Übungen zu EDV für Physikerinnen und Physiker (physik131) WS 2011/2012

Jörg Pretz und Daniel Elsner

8. Übung Woche: 5.12.-9.12.2011

Lernziele

Programmieren in C++: Klassen, Zeiger (Pointer)

Präsenzübungen

Anwendung der String Klasse

Die C++ Standardbibliothek stellt eine *string* Klasse zum Arbeiten mit Zeichenketten wie z.B. `Hello World` zur Verfügung. Schreiben Sie den unten aufgeführten Programmcode in eine Datei. Kompilieren Sie das Programm und führen Sie es aus. Versuchen Sie zunächst die einzelnen Befehle und die Ausgabe dieses Programms nachzuvollziehen.

```
#include <iostream>
#include <string>
using namespace std;

int main()
{

string s1 = "abc";
string s2 = "def";
string s;

cout << s1.length() << endl;

s=s1+s2;
cout << "s= "<< s << endl;

s1.append(s2);
cout << "s1= "<< s1 << endl;

string s_sub = s.substr(1,4);
cout << s_sub << endl;

return(0);
}
```

Wie Sie sehen ist in dieser Klasse für den Umgang mit strings ein + Operator definiert. Darüberhinaus hängt der Operator += eine Kopie des Argumentes an den string links vom Operator. Das gleiche Ergebnis erhalten Sie auch wenn Sie die *append* Funktion benutzen. Machen Sie sich unter http://www.cplusplus.com/reference/string/string/ grob mit der string Klasse vertraut. Erweitern Sie nun das Programm indem Sie

- überprüfen ob `s, s1, s2` gleich sind
- die Buchstaben `cd` im String durch `ZZ` ersetzen
- die Position ausgeben an der der Substring `ef` steht.

Klassen selber schreiben

In der Vorlesung wurde das Konzept der Klassen im Allgemeinen und in C++ im Besonderen besprochen. Die Klasse (class) ist die zentrale Datenstruktur in C++. Sie kapselt zusammengehörige Daten und Funktionen vom Rest des Programmes ab. Mit anderen Worten ist eine Klasse ein Bauplan, welcher die Attribute und Methoden definiert, die alle Objekte einer bestimmten Art besitzen. Im Folgenden machen Sie sich mit einem Beispielprogramm vertraut, welches Sie dann erweitern werden. Auf **eCampus** finden Sie die Quelldateien zu einem Programm welches die Eigenschaften von Studenten in der Klasse Student zusammenfassen soll. In den Dateien selbst ist eine kleine Dokumentation (zeilenweise) eingebaut.

- `student.h` Hier sind alle Deklarationen der Klasse Student zusammengefasst, d.h. hier werden ihre Elemente (members, das umfasst Daten und Funktionen) festgelegt.
- `student.cpp` Die Implementierung, d.h. der Code für die jeweiligen Memberfunktionen, erfolgt in einer CPP-Datei.
- `main.cpp` Das Hauptprogramm welches nun auf die Klasse Student zugreifen kann.

Die Klasse Student besitzt bisher die Funktionalität den Vor- und Nachnamen in Membervariablen zu speichern und aus dem angegebenen Geburtsdatum das Alter des Studenten zu berechnen (nicht exakt, da die Schaltjahre vernachlässigt werden.) Des Weiteren wurde bei den Eingabeoperationen der Übersichts halber darauf verzichtet, die Sinnhaftigkeit der Eingabewerte zu überprüfen (z.B. kein negatives Geburtsdatum usw.)

Nachdem Sie sich mit den drei Dateien vertraut gemacht haben, probieren Sie folgende Funktionalitäten zu ergänzen.

1. Analog dem Vorgehen mit dem Vornamen, fragen Sie nun den Nachnamen ab, speichern Sie diesen in einer Membervariable (bereits im Headerfile deklariert) und fragen Sie den neuen vollen Namen ab (bereits angelegt).
2. Ergänzen Sie eine Funktion mit der Sie das Geburtsdatum aus den Membervariablen (geb_tag, geb_monat, geb_jahr) auslesen können (z.B. getGeburtstag()).
3. Erweitern Sie die Klasse um eine weitere nützliche Information (z.B. Matrikelnummer oder Studienfach). Hierfür legen Sie ein neue Membervariable an, und Funktionen um diese zu setzen und auszulesen. Denken Sie daran einen Defaultwert Kontruktor zu setzen.

Zum compilieren des Programmes gehen Sie folgendermassen vor:

```
g++ -c student.cpp
g++ -c main.cpp
g++ student.o main.o -o mymain
```

Zum Ausführen rufen Sie das Programm auf mit: `./mymain`

Berichtsaufgaben

In der letzten Woche (Übungslatt 7) wurde die Ein- und Ausgabe von Dateien besprochen. Hierbei waren die Dateinamen im Quellcode fest vorgegeben und ließen sich somit nicht ohne erneutes Kompilieren verändern. In der Vorlesung in dieser Woche wurde gezeigt, wie man beim Programmstart Parameter an das Programm übergeben kann:

```
int main ( int argc , char **argv )
{
  for ( int i=0; i < argc ; ++i ) {
    //Ausgabe der uebergebenen Parameter
    cout << i << " " << argv[i] << endl;
}
```

Schreiben Sie ein Programm welche als ersten Eingabeparameter den Namen einer Ausgabedatei hat und alle weiteren Eingabeparameter in diese Datei schreibt.
Ein Beispiel ist der Aufruf von:
`> meinprogramm output.dat Vorname Nachname Strasse Stadt`
erzeugt die Datei `output.dat` mit dem Inhalt

```
Vorname
Nachname
Strasse
Stadt
```

.