# Strings in C++

Brian R.Grell

OO Get Together
2. Oktober 2006

# Im Überblick

- Wie werden Strings in C behandelt
- Wie in C++
- Was will ROOT

#include <cstring.h>

# C strings

**Stringliteral:**

**Ein Array mit Elementen vom Typ** const char

**C:**

**Das Array muss mit einer null abgeschlossen werden**

**C++:**

**Compiler hängt an ein Stringliteral eine null und rettet Kompatibilität**

# C strings: Initialisierungs-Übung

```
char ca1[] = {'C', '+', '+'};

char ca1[] = {'C', '+', '+', '\0'};

char ca3[] = “C++”;

const char *cp = “C++”;

char *cp1 = ca1;

char *cp2 = ca2;
```

**Welche sind strings im C-Sinne?**

# Funktionen für strings (C style)

| | |
|---|---|
| strlen(s) | **Gibt Länge von** s **ohne das Null-Zeichen zurück** |
| strcmp(s1,s2) | **Prüft** s1 **und** s2 **auf Gleichheit (dann** 0**)** |
| strcat(s1,s2) | **Hängt** s2 **an** s1 **an und gibt** s1 **zurück** |
| strcpy(s1,s2) | **Kopiert** s2 **in** s1 **und gibt** s1 **zurück** |
| strncat(s1,s2,n) | **Hängt** n **Zeichen von** s2 **an** s1 **an und gibt** s1 **zurück** |
| strncpy(s1,s2,n) | **Kopiert** n **Zeichen von** s2 **in** s1 **und gibt** s1 **zurück** |
| sprintf | **Schreibt in string hinein** |

# C++ strings

- **In C++ gibt es einen eigenen Bibliothekstyp string (**#include <string.h>**)**
- **Die Bibliothek erledigt die Speicherverwaltung**
- **Der Typ string ist nicht mehr identisch mit Zeichenstringliteralen**

# C++: Initialisierung

| | |
|---|---|
| string s1; | **Standardkonstruktor:** s1 **ist leerer String** |
| string s2(s1); | **Initialisiert** s2 **als Kopie von** s1 |
| string s3(“Wort”); | **Initialisiert** s3 **als Kopie des Stringliterals** |
| string s4(n, 'c') | **Initialisiert** s4 **mit** n **Kopien des Zeichens '**c**'** |

# Stringoperationen

| | |
|---|---|
| s.empty() | **Gibt** true **zurück, wenn** s **leer, sonst** false**.** |
| s.size() | **Gibt die Anzahl der Zeichen in** s **zurück.** |
| s[n] | **Gibt das Zeichen an Position** n **in** s **zurück.** |
| s1 + s2 | **Gibt Verkettung von** s1 **und** s2 **zurück.** |
| s1 = s2 | **Ersetzt die Zeichen in** s1 **durch Kopie von** s2**.** |
| s1 == s2 | **Gibt** true **zurück, wenn** s1 **und** s2 **gleich sind, sonst** false**.** |

# Die Zeichen eines strings

| | |
|---|---|
| isalnum(c) | true**, wenn** c **ein Buchstabe oder eine Ziffer ist** |
| isalpha(c) | true**, wenn** c **ein Buchstabe ist** |
| iscntrl(c) | true**, wenn** c **ein Steuerzeichen ist** |
| isdigit(c) | true**, wenn** c **eine Ziffer ist** |
| isgraph(c) | true**, wenn** c **druckbar, aber kein Leerzeichen ist** |
| isprint(c) | true**, wenn** c **ein druckbares Zeichen ist** |
| ispunct(c) | true**, wenn** c **ein Satzzeichen ist** |
| isspace(c) | true**, wenn** c **ein Leerraumzeichen ist** |
| isupper(c) | true**, wenn** c **ein Grossbuchstabe ist** |
| isxdigit(c) | true**, wenn** c **eine Hexadezimalziffer ist** |
| toupper(c) | **gibt den entsprechenden Grossbuchstaben zurück, wenn** c **ein Kleinbuchstabe ist, sonst** c **unverändert** |

# Zurück zu Stringliteralen

- **ROOT will meist keinen C++ string-Typ, sondern Zeichenstringliterale**
- **Die Funktion** c_str **liefert eine Darstellung des C++ strings im C-Stil (Zeiger auf einen mit null abgeschlossenen Zeichenarray, der dieselben Daten enthält wie die Zeichen des strings)**

```
const char *str = s1.c_str();
```

# Zusammenfassung

- Zeichenkettenarithmetik effizienter in C++ mit Stringbibliothekstyp
- Häufig werden Stringliterale, und keine C++ strings gebraucht
- Die Funktion c_str() ermöglicht Übergang zum C-Stil-Literal