Kapitel 12

Strings

Konstante Zeichenfolgen

- Zeichenfolgen, die in *char*-Arrays abgelegt werden, werden C-String genannt.
- Dies ist in der Programmiersprache C die einzige Möglichkeit Zeichenfolgen abzulegen.
- C-String Konstanten bestehen aus einer Folge von Zeichen, die in Anführungszeichen eingeschlossen sind.
- Im Speicher wird der C-String mit einem String-Ende Zeichen ('\0') am Ende abgespeichert.
- Man beachte den Unterschied zwischen z.B. 'Z' und "Z".
- 'Z' entspricht also einem Zeichen und "Z" zwei Zeichen.

Die Klasse 'string'

- Statt der C-Strings beinhaltet die C++ Standardbibliothek eine Klasse namens *string*.
- Sie bietet erhebliche Vorteile gegenüber der Verwendung der C-Strings.
- Zur Verwendung der Klasse *string* muss die Header-Datei *string* eingebunden werden.

```
#include <string>
```

Definition eines *string* Objekts

- Im folgenden Beispiel wird das *string* Objekt *text* definiert und initialisiert.
- Die Zeichenfolge nach dem Zuweisungsoperator ist eine C-String Konstante.

```
string text = "Ich bin ein string Beispiel.\n";
```

Definition von *string* Feldern

- Es können auch *string* Felder (Arrays) definiert werden.

```
string tage[] = {"Montag", "Dienstag", "Mittwoch",
                 "Donnerstag", "Freitag", "Samstag"
                 "Sonntag"};
cout << tage[0] << endl;
```

```
Montag
```

Zuweisung von *string* Objekten

- Ein *string* Objekt kann einem anderen zugewiesen werden.

```
string text;
string tage[] = {"Montag", "Dienstag", "Mittwoch",
                 "Donnerstag", "Freitag", "Samstag"
                 "Sonntag"};
text = tage[2];
cout << text << endl;
```

```
Mittwoch
```

Eingabe von *string* Objekten

- Bei der Eingabe unterscheidet man die Eingabe eines Wortes ohne Leerzeichen und die Eingabe einer Zeile mit Leerzeichen.

Eingabe von *string* Objekten - Wort

- Im folgenden Beispiel wird das zweite Wort *Maier* wegen des Leerzeichens nicht übergeben.

```
string name;
cin >> name;
cout << name << endl;
```

Eingabe: Otto Maier

Ausgabe: Otto

Eingabe von *string* Objekten - Zeile

- Zum Einlesen einer ganzen Zeile verwendet man die Funktion *getline()*.

```
getline(cin, name);
cout << name << endl;
```

Eingabe: Otto Maier

Ausgabe: Otto Maier

Strings zusammensetzen

- Die Operatoren + und += sind auf *string* Objekte anwendbar.

```
string  text1 = "Ich ";
text1 = text1 + "bin ";
text1 += "ein Beispiel!";
cout << text1 << endl;
```

Ausgabe:

```
Ich bin ein Beispiel!
```

Strings zusammensetzen

- Das direkte Zusammensetzen zweier String-Konstanten ist hingegen **nicht** möglich.

```
text = "Ich " + "bin";
```

Strings vergleichen

- Die Vergleichsoperatoren sind auf *string* Objekte anwendbar.
- Es erfolgt ein lexikalischer Vergleich.

```
string a = "Maier", b = "Mayer";
if (a < b)
    cout << "Maier < Mayer" << endl;
```

Zugriff auf einzelne Zeichen

- Mit dem Indexoperator [] kann auf einzelne Zeichen eines Strings zugegriffen werden.

```
string a = "Maier";
cout << a[0] << a[4] << endl;
```

Ausgabe:

```
Mr
```

Länge eines *Strings*

- Mit der Funktion *size()* kann die Länge eines Strings ermittelt werden.
- Die Länge eines Strings entspricht der Anzahl der Zeichen des Stings.

```
string text = "Ich bin ein string Beispiel!";
cout << text.size() << endl;
```

Ausgabe:

```
28
```

Beispiel

```
string text = "Ich bin ein string Beispiel!";
int eCount = 0;
for (int i=0; i< (int)text.size(); i++)
{
    if (text[i] == 'e')
    {
        eCount++;
    }
}
cout << eCount << endl;
```

```
3
```

Groß- und Kleinschreibung

- Mit der Funktion *toupper()* kann ein Kleinbuchstabe in einen Großbuchstaben umgewandelt werden.
- Mit der Funktion *tolower()* kann ein Großbuchstabe in einen Kleinbuchstaben umgewandelt werden.

```
string s1 = "abc", s2 = "XYZ";
s1[0] = tolower(s2[0]);
s2[1] = toupper(s1[1]);
cout << s1 << endl << s2 << endl;
```

Ausgabe:

```
xbc
XBZ
```

Weitere *string* Funktionen

compare	Mit der Funktion *compare()* können Strings verglichen werden.
find	Mit der Funktion *find()* kann ein String in einem anderen String gesucht werden. *find()* gibt die Position des ersten Auftretens des Strings zurück.
erase	Mit der Funktion *erase()* kann ein Teilstring eines Strings entfernt werden.
c_str	Zur Konvertierung eines string Objekts in einen C-String wird die Funktion *c_str()* verwendet

Vergleich von *Strings*

- *compare()* liefert 0 als Rückgabewert, wenn die Zeichenketten gleich sind.
- *compare()* liefert einen negativen Wert, wenn s1 kleiner als s2 ist und einen positiven Wert, wenn s1 größer als s2 ist.

```
string s1 = "efg";
string s2 = "abc";
int r = s1.compare(s2);
```

Vergleich von *Strings*

- Beispiel mit Rückgabewert 0:

```
string s;
cin >> s;
if (s.compare("Ja") == 0)
{
    cout << "Ja eingegeben!" << endl;
}
```

Vergleich von *Strings*

- Beispiel mit Rückgabewert ungleich 0:

```
string s1, s2;
cin >> s1 >> s2;
if (s1.compare(s2) == 0)
    cout << "s1 ist gleich s2" << endl;
else if (s1.compare(s2) < 0)
    cout << "s1 ist kleiner als s2" << endl;
else if (s1.compare(s2) > 0)
    cout << "s1 ist größer als s2" << endl;
```

Vergleich von *Strings*

- Die Funktion *compare()* bietet noch weitere Möglichkeiten:

```
string s = "Kennzeichen: DA-FH 1000";
if (s.compare(13,2,"DA") == 0)
{
    cout << "Darmstadt" << endl;
}
```

- Im String s werden nach dem 13. Zeichen 2 Zeichen mit „DA“ verglichen.

Suchen in *Strings*

- Mit der Funktion *find()* kann ein String in einem anderen String gesucht werden. *find()* gibt die Position des ersten Auftretens des Strings zurück. Bei Nichtauftreten wird der Wert *string::npos* zurückgegeben. *string::npos* steht für den Wert -1.

```
string zeile;
getline(cin, zeile);
if (zeile.find("Autor") != string::npos)
{
    cout << "Autor enthalten" << endl;
}
```

Entfernen von Teilstrings

- Mit der Funktion *erase()* kann ein Teilstring entfernt werden.

```
string s1 = "ABCDEFGHIJ";
s1.erase(2,6);
cout << s1 << endl;
cout << "Restzeichen: " << s1.size() << endl;
```

```
ABIJ
Restzeichen: 4
```

- In s1 werden ab Position 2 insgesamt 6 Zeichen gelöscht.

Konvertierung in C-String

- Zur Konvertierung in einen C-String (= char-Array) kann die Methode *c_str()* verwendet werden.

```
char cs[20] = "C-String";
string s = "String-Objekt";
strcpy(cs, s.c_str());
cout << cs << endl;
```

- Die C-Funktion strcpy erwartet als 2. Parameter ein const char * Element.

Lesen von *Strings*

- Deklariert man eine Variable vom Typ istringstream, so hat man einen Stream Objekt mit dem man von einem String lesen kann.
- Um istringstream benutzen zu können, muss ein "#include <sstream>" erfolgen.
- Im folgenden Beispiel wird die Schreibweise eines Namens durch Vertauschung des Vor- und Nachnamens innerhalb eines Strings verändert.
- Hierzu wird der String elementweise ausgelesen.

Lesen von *Strings*

```
istringstream iss;
string zeile = "Wilhelm Maier 674529";
string vorname, nachname;
int ident;
iss.str(zeile);
iss >> vorname;
iss >> nachname;
iss >> ident;
iss.clear();
cout << "Name: " << nachname << ", " << vorname <<
" Identnr.: " << ident << endl;
```

Lesen von *Strings*

- Wenn iss zum zweiten Mal ein String zugewiesen werden soll, muss zunächst die Funktion *clear()* aufgerufen werden.
- Das Lesen von einem String kann auch zur Datentypumwandlung von einem String in eine arithmetische Größe verwendet werden.

Schreiben in *Strings*

- Mit dem Stream ostringstream kann in einen String geschrieben werden.
- Im folgenden Beispiel werden zwei Zahlenwerte kombiniert und mit Text in einen String geschrieben.

Schreiben in *Strings*

```
ostringstream oss;
string zeile;
int i = 100, j = 200;
oss << "i = " << setw(3) << i << " j = " << setw(3) << j;
zeile = oss.str();
cout << zeile << endl;
oss.str("");                    // vor der Wiederverwendung leeren
oss << "i + j = " << setw(3) << i + j;
zeile = oss.str();
cout << zeile << endl;
```