dynamisch angelegten Array.
Eine verkettete Liste kann einfach in der Größe dynamisch verändert werden, wohingegen ein Array weniger Speicherplatz belegt.
- f) Beim Einfügen und Löschen eines Elements in einer verketteten Liste werden keine Elemente verschoben, sondern lediglich Zeiger versetzt.
- g) Die C++ Standard Template Library (STL) ist eine Sammlung von Klassen Templates, die es dem Programmierer erlauben einfache Standarddatenstrukturen und Algorithmen zu verwenden. → Richtig

Aufg. 6.01: Verständnisfragen Lsg. (3)

- h) Die STL bietet „Container“, welche nur bei bestimmten Datentypen angewendet werden können. → Falsch
- i) Nennen Sie 3 Klassen der STL.
map, list, stack, queue,
- j) Erklären Sie kurz die Begriffe FIFO und LIFO in Bezug auf die STL.
FIFO-Prinzip (First In, First Out): Lesen nur in selber Reihenfolge wie beim Schreiben
LIFO-Prinzip (Last In, First Out): Lesen nur in umgekehrter Reihenfolge wie beim Schreiben

Aufg. 6.01: Verständnisfragen Lsg. (4)

- k) Welches Prinzip wird in einer Warteschlange (Queue) verwendet?
Beschreiben Sie dieses kurz.

FIFO Prinzip (First-In First-Out), in einer Warteschlange kann eine beliebige Anzahl von Objekten gespeichert werden, die gespeicherten Objekte können nur in der gleichen Reihenfolge wieder gelesen werden, wie sie gespeichert wurden.

- Aufg. 6.01: Verständnisfragen



Aufg. 6.02: Dateiverarbeitung

- Erstellen Sie ein Programm, das als Memoblock verwendet werden kann. Dabei soll das Programm ein Menü bereitstellen, im welchem der Benutzer auswählen kann, ob er seine bisherigen Memos lesen oder eine neue Memo hinzufügen möchte.
 - Beim Hinzufügen soll das Programm die Datei „memo.txt“ zum Schreiben ab Dateiende öffnen. Wenn die Datei nicht vorhanden ist, soll sie angelegt werden. Anschließend soll das Programm eine Zeile vom Benutzer abfragen und diese mit einem führenden Zeitstempel (siehe Hinweis) in die Datei schreiben. Anschließend wird die Datei wieder geschlossen.
 - Beim Lesen soll die Datei „memo.txt“ geöffnet werden, jede Zeile aus der Datei gelesen werden und mit einer führenden Zeilennummer auf dem Bildschirm ausgegeben werden. Die Zeile enthält dabei den Zeitstempel und den jeweiligen Text. Anschließend wird die Datei wieder geschlossen.
 - Falls ein Fehler beim Dateizugriff auftritt, soll dieser mit einer entsprechenden Fehlermeldung auf dem Bildschirm angezeigt werden.
- Hinweis: Die aktuelle Zeit als String erhalten Sie durch den Aufruf der Standardfunktion `time()` und `ctime_s()` der Bibliothek `<ctime>`.

Aufg. 6.02: Dateiverarbeitung Lsg. (1)

```
#include <iostream>
#include <fstream>
#include <string>
#include <ctime>
```

Zur Ermittlung / Ausgabe der Zeit

```
using namespace std;
```

Speicherung des Dateinamens

```
class MemoBlock{
```

```
private:
```

Fehlerfunktionen

```
    string filename;
```

```
    inline void openError();
```

```
    inline void writeError();
```

```
    inline void readError();
```

```
    void schreiben();
```

Eingabe und speichern eines Memos

```
    void lesen();
```

Lesen aller Memos

```
    void loeschen();
```

Löschen aller Memos

```
public:
```

```
    MemoBlock();
```

```
    ~MemoBlock();
```

```
    void setfilename( string _filename );
```

```
    void menu();
```

Benutzermenü

```
};
```

Aufg. 6.02: Dateiverarbeitung Lsg. (2)

■ Fehlerfunktionen

Jeweils Fehlermeldung ausgeben und Programm beenden

```
inline void MemoBlock::openError() {  
    cerr << "Fehler beim Oeffnen der Datei " << filename << endl;  
    exit( 1 );  
}
```

```
inline void MemoBlock::writeError() {  
    cerr << "Fehler beim Schreiben in die Datei " << filename << endl;  
    exit( 2 );  
}
```

```
inline void MemoBlock::readError() {  
    cerr << "Fehler beim Lesen der Datei " << filename << endl;  
    exit( 3 );  
}
```

Aufg. 6.02: Dateiverarbeitung Lsg. (3)

```

void MemoBlock::schreiben() {
    string line;
    char zeit [26];
    fstream memoFile( filename.c_str(), ios::out | ios::app );
    if( !memoFile ) {
        openError();
    }

    time_t sec = time( NULL );
    ctime_s( zeit, 26, &sec );
    zeit[24] = '\0';

    cout << "Geben Sie Ihre Notiz ein:\n";
    getline( cin, line );

    if( !( memoFile << zeit << ": " << line << endl ) ) {
        writeError();
    }

    memoFile.close();
}

```

Einlesen und Abspeichern eines Memos

Datei zum Schreiben „ab Dateende“ öffnen

Fehler beim Öffnen?

Ermittlung und Umwandlung der Zeit in C-String

Löschen des '`\n`'-Characters des C-Strings

Notiz vom User abfragen

Zeit und Memo in die Datei schreiben mit Fehlerabfrage

Datei schließen

Aufg. 6.02: Dateiverarbeitung Lsg. (4)

```
void MemoBlock::lesen() {  
    string line;  
    int number = 0;  
  
    ifstream memoFile( filename.c_str(), ios::in );  
  
    if( !memoFile ) {  
        openError();  
    }  
  
    while( getline( memoFile, line ) ) {  
        cout.width( 5 );  
        cout << ++number << ": " << line << endl;  
    }  
  
    if( !memoFile.eof() ) {  
        readError();  
    }  
  
    memoFile.close();  
}
```

Lesen aller Memos

Datei zum Lesen öffnen

Fehler beim Öffnen?

Lesen einer Zeile aus der Datei

Ausgabebreite für die Nummer festlegen

Nummer und gelesene Zeile ausgeben

Fehler beim Lesen? (nicht am Dateiende)

Datei schließen

Aufg. 6.02: Dateiverarbeitung Lsg. (5)

```
void MemoBlock::loeschen() {  
    fstream memoFile( filename.c_str(), ios::out );
```

Inhalt der Datei löschen (nicht Datei selbst)

```
    if( !memoFile ) {  
        openError();  
    }
```

Datei zum Schreiben (ab Dateianfang) öffnen und wieder schließen

```
    memoFile.close();
```

```
}
```

```
MemoBlock::MemoBlock() {  
}
```

Leerer Konstruktor

```
MemoBlock::~MemoBlock() {  
}
```

Leerer Destruktor

```
void MemoBlock::setfilename( string _filename ) {  
    filename = _filename;  
}
```

Festlegen des Dateinamen (privates Attribut)

Aufg. 6.02: Dateiverarbeitung Lsg. (6)

```

void MemoBlock::menu() {
    char auswahl = 'a';

    do {
        cout << endl;
        cout << "Memoblock" << endl;
        cout << "-----" << endl << endl;
        cout << "(N)eues Memo hinzufuegen" << endl;
        cout << "(V)orhandene Memos ausgeben" << endl;
        cout << "(L)oechen der Memodatei" << endl;
        cout << "(E)nde" << endl;
        cout << "bitte auswaehlen:";
        cin >> auswahl;
        cin.sync();
        cin.ignore();

        switch( auswahl ) {
            case 'N':
            case 'n': schreiben(); break;
            case 'V':
            case 'v': lesen(); break;
            case 'L':
            case 'l': loeschen(); break;
            case 'e':
            case 'E': break;
            default: cout << "Keine gueltige Eingabe!";
        }
    } while( auswahl != 'e' && auswahl != 'E' );
}
  
```

Alles läuft in der Schleife bis zur Eingabe von 'e' oder 'E'

Menü ausgeben

Ignorieren des '\n', welches noch im Eingabepuffer steht (wichtig für späteres getline())

Nach der Auswahl entsprechend die Methoden aufrufen

Fehlermeldung für ungültige Eingabe

Aufg. 6.02: Dateiverarbeitung Lsg. (7)

```
int main() {
```

```
    MemoBlock Memo;
```

Objekt der Klasse anlegen

```
    Memo.setfilename( "memo.txt" );
```

Dateinamen festlegen

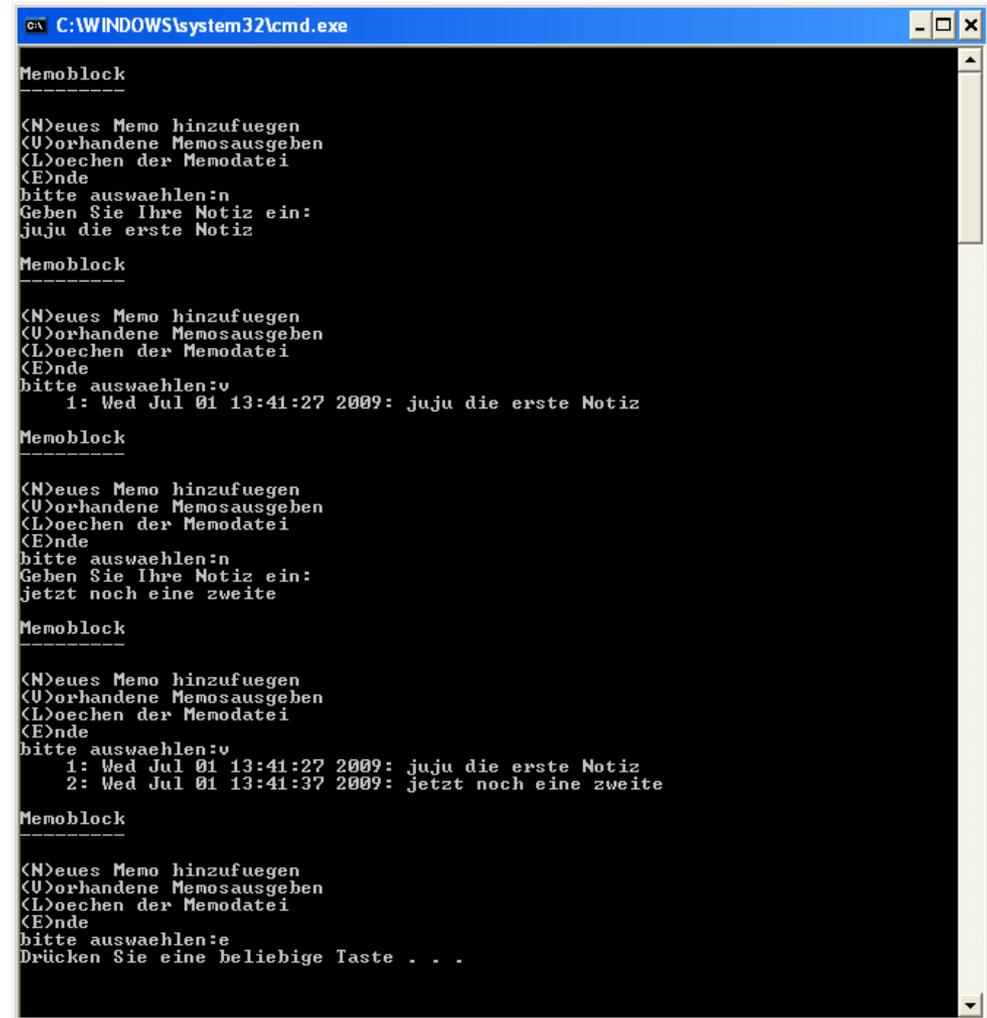
```
    Memo.menu();
```

Menü starten

```
    return 0;
```

```
}
```

Hinweis: mit dem Befehl
`system("cls");` kann man
 die Bildschirmausgabe löschen



```

C:\WINDOWS\system32\cmd.exe
Memoblock
-----
(N)eues Memo hinzufuegen
(U)orhandene Memosausgeben
(L)oechen der Memodatei
(E)nde
bitte auswaehlen:n
Geben Sie Ihre Notiz ein:
juju die erste Notiz

Memoblock
-----
(N)eues Memo hinzufuegen
(U)orhandene Memosausgeben
(L)oechen der Memodatei
(E)nde
bitte auswaehlen:v
1: Wed Jul 01 13:41:27 2009: juju die erste Notiz

Memoblock
-----
(N)eues Memo hinzufuegen
(U)orhandene Memosausgeben
(L)oechen der Memodatei
(E)nde
bitte auswaehlen:n
Geben Sie Ihre Notiz ein:
jetzt noch eine zweite

Memoblock
-----
(N)eues Memo hinzufuegen
(U)orhandene Memosausgeben
(L)oechen der Memodatei
(E)nde
bitte auswaehlen:v
2: Wed Jul 01 13:41:37 2009: jetzt noch eine zweite

Memoblock
-----
(N)eues Memo hinzufuegen
(U)orhandene Memosausgeben
(L)oechen der Memodatei
(E)nde
bitte auswaehlen:e
Drücken Sie eine beliebige Taste . . .
  
```

- Aufg. 6.02: Dateiverarbeitung



Aufg. 6.03: Studentendatenbank

- In dieser Aufgabe soll eine Studentendatenbank in Form einer verketteten Liste unter Zuhilfenahme der STL-Klasse `list` aufgebaut werden. Verwenden Sie zudem Iteratoren. Die elementare `Student` Datenstruktur (Objekt einer Klasse) soll folgende Attribute über den jeweiligen Studenten beinhalten: Vorname, Nachname, Matrikelnummer und Note.

```

C:\WINDOWS\system32\cmd.exe
Studentendatenbank
=====
<1> Eintrag hinzufuegen
<2> Eintrag suchen
<3> Eintrag loeschen
<4> Datenbank ausgeben
<5> Ende
Bitte auswaehlen:
  
```

```

C:\WINDOWS\system32\cmd.exe
Studentendatenbank
=====
<1> Eintrag hinzufuegen
<2> Eintrag suchen
<3> Eintrag loeschen
<4> Datenbank ausgeben
<5> Ende
Bitte auswaehlen: 4

      Name      Vorname  Matrikelnr      Note
-----
      Zander      Adam      568235          2.3
      Mueller     Peter     765328          1.7

Studentendatenbank
=====
<1> Eintrag hinzufuegen
<2> Eintrag suchen
<3> Eintrag loeschen
<4> Datenbank ausgeben
<5> Ende
Bitte auswaehlen:
  
```

Aufg. 6.03: Studentendatenbank Lsg. (1)

//Klasse zum Speichern der Daten eines Studenten

```
class Student {  
    private:  
        string name;  
        string vorname;  
        int matrikelnr;  
        double note;  
  
    public:  
        Student();  
        ~Student();  
  
        const string& getName() { return name; };  
        const string& getVorname() { return vorname; };  
        int getMatrikelnr(){ return matrikelnr; };  
        double getNote() { return note; };  
  
        void setData( string vorname, string name, int matrikelnr, double note );  
};
```

student.h

Attribute zum Speichern der
spezifischen Daten

get-Methoden

Aufg. 6.03: Studentendatenbank Lsg. (2)

student.cpp

```
Student::Student() {  
    name = "";  
    vorname = "";  
    matrikelnr = 0;  
    note = 0;  
}
```

Konstruktor

Initialisierung der Attribute

```
Student::~~Student() {  
}
```

Leerer Destruktor

```
void Student::setData( string vorname, string name, int matrikelnr, double note ) {  
    this->name = name;  
    this->vorname = vorname;  
    this->matrikelnr = matrikelnr;  
    this->note = note;  
}
```

Einzelne Set-Methode für alle Werte

Zugriff auf Klassenattribute über `this`-Zeiger

Aufg. 6.03: Studentendatenbank Lsg. (3)

```

//Klasse zum Speichern der Datenbank
#include "student.h"
#include <list>

class Studentenverwaltung {
private:
    list<Student> studentenliste;

public:
    void addStudent( const string& vorname, const string& name,
                    int matr, double note );
    void deleteStudent( list<Student>::iterator it );
    void printStudent( list<Student>::iterator it ) const;
    void print();
    list<Student>::iterator findStudent( int matrikel );
};
  
```

studentenverwaltung.h

Einbinden der `<list>` STL Bibliothek

`list` Container als Datenbank

Methoden zum Verwalten der Datenbank

Gibt Iterator auf den gefundenen Studenten zurück. Iterator „zeigt“ auf Student

Aufg. 6.03: Studentendatenbank Lsg. (4)

studentenverwaltung.cpp

```
void Studentenverwaltung::addStudent( const string& vorname, const string& name,  
                                     int matr, double note ) {
```

```
    Student s;
```

```
    s.setData( vorname, name, matr, note );
```

```
    studentenliste.push_back( s );
```

```
}
```

Daten dem neuem Element zuweisen

Student am Ende der Liste hinzufügen

```
void Studentenverwaltung::deleteStudent( list<Student>::iterator it ) {
```

```
    if( it != studentenliste.end() ){
```

```
        studentenliste.erase(it);
```

```
    } else {
```

```
        cout << "Student nicht gefunden" << endl;
```

```
    }
```

```
}
```

Student auf den Iterator `it` zeigt aus der Liste löschen, wenn `it` nicht auf Listenende zeigt (`end()`)

Aufg. 6.03: Studentendatenbank Lsg. (5)

studentenverwaltung.cpp

```
list<Student>::iterator Studentenverwaltung::findStudent( int matrikel ) {
    list<Student>::iterator it;
    for ( it = studentenliste.begin(); it != studentenliste.end(); ++it ) {
        if ( it->getMatrikelnr() == matrikel ) {
            return it;
        }
    }
    return studentenliste.end();
}
```

Elemente nacheinander durchgehen

Matrikelnummer überprüfen

Student nicht gefunden → Rückgabe des Iterators auf Listenende

```
void Studentenverwaltung::printStudent( list<Student>::iterator it ) const {
    if ( it != studentenliste.end() ) {
        cout << "Vorname: " << it->getVorname() << ", Name: "
            << it->getName() << ", Matrikel-Nr.: " << it->getMatrikelnr()
            << ", Note: " << it->getNote() << endl;
    } else {
        cout << "Student nicht gefunden" << endl;
    }
}
```

Attribute des Studenten an Position *it* ausgeben

Aufg. 6.03: Studentendatenbank Lsg. (6)

studentenverwaltung.cpp

```

void Studentenverwaltung::print () const {
    cout << endl << setw(12) << "Name" << setw(12) << "Vorname" << setw(12)
        << "Matrikelnr" << setw(12) << "Note" << endl;
    cout << "-----" << endl;

    if ( ! studentenliste.empty() ) {
        for ( it = studentenliste.begin(); it != studentenliste.end(); ++it ) {
            cout << setw(12) << it->getName() << setw(12)
                << it->getVorname() << setw(12)
                << it->getMatrikelnr() << setw(12)
                << it->getNote() << endl;
        }
    } else {
        cout << "Die Datenbank ist leer" << endl;
    }
}

```

Tabellenkopf ausgeben

Liste nicht leer?

Daten-Attribute jedes Studenten ausgeben

Aufg. 6.03: Studentendatenbank Lsg. (7)

```

int main() {
  Studentenverwaltung* database = new Studentenverwaltung;
  int auswahl;

  do {
    cout << endl;
    cout << "Studentendatenbank" << endl;
    cout << "-----" << endl << endl;
    cout << "(1) Eintrag hinzufuegen" << endl;
    cout << "(2) Eintrag suchen" << endl;
    cout << "(3) Eintrag loeschen" << endl;
    cout << "(4) Datenbank ausgeben" << endl;
    cout << "(5) Ende" << endl;
    cout << "Bitte auswaehlen: ";
    if ( cin >> auswahl ) {
      switch( auswahl ) {
        case 1: daten_hinzufuegen( database ); break;
        case 2: daten_suchen( database ); break;
        case 3: daten_loeschen( database ); break;
        case 4: daten_ausgabe( database ); break;
        case 5: break;
        default: cout << "Ihre Eingabe ist falsch, die Option steht nicht zur Verfuegung" << endl;
      }
    } else {
      cout << "Sie haben keine Zahl eingegeben!" << endl;
    }
    cin.clear();
    cin.ignore( numeric_limits<streamsize>::max(), '\n' );
  } while( auswahl != 5 );
  delete database;
}

```

main.cpp

Ausgabe des Menüs

Einlesen der Auswahl des Benutzers

Je nach der Auswahl des Benutzers wird die entsprechende Funktion aufgerufen

Bei Fehleingabe wird eine Fehlermeldung ausgegeben

Wichtig: Eingabepuffer nach Fehleingabe leeren (Einbinden von `<limits>`)

Abbruchbedingung der `while`-Schleife

Freigeben des Datenbankspeichers

Aufg. 6.03: Studentendatenbank Lsg. (8)

```
void daten_hinzufuegen( Studentenverwaltung* database ) {  
    string name;  
    string vorname;  
    int matrikelNr;  
    double note;  
  
    cout << endl << "Bitte die neuen Daten eingeben" << endl;  
    cout << "Name: ";  
    cin >> name;  
    cout << "Vorname: ";  
    cin >> vorname;  
    cout << "Matrikelnummer: ";  
    cin >> matrikelNr;  
    cout << "Note: ";  
    cin >> note;  
  
    database->addStudent( vorname, name, matrikelNr, note );  
}
```

main.cpp

Einlesen der Daten von der Tastatur

Aufruf der Methode zum Hinzufügen
der Daten zur verketteten Liste

Aufg. 6.03: Studentendatenbank Lsg. (9)

```
void daten_suchen( Studentenverwaltung* database ) {  
    int matrikelNr;  
  
    cout << "Bitte zu suchende Matrikelnummer eingeben: ";  
    cin >> matrikelNr;  
  
    list<Student>::iterator gesucht = database->findStudent( matrikelNr );  
    database->printStudent( gesucht );  
}
```

main.cpp

Student mit entsprechender Matrikelnummer
in der Liste vorhanden?

Ausgabe der Daten des gefundenen
Studenten durch Delegation an die Methode
der Datenbank

Aufg. 6.03: Studentendatenbank Lsg. (10)

```
void daten_loeschen( Studentenverwaltung* database ){  
    int matrikelNr;
```

main.cpp

```
    cout << "Bitte zu loeschende Matrikelnummer eingeben: ";  
    cin >> matrikelNr;
```

Student mit entsprechender Matrikelnummer
in der Liste vorhanden?

```
    list<Student>::iterator gesucht = database->findStudent( matrikelNr );  
    database->deleteStudent( gesucht );  
}
```

Löschen des gefundenen Elements

```
void daten_ausgabe( Studentenverwaltung* database ) {  
    database->print();  
}
```

Weiterleitung der Methode an die Datenbank

- Aufg. 6.03: Studentendatenbank



Aufg. 6.04: STL Dictionary

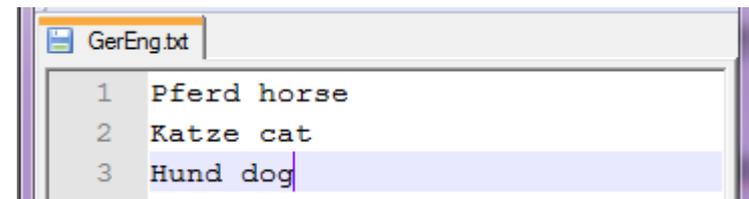
- Programmieren Sie unter Zuhilfenahme der STL-Klasse `map` ein Wörterbuch englisch-deutsch / deutsch-englisch. Verwenden Sie hierbei **Iteratoren**. Der Nutzer gibt ein Suchwort ein und dazu wird die passende Übersetzung angezeigt. Immer wenn der Nutzer ein Suchwort eingibt, das nicht im Wörterbuch steht, bekommt er die Gelegenheit, das Wort zusammen mit der Übersetzung in das Wörterbuch aufzunehmen. Beim Beenden des Programms sollen alle Wortpaare in einer Textdatei gespeichert werden. Zu Beginn des Programms sollen alle Wortpaare aus dieser Textdatei eingelesen werden, falls sie bereits existiert.

Aufg. 6.04: STL Dictionary Lsg. (1)

- Speicherung während der Laufzeit des Programms
- Speicherung nach Beendigung des Programms
- `map<key, value>`
 - `key` = deutsches Wort
 - `value` = englisches Wort
 - Zugriff über Iterator
- Textdatei - Protokoll
 - Ein Wortpaar pro Zeile
 - 1. deutsches Wort
 - 2. Leerzeichen
 - 3. englisches Wort
 - 4. end line
- Beispiel:

```
map<string, string> myMap;
myMap["Hund"] = "dog";
map<string, string>::iterator
  myIterator;
myIterator = myMap.begin();
cout << myIterator->first; // "Hund"
cout << myIterator->second; // "dog"
```

- Beispiel:



Aufg. 6.04: STL Dictionary Lsg. (2)



TXT

Textdatei

Interaktion
mit Textdatei



- lesen der Textdatei
- speichern in **map**
- suchen in **map**
- speichern in Textdatei



Benutzer

Interaktion
mit Benutzer



verwaltet



- übergibt Suchbegriff und neue Wortpaare an Dictionary
- gibt Übersetzung aus

Aufg. 6.04: STL Dictionary Lsg. (3)

```
class Dictionary {  
  
private:  
    map<string, string> gerEngMap;  
    map<string, string> engGerMap;  
    string filename;  
  
public:  
    Dictionary( string filename );  
    string searchGerman( string searchString );  
    string searchEnglish( string searchString );  
    void add( string germanWord, string englishWord );  
    void readFile();  
    void saveToFile();  
};
```

dictionary.h

map<deutsch, englisch>

map<englisch, deutsch>

Aufg. 6.04: STL Dictionary Lsg. (4)

dictionary.cpp

```

Dictionary::Dictionary( string _filename ) {
    filename = _filename;
    readFile();
}

void Dictionary::readFile() {
    ifstream myFile( filename.c_str(), ios::in );
    if( myFile ) {
        string germanWord, englishWord;
        while( myFile >> germanWord >> englishWord ){
            gerEngMap[germanWord] = englishWord;
            engGerMap[englishWord] = germanWord;
        }
        myFile.clear();
    } else {
        myFile.open( filename.c_str(), ios::out );
    }
    myFile.close();
}

```

öffne Datei zum Lesen

falls Datei existiert

lese alle Wertepaare

lösche error flags

falls Datei noch nicht existiert

erzeuge Datei

schließe Datei

Aufg. 6.04: STL Dictionary Lsg. (5)

```
string Dictionary::searchEnglish( string searchString ) {
    map<string, string>::iterator mapIterator;
    mapIterator = engGerMap.find( searchString );

    if ( mapIterator != engGerMap.end() ) {
        return mapIterator->second;
    }
    else {
        return "not in map";
    }
}
```

```
string Dictionary::searchGerman( string searchString ) {
    map<string, string>::iterator mapIterator;
    mapIterator = gerEngMap.find( searchString );

    if ( mapIterator != gerEngMap.end() ) {
        return mapIterator->second;
    } else {
        return "not in map";
    }
}
```

dictionary.cpp

passender Iterator

`find(..)` gibt Position des Suchbegriffs (**key**) oder `end()` zurück

Aufg. 6.04: STL Dictionary Lsg. (6)

dictionary.cpp

```
void Dictionary::add( string germanWord, string englishWord ) {  
    gerEngMap[germanWord] = englishWord;  
    engGerMap[englishWord] = germanWord;  
}
```

```
void Dictionary::saveToFile() {  
    ofstream myFile( filename.c_str(), ios::out );  
    map<string, string>::iterator mapIterator;  
  
    for( mapIterator = gerEngMap.begin();  
         mapIterator != gerEngMap.end();  
         ++mapIterator ) {  
        myFile << mapIterator->first << " " << mapIterator->second << endl;  
    }  
    myFile.close();  
}
```

`begin()` zeigt auf das 1. Element der map

`end()` auf das Ende der map und **nicht** auf das letzte Element

über `first` und `second` wird der `key` bzw. der `value` ausgelesen und durch ein Leerzeichen getrennt in eine Zeile geschrieben

Aufg. 6.04: STL Dictionary Lsg. (7)

```
class Translator{  
public:  
    Translator( string _filename );  
    ~Translator();  
    void menu();  
    void save();  
private:  
    string filename;  
    Dictionary* myDictionary;  
    void search();  
};
```

translator.h

Aufg. 6.04: STL Dictionary Lsg. (8)

```
Translator::Translator( string _filename ) {  
    myDictionary = new Dictionary( _filename );  
}  
  
void Translator::save() {  
    myDictionary->saveToFile();  
}  
  
Translator::~~Translator(){  
    /*Beim Zerstören des Translators auch Speicher  
    * des Dictionary wieder freigeben  
    */  
    delete myDictionary;  
}
```

translator.cpp

Auch Speicher des Dictionary
wieder freigeben!

Aufg. 6.04: STL Dictionary Lsg. (9)

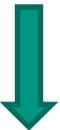
translator.cpp

```
void Translator::search() {
    string searchWord;
    bool wordFound = false;
    cout << "Bitte deutsches oder englisches Suchwort eingeben: ";
    getline( cin, searchWord );
    cin.sync();
    cout << endl;

    string germanTranslation=myDictionary->searchEnglish( searchWord );
    string englishTranslation=myDictionary->searchGerman( searchWord );

    if( germanTranslation != "not in map" ) {
        cout << "Die deutsche \x9A \bbersetzung lautet: " << germanTranslation << endl;
        wordFound = true;
    }

    if( englishTranslation != "not in map" ) {
        cout << "Die englische \x9A \bbersetzung lautet: " << englishTranslation << endl;
        wordFound = true;
    }
}
```



Aufg. 6.04: STL Dictionary Lsg. (10)

translator.cpp

```
if( !wordFound ) {
    char add = 'a';
    string germanWord, englishWord;
    do {
        cout << "Begriff nicht im W\u00f6rterbuch! Hinzuf\u00fcgen? (y/n)";
        cin >> add;
        cin.ignore();
        cout << endl;
    } while( ( add != 'y' ) && ( add != 'n' ) );

    if( add == 'y' ) {
        cout << "Bitte das deutsche Wort eingeben: ";
        getline( cin, germanWord );
        cin.sync();
        cout << "Bitte das englische Wort eingeben: ";
        getline( cin, englishWord );
        cin.sync();
        myDictionary->add( germanWord, englishWord );
        cout << "Neues Wortpaar hinzugef\u00fcgt" << endl;
    }
}
```

Aufg. 6.04: STL Dictionary Lsg. (11)

translator.cpp

```

void Translator::menu() {
    char auswahl = 'a';
    do {
        cout << endl << "Wörterbuch" << endl;
        cout << "-----" << endl << endl;
        cout << "(S)uche \bbersetzung" << endl;
        cout << "(E)nde" << endl << endl;
        cout << "Ihre Wahl: ";
        cin >> auswahl;
        cin.ignore();
        switch( auswahl ) {
            case 'S':
            case 's':
                search();
                system( "pause" );
                break;
            case 'e':
            case 'E': break;
            default:
                cout << "Keine gültige Eingabe!" << endl;
                system( "pause" );
        }
        system( "cls" );
    } while( auswahl != 'e' && auswahl != 'E' );
}
  
```

```

Wörterbuch
-----
(S)uche Übersetzung
(E)nde
Ihre Wahl: _
  
```

Überprüfung, ob es beim Einlesen einen Fehler gab, fehlt hier (und an manchen anderen Stellen)!

Aufg. 6.04: STL Dictionary Lsg. (12)

```
int main() {  
    Translator* myTranslator;  
    myTranslator = new Translator( "GerEng.txt" );  
    myTranslator->menu();  
    myTranslator->save();  
    delete myTranslator;  
    return 0;  
}
```

main.cpp

Speicher des dynamisch
erzeugten Translators freigeben!

Der hier gezeigte Quellcode ist nicht 100% vollständig → Alle Quellcode-Dateien zu allen Aufgaben finden Sie wie immer auf der Lernplattform ILIAS

Aufg. 6.04: STL Dictionary Lsg. (13)

```
Wörterbuch
-----
<S>uche Übersetzung
<E>nde

Ihre Wahl: s
Bitte deutsches oder englisches Suchwort eingeben: Hund
Die englische Übersetzung lautet: dog
Drücken Sie eine beliebige Taste . . . _
```

Begriff ist bereits im Wörterbuch

```
Wörterbuch
-----
<S>uche Übersetzung
<E>nde

Ihre Wahl: s
Bitte deutsches oder englisches Suchwort eingeben: Katze
Begriff nicht im Wörterbuch! Hinzufügen? <y/n>y
Bitte das deutsche Wort eingeben:Katze
Bitte das englische Wort eingeben:cat
Neues Wortpaar hinzugefügt
Drücken Sie eine beliebige Taste . . .
```

Begriff nicht im Wörterbuch und wird auf Nachfrage hinzugefügt

- Aufg. 6.04: STL Dictionary



Fragen?

Vielen Dank für Ihre Aufmerksamkeit



Marc Weber
Karlsruher Institut für Technologie (KIT) – ITIV
marc.weber3@kit.edu