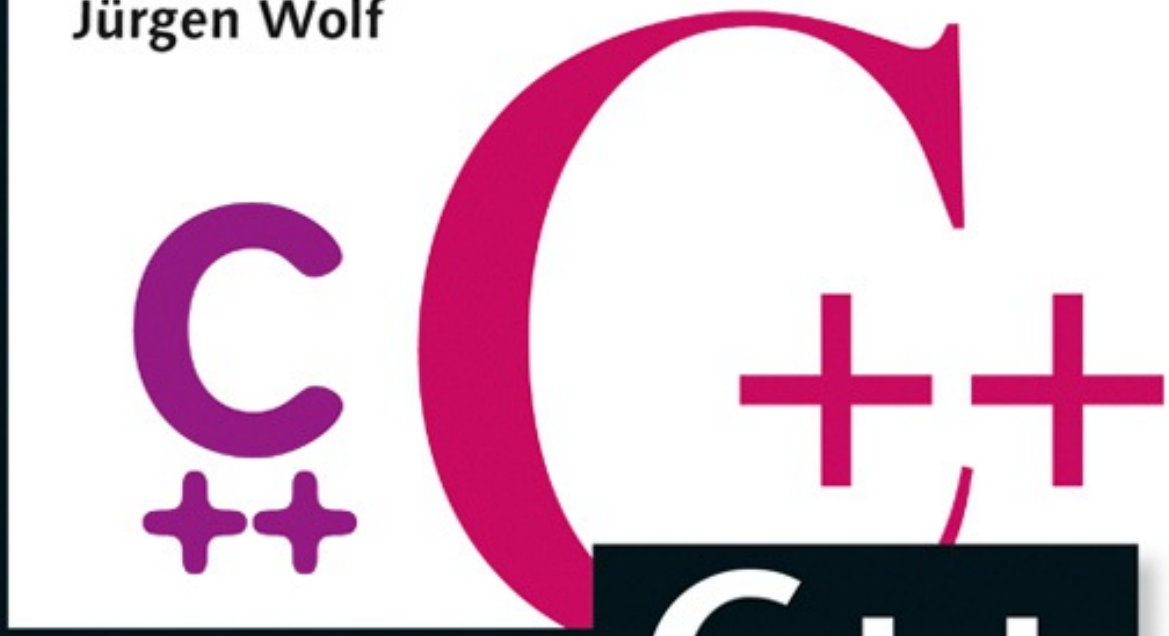
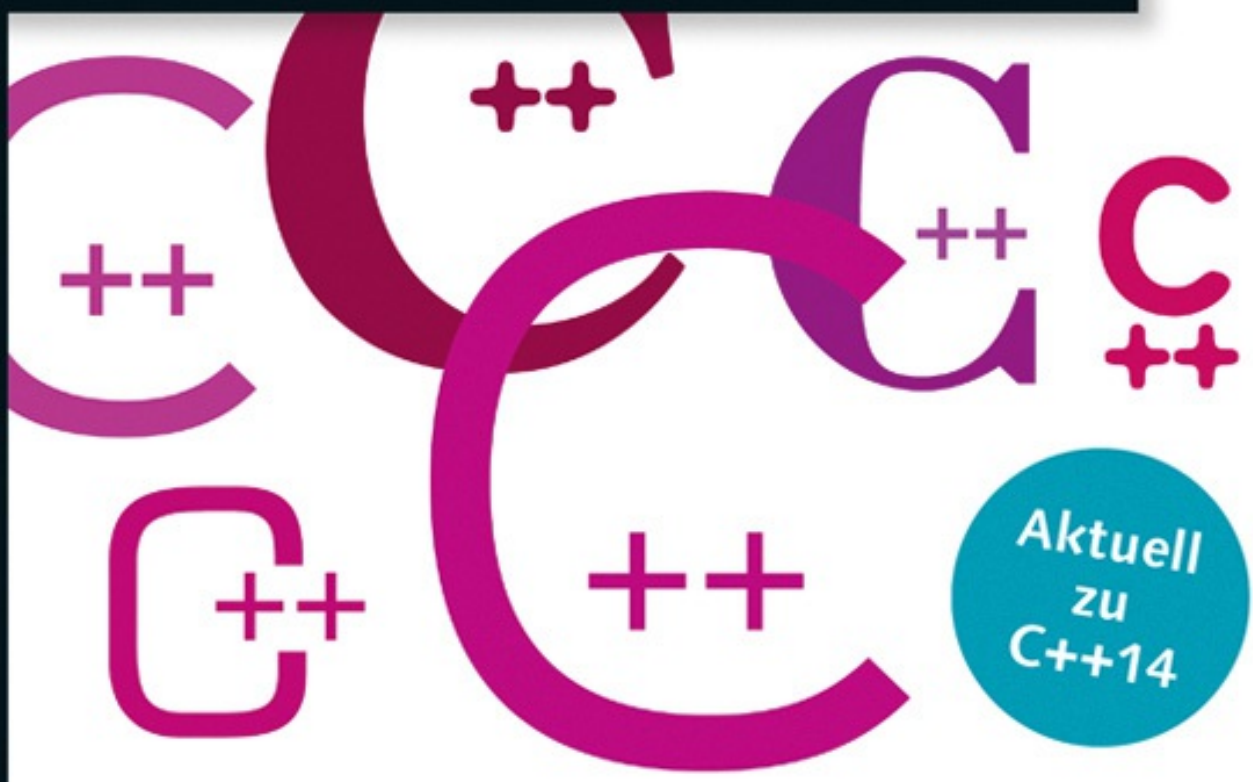
Jürgen Wolf



Grundkurs C++



Aktuell zu C++14

- Eine kompakte Einführung in die Programmiersprache C++
- Vom ersten Schritt bis zum komplexen Programm
- Mit Übungen und Musterlösungen zur Lernkontrolle

3., aktualisierte Auflage

Rheinwerk Computing

1.4 Übersetzen mit g++ und clang++

In diesem Abschnitt möchte ich Ihnen erläutern, wie Sie Kommandozeilen-Compiler wie `g++` oder `clang++` verwenden. Hierbei schreiben Sie Ihren Quelltext mit einem beliebigen ASCII-Editor und speichern diesen mit der Endung **.cpp* ab. In der Kommandozeile übersetzen Sie diesen Quelltext dann mit `g++` oder `clang++`. Gerade bei kleineren Listings wie in diesem Buch ist dieser Vorgang weniger aufwendig, als den Quelltext mit einer Entwicklungsumgebung zu übersetzen.

An dieser Stelle setze ich voraus, dass Sie wissen, was eine Kommandozeile ist und wie Sie sich mit unterschiedlichen Kommandos durch die Verzeichnisse bewegen können. Ein einfache Einführung hierzu finden Sie beispielsweise unter *https://en.wikibooks.org/wiki/Linux_For_Newbies/Command_Line*. Wenn Sie also den Quelltext geschrieben und abgespeichert haben, öffnen Sie die Kommandozeile und wechseln in das Verzeichnis, in dem Sie den Quelltext gespeichert haben. Tippen Sie Folgendes ein (*listing001.cpp* steht hier für den abgespeicherten C++-Quelltext):

```
$ g++ -o listing001 listing001.cpp
$ ./listing001
…
```

Mit dem Schalter `-o` wurde hier bewirkt, dass der Compiler aus der Quelldatei *listing001.cpp* die ausführbare Datei *listing001* macht. Sie können natürlich auch einen anderen Programmnamen wie *listing001* verwenden. In der folgenden Zeile wurde die ausführbare Datei in der Kommandozeile mit dem Programmnamen gestartet.

In der Praxis empfiehlt es sich aber, mehrere solcher Flags wie `-o` zu verwenden, um dem Compiler auch mehr Informationen bei einer Warnung oder einem Fehler zu entlocken. Die von mir favorisierte minimale Zeile zum Übersetzen eines Quelltextes lautet daher mindestens:

```
$ g++ -Wall –pedantic -o listing001 listing001.cpp
$ ./listing001
…
```

Der Schalter `-Wall` zeigt alle sinnvollen Warnungen des Compilers an, und `-pedantic` gibt Warnungen aus, die vom ANSI C++-Standard gefordert werden. Sollten Sie den Quellcode nur kompilieren wollen, ohne ihn zu linken, müssen Sie anstatt `-o` den Schalter `-c` verwenden.

Schalter für C++11 und C++14

Wenn der Compiler nicht schon von Haus aus den C++11-Standard verwendet, können Sie diesen Standard mit dem Flag `-std=c++11` aktivieren. Dasselbe gilt auch für den neueren C++14-Standard, den Sie mit `-std=c++14` oder bei GCC auch mit `-std=gnu++14` aktivieren können. Beispiel:

```
$ g++ -Wall –pedantic -o prg prg.cpp –std=c++14
$ ./prg
…
```

1.5 Listings zum Buch

Zwar wäre es empfehlenswert, wenn Sie die Beispiele im Buch selbst eintippten, aber aus eigener Erfahrung weiß ich, dass bei dem einen oder anderen Beispiel einfach die Lust oder Zeit dazu fehlt. Außerdem habe ich an manchen Stellen auf seitenlangen Code verzichtet und nur die zum Thema passenden Codezeilen ins Buch gedruckt. Daher können Sie die (kompletten) Listings aus dem Buch und auch die Musterlösungen zu den Aufgaben von der Webseite *https://www.rheinwerk-verlag.de/3981* herunterladen. Die Listings zum Buch wurden auf vielen gängigen Systemen (Windows, Linux, OS X) und Compilern getestet.

1.6 Kontrollfragen und Aufgaben im Buch

Sie finden am Ende jedes Kapitels einige Kontrollfragen und Aufgaben. Ziel und Zweck dieser Tests ist es, zunächst einmal zu überprüfen, ob Sie das Gelesene grundsätzlich verstanden haben, und anschließend zu üben, wie Sie das erworbene Wissen anwenden.

1.7 Aufgabe

In diesem Kapitel haben Sie eine sehr wichtige Aufgabe, bevor Sie mit dem Buch fortfahren. Tippen Sie den im Folgenden abgedruckten C++-Quellcode in eine Entwicklungsumgebung oder in einen ASCII-Editor Ihrer Wahl ein. Übersetzen Sie dann das Programm, und bringen Sie es zur Ausführung. Wenden Sie sich dem nächsten Kapitel erst zu, wenn Sie das Programm eingetippt, gespeichert und zur Ausführung gebracht haben. Hierzu das entsprechende Listing:

```
#include <iostream>
using namespace std;

int main() {
  cout << "Das Programm wurde gestartet" << endl;
  return 0;
}
```

Aufpoppende Fenster unter Windows

Wenn Sie ein Programm in einer Entwicklungsumgebung übersetzt haben und es ausführen, kommt es häufig vor, dass hierbei nur kurz die Eingabeaufforderung aufpoppt, das Programm ausgeführt und die Eingabeaufforderung wieder beendet wird. Es gibt jedoch auch Entwicklungsumgebungen, die beim Beenden des Programms die Eingabeaufforderung offen halten. Sollten Sie unter Windows nur ein schnell aufpoppendes Windows-Fenster zu Gesicht bekommen, können Sie die Eingabeaufforderung auch kurz anhalten, indem Sie die dick hervorgehobenen Zeilen zu Ihrem Programm hinzufügen:

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main() {
  cout << "Das Programm wurde gestartet" << endl;
  system("PAUSE");
  return 0;
}
```

Hier wird der alte DOS-Befehl PAUSE eingefügt, der die Ausführung des Programms bis zum nächsten Tastendruck pausieren lässt. Wollen Sie keine solchen Eselsbrücken verwenden, würde es sich auch empfehlen, gleich die Eingabeaufforderung von Windows (*cmd.exe*) zu starten, in das entsprechende Verzeichnis zu wechseln und das Programm dort manuell mit dem Programmnamen zu starten. Zudem bietet beispielsweise MS Visual Studio eine Developer-Eingabeaufforderung an, mit der Sie Ihre Anwendung