Hydroinformatik I - WiSe 2019/2020

V7: Strings (Textverarbeitung)

Prof. Dr.-Ing. habil. Olaf Kolditz

[1]Helmholtz Centre for Environmental Research – UFZ, Leipzig

[2]Technische Universität Dresden – TUD, Dresden

[3]Center for Advanced Water Research – CAWR

Dresden, 29.11.2019

Semesterfahrplan

WiSe 2019/2020: Hydroinformatik I, Freitag (3. DS) 11:10-12:40, HÜL/S186/H

No	KW	Datum	ID	Vorlesung	Dozent
1a	42	19.10.2019	BHYWI-08-01	Hydroinformatik - Einführung	Kolditz
1b	42	19.10.2019	BHYWI-08-02	Compiler (Installation)	Kolditz
2	43	25.10.2019	BHYWI-08-03	Datentypen	Kolditz
3	44	01.11.2019	BHYWI-08-05	Hausaufgaben	Kolditz
5	45	08.11.2019	BHYWI-08-04	Einführung in Python	Kolditz
4	46	15.11.2019	BHYWI-08-06	Programmieren in Nat-Ing	Kalbacher
6	47	22.11.2019	BHYWI-08-07	Klassen	Kolditz
7	48	29.11.2019	BHYWI-08-08	Input-Output (I/O)	Kolditz
8	49	06.12.2019	BHYWI-08-09	Strings - Textverarbeitung	NN
9	50	13.12.2019	BHYWI-08-10	Pointer	NN
10	51	20.12.2019	BHYWI-08-11	Container	Kolditz
11	1	03.01.2020	BHYWI-08-12	BigData & Water 4.0	Kolditz
12	2	10.01.2020	BHYWI-08-13	Hydrologische Modellierung	Kolditz
13	3	17.01.2020	BHYWI-08-14	Neuronale Netzwerke	Kolditz
14	4	24.01.2020	BHYWI-08-15	ANN / Bayes'sche Netzwerke	Kolditz
15	5	31.01.2020	BHYWI-08-16	BN / Maschinelles Lernen	Kolditz
16	6	07.02.2020	BHYWI-08-17	Klausurvorbereitung	Kolditz

Strings?

Abbildung: Quelle:
http://z.about.com/d/guitar/1/5/r/r/string-change-electric059.jpg

Strings!

Wir haben schon in mehreren Übungen den Datentyp string benutzt, ohne diesen Datentyp etwas näher zu beleuchten. Dieses Versäumnis soll in diesem Kapitel nachgeholt werden.
String (in Deutsch Zeichenkette) ist vielmehr als nur ein Datentyp, string ist eine Standard-Klasse in C++. String ist eine der genialsten Weiterentwicklungen des C-Datentyps char, die das Hantieren mit Zeichen und Zeichenketten zu einem Kinderspiel macht, na sagen wir mal - uns das Programmierleben erheblich vereinfachen wird, wenn wir mit Zeichenketten operieren werden.
So wird z.B. der erforderliche Speicherplatz für Zeichenketten automatisch reserviert und bei Veränderungen angepasst.
Wenn wir strings benutzen wollen, müssen den Header der string-Klasse wie folgt inkludieren.

```
#include <string>    // for using strings
using namespace std; // for using standard names
```

String-Methoden

Methode	Erläuterung
`.append()`	verlängert den String
`.c_str()`	erzeugt Zeichenfeld mit dem Inhalt des Strings
`.clear()`	löscht den Inhalt des Strings
`.compare()`	vergleicht Strings
`.erase()`	löscht Zeichen im String
`.find()`	sucht Zeichen in einem String
`.insert()`	fügt in den String ein
`.length()`	ermittelt die Länge des Strings
`.replace()`	ersetzt Zeichen im String
	wichtig für die Konvertierung von string zu char*
`.resize()`	ändert Länge des Strings
`.substr()`	gibt einen Substring zurück
>>	Eingabeoperator für Strings
<<	Ausgabeoperator für Strings
`getline()`	liest Zeichen aus der Eingabe

Tabelle: string Methoden

Initialisieren von Strings

Eine Möglichkeit für die Initialisierung von strings haben wir uns bereits in der Exercise E45 angesehen bei der Verwendung von Klassen-Konstruktoren. Der Standard-Konstruktor `string()` erzeugt einen leeren String. Eine zweite Möglichkeit besteht direkt bei der Deklaration des strings, wie folgt:
Exercise E521:

```
string exercise("Exercise: string initialisation");
cout << exercise.length() << endl;
```

Zuweisen von Strings

In der folgenden Übung schauen wir uns an, wie wir mittels Tastatureingabe (Standard-Eingabe-Gerät) strings zuweisen und Teile von strings in andere kopieren zu können.

Zuweisen von Strings: Exercise 522

```
#include ... // Bitte fügen sie die notwendigen Header selbst ein
main()
{...
string eingabe;
string eingabe_anfang;
string message("Bitte geben Sie eine Zeile mit der Tastatur ein.
                Schliessen Sie die Eingabe mit Enter ab");
//-------------------------------------------------------------
cout << message << endl; // Ausgabe der Eingabeaufforderung
getline(cin,eingabe); // Eingabe einer Zeile über Tastatur
eingabe_anfang(eingabe,0,10); // die ersten 10 Zeichen von eingabe werd
                              nach eingabe_anfang kopiert
cout << "Ihr Eingagetext: " << eingabe << endl;
cout << "Die ersten 10 Zeichen des Eingagetextes: " << eingabe_anfang <
...}
```

Verketten von Strings

Mit dem Operator + können Strings miteinander verknüpft werden und in einem neuen `string name` kopiert. In der folgenden Übung produzieren wird aus Vor- und Nachnamen den ganzen Namen. Exercise E523:

```
string name;
CStudent* m_std = new CStudent();
name = m_std->name_first + m_std->name_last;
// oder
name = m_std->name_first;
name += m_std->name_last;
```

Wie bekommen wir einen Zwischenraum (Leerzeichen) zwischen Vor- und Nachnamen ?

Vergleichen von Strings

Oft sind Abfragen notwendig, ob gewisse Zeichenketten gefunden wurden, um dann gewisse Operationen durchzuführen. Hierfür bietet die String-Klasse mehrere Möglichkeiten an, z.B. den exakten Vergleich (string::compare) oder einen Teil von Zeichenketten (string::find). Die nachfolgende Übung zeigt, wenn der Nachname BOND gefunden wurde, dann wird der Vorname auf JAMES gesetzt (und natürlich der russische Geheimdienst informiert).
Exercise E524:

```
if(m_std->name_last.compare("BOND")==0)
{
  m_std->name_first = "JAMES";
}
```

Vergleichen von Strings

Die nachfolgende Übung zeigt, wie ein Programm beendet werden kann, wenn eine bestimmte Taste gedrückt wird. Hierfür werden einzelne Zeichen mit dem Operator == verglichen.

```
string Taste("N");
while(Taste == "J")
{
  cout << "Soll dieses Programm endlich beendet werden? (J/N)" << endl;
  getline(cin,Taste);
}
cout << "Programm-Ende" << endl;
```

Suchen in Strings

Diese Übung zeigt ihnen, wie nach Zeichenketten in string suchen können. Sie sehen, je nach Sorgfalt des Programmierers haben sie eventuell schlechte Karten, wenn ihr Vorname die Zeichenkette 'BON' enthält.

Exercise E525:

```
if(m_std->name_last.find("BON")!=string::npos)
{
  m_std->name_first = "JAMES";
}
```

Einfügen von Strings

Nun benutzen wir die Einfüge-Funktion von strings (`string::insert`), um Vor- und Nachnamen zusammenzusetzen. Dabei ermitteln wir zunächst die Länge des Vornamen mit `string::length`, setzen dann den Positionszähler pos um Eins hoch (Leerzeichen zwischen Vor- und Nachnamen) und fügen dann den Nachnamen mit string::insert ein.
Exercise E526:

```
string name;
int pos;
if(m_std->name_first.find("JAMES")!=string::npos)
{
  pos = m_std->name_first.length();
  name.insert(pos+1,"BOND");
}
```

Ersetzen in Strings

Eine weitere nützliche String-Funktion ist das Ersetzen von Zeichen. In der nachfolgenden Übung ändern wir den Nachnamen. Dazu wird zunächst wieder die Länge des Vornamens mit string::length ermittelt und dann der neue Nachname eingefügt. So können sie ihre Spuren verwischen ... ist auch praktisch bei Namensänderungen z.B. infolge Heiraten (Beachten sie, dass Triple-Namen wie Müller-Graf-Kleditsch nicht mehr zulässig sind).
Exercise E527:

```
string name;
int pos;
name = "JAMES" + " " + "CHRISTIE"
if(m_std->name_first.find("JAMES")!=string::npos)
{
  pos = m_std->name_first.length();
  name.replace(pos+1,"BOND");
}
```

Was passiert, wenn der neue Nachname länger ist als der alte ?

Löschen in Strings

Natürlich können auch Zeichen in einem string gelöscht werden. Diese Funktion passt den Speicherbedarf des gekürzten Strings automatisch an.
Exercise E528:

```
string name;
int pos;
name = "JAMES" + " " + "CHRISTIE"
if(m_std->name_first.find("JAMES")!=string::npos)
{
  pos = m_std->name_first.length();
  name.erase(pos);
  name.replace(pos+1,"BOND");
}
```

Umwandeln von strings in char

Manchmal ist es notwendig C++ strings in C char umzuwandeln (wir benötigen dies später, wenn wir mit der MFC Klasse CStrings für grafische Benutzeroberflächen arbeiten). Die String-Methoden c_str() und data() wandeln strings in char um. Aufgepasst, ab char müssen wir uns selber um das Speichermanagement kümmern.

```
fprintf(f,"  %s\n",name.c_str());
name.clear();
const char *char_string;
char_string = name.data();
```

Auswerten von Strings: Stringstreams

Stringstreams ... lässt sich kaum aussprechen .. es handelt sich aber definitiv nicht um stringdreams ... (sie erinnern sich an die Eingangsfolie der Vorlesung). Stringstreams sind eine sehr nützliche Sache, damit lassen sich Eingabedaten (von Tastatur und Datei) bequem als Stream auswerten. Um Stringstreams nutzen zu können, muss die Klasse `sstream` inkludiert werden. Die Übung zeigt, wie man eine eingegebene Zeile (Vor- und Nachnahme) elegant zerlegen kann. Dabei wird die eingebene Zeile zunächst in den `stringstream` kopiert, danach wird `input_line` wie ein normaler stream ausgelesen.

Stringstreams: Exercise E5210

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;
int main()
{
  string name;
  string name_first;
  string name_last;
  stringstream input_line;
  cout << "Geben Sie bitte Ihren Namen (Vor- und Nachnamen) ein: ";
  getline(cin,name);
  input_line.str(name);
  // Der Name wird nun zerlegt
  input_line >> name_first;
  cout << "Vorname:\t" << name_first << endl;
  input_line >> name_last;
  cout << "Nachname:\t" << name_last << endl;
  input_line.clear();
  return 0;
}
```

Profile

Wir nähern uns der Lösung, einfach in das Verzeichnis zu gelangen, in dem sich unsere Übungen befinden. Ähnlich wie in DOS die Programme autoexec.bat und config.sys beim Start des Betriebsystems automatisch ausgeführt werden, ist es bei LINUX ein so genanntes Profile: `.bash_profile`. In diese Datei können eigene Befehle eingetragen werden.
Eigentlich ist es ganz einfach ....

```
cd C:/User/TEACHING/C++/EXERCISES
```

Durch diese Instruktion in der `.bash_profile` wechseln wir direkt in das Verzeichnis, in dem sich unsere Übungen befinden.
Nachdem wir verschiedene Editoren ausprobiert haben (und erhebliche Unterschiede in der Behandlung von Zeilenenden (CR) gesehen haben), ließ sich zu allem Übel unsere mühsam editierte .bash_profile mit dem Windows-Explorer nicht speichern (es liegt an dem Punkt am Anfang des Datei-Namens). Ich kann ihre Enttäuschung gut verstehen, nicht umsonst gibt es Windows und Linux-Anhänger. Dennoch müssen wir eine Lösung finden oder ?