Praxis der Programmierung

Von C zu C++

Institut für Informatik und Computational Science
Universität Potsdam

Henning Bordihn

Einführung in C++

Weiterentwicklung von C

- **C++** übernimmt die Konzepte von **C**:
  - Compilerarchitektur (Präprozessor, Compiler, Linker)
  - Header-Dateien, (stark erweiterte) Standardbibliothek
  - Variablen, Pointer, Datentypen (einschl. Arrays, Strukturen, Unionen, ...)
  - Kontrollstrukturen, Funktionen, Konstanten
  - Kommentare, ...
- weitere Datentypen (`bool`, erweiterte Zeichensätze, z.B. `wchar_t`)
- Namensräume
- Unterstützung des objektorientierten Paradigmas
- Templates
- geringe syntaktische Varianten

Bibliotheken

- Einbinden mit `#include <...>` (ohne Endung `.h`)
- sehr mächtige Bibliothek mit Lösungen im **C**++-Standard
- **C**-Bibliotheken vorhanden:
  `<name.h>` $\rightsquigarrow$ `<cname>` , *z.B.* `<cstdio>`, `<cstdlib>`, ...
- Ein- und Ausgabe mit `<iostream>`
  - Erlaubt Lenken von Datenströmen in Ein- und Ausgabeobjekte
  - `cout` ist Standardausgabe
  - `cerr` ist Standardfehlerausgabe
  - `cin` ist Standardeingabe
  - `cout, cin, cerr` im Namensraum std definiert

Das Hello World-Beispiel

```
/* Programmname.cpp
 * Quellcodedateien enden auf .cpp
 */

#include <iostream>          // Praeprozessoranweisung
using namespace std;         // Anweisung: Benutzen des Namensraums std

int main() {

   cout << "Hello World!";  // << fuer Umlenkung des Datenstroms
   cout << endl;            // endl ist Konstante fuer das Zeilenende in std

   // alternatives einzeiliges Kommando:
   cout << "Hello World!" << endl;

}  // fehlendes return-Kommando bei main() wird vom Compiler toleriert
```

Namensräume

- definieren Bereiche, in denen Namen / Bezeichner eindeutig sein müssen
- in verschiedenen Namensräumen kann der gleiche Name verwendet werden
- *Beispiele*:
  - Telefonnummern in Vorwahlbereichen: `0331 123456`, `030 123456`
  - Telefonnummern mit Vorwahl in Ländernetzen: `+49 30 123456`, `+36 30 123456`
  - Dateinamen in Ordnern; diese in übergeordneten Ordnern, ...
  - in Netzwerken absolute Pfadnamen auf Hosts (Rechnernamen), ...

Qualifizierte Namen

- unqualifizierte Namen: die Bezeichner selbst

  Beispiel: `meineDatei`

- qualifizierte Namen: mit Angabe des Namensraums

  Beispiel: `/home/rlehre/meineDatei`

- in **C++**: `namensraum::bezeichner`

  Beispiel: `std::cout`, `std::cin`, `std::endl`

- Namensraum `std` enthält Bezeichner der Standardbibliothek

- Definition eigener Namensräume möglich ... *später*

Das Hello World-Beispiel (2)

```
/* Programmname.cpp
 * Quellcodedateien enden auf .cpp
 */

#include <iostream>          // Praeprozessoranweisung
/* using namespace std; auskommentiert */

int main() {

   std::cout << "Hello World!";  // << fuer Umlenkung des Datenstroms
   std::cout << std::endl;       // endl ist Konstante fuer das Zeilenende in std

   // alternatives einzeiliges Kommando:
   std::cout << "Hello World!" << std::endl;

}  // fehlendes return-Kommando bei main() wird vom Compiler toleriert
```

Aufgabe 1

Schreiben Sie ein **C++**-Programm, das den Benutzer auffordert, seinen Alter einzugeben, dieses von der Standardeingabe entgegennimmt und wieder auf die Standardausgabe ausgibt:

```
Geben Sie Ihr Alter ein: |  // Eingabe z.B. von "22"
Sie sind 22 Jahre alt.
```

- Realisieren Sie auch den konstanten Text und die Zeilenumbrüche.
- Benutzen Sie keine **C**-Bibliotheken. Versuchen Sie, `cin` durch Analogieschluss korrekt zu verwenden.
- Compileraufruf mit `g++` oder `c++`, *z.B.* `g++ -Wall Name.cpp -o Name`

Ein- und Ausgabe

Formatierte Ausgabe

- Verwendung von Funktionen aus `<cstdio>`
  $\rightsquigarrow$ z.B. bei formatierter Ausgabe von Zahlen

- Verwendung von Manipulatoren, z.B.:
  - `endl` (`<iostream>`)
    $\rightsquigarrow$ erzwingt Ausgabe der bisher eingegebenen Zeile und einen Zeilenvorschub
  - `setw(breite)` (`<iomanip>`)
    $\rightsquigarrow$ rechtsbündige Ausgabe des nächsten Ausgabestroms
    und Auffüllen mit Leerzeichen auf `breite`
  - `setfill(chr)` (`<iomanip>`)
    $\rightsquigarrow$ ab jetzt Ersetzen des Leerzeichens zum Auffüllen durch `chr`
  - `left` und `right` (`<iostream>`)
    $\rightsquigarrow$ ab jetzt Umstellen auf Links- bzw. Rechtsbündigkeit

Aufgabe 2

Schreiben Sie ein **C**++-Programm, das mit Hilfe von Manipulatoren folgende Ausgabe erzeugt:

```
      42
-----42
42-----
```

Weitere Manipulatoren

- `dec, oct, hex` (<iostream>)
  ⇝ ab jetzt Darstellung ganzer Zahlen als Dezimal-, Oktal- bzw. Hexadezimalzahl

- `showbase` (<iostream>)
  ⇝ ab jetzt Zeigen von `0x` bei Hexadezimalzahlen und `0` bei Oktalzahlen

- `scientific, fixed` (<iostream>)
  ⇝ ab jetzt in Exponential- bzw. Festkommadarstellung

- `setprecision(n)` (<iomanip>)
  ⇝ ab jetzt `n` Nachkommastellen

Elementare Datentypen

Der Datentyp `bool`

- Literale `true` und `false`
- Ausgabe als `1` bzw. `0`
  Umstellung mit den `iostream`-Manipulatoren `boolalpha` und `noboolalpha`
- Boolesche Ausdrücke durch Vergleich arithmetischer Ausdrücke, `(a < b+2)` und durch Funktionen mit Rückgabetyp `bool`
- logische Verknüpfung mit `!` (Negation), `&&` (Und), `||` (Oder)

Aufgabe 3

Schreiben Sie ein **C++**-Programm mit drei Funktionen zum

- Feststellen, ob eine ganze Zahl gerade ist,
- Feststellen, ob eine ganze Zahl negtiv ist,
- Feststellen, ob eine ganze Zahl gleich 20 ist.

Das Hauptprogramm fragt solange den Benutzer nach einer ganzen Zahl, bis die 0 eingegeben wird. Für jede eingegebene Zahl gibt es aus, ob die Zahl größer gleich 0 und gerade ist und außerdem ob die Zahl negativ oder 20 ist.

Aufgabe 4

Kopieren Sie die Datei `/home/rlehre/Woche_08/Rechte.cpp`, übersetzen Sie sie und führen Sie das Programm aus.

Analysieren Sie anschließend den Quellcode! Hier wird bitweise gerechnet.

1. Was bedeuten die Operatoren `&` und `~` ?

2. Weshalb steht `"j"` in doppelten Anführungszeichen?

Zugriffsrechte für neue Dateien

- `umask` <*Oktalzahl*> gibt an, welche Rechte nicht vergeben werden sollen.
- Als Grundeinstellung (mit `umask 0`) werden Dateien mit Oktal 666 (`rw-rw-rw-`) und Verzeichisse mit Oktal 777 (`rwxrwxrwx`) angelegt.
- In der Oktalzahl von `umask` auftretende Rechte werden von der Grundeinstellung abgezogen.
  (genauer: bitweise UND-Verknüpfung des Wertes der Grundeinstellung und des negierten `umask`-Wertes)
- Beispiel: `umask 022` bewirkt, dass Dateien mit Oktal 644 (`rw-r--r--`) und Verzeichnisse mit Oktal 755 (`rwxr-xr-x`) angelegt werden.
- `umask` (ohne Argument) zeigt die aktuelle Einstellung an.

Bit-Operationen

`&`	AND	1 wenn beide Operanden 1
`|`	OR	1 wenn einer der Operanden 1
`~`	NOT	kippt das Bit
`^`	XOR	1 wenn genau einer der Operanden 1
`>> n`	R-Shift	verschiebt ein Bitmuster um n Bits nach rechts
`<< n`	L-Shift	verschiebt ein Bitmuster um n Bits nach links

Beispiel: `00011000 >> 1` ergibt `00001100`

$\rightsquigarrow$ >> bewirkt Halbieren einer ganzen Zahl

$\rightsquigarrow$ << bewirkt Verdoppeln einer ganzen Zahl

Aufgabe 5

Schreiben Sie ein **C++**-Programm, dass mit Hilfe von Shiftoperationen die Zahl 2^n berechnet, wobei n als Programmparameter übergeben wird.

Zeichentypen

- `char` (1 Byte für ASCII-Zeichen)
- `wchar_t` (internationaler Zeichensatz)
- `char16_t` (16-Bit UNICODE) im **C++11-Standard**
- `char32_t` (32-Bit UNICODE) im **C++11-Standard**
  (schließt auch asiatische Zeichen ein)

Übersetzung im C++11-Standard mit `g++ -std=c++11`

Einige syntaktische Varianten

- **Initialisierung von Variablen**

```
1. int zahl = 0;        // wie in C
2. int zahl(0);         // wie Funktionsaufruf
3. int zahl{0};         // direkte Initialisierung im C++11-Standard
4. int zahl = {0};      // direkte Initialisierung im C++11-Standard
```

- **Typkonvertierung**

```
1. int n = 200;
   (char) n             // wie in C
2. char(n)              // wie Funktionsaufruf
```