IN DIESEM KAPITEL

Das erste Programm und die Methode main

Raus und rein: Bildschirm und Tastatur

Überraschung: Der Computer kann rechnen

Kapitel 1
Grundgerüst eines Programms

Es spricht einiges dafür, dass Sie dieses Buch lesen, um zu lernen, wie Sie Ihren Computer in C++ programmieren können. So etwas geschieht in folgenden Schritten:

1. Sie werden Ihr Programm eintippen müssen.
2. Dabei werden Sie sich wohl oder übel an die *Syntax* halten müssen, die C++ vorgibt und dieses Buch beschreibt.
3. Leider versteht Ihr Computer die Programme so nicht. Der Computer mag wilde Zahlenfolgen, die Menschen nicht eintippen wollen. Ein *Compiler* übersetzt menschengeschriebene Programme in ausführbare Programmdateien für den Computer.
4. Sie starten die ausführbare Programmdatei und hoffen, dass das Programm das tut, was Sie dem Computer sagen wollten. Falls Sie unzufrieden sind, stimmt die *Semantik* nicht.

Sie werden einen Compiler brauchen. Davon gibt es jede Menge für alle möglichen Betriebssysteme. Ihr Chef wird Ihnen sicher gern auf Ihrem Arbeitsplatz einen Compiler zur Verfügung stellen. Sollten Sie keinen so fürsorglichen Chef haben, der sich so rührend um Sie kümmert, sollten Sie Kapitel 24 lesen. Ich warte hier solange auf Sie. Dann können wir anfangen.

Die Funktion main()

Ein C++-Programm besteht mindestens aus der Funktion `main`. Keine Sorge, was eine Funktion ist, wird in Kapitel 7 genau erklärt. In jedem C++-Programm muss irgendwo der Name `main` mit einem Klammernpaar stehen, dem die Silbe `int` voransteht.

```
int main()
{
}
```

Listing 1.1: Ein Minimalprogramm, das nichts tut

In manchen Fällen stehen zwischen den Klammern ein paar kryptisch anmutende Zeichen. Lassen Sie sich nicht verwirren. Was diese Zeichen zwischen den Klammern bewirken, werden Sie noch rechtzeitig erfahren.

```
int main(int argc, char** argv)
{
}
```

Listing 1.2: Ein Minimalprogramm, das nichts tut und komische Zeichen zwischen den Klammern hat

Die Hauptfunktion beginnt direkt nach einer öffnenden geschweiften Klammer und endet mit der passenden schließenden Klammer. Was dazwischen steht, nennt man einen *Block*. In diesem Block zwischen den geschweiften Klammern stehen Anweisungen für den Computer. Sie werden nacheinander ausgeführt.

Kommentare

Wenn wir schon einmal ein Programm haben, das nichts tut, wollen wir es auch um Befehle erweitern, die nichts tun: Kommentare. Sie dienen zur Dokumentation des Programms. Unverständliche Programme werden zu Fehlern führen. Gute Kommentare helfen beim Verständnis.

```
/* Ganz tolles Programm!
Von mir geschrieben! Bitte nicht ohne Lizenz kopieren!
*/
int /* Das ist der Anfang */  main()
{
} // Und das ist das Ende
```

Listing 1.3: Ein Minimalprogramm, das nichts tut

Sie sehen, es gibt zwei Arten von Kommentaren.

- ✔ // Der Kommentar gilt bis zum Ende der Zeile.
- ✔ /* Alles dazwischen ist Kommentar. */

Nicht verschachtelbar

/* und */ umschließen einen Kommentar. Sie können die Kommentarklammern allerdings leider nicht verschachteln. Mit dem ersten Auftreten der Kombination /* beginnt der Kommentar. Er endet mit dem ersten Auftreten der Kombination */. Dazwischen liegende Kommentaranfangzeichen gehören zum Kommentar. Der Compiler wird den folgenden Text als Programm ansehen und versuchen, ihn zu übersetzen.

```
int main()
{
  /* Hier beginnt der Kommentar
     /*
       Die nächste Zeile braucht kein
       eigenes Kommentarzeichen
      */
     Dies wird der Compiler wieder übersetzen wollen.
  */
}
```

Listing 1.4: Das geht schief!

Die Kommentare sollen dokumentieren, warum das Programm so und nicht anders geschrieben wurde, und richten sich an Programmierer. Dies können Kollegen sein, die Ihr Programm korrigieren oder erweitern sollen. Aber noch öfter helfen Sie sich selbst damit. In der Praxis ist es so, dass Sie vermutlich auch herangezogen werden, wenn eines Ihrer Programme nicht korrekt läuft oder ergänzt werden soll. Da Sie zwischendurch andere Programme geschrieben haben werden, vielleicht geheiratet haben, umgezogen sind und noch drei weitere Programmiersprachen gelernt haben, werden Sie dankbar sein, wenn Sie in den Programmen ausführliche Hinweise finden, wie und warum es so funktioniert oder zumindest funktionieren soll.

Goldene Regel der Kommentierung: Kommentieren Sie nicht, *was* Sie gemacht haben, sondern *warum* Sie es so gemacht haben!

Ausgabe für Anfänger

Das erste Fachbuch über C zeigte als erstes Programm eines, das »Hello World!« auf dem Bildschirm ausgab. Seither grüßen fast alle Programmierbücher erst einmal die Welt. Ich finde, dass Munkbrarup in den Fachbüchern dieser Welt zu kurz kommt.

```
#include <iostream>

int main()
{
    std::cout << "Hallo Munkbrarup" << std::endl;
}
```

Listing 1.5: Kleiner Gruß

In der ersten Zeile sehen Sie die Anweisung an den Compiler, die Datei *iostream* an dieser Stelle einzufügen. Solche `#include`-Anweisungen führen dazu, dass das Programm die Funktionen einer Bibliothek kennenlernt. Das Grußprogramm findet dort heraus, wer eigentlich `std::cout` ist.

Nachdem der Compiler nun weiß, wer `std::cout` ist, wollen Sie es vermutlich auch wissen.

- ✔ `cout` steht für »character out«. Es handelt sich um den Ausgang, zu dem die Zeichen auf den Bildschirm geschickt werden. Und damit sie den Weg auch sicher finden, weisen zwei Kleinerzeiichen dem Gruß an Munkbrarup den Weg.
- ✔ `std` ist ein Namensraum und besagt, dass `cout` unter dem Suchbegriff `std` zu finden ist.
- ✔ `::` ist der Trenner zwischen Namensraum und Bezeichner.

Am Ende der Zeile finden Sie mit `std::endl` noch das Zeilenendezeichen. Damit werden nachfolgende Ausgaben in die nächste Zeile geschickt.

Ansonsten sind die Anführungszeichen noch bemerkenswert. Sie umrahmen jedes *Literal* einer Zeichenkette. Damit weiß der Compiler, dass Hallo und Munkbrarup keine Befehle sind, sondern zu einem Text zusammengefasst werden sollen.

Variante mit using namespace

Die Symbole in *iostream* beginnen mit `std::`. Das bedeutet, dass sich `cout` und `endl` im Namensraum `std` befindet. Sie können sich das Eintippen von `std::` sparen, wenn Sie mit der Anweisung `using namespace` den Namensraum `std` einbinden.

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

int main()
{
    cout << "Hallo Munkbrarup" << endl;
}
```

Listing 1.6: Ein kleiner Gruß mit Namensraum

Zahlenspielereien

Sie können nun zwischen die Anführungszeichen (beinahe) beliebige Texte setzen. Aber noch interessanter ist es, Zahlen einzusetzen. Wenn Sie zwischen die Zahlen noch Operatoren wie das Plus- oder Minuszeichen setzen, kann Ihr Programm sogar rechnen.

```
#include <iostream>

int main()
{
    std::cout << 1268 + 973 << std::endl;
}
```

Listing 1.7: C++ kann rechnen

Wenn Sie das Programm übersetzen und starten, wirft es die Zahl 2241 auf dem Bildschirm aus. Da es auf Tastaturen keinen mittigen Punkt gibt, behelfen sich die meisten Programmiersprachen mit einem Stern als Operator für die Multiplikation. Für die Division hat sich der Schrägstrich durchgesetzt. Auch die Klammern können Sie genau wie in der Mathematik üblich einsetzen.

Sogar Nachkommastellen sind möglich, sofern Sie damit zurechtkommen, dass das Komma ein Punkt ist.

```
#include <iostream>

int main()
{
    std::cout << (1.2 + 9.3) * 3.14159 << std::endl;
}
```

Listing 1.8: C++ kann auch Nachkommastellen berechnen.

Übungen

So wie nur selbst essen fett macht, lernen Sie das Programmieren nur dadurch, dass Sie es selbst tun. Aus diesem Grund habe ich mir einige Aufgaben ausgedacht. Aber werden Sie kreativ und verwirklichen Sie auch eigene Programmideen!

1. Schreiben Sie Listing 1.5 ab. Übersetzen und starten Sie es. In Kapitel 24 ist beschrieben, wie Sie Ihre Arbeitsumgebung einrichten.
2. Schreiben Sie ein neues Programm, das die beiden Wörter »Hallo Benutzer« auf dem Bildschirm ausgibt.
3. Ändern Sie das Programm in Listing 1.7, indem Sie andere Zahlen, Operatoren und Klammern einsetzen. Überlegen Sie, was Ihre Änderung bewirken, und probieren Sie, ob Ihre Erwartungen erfüllt werden. Falls Sie eine Fehlermeldung des Compilers erhalten, lesen Sie die Meldung. Versuchen Sie, den Zusammenhang zwischen Ihrer Änderung und der Compiler-Meldung herzustellen.