Notizen AIK ProgTech 3: C++ Strings: Klasse `string`

Klaus Kusche

In C++ wurde für Strings eine eigene Klasse **`string`** eingeführt. Sie bietet mehr Komfort und mehr Möglichkeiten als die klassischen C-Strings (**`char *`** bzw. **`char []`**, Header **`<string.h>`** bzw. **`<cstring>`**), braucht aber etwas mehr Speicher und Rechenzeit.

Der wichtigste Vorteil von C++-Strings ist, dass sie intern automatisch Speicher in der richtigen Größe anlegen und bei Bedarf wachsen: Man muss bei der Deklaration bzw. beim Erzeugen keine Größe angeben, und es besteht keine Gefahr eines String-Überlaufs.

Daneben können die alten C-Strings auch in C++ wie bisher verwendet werden. In vielen C++-Programmen kommt beides gemischt vor, manche Firmen schreiben in ihren Programmier-Richtlinien auch für C++ nur C-Strings vor, andere wieder nur die neuen C++-Strings. Bei Funktionen und Methoden, die Strings als Parameter oder Returnwert haben, ist jedenfalls darauf zu achten, um welche Art String es sich dabei handelt.

Operationen auf **`string`**-Objekten (Header **`<string>`**):

- Konstruktoren:
  - Der Standard-Konstruktor (ohne Argumente) liefert einen leeren String .
  - Es gibt einen Copy-Konstruktor, einen Konstruktor mit einem C-String als Argument, und einen Konstruktor, der n Mal ein einzelnes Zeichen wiederholt.
- Zuweisungs-Operator **=**: Von C++-String, von C-String, von einzelnem **`char`** .
- Umwandlung in Gegenrichtung: C-String aus C++-String **`s`** extrahieren: **`s.c_str()`**
- **`s.clear()`**: Setzt **`s`** auf den Leerstring.
- **`s.length()`** oder **`s.size()`**: Tatsächliche Länge von **`s`**.
- **`s.empty()`**: Ist **`s`** leer?
- Zugriff auf einzelne Buchstaben mit **`[]`**:
  Lesend und schreibend (links und rechts von **=**) möglich. Ohne Längenprüfung!
- Zusammenhängen mit Operator **+**: String, C-String oder **`char`**
  Alle 3 Möglichkeiten für links und rechts, eine Seite muss **`string`** sein !
- Anhängen an bestehenden String:
  Mit **+=** oder **`s.append`**: String, C-String oder **`char`**
- Teilstring extrahieren (**`n`** Zeichen ab **`pos`**) : **`s.substr(pos, n)`**
  Append und Konstruktor gibt es auch mit Ausschneiden eines Substring
- 2 Möglichkeiten für Vergleiche:
  - Normale Operatoren **`== != < > <= >=`** (wie bei Zahlen), auch mit C-String
  - **`s1.compare(s2)`** (**`s2`**: C++-String oder C-String), Ergebnis **<0 ==0 >0** wie **`strcmp`**
- **`s1.find(s2, pos = 0)`** (**`s2`**: C++-String, C-String oder **`char`**): Sucht ab **`pos`**
- Weitere Funktionen: **`insert`**, **`erase`**, **`replace`**
- I/O mit **<<** und **>>** oder Funktion **`getline`** (liest eine Zeile)
- Und einige mehr...