4.8 Von C nach C++/C++-Strings

4.8.1 Ein neuer Datentyp für Zeichenketten

In der Programmiersprache C gibt es für Zeichenketten keinen eigenen Datentyp; stattdessen wird ein Array von `char`-Elementen verwendet. Innerhalb des Arrays befindet sich ein `'\0'`-Element, der Terminator. Die C-Standard-Bibliothek hilft dabei, die so erstellten Zeichenketten einzugeben, zu verarbeiten und auszugeben.

In C++ stellt die C++-Standard-Bibliothek einen eigenen Datentyp (eine eigene Klasse) bereit: Eine Variable (ein Objekt) des Datentyps `string` ist in etwa ein Record, das mit Hilfe dynamischen Speichers eine beliebig lange Zeichenkette aufnehmen kann. Man muss also nicht mehr auf die maximale Länge und die Terminierung achten. Außerdem kann man mit diesem Objekt einfacher umgehen als mit einer C-Zeichenkette.

4.8.2 Zeichenketten anlegen, zuweisen und vergleichen

Im folgenden Beispiel werden `string`-Objekte angelegt und benutzt:

string1.cpp

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main(void)
6  {
7     string s;          // vereinbaren
8     string t="blubb"; // initialis. (wie in C)
9     s = "hallo";       // NEU: einfache Zuweisung
10    cout << "Bitte hallo eingeben: ";
11    cin >> t;
12    if(s==t)           // NEU: Vergleich moeglich
13       cout << "gleich" << endl;
14    else
15       cout << "verschieden" << endl;
16    return 0;
17 }
```

- Zeile 2: Diese Einbindung braucht man für die Klasse `string`.
- Zeile 8: Man kann ein `string`-Objekt wie hier mit einer C-Zeichenkette initialisieren oder aber mit einem anderem `string`-Objekt, also `string t=s`.
- Zeile 9: Ein `string`-Objekt erlaubt eine direkte Zuweisung von einer C-Zeichenkette. `strcpy()` ist hier nicht nötig. Die Zuweisung von einem anderen `string`-Objekt, also `s=t`, ist auch möglich.
- Zeile 10: Mit `cin` kann man von der Tastatur ein Wort in ein `string`-Objekt einlesen. Wie in C bleibt das Leerzeichen oder der Zeilenumbruch anschließend im Tastaturpuffer.
- Zeile 12: Man kann `string`-Objekte direkt miteinander vergleichen.

Terminal

```
schueler@debian964:~$ g++ string1.cpp
schueler@debian964:~$ a.out
Bitte hallo eingeben: hallo
gleich
```

4.8.3 Zeichenketten verändern und ausgeben

Und hier ein anderes Beispiel, in dem eine Zeichenkette verlängert wird:

string2.cpp

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main(void)
6  {
7      string s="hallo";
8      s=s+", "; // C-Zeichenkette anhaengen mit +
9      s+="Welt"; // C-Zeichenkette anhaengen mit +=
10     s=s+'!';  // einzelnes Zeichen anhaengen
11
12     s[0]='H'; // einzelnes Zeichen ersetzen
13     cout << s << endl; // Ausgabe mit cout
14     return 0;
15 }
```

- Zeile 8: Man kann ein `string`-Objekt und eine C-Zeichenkette addieren, ebenso zwei `string`-Objekte.
- Zeile 9: Das funktioniert auch mit dem Operator `+=`.
- Zeile 10: Man kann ein `string`-Objekt und eine `char`-Variable oder eine `char`-Konstante addieren.
- Zeile 12: Das `string`-Objekt erlaubt einen Zugriff auf ein einzelnes Zeichen wie eine C-Zeichenkette! Das ist im Vergleich zu anderen Sprachen (Javascript) ein Vorteil.
- Zeile 13: Die Ausgabe mit `cout` verläuft problemlos.

Terminal

```
schueler@debian964:~$ g++ string2.cpp
schueler@debian964:~$ a.out
Hallo, Welt!
```

4.8.4 C++-String in C-String umwandeln

Mit dem Methodenaufruf `s.c_str()` kann man aus dem `string`-Objekt `s` eine C-Zeichenkette erhalten. Ein Methodenaufruf ist von der Schreibweise her wie ein Funktionsaufruf, der zu einer Record-Variablen gehört; deshalb benutzt er den Punkt-Operator, den man von Records her kennt.

string3.cpp

```
1  #include <iostream>
2  #include <string>
3  #include <cstdlib> // C++-Version von <stdlib.h> fuer system()
4  #include <unistd.h> // fuer sleep()
5  using namespace std;
6
7  int main(void)
8  {
9      string s="xeyes &"; // von C nach C++
10     system(s.c_str());  // von C++ nach C
```

```
11      sleep(3);
12      s="killall "+s;
13      system(s.c_str());   // system("killall xeyes &");
14      return 0;
15  }
```

- Zeile 9: Das C++-Objekt wird durch die C-Zeichenkette initialisiert
- Zeile 10: Durch den Methodenaufruf `c_str()` wird der Inhalt von `s` als C-Zeichenkette zurückgegeben.

Hier sind ein paar weitere Methodenaufrufe für `string`-Objekte:

string4.cpp

```
1   #include <iostream>
2   #include <string>
3   using namespace std;
4
5   int main(void)
6   {
7       int le, sz;
8       string s;
9       cout << "Ein Wort eingeben: ";
10      cin >> s;
11
12      le=s.length();
13      cout << "Inhalt hat " << le << " Bytes." << endl;
14
15      sz=s.size();
16      cout << "Objekt hat " << sz << " Bytes." << endl;
17
18      cout << "Fuege Wort ab Byte 4 ein: " << endl;
19      s=s.insert(4, "Einschub");
20      cout << s << endl;
21
22      cout << "Suche und ersetze Wort: " <<  endl;
23      s=s.replace(s.find("schub"), 5, "bau");
24      cout << s << endl;
25
26      cout << "Entnehme 5 Zeichen ab Byte 3:" << endl;
27      s=s.substr(3, 5);
28      cout << s << endl;
29      return 0;
30  }
```

Terminal

```
schueler@debian964:~$ g++ string4.cpp
schueler@debian964:~$ a.out
Ein Wort eingeben: Zitronenfalter
Inhalt hat 14 Bytes.
Objekt hat 14 Bytes.
Fuege Wort ab Byte 4 ein:
ZitrEinschubonenfalter
Suche und ersetze Wort:
```

```
ZitrEinbauonenfalter
Entnehme 5 Zeichen ab Byte 3:
rEinb
```

4.8.5 Ganze Zeile in C++-String eingeben

Will man eine ganze Textzeile von der Tastatur in ein `string`-Objekt eingeben, braucht man dazu die Funktion `getline()`: Wenn man im gleichen Programm `cin >>` und `getline` benutzt, ist es wichtig, nach jedem `cin >>` die oben aufgeführten Aufrufe `cin.clear()` und `cin.ignore(128, '\n')` zu benutzen, um den Eingabepuffer zu leeren.

string5.cpp

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main(void)
6  {
7      string s;
8      cout << "Wort eingeben: ";
9      cin >> s;
10     cin.clear();
11     cin.ignore(128, '\n');
12     cout << "Ihre Eingabe war: '" << s << "'" << endl;
13
14     cout << "Textzeile eingeben: ";
15     getline(cin, s);
16     cout << "Ihre Eingabe war: '" << s << "'" << endl;
17     return 0;
18 }
```

- Zeile 10: Falls die `cin`-Benutzung nicht erfolgreich war, wird die Eingabe wieder in Ordnung gebracht.
- Zeile 11: Hier werden maximal 128 Zeichen eingelesen bis zum Zeilenumbruch. Vergleichbar mit `while(getchar()!='\n'){}`. Dies sollte man unbedingt tun, falls noch weitere Eingaben in diesem Programm folgen.
- Zeile 15: So lässt man den Benutzer eine ganze Zeile eingeben.

Terminal

```
schueler@debian964:~$ g++ string4.cpp
schueler@debian964:~$ a.out
Wort eingeben: Hallo
Ihre Eingabe war: 'Hallo'
Textzeile eingeben: Hallo, Welt!
Ihre Eingabe war: 'Hallo, Welt!'
```