# Motivation und Hilfestellung beim Programmieren allein zu Hause

Programmieren ist wirklich nicht schwer, man muss sich nur ein bisschen Zeit nehmen und darf nicht zu früh aufgeben. Dennoch sollte man auch nicht zu lange (mehr als 15 Minuten) an einem Fehler hängen bleiben, sondern sollte hier in diesem Dokument nach Hilfe suchen. Ich bin mir ziemlich sicher, Sie finden eine Lösung.
Hier ein Startvideo zur Einführung: https://www.youtube.com/watch?v=-7Xxokla4UU


Coco startet mit der C Programmierung Teil 3 - Das Flussdiagramm

Wie komme ich von der Aufgabe zum Programm?

1. Ich versuche zu verstehen, was die Aufgabe ist, was das Programm also machen soll. Dies schreibe ich als erste Zeile in mein Programm oben rein hinter Kommentarstriche: //
   Beispiel: `//es sollen 2 Zahlen dividiert werden`
2. Ich denke mir einen Testfall aus, wie ich das Programm, wenn es fertig ist, testen will. Dies schreibe ich als zweite Zeile oben in mein Programm hinter Kommentarstriche: //
   Beispiel: `//a=3, b=5, Ergebnis=0.6`
3. Ich überlege mir, welche Eingaben es geben muss und von welchem Datentyp sie sein müssen. Dies hat nichts mit Programmierung zu tun, sondern meistens mit Mathe. Weiterhin denke ich mir schöne Variablennamen aus, in die ich die Eingaben speichern will.

Beispiel:

```
//beide Werte in double, da Division nicht ganzzahlig
double dividend, divisor;
```

4. Ich überlege mir, welche Ausgabe mein Programm haben soll und von welchem Datentyp diese sind. Weiterhin denke ich mir schöne Variablennamen aus, in die ich die Ausgaben speichern will, bevor ich sie am Bildschirm anzeige.
   Beispiel:

```
//Division nicht ganzzahlig
double Division;
```

5. Ich programmiere die Eingaben und die Ausgaben.
   Beispiel:

```
printf("Dividend: ");
scanf("%lf",&dividend);
printf("Divisor: ");
scanf("%lf",&divisor);
…
printf("Die Division ist %lf:", division);
```

6. Jetzt programmiere ich den Rest ganz einfach, in dem ich
   - einem Flussdiagramm folge oder
   - nach Schleifen suche (Wiederholungen → Verwendung von `while` oder `for`) und nach Verzweigungen (Ja/Nein-Fragen→ Verwendung von `if/else`)
   - eine Formel von außen nach innen programmiere oder
   - per Hand die Aufgabe löse und dann alle Schritte nachprogrammiere
   - oder …

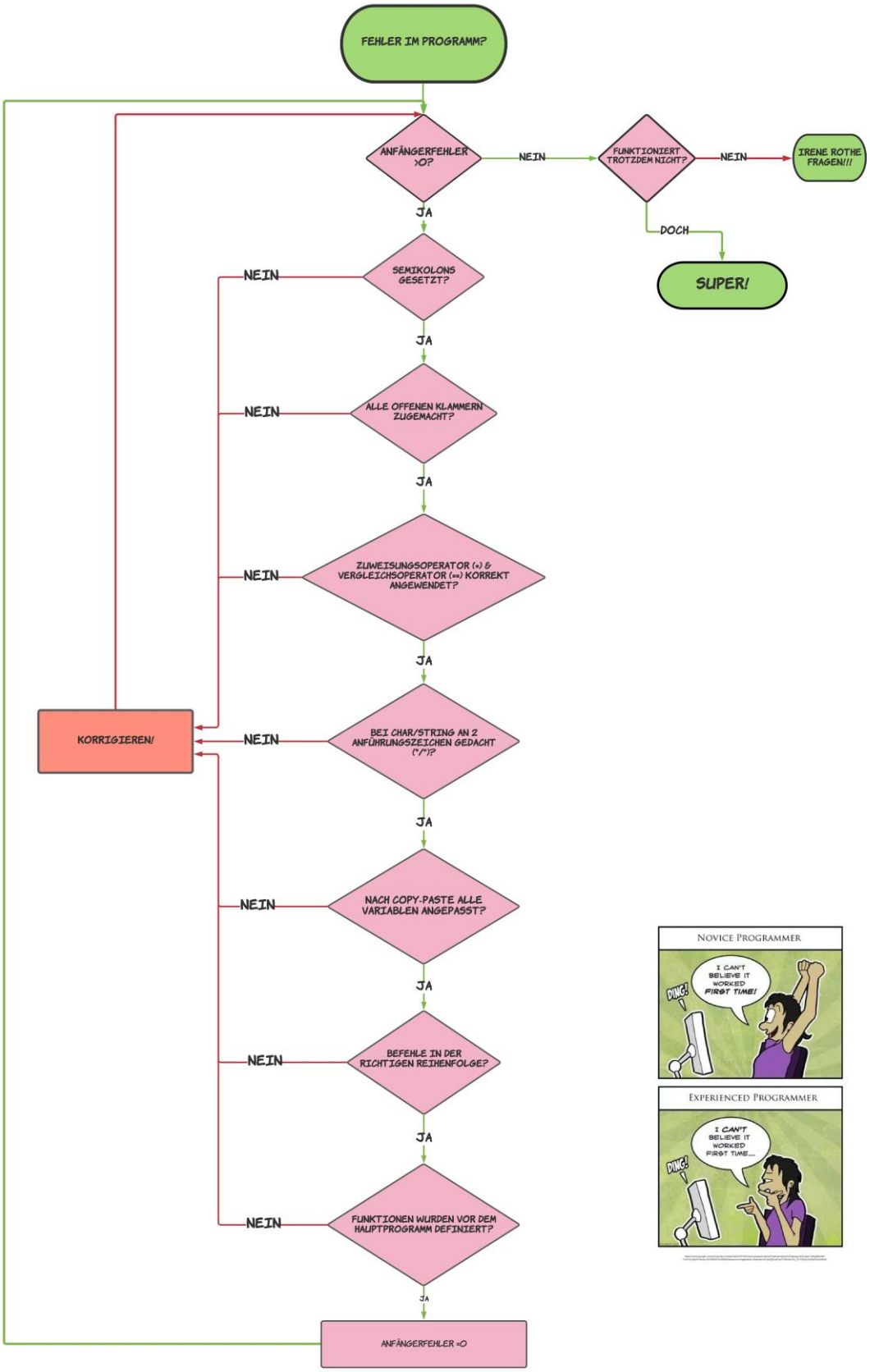
   Dabei immer an die Mathematikregeln denken!
   Beispiel:

```
//Division durch Null nicht erlaubt
if(divisor==0){
  printf("Division durch Null nicht möglich! ");
}
else{
  division=divident/divisor;
  printf("Die Division ist %lf: ", division);

}
```

Flussdiagramm: Malen Sie eins, das spart viel Arbeit und hilft total! Ich sage das aus Erfahrung! Wie male ich ein Flussdiagramm → siehe Vorlesung_Grundlagen am Ende!

# Fehlersuche = Lösen von Problemen!



Hilfe zur Fehlersuche in C-Programmen:

- Sehen Sie sich immer nur den ersten Fehler in Ihren Fehlermeldungen im Compiler an, alle anderen Fehler sind in der Regel Folgefehler. Haben Sie den ersten Fehler repariert, übersetzen Sie Ihr Programm erneut.

- Sind alle C-Schlüsselwörter richtig geschrieben? (Wenn ja, zeigt der Editor sie in einer bestimmten Farbe an.)

- Gibt es eine *main*-Funktion? (`int main(){ return 0; }`)

- Steht nach jeder Anweisung ein Semikolon ;? (Keine Anweisungen sind: Bedingungen/Fragen)
  **Tipp:** Es steht niemals ein Semikolon VOR einer offenen geschweiften Klammer!

- Gibt es gleich viele offene und geschlossene Klammern (egal ob runde oder geschweifte)? (In der Regel vergisst man sehr schnell geschweifte Klammern jeder Art.)

- Bei der *while*- oder *do-while*- Kontrollstruktur unbedingt auf das Abbruchskriterium achten! Es sollte irgendwann eintreten! (Sonst erhält man eine Endlosschleife, also gar keinen Algorithmus.)

- Alle Initialisierungen (Anfangsbelegungen von Variablen) müssen wohlüberlegt sein! (Setzt man eine Variable gleich Null und multipliziert dann mit ihr, darf man sich nicht wundern, dass die Variable auch in Zukunft gleich Null bleibt.)

- Häufige Fehler bei `scanf`:
  - NICHT mit %g arbeiten
  - NICHT \n zwischen "…" setzen
  - **& vergessen**
  - Formatierer passt nicht zur Variable nach dem Komma

Tipps fürs Programmtesten**:**

- Programmtests vor Beginn der Programmierung planen
- Sinnvolle Variablennamen, wie z.B `ergebnis, eingabe`
- Eventuell andere Programmierkonstrukte verwenden (`for-` statt `while`-Schleife)
- Bestimmte Stücke mit Papier und Bleistift nachrechnen!!
- Kommentare überarbeiten, z.B. Beschreibung dessen, was dort passieren *sollte*
- Zeilennummerierung im Editor einschalten

**Absolute Notfall-Programmtesttipps:**

- Bei der **scanf**-Funktion das & vergessen?
- Ein Semikolon vor einer offenen geschweiften Klammer?

- Geben Sie sich mit **printf** ALLE Variablen aus (kommentieren Sie diese **printfs** später aus, nicht löschen!)!
- Der ultimative Tipp: rechnen Sie bestimmte Programmstücke mit Papier und Bleistift nach (der Lerneffekt und Erfolg ist phänomenal!)
- Der noch ultimativerer Tipp: Schlafen Sie drüber und am nächsten Tag sehen Sie sofort den Fehler/das Problem, weil Ihr Gehirn nachts für Sie gearbeitet hat!
- Auskommentieren (`/* */` oder //) von eventuell kritischen Teilen des Codes

Programmieren mit dem gcc- Compiler (Dev-Cpp):

1. Laden Sie sich den **Dev-Cpp**-Compiler hier herrunter: http://sourceforge.net/projects/orwelldevcpp/ und installieren Sie alles auf Ihrem Rechner
2. Start des Compilers: Gehen Sie in Ordner **Dev-Cpp** und führen Sie einen Doppelklick auf der **Dev-Cpp-Anwendung** aus.
3. Arbeiten mit dem Compiler: Anlegen einer neuen Datei an: **File**->**Neu**->**SourceFile**
4. Speichern Sie Ihre C-Programme mit der Endung **.c** ab.
5. Übersetzen (Übersetzen des Textes aus der C-Sprache in die Maschinensprache Ihres Rechners) Sie Ihr Programm durch Klicken auf das bunte quadratische Symbol ganz links

Programmausführung im Eingabeaufforderungsfenster (Dos-Fenster):

Zum Ausführen von Programmen empfehle ich die Arbeit im Eingabeaufforderungsfenster (Bemerkung: die Kommandozeile ist Ihr Freund, Arbeit mit der Kommandozeile ist viel schneller als Arbeit mit der Maus, alte Kommandos können einfach wiederholt werden, alte Programmdurchläufe sind später noch einsehbar, Entwicklungsumgebung = what you see is all you get!):

1. Öffnen des Fensters in Windows: **Start** -> **Programme**-> **Zubehör** -> **Eingabeaufforderung (**oder links unten bei Windows im kleinen Eingabefenster **cmd** eingeben**)**
2. Im Fenster das Laufwerk wechseln, z.B. nach H: **H:**
3. Im Fenster in den Ordner wechseln, wo man ein Programm ausführen möchte: **cd** *Ordnername*
4. Aus einen Ordner wieder raus gehen: **cd ..**
5. Im Fenster ein Programm „*Name*“ ausführen: *Name* und Enter

**Achtung**: dieses Fenster erst schließen, wenn Sie zum Programmieren keine Lust mehr haben. Bleibt Ihr Programm bei der Ausführung hängen oder gerät in eine Endlosschleife, dann das Fenster NICHT schließen, sondern **Strg C** tippen (Ihr Programm wird brutal vom Betriebssystem beendet)

Weitere hilfreiche Kommandos im Eingabeaufforderungsfenster:

- Auflisten aller Dateien des Ordners (engl.: directory), indem man sich gerade befindet: **dir**
- Wiederholen alter Befehle: Pfeiltasten nach oben oder unten
- Wortergänzung macht es sehr gemütlich: →|- Taste (links und ziemlich weit oben auf der Tastatur): ersten Buchstaben tippen und dann die Wortergänzungstaste drücken