C für Kids

von
Hans-Georg Schumann

1. Auflage

Thematische Gliederung:

Prozedurorientierte Programmierung

mitp/bhv 2009

Verlag C.H. Beck im Internet:
www.beck.de
ISBN 978 3 8266 8670 2

C

FÜR KIDS

Hans-Georg
Schumann

Auf der CD:
Alle Beispieldateien sowie eine vollständige Entwicklungsumgebung für Windows und Linux

mit CD

bhv

1 Rezepte und Algorithmen

Programmieren kannst du in diesem Kapitel leider noch nicht. Hier geht es erst einmal nur um Gebrauchsanleitungen und Rezepte. Denn ähnlich müssen auch die Anweisungen an den Computer aussehen, damit der versteht, was wir ihm mitteilen wollen. Gemeint sind natürlich nur Anleitungen und Rezepte, die aus klaren Anweisungen bestehen. Während wir uns auf so manche verschwommene Formulierung noch »einen Reim« machen können, kommt ein Computer mit Unklarheiten nicht zurecht.

Um auch dem Computer etwas klarmachen zu können, brauchen wir eine entsprechende Programmierumgebung. Die gibt es auf der CD zum Buch.

In diesem Kapitel lernst du

- etwas über Gebrauchsanleitungen und Rezepte
- was Algorithmen und Flussdiagramme sind
- was ein Programm ist
- wozu ein Entwicklungssystem gut ist
- was Editor, Compiler und Linker sind

A wie Auto

Vor dir steht ein Auto. Du darfst auch mal einsteigen und es dir auf dem Fahrersitz gemütlich machen. Bisher bist du wohl noch nicht selbst gefahren, hast aber schon oft danebengesessen, wenn Vater oder Mutter mit dem Auto fuhr.

Da siehst du am Armaturenbrett einen Zettel:

Losfahren:
1. *Kupplung rein*
2. *Gang rein*
3. *Kupplung raus und Gas geben*

Anhalten:
1. *Kupplung rein und bremsen*
2. *Gang raus*

Um diese Gebrauchsanleitung zu verstehen, muss man erst einmal wissen, was »Kupplung« und »Gang«, »Gas« und »Bremse« bedeuten. Du meinst, das weiß doch jedes Kind? Und trotzdem kann es sein, dass man für die Bedienung dieser Elemente noch ein paar genauere Anweisungen braucht.

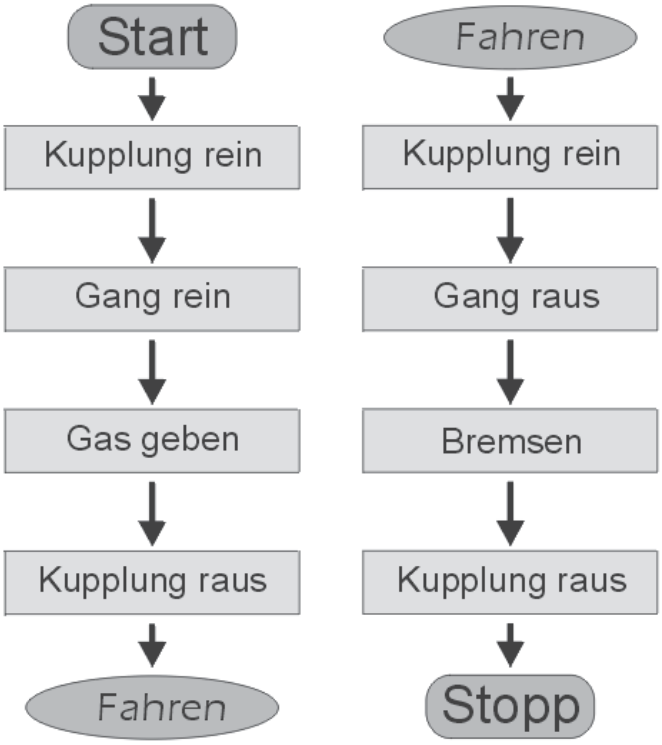
Vielleicht könnte dann diese Tabelle weiterhelfen:

Kupplung rein	Linkes Pedal (langsam) treten
Kupplung raus	Linkes Pedal (langsam) loslassen
Gang rein	Schalthebel nach vorn drücken
Gang raus	Schalthebel in die Mitte ziehen
Gas geben	Rechtes Pedal (langsam) treten
Bremsen	Mittleres Pedal (langsam) treten

Offenbar handelt es sich um drei Pedale und einen Schalthebel, mit dem man ein Auto zum Fahren bringt und wieder anhält. Das ist aber natürlich nicht alles, denn lenken muss man das Auto auch noch:

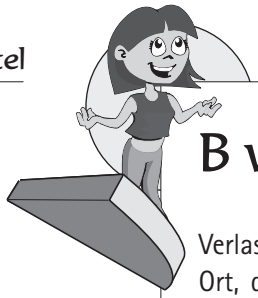
Geradeaus fahren	Lenkrad in der Mittelposition halten
Nach links fahren	Lenkrad nach links drehen
Nach rechts fahren	Lenkrad nach rechts drehen

Wenn wir wissen, was die einzelnen Begriffe bedeuten, und etwas mit den Anweisungen anfangen können, lassen sich die Informationen fürs Anfahren und Anhalten in einem solchen *Schaubild* zusammenfassen – auch *Flussdiagramm* genannt:



Eine Folge von Anweisungen bezeichnet man als *Algorithmus*, wenn sämtliche Anweisungen *eindeutig* ausführbar sind. Das kann man von den obigen Anweisungen zum Fahren eines Autos aber nicht unbedingt sagen. Es hängt eben davon ab, wie weit der Ausführende (also z. B. du als Fahrer) Bescheid weiß.

Was heißt hier eindeutig? Wenn jemand *genau* weiß, wie er eine Anweisung ausführen soll, ist die Aussage für ihn eindeutig. So wäre eine Aussage wie »Fahr nicht so schnell« sicherlich unklar. Mit »Fahr nicht mehr als 50 km/h« dagegen weiß man schon eher etwas anzufangen. Ganz eindeutig wäre die Anweisung »Fahr exakt 50 km/h.« Voraussetzung ist in jedem Falle, dass man mit einem Auto umgehen kann.



B wie Brot

Verlassen wir jetzt wieder das Auto und besuchen wir mal die Küche, einen Ort, der dir sicher vertraut ist. Auch hier liegt wieder einer dieser Zettel herum:

Wir brauchen:
200g Roggenmehl, 400g Weizenmehl,
1 Päckchen Hefe, 1 Esslöffel Salz,
400 ml Wasser (40 Grad heiß)

Um eine Anleitung handelt es sich diesmal nicht. Eher um eine *Aufzählung* von Zutaten. Allzu viel damit anfangen können wir allerdings nicht. Einfach alles in eine Schüssel tun und verrühren dürfte wohl nicht reichen.

Wenn man noch nie versucht hat, zu kochen oder zu backen, weiß man wahrscheinlich nicht einmal, dass es sich hier um Zutaten fürs Brotbacken handelt. Und erst wenn man noch die Backanleitung hat, ist das *Rezept* komplett:

- Mische die beiden Mehlsorten in einer Schüssel.
- Löse die Hefe in etwas heißem Wasser auf, gieße sie über das Mehl und verrühre das Ganze.
- Gieße das restliche heiße Wasser dazu und rühre dabei weiter.
- Streue das Salz in die Schüssel und verrühre es mit dem Teig.
- Am Ende muss sich der Teig von der Schüssel lösen lassen und darf nicht an den Fingern kleben. Sonst musst du entweder noch etwas Mehl oder etwas Wasser dazugeben.
- Decke den Teig mit einem Tuch zu und lass ihn eine Dreiviertelstunde stehen (und »gehen«).
- Nimm den Teig aus der Schüssel, knete ihn nochmals durch und forme daraus auf einem Backblech ein Brot (oder mehrere Brötchen).
- Decke das Ganze wieder mit einem Tuch zu und lass es eine Dreiviertelstunde stehen (und »gehen«).

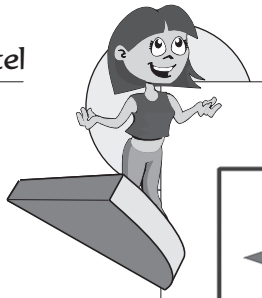
- Schiebe das Backblech in den vorgeheizten Ofen.
- Wenn es ein Brot werden soll, lass es 60 Minuten backen,
- wenn es Brötchen werden sollen, lass sie 30 Minuten backen.
- Hol das Backwerk aus dem Ofen und lass es abkühlen.

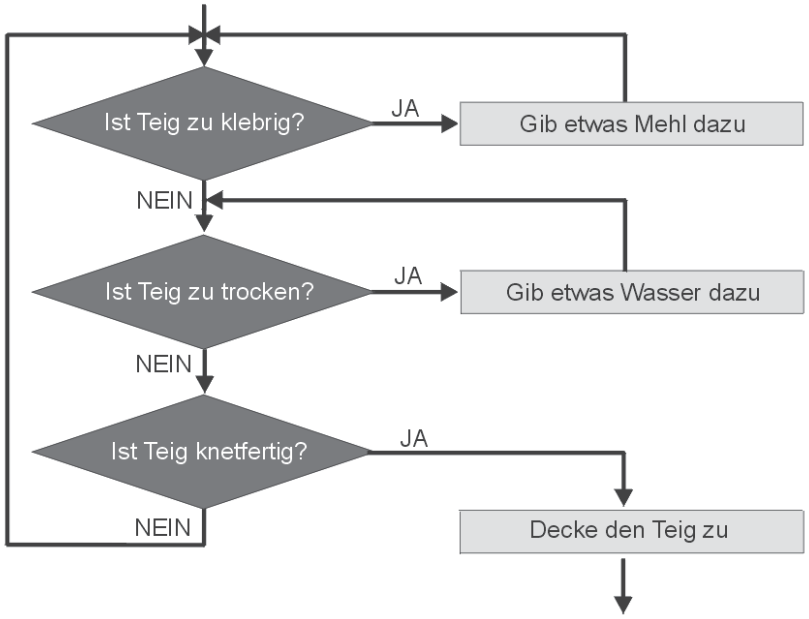
Ich hätte diese Anleitung natürlich viel eleganter formulieren können. Aber es kommt hier nicht auf eine schicke Wortwahl an, sondern an erster Stelle steht die Eindeutigkeit. Und trotz dieser Anleitung ist es noch gar nicht sicher, ob man damit auch ein schmackhaftes Brot (oder Brötchen) hinbekommt. (Wenn du willst und deine Eltern einverstanden sind, kannst du es ja mal selber ausprobieren.)

Auch hier fassen wir alle Anweisungen wieder in einem *Flussdiagramm* zusammen. Weil dieses recht lang gerät, machen wir daraus mehrere Teile. Im ersten Abschnitt werden erst einmal die Zutaten vermischt:

Zum Schluss musst du entscheiden, in welchem Zustand sich der Teig befindet:

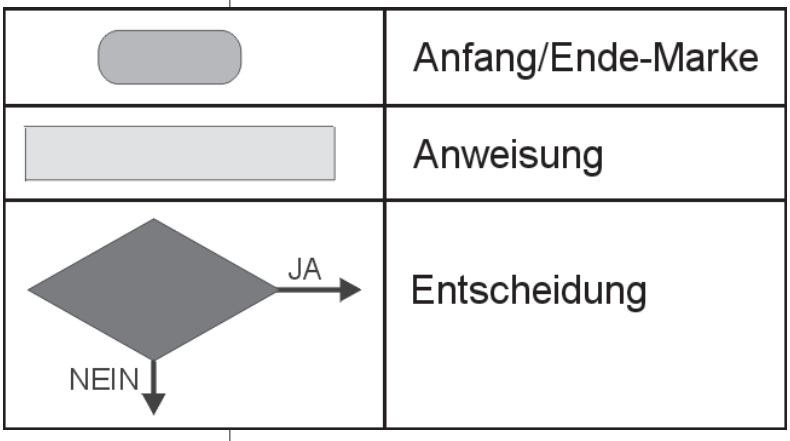
- Ist er zu klebrig, muss noch etwas Mehl hinzu.
- Ist er zu trocken, muss noch etwas Wasser hinzu.
- Ist er fertig, kann er zugedeckt werden (und es geht weiter).





Im Vergleich zum Anfangsteil sieht es hier doch ein wenig verwirrend aus – oder?

Bekannt sind bis jetzt zwei Symbole: Eines markiert entweder Anfang und Ende des gesamten Prozesses, das zweite veranschaulicht einen Ausführungsschritt (bzw. eine Anweisung). Nun taucht ein neues Symbol auf, das ein bisschen komplizierter ist:

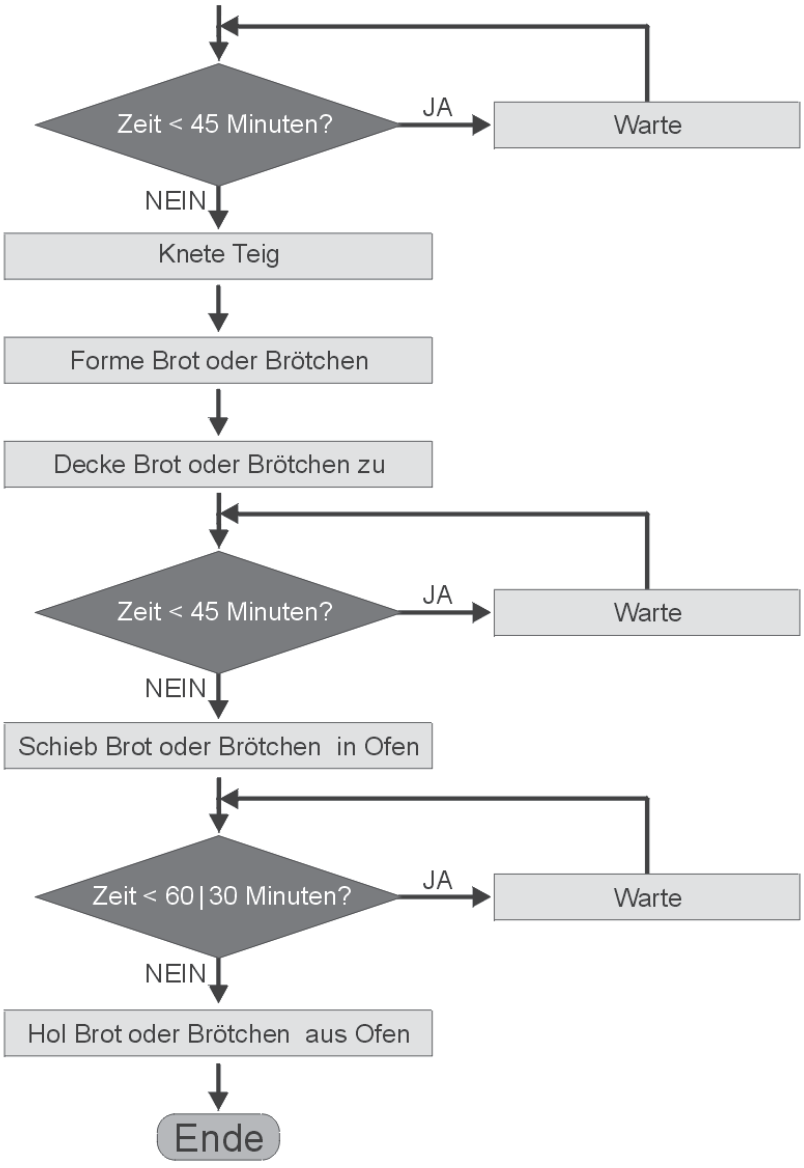


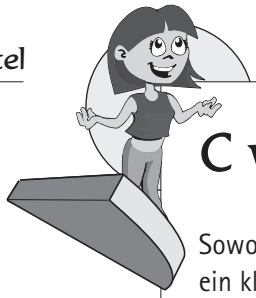
Was bedeutet hier *Entscheidung*? Zuerst wird eine Frage gestellt, die sich eindeutig mit JA oder NEIN beantworten lassen muss – z. B. »Ist der Teig knetfertig?« Dann wird je nach Lage entschieden (den Weg weisen die Pfeile):

- Wenn *NEIN*, dann wird der ganze Prozess wieder nach oben gelenkt (und alle Fragen müssen noch einmal gestellt und beantwortet werden).
- Wenn *JA*, kann es weitergehen zum nächsten Abschnitt.

Solltest du also so ungeschickt sein, mal zu viel Mehl, mal zu viel Wasser hinzuzuschütten, dann nimmt dieser Prozess theoretisch kein Ende. (In der Praxis könnte aber dann der Mehlvorrat erschöpft sein oder du könntest keine Lust mehr haben.)

Ähnlich verlaufen dann die restlichen Schritte bis zum fertigen Brot (bzw. den Brötchen) aus dem Backofen:





C wie Computer

Sowohl beim Auto als auch beim Brotbacken haben wir eigentlich schon ein kleines Programm, wie es auch von einem *Computer* verarbeitet werden könnte.

Computer gibt es ja schon fürs Autofahren und auch fürs Brotbacken. Man braucht nur noch je ein Pedal zum Anfahren und zum Anhalten. Alles andere wie den Umgang mit Kupplung und Schaltung regelt ein Computer. Neue Autos werden nur noch mit Computern ausgeliefert, die viele Dinge übernehmen, von denen wir oft gar nichts merken.

Und in Großbäckereien werden Computer mit Rezepten gefüttert und steuern Maschinen, die die passenden Brote, Brötchen und Kuchen backen.

Du weißt natürlich genau, was ein Computer ist, kurz meist auch *PC* genannt? Dennoch gibt es hier eine kurze Erläuterung, die du auch überspringen kannst:

Der Name kommt aus dem Englischen und heißt auf Deutsch eigentlich so viel wie Rechner. Man könnte einen Computer also auch *Rechenmaschine* nennen. Alles, womit man ihn füttert, wandelt er in ein eigenes Zahlensystem um. Damit rechnet er dann. Und was am Ende dabei herauskommt, kann z. B. ein Bild sein oder ein Text (aber natürlich auch Zahlen). Genannt wird das Ganze *Daten*. Ganz schlaue Leute sagen daher zum Computer auch *Datenverarbeitungsanlage*.

Damit ein Computer überhaupt rechnen kann, braucht er ein *Programm*, das ihm vorschreibt, wie er zu rechnen hat. Unter einem Programm versteht man einfach eine Sammlung von (ausführbaren) Anweisungen bzw. einen Algorithmus, der an einen Computer angepasst ist.

Weil im Grunde genommen damit das Gleiche gemeint ist, werde ich immer mal wieder *PC* statt *Computer* schreiben.

Wenn wir unser Rezept zum Brotbacken noch mal genauer unter die Lupe nehmen, so besteht dieses »Programm« aus zwei wesentlichen Teilen:

WAS	Vereinbarungsteil (die Zutaten)
WIE	Anweisungsteil (die Verarbeitung)

Für einen Computer ergibt sich dann dieses Schema:

ZUTATEN:

Mehl, Wasser, Hefe, Salz ...

START

Verarbeitung der Zutaten zu Teig

Verarbeitung des Teigs zu Brot/Brötchen

ENDE

Weder damit noch mit dem obigen Rezept noch einem Flussdiagramm jedoch kann ein Computer etwas anfangen. Die Anweisungen sind zu unklar und ungenau. Mischen und Rühren wären wohl noch recht eindeutig (über entsprechende Maschinen) zu erledigen, und auch das Messen von Mengen, Temperaturen und Zeit ließe sich ziemlich genau durchführen.

Aber schon beim Unterschied zwischen Kneten und Formen sowie bei Begriffen wie »Gehen lassen« wird es knifflig: Der Computer wäre überfordert.

Ein Programm muss also

- in seinen *Anweisungen eindeutig* ausführbar und
- in seinen *Bedingungen klar* testbar sein, um dann entsprechende Entscheidungen treffen zu können

Unser PC muss ja auch gar kein Brot backen können (oder ein Auto zum Fahren bringen). Deshalb nehmen wir uns ein einfacheres Beispiel vor, zu dem jeder Computer fähig sein sollte.

Computer sind eigentlich strohdumm, auch wenn sie eine Menge (einfacher Dinge) können und das oft ausgesprochen schnell und genau. Und wenn du dennoch meinst, Computer seien besonders schlau, so läuft eben gerade ein tolles Programm ab, das ein intelligenter Programmierer erstellt und dem PC zur Ausführung gegeben hat.

Wie wäre es mit einem kleinen Programm, das es dem PC ermöglicht, mit uns ein kurzes Gespräch zu führen?

Zuerst soll der PC etwas sagen, nämlich »Hallo, ich bin dein PC.« Dann soll er fragen: »Und wer bist du?«

Dazu braucht er einen `SCHREIB`-Befehl. Das heißt: Er soll etwas auf dem *Bildschirm* anzeigen. Denn das ist sozusagen sein »Mund«, mit dem er zu uns spricht.

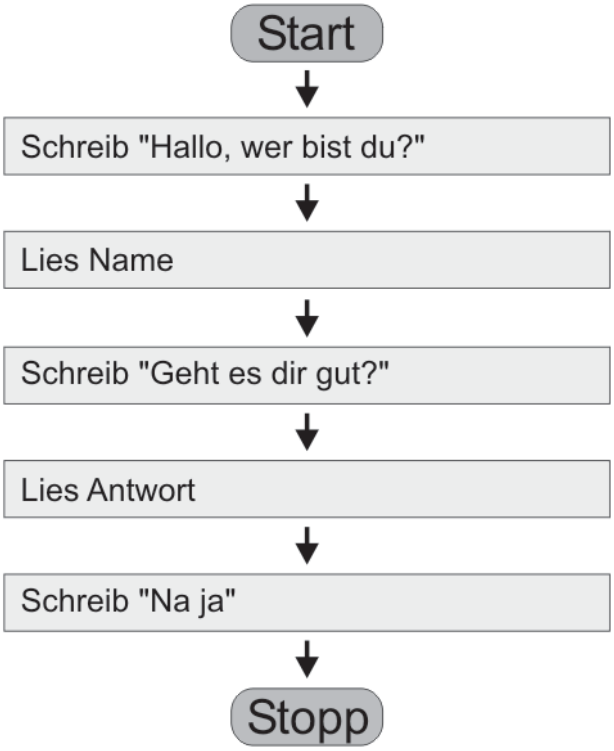
```
Hallo, wer bist du?
Georg
Geht es dir gut?
nein
Na ja
```

Dann soll der PC uns zuhören. Dazu bekommt er einen LESE-Befehl. Das heißt: Wir teilen ihm etwas über die *Tastatur* mit. Denn dies ist sozusagen sein »Ohr«, über das er uns hört.

Anschließend soll er uns fragen, ob es uns gut geht (Schreiben), unsere Antwort dazu hören (Lesen) und abschließend noch etwas sagen (Schreiben).

Diesen ständigen Wechsel zwischen Schreiben und Lesen bzw. Sagen und Hören kann man eigentlich schon als Gespräch bezeichnen.

Sehen wir uns dazu doch mal das Flussdiagramm an:

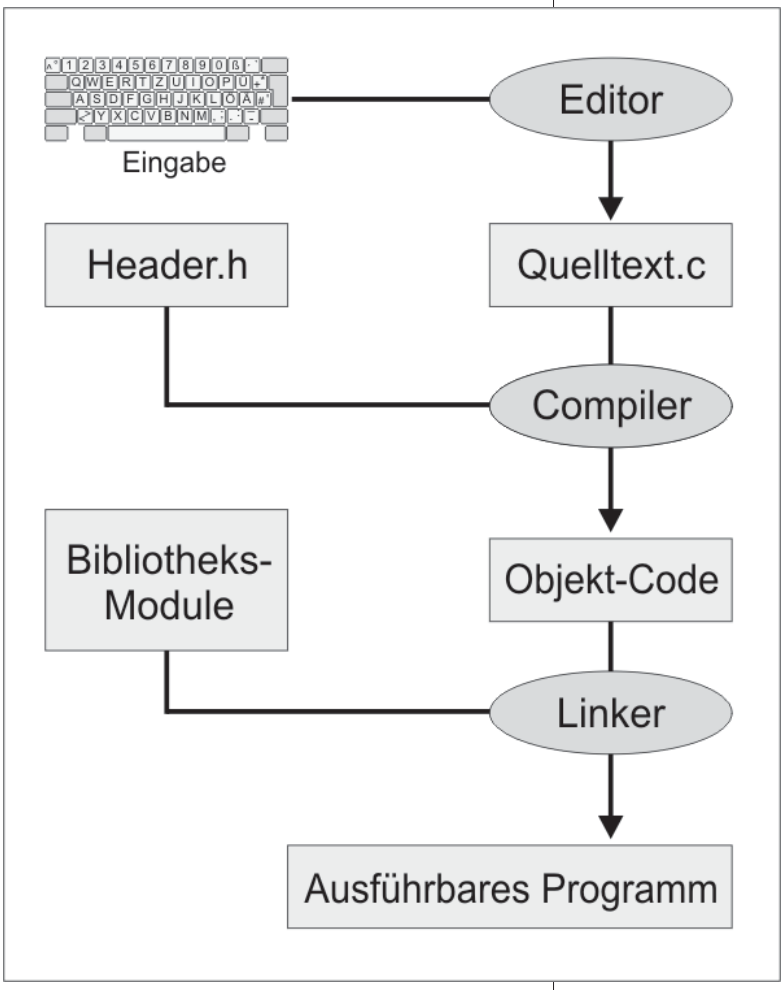


Um dem Computer so etwas klarzumachen, müssten wir mit ihm in einer Sprache sprechen, die *er* versteht. Die aber verstehen *wir* nicht. Aber wozu gibt es denn Dolmetscher, die zwischen uns beiden vermitteln können? Ein solcher Dolmetscher gehört zu einem *Programmiersystem* oder *Entwicklungssystem*. In diesem Buch benutzen wir ein System mit dem Namen *Eclipse*. Das kannst du dir von der beigefügten CD installieren. Hilfe bekommst du im Anhang B (für Windows) oder C (für Linux).

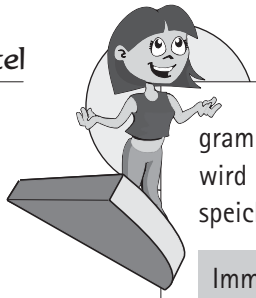
Von der Eingabe bis zum Programmstart

So wie im oben stehenden Beispiel kann man nicht programmieren. Man benötigt dazu eine besondere Sprache. Davon gibt es eine ganze Menge. Wir haben es hier mit der Programmiersprache C zu tun, die ja auch zur Grundlage für viele weitere Sprachen geworden ist.

Mit der Eingabe eines Textes ist es nicht getan. Deshalb sorgt ein Entwicklungssystem dafür, dass daraus schließlich ein funktionierendes Programm wird. Wie du im folgenden Diagramm sehen kannst, sind dazu einige Stationen nötig – ähnlich wie auf dem Weg bis zum fertigen Brot:



Im *Editor* kannst du einen Text eingeben und bearbeiten, also z. B. Teile davon löschen, kopieren und verschieben oder neu eintippen. Der Pro-



grammtext wird auch *Quelltext* genannt. Da wir hier in C programmieren, wird das Ergebnis unserer Arbeit in einer Datei mit der Kennung »C« gespeichert, z. B. PROJEKT1.C.

Immer wieder mal wirst du beim Programmieren dem Begriff *Syntax* oder einer Fehlermeldung wie »Syntax Error« begegnen.

Die Syntax ist für einen Algorithmus oder einen Programmtext so etwas wie Rechtschreibung und Grammatik. Wird z. B. eine Anweisung falsch geschrieben oder ein Symbol oder Zeichen vergessen, dann führt das zu einem Syntaxfehler: Der Dolmetscher weiß nicht, wie er den Text für den PC übersetzen soll, wenn er ihn nicht mal selber versteht.

Eine Quelltextdatei allein ist noch ziemlich mager, sie enthält ja nur unsere Anweisungen. Damit die auch für einen Computer ausführbar sind, brauchen wir fertige Module, in denen z. B. genau drinsteht, wie ein PC etwas von der Tastatur lesen und etwas auf den Monitor schreiben kann.

Deshalb gibt es mit der so genannten *Headerdatei* eine weitere Datei, dort stehen u. a. Hinweise auf zusätzlich benötigte Module. Während die Quelltextdatei also das Mehl enthält, bringt eine Headerdatei das Salz mit.

Beides wird vom *Compiler* übernommen und in eine Form umgewandelt, die der Computer versteht. Sie wird als *Objekt-Code* bezeichnet. Das ist sozusagen die Basismischung, aus dem nachher das fertige Programm gebacken wird.

Man kann auch sagen: Der Compiler übersetzt unsere Hochsprache (die wir verstehen) in die so genannte *Maschinensprache* (die der PC versteht).

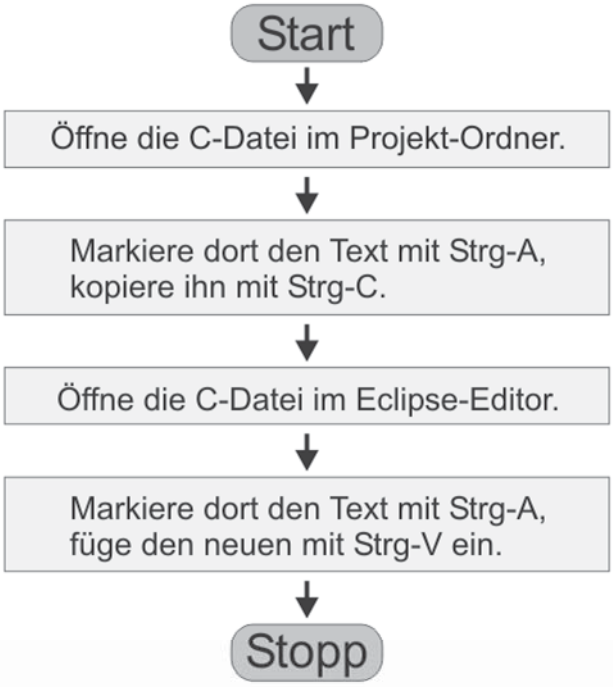
Damit nachher auch ein fertiges Brot, äh ein lauffähiges Programm daraus wird, muss nun noch der *Linker* weitere Module mit der Objektdatei verknüpfen, den so genannten Bibliothek-Modulen (englisch: Libraries). Erst jetzt kann ein Programm richtig funktionieren. Um im Bild mit dem Brot zu bleiben: Die Bibliotheken bringen das Wasser und der Linker formt den Teig und backt ihn.

Und während uns am Schluss ein frischgebackenes Brot erwartet, kommt für den Computer ein ausführbares Programm heraus.

Mit Eclipse und ein paar damit verbundenen Hilfsmitteln haben wir ein Entwicklungssystem, mit dem sich nach der Eingabe des Quelltextes alles Weitere automatisieren lässt – wie du schon bald sehen wirst.

Allerdings muss das Ganze zuvor von der CD auf deinem PC installiert werden. Näheres dazu findest du in Anhang B (Windows) oder C (Linux).

Hier aber ist schon mal der Algorithmus für die Verwendung der Projektdateien von der CD:



Genaueres dazu steht im Anhang D.

Zusammenfassung

Erst einmal verdienst du eine Pause. Immerhin hast du einen Trockenkurs in Sachen Algorithmen und Programme hinter dir. Erst im nächsten Kapitel machen wir uns endlich daran, dem Computer unser Programm näher zu bringen.

Du weißt, dass ein Computer immer nur eindeutige Anweisungen akzeptiert und dir ein Entwicklungssystem wie Eclipse beim Erstellen und Übersetzen deiner Programme behilflich sein kann – was du im nächsten Kapitel ja selbst ausprobieren sollst. Wenn du vorher aber noch etwas Arbeit suchst, dann gibt es die schon hier:

Ein paar Fragen ...

1. Welche Unterschiede bestehen für dich zwischen einem Rezept und einer Gebrauchsanleitung?
2. Was ist ein Algorithmus, was ein Programm?
3. Was ist ein Compiler und was ist ein Linker?

... und ein paar Aufgaben

1. Entwirf Algorithmen für folgende Probleme:
   a) Wie fahre ich Fahrrad?
   b) Wie fahre ich Mofa, Moped oder Motorrad?
   c) Wie wechsle ich ein Rad bzw. einen Reifen?
   d) Wie tanke ich?
   e) Wie koche ich Kaffee, Tee oder Eier?
2. Beschreibe in einem Algorithmus den Weg von deinem Zuhause zur Schule oder Arbeitsstelle.