

Programmierpraktikum im WiSe



„Würden Architekten Häuser bauen, wie Programmierer ihre Programme, dann könnte ein Specht die ganze Zivilisation zerstören.“

Wichtig: Das Hauptziel der Informatikveranstaltung ist, dass Sie lernen, eigenständig Programme zu schreiben und zu lesen.

Programmierung ist eine Fertigkeit. Klavierspielen lernt man nicht durch youtube-Filmchen ansehen oder Folien, sondern durch Üben!

Der wichtigste Schritt ist das Verstehen der Aufgabe. Das Nachdenken über die Aufgabe wird Ihrem Gehirn helfen, eine Lösung zu finden. Sprechen Sie mit anderen über die Aufgaben. Machen Sie Pausen und gehen Sie die Aufgaben immer wieder an. Schlafen Sie drüber, oft sieht man am nächsten Morgen sofort die Lösung. Das ist alles schon recht gut erforscht: im Wachzustand füttern Sie Ihr Gehirn mit Wissen, Regeln und den Aufgaben, nachts, wenn Sie schlafen, löst das Gehirn dann alle Aufgaben für Sie. Sie müssen die Lösungen dann nur noch aufschreiben.

Suchen Sie nach der einfachsten Lösung, die die Aufgabenstellung hergibt!

Dieses Semester programmieren wir wie folgt:

- Suchen Sie sich einen Programmierpartner und programmieren Sie zu zweit. Wechseln Sie sich beim Tippen der Programme aller 15 Minuten ab.
- Programmieren Sie alle Aufgaben weiter unten in diesem Dokument in einem offline-Compiler (den Sie sich auf Ihren Rechner vorher installiert haben - weiter unten sind Beschreibungen zur Installation und Nutzung für Devcpp und Codeblocks).
- **Achtung: Wir programmieren in diesem Praktikum in C89! Das ist beim Devcpp-Compiler z.B. (mehr oder weniger) automatisch voreingestellt!**
- Heben Sie sich alle Programme auf Ihrem Rechner auf, denn sie sind Ihre „Schätze“ und helfen später sehr bei der Vorbereitung auf die Klausur.
- **IMMER** sollten Ihre Programme wie folgt beginnen:

// LEA-Benutzername: z.B. irothe3m

// Inhalt: Schreiben Sie, was Ihr Programm machen soll, z.B. Summe zweier Zahlen

// Testfall: Geben Sie hier Ihre Eingabewerte ein und was Sie hoffen, als Ergebnis zu erhalten

//z.B. 5, 6

// Ergebnis: 11

- **IMMER** kommentieren Sie wie folgt:
 - Not all programmers can write really obvious code.
 - Some comments are like titles and subtitles in articles, they guide, provide context and convey overall meaning
 - Kommentieren Sie alle Variablen auf einer Zeile über der eigentlichen Deklaration der Variablen, wofür sie da sind. Nur wenn Sie wissen, wofür Sie die Variable benutzen wollen, hat es Sinn, die Variable zu deklarieren!
 - Wenn man das Programm nach einer Woche wieder öffnet, fragen Sie sich: is still everything obvious to you or would you wish for more comments?

- Testen Sie Ihre Programme mit den Testfällen, die Sie am Anfang des Programms notiert haben. Versuchen Sie an alle möglichen Testfälle zu denken.
- Beim Übersetzen des Programms dürfen keine Warnungen auftreten. **Alle Warnungen müssen beseitigt werden.**
- Halten Sie sich bitte nach Möglichkeit an folgenden Bearbeitungsplan:

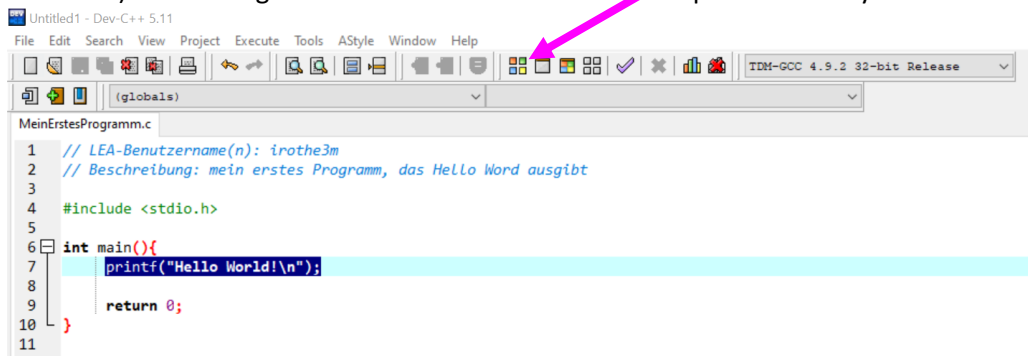
Aufgaben	Abgabe bis
1-4	Mitte November
5-8	Ende Dezember
9-12	Ende Januar
13	Mitte Februar

- Sind Sie zufrieden mit Ihrem Programm (Wie ein perfektes Programm aussieht: siehe Link in Lea **Roths Informatik 1** → **Wann ist mein Programm einreichbar** (ziemlich weit unten)), zeigen Sie es mir *im Praktikum* (über <https://h-brs.webex.com/meet/irene.rothe>). Ich führe eine Liste und hake das Programm dann ab. Ich hake Programme nur *innerhalb* der Praktikumszeiten ab und bevorzuge dabei alle, die regelmäßig anwesend sind. Bei unregelmäßiger Teilnahme kann ich nicht garantieren, dass ich Zeit zum Abnehmen der Aufgaben habe.
- Brauchen Sie *außerhalb* der Praktikumszeiten Hilfe beim Programmieren, können Sie mich gerne per email irene.rothe@h-brs.de fragen. Wenn es dafür nötig ist, Code zu schicken, schicken Sie bitte niemals Anhänge, sondern kopieren Ihren Code direkt in die email rein. Als Betreff können Sie unter anderem den Tag Ihres Praktikums nennen und die Uhrzeit.
Bemerkung: Ich antworte sehr gerne auf Fragen. Bitte entschuldigen Sie aber schon jetzt meinen möglicherweise sehr direkten Ton. Wenn es um Programmierung geht, schalte ich scheinbar in eine Hälfte meines Gehirns, wo es nur noch um Programmierung geht. Ich meckere gerne über Programme, das betrifft aber nie Sie, sondern nur Ihre Programme. Also bitte nie persönlich nehmen. Mein Wunsch ist, dass Sie schöne Programme schreiben. Sie sollten **innerhalb von 24 Stunden** eine Antwort von mir erhalten, ansonsten schicken Sie die email noch mal.

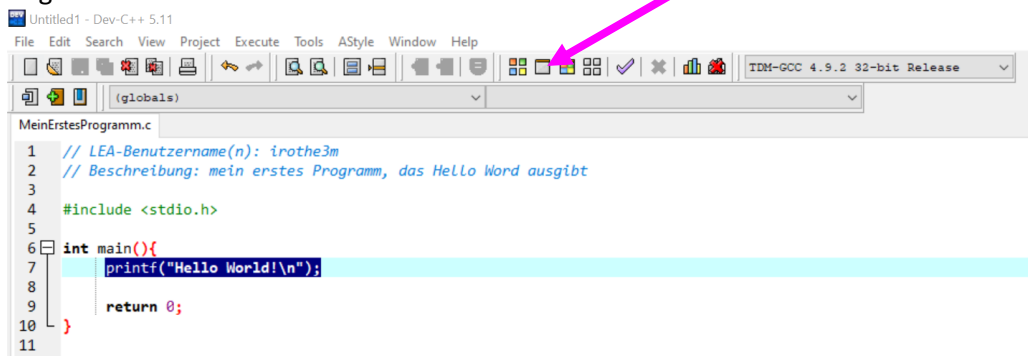
Programmieren mit dem gcc- Compiler (Dev-Cpp), ihn gibt es auch portable (man muss ihn nicht installieren!):

1. Bemerkung: der Compiler ist leider veraltet, aber ich liebe ihn sehr. Es gibt keine Updates mehr. Es kann sein, dass Ihr Virens scanner z.B. anschlägt, wenn Sie Programme übersetzen.
2. Laden Sie sich den **Dev-Cpp**-Compiler hier herunter:
<http://sourceforge.net/projects/orwelldvcpp/> und installieren Sie alles auf Ihrem Rechner
3. Start des Compilers: Gehen Sie in Ordner **Dev-Cpp** und führen Sie einen Doppelklick auf der **Dev-Cpp-Anwendung** aus.
4. Arbeiten mit dem Compiler: Anlegen einer neuen Datei an: **File->Neu->SourceFile** (Kein Projekt anlegen!)
5. Speichern Sie Ihre C-Programme mit der Endung **.c** ab.

6. Übersetzen (Übersetzen des Textes aus der C-Sprache in die Maschinensprache Ihres Rechners) Sie Ihr Programm durch Klicken auf das bunte quadratische Symbol in der Mitte.



7. Ausführen können Sie Ihr Programm durch Klicken auf den grünen Pfeil oben in der Mitte ungefähr.

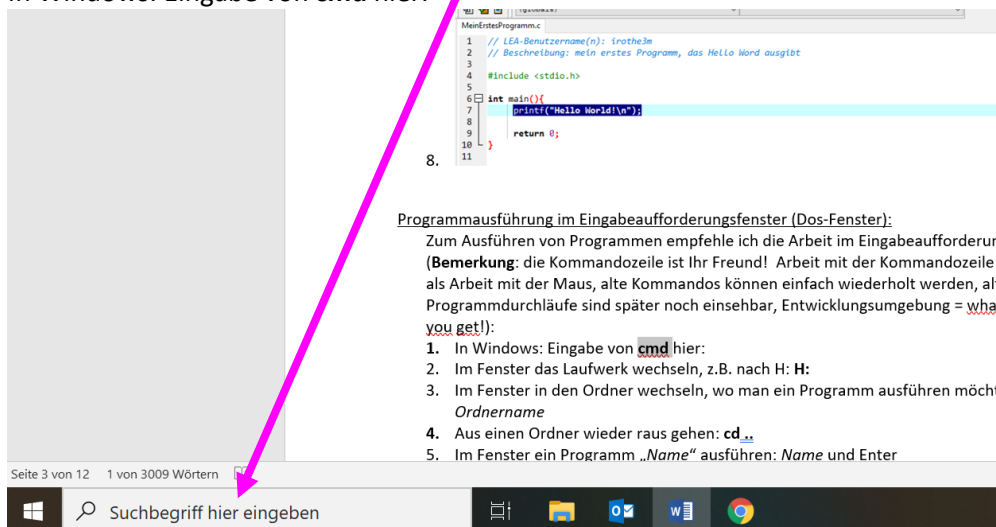


8.

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. In Windows: Eingabe von **cmd** hier:



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 get!):

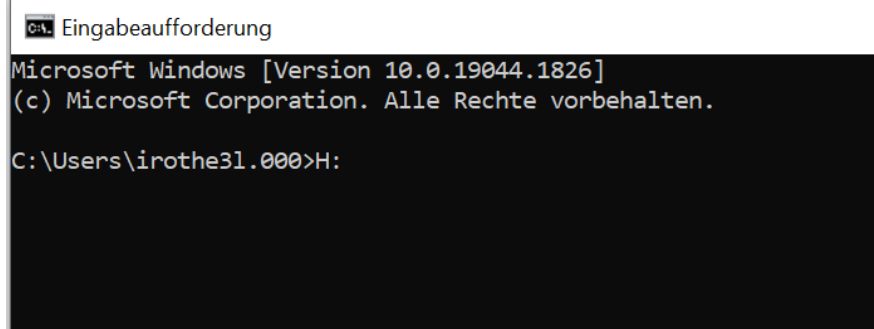
1. In Windows: Eingabe von **cmd** hier:
2. Im Fenster das Laufwerk wechseln, z.B. nach H: **H:**
3. Im Fenster in den Ordner wechseln, wo man ein Programm ausführen möchte **Ordnername**
4. Aus einen Ordner wieder raus gehen: **cd ..**
5. Im Fenster ein Programm „Name“ ausführen: **Name** und Enter

2. So was Ähnliches öffnet sich dann:



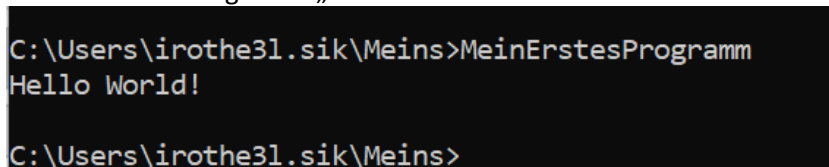
```
C:\Users\irothe31.000>
```

3. Im Fenster das Laufwerk wechseln, z.B. nach H: **H:**



```
C:\Users\irothe31.000>H:
```

4. Im Fenster in den Ordner wechseln, wo man ein Programm ausführen möchte: **cd** *Ordnername*
5. Aus einen Ordner wieder raus gehen: **cd ..**
6. Im Fenster ein Programm „Name“ ausführen: *Name* und Enter



```
C:\Users\irothe31.sik\Meins>MeinErstesProgramm
Hello World!
C:\Users\irothe31.sik\Meins>
```

Achtung: Die Eingabeaufforderung 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

Programmieren mit Codeblocks:

Video zur Installation von **Codeblocks**: <https://www.youtube.com/watch?v=clx8HthehKs>



Coco startet mit der C Programmierung Teil 1 - Der Compiler

Bemerkung: Am Ende dieses Dokuments finden Sie Tipps zur Syntaxfehlersuche und zum Testen!

Aufgaben zur Veranstaltung Informatik 1:

Aufgabe 1 (Vorlesung Einführung):

Wir starten mit dem berühmten HelloWorld-Programm, typischerweise dem ersten Programm in jeder Programmiersprache:

```
// LEA-Benutzername(n)
// Beschreibung

#include <stdio.h>
int main() {
    printf("Hello World!\n");

    return 0;
}
```

- Führen Sie Ihr Programm aus. Funktioniert alles? Können Sie die Ausgabe sehen?
- Löschen Sie einfach mal ein Semikolon und übersetzen das Programm noch einmal - was passiert?
- Ergänzen Sie das Programm nun so, dass folgende Ausgabe erzeugt wird:

Hello World!

Bonjour le monde!

Hallo Welt!

Tragen Sie am Anfang des Programms (im 'Header') die geforderten Informationen ein.

Aufgabe 2 (Vorlesung Grundlagen):

Schreiben Sie ein C-Programm, das zunächst die Eingabe eines Alters ermöglicht. Danach soll als Ausgabe die Anzahl der gelebten Monate, Tage, Stunden, Minuten und Sekunden bzgl. dieses Alters erfolgen. Schaltjahre und Anzahl der Tage vom letzten Geburtstag bis heute sollen vernachlässigt werden.

- Testen Sie Ihr Programm mit Ihrem selbst vorgegebenen Testfall.
- Testen Sie Ihr Programm mit einem Alter von 80 Jahren. Was fällt Ihnen beim Wert für die Sekunden auf? Warum?
- Testen Sie Ihr Programm mit einem Alter von 170 (Schildkröte) Jahren. Was fällt Ihnen auf? Warum?

- Überlegen Sie, wie das Programm geändert werden könnte, damit es auch mit einem hohen Alter zurechtkommt. Schreiben Sie Ihre Vermutung in den Header. Die von Ihnen vorgeschlagene Änderung müssen Sie im Programcode selbst erst einmal NICHT machen.
- Der Programmheader sollte ungefähr wie folgt aussehen:

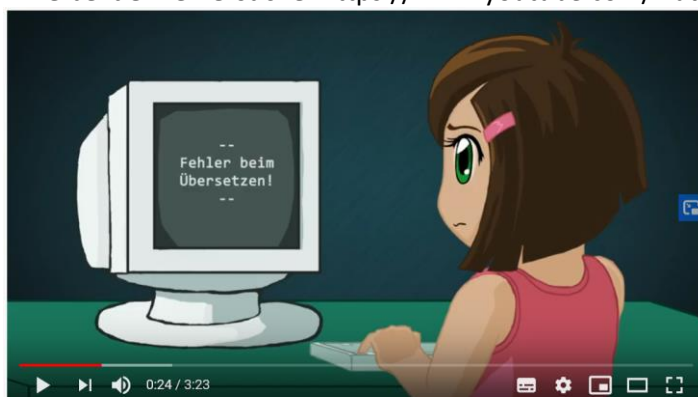
```
// LEA-Benutzername (n)
// Beschreibung
// Testfall: <Beispiel für einen Eingabewert>/<erhoffte Ausgabewerte>
// Was passiert für ein Alter von 80?
// Warum?
// Was passiert für ein Alter von 170?
// Warum?
// mögliche Änderung für hohes Alter?
```

Lerntagebuch:

Was habe ich gelernt bzw. ausgeführt?	Kreuz bei JA
Ich habe das Programm begonnen mit einem Header (Name, Inhalt, Testbeispiel)	
Ich weiß, wie man eine Eingabe programmiert in C.	
Ich weiß, wie man eine Ausgabe programmiert.	
Ich weiß über den Datentyp int Bescheid.	
Ich weiß, was Variablen sind.	

„Irren ist menschlich, aber wenn man richtigen Mist bauen will, braucht man einen Computer.“

Hilfe bei der Fehlersuche: <https://www.youtube.com/watch?v=Ji6bMBDuqKU>



Coco startet mit der C Programmierung Teil 2 - Fehler beim Programmieren

Lernziele: Anwendung verschiedener Datentypen, Erfahrungen mit Schleifen und Verzweigungen

Aufgabe 3 (Vorlesung Datentypen, Kontrollstrukturen, Flussdiagramm):

Für das, vor allem in Vertretungsstunden beliebte, Ratespiel 'Hangman' soll die Anzahl der (Fehl-)versuche, die ein*e Mitspieler*in bekommt, von dem jeweiligen Alter abhängig sein. Hierbei gelten folgende Regeln:

- Kinder, die jünger sind als 10 Jahre, sollen 12 Versuche bekommen (mehr Chancen für die Kleinen).
- Ist jemand tatsächlich 40 Jahre oder älter, soll es 10 Versuche geben (im Alter lässt die Gehirnleistung nach).
- Alle anderen haben 8 Versuche.
- Um eine gewisse Ungerechtigkeit in das Spiel zu bringen, soll bei denjenigen, deren Alter mathematisch gerade ist, die Anzahl der Versuche um 1 erhöht werden.

Schreiben Sie ein Programm, das abhängig von einem eingegebenen Alter, eine Variable **rateVersuche** nach den genannten Regeln setzt und in Folge ausgibt.

Achtung: Nutzen Sie Ihren Header aus Aufgabe 2 und ändern alle nicht mehr passenden Informationen ab, z. B. mit Testfällen für die verschiedenen Möglichkeiten, die in dieser Aufgabe auftreten können.

Aufgabe 4 (Vorlesung Datentypen, Kontrollstrukturen, Flussdiagramm):

Bis jetzt haben Sie sicher die Nullstellen (Schnittpunkte des Graphen einer Funktion mit der x-Achse) von quadratischen Funktionen mit der Hand oder einem programmierbaren Taschenrechner berechnet.

Schreiben Sie nun ein C-Programm, das Nullstellen einer quadratischen Gleichung ($x^2 + px + q = 0$) berechnet.

Sie können die Funktion `sqrt(...)` aus der `math.h`-Bibliothek fürs Wurzelziehen benutzen. Informieren Sie sich im Internet darüber, wie diese Funktion verwendet wird.

Achtung: Vergessen Sie nicht, den Header auszufüllen, besonders die Testfälle: Eingabe(n) und die erhofften Ausgaben (eventuell mehrere Testfälle).

Lerntagebuch:

Was habe ich gelernt bzw. ausgeführt?	Kreuz bei JA
Ich habe das Programm begonnen mit einem Header (Name, Inhalt, Testbeispiel)	
Ich weiß, dass man <code>if</code> immer mit ... programmiert.	
Ich habe den Datentyp <code>double</code> kennengelernt.	

„Fehler in seinem Quelltext zu machen, ist nicht schwer – sie zu finden dagegen sehr.“

Aufgabe 5 (Vorlesung Datentypen, Kontrollstrukturen, Flussdiagramm):

Schreiben Sie ein Programm, das abhängig von einer eingegebenen Zahl `fehler` verschiedene Ausgaben macht:

- `fehler = 1` → Ausgabe: Der Galgenfuss wurde gebaut.
- `fehler = 2` → Ausgabe: Der Galgenpfahl wurde errichtet.
- ...
- Bis `fehler = 12` → Ausgabe: Sie hängen am Galgen und sind tot.

Nutzen Sie das `switch`-Programmierkonstrukt

Vergessen Sie nicht, den Header auszufüllen mit mindestens einem konkreten Testfall.

Aufgabe 6 (Vorlesung Datentypen, Kontrollstrukturen, Flussdiagramm):

Schreiben Sie ein Programm, das Folgendes ausgibt:

```
  A B C D E F G H I J
1  . . . . .
2  . . . . .
3  . . . . .
4  . . . . .
5  . . . . .
6  . . . . .
7  . . . . .
8  . . . . .
9  . . . . .
10 . . . . .
```

Nutzen Sie ineinander geschachtelte `for`-Schleifen. Übrigens wird dieses Programm im Sommersemester gebraucht für das Spiel Schiffe versenken.

Aufgabe 7 (Vorlesung Datentypen, Kontrollstrukturen, Flussdiagramm):

Berechnen Sie die Fakultät einer beliebigen Zahl (Beispiel: $4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$) zweimal, einmal mit:

- einer `while` - Schleife (siehe Vorlesung_B) und geben das Ergebnis aus und einmal mit
- einer `for`-Schleife (siehe Vorlesung_B) und geben das Ergebnis aus.
- Die `for`- und die `while`-Schleife sollen die gleiche Logik bzgl. der Abarbeitung benutzen.
- Ab welcher Eingabezahl liefert die Fakultätsberechnung nur noch Quatsch als Ergebnis? Warum?

```
// LEA-Benutzername(n)
// Beschreibung
// Testfall
// Bis zu welcher Eingabe ist das Ergebnis richtig?
// Warum danach nicht mehr?
```

Aufgabe 8 (Vorlesung Datentypen, Kontrollstrukturen, Flussdiagramm):

Schreiben Sie ein Programm, das eine Schleife beinhaltet, die abbricht, wenn eine Variable `fehler` gleich oder größer `rateVersuche` ist oder wenn eine Variable `richtigGeraten` gleich oder größer `anzahlBuchstaben` ist. Dies wird für die große Spielschleife beim Programmieren des Galgenmännchenspiels gebraucht.

- Geben Sie im Programm die folgenden festen Werte für `rateVersuche` und `anzahlBuchstaben` vor:
`rateVersuche = 8;`
`anzahlBuchstaben = 6;`
- Die Variablen `fehler` und `richtigGeraten` sollen jeweils mit 0 initialisiert werden.
- Bei Eingabe einer 1 soll die Variable `fehler` hochgezählt werden, bei Eingabe einer 2 zählen Sie die Variable `richtigGeraten` hoch.
- Geben Sie nach dem Abbruch der Schleife die Werte von `fehler` und `richtigGeraten` aus.

Aufgabe 9 (Vorlesung Funktionen):

Wie rechnet der Taschenrechner eigentlich *sinus* oder *cos* oder π aus? Der *sinus* von x (eine sehr wichtige Funktion in der Mathematik und im Ingenieurwesen) kann mit folgender Summe berechnet werden:

$$\sin(x) = \sum_{i=0}^n \left((-1)^i \frac{x^{2i+1}}{(2i+1)!} \right)$$

Dabei gibt das Summenende n die Genauigkeit an, mit der der sinus (im Bogenmaß) berechnet werden soll.

Programmieren Sie diese Formel und testen Sie, wie weit Sie die Summe berechnen müssen, um mit der in der C-Bibliothek <math.h> mitgelieferten Funktion sin mithalten zu können. Wenn Sie möchten, können Sie die pow(...)-Funktion aus dieser Bibliothek benutzen, die wie folgt benutzt werden kann: z.B. x hoch 2 → pow(x,2). Die Genauigkeit n soll einlesbar sein.

Achtung: Benutzen Sie die Fakultätsberechnung aus Aufgabe 7, in dem Sie daraus eine Funktion machen, die Sie bei der sinus-Berechnung aufrufen. Wie Funktionen in C implementiert werden, finden Sie in Vorlesung_C.

Vergessen Sie die Testfälle nicht.

```
// LEA-Benutzername(n)
// Beschreibung
// Testfaelle
// Ab welchem n erreichen Sie die Genauigkeit der sinus-Funktion aus
// der math-Bibliothek?
```

Lerntagebuch:

Was habe ich gelernt bzw. ausgeführt?	Kreuz bei JA
Ich weiß, wie man switch benutzt.	
Ich weiß, wofür das default bei switch da ist.	
Ich weiß, wie man mit for und while Schleifen programmieren kann.	
Ich kenne den Unterschied zwischen einer Verzweigung und einer Schleife.	

„Es ist einfach, ein Programm zu schreiben, das innerhalb kürzester Zeit und mit wenig Speicherplatz ein gegebenes Problem nicht löst.“

Aufgabe 10 (Vorlesung Datentypen, Kontrollstrukturen, Flussdiagramm):

Malen Sie ein Flussdiagramm für das Spiel Hangman (z.B. mit Stift auf Papier oder mit draw.io) und schicken es als Foto oder png-Bilddatei an die email-Adresse Ihres Praktikumbetreuers:

irene.rothe@h-brs.de

Achtung: In Aufgabe 8 haben Sie sich schon Gedanken gemacht, wann das Spiel zu Ende ist.

Sie können das Flussdiagramm mit Stift und Papier erstellen oder mit folgender Webseite:

<https://app.diagrams.net/> oder ... Hauptsache es sieht hübsch aus und gefällt mir.

Hilfreiches Lernvideo für Flussdiagramme: https://www.youtube.com/watch?v=OR1I_xerHsk



Coco startet mit der C Programmierung Teil 3 - Das Flussdiagramm

Aufgabe 11 (Vorlesung Funktionen):

Programmieren Sie jetzt Ihre erste Rekursion. Die Quadratwurzel einer Zahl berechnet sich wie folgt (Verfahren nach Heron: <https://de.wikipedia.org/wiki/Heron-Verfahren> unter Verfahren):

$$w(n, x) = \frac{1}{2} \left[w(n-1, x) + \frac{x}{w(n-1, x)} \right] \quad \text{für} \quad n \geq 1$$

$$w(n, x) = 1 \quad \text{für} \quad n = 0$$

wobei n die Genauigkeit der Berechnung bedeutet und x der Wert, von der die Wurzel gezogen werden soll, ist.

Was fällt Ihnen bei wachsendem n auf?

```
// LEA-Benutzername(n)
// Beschreibung
// Testfaelle
// Was passiert bei wachsendem n?
```

Aufgabe 12 (Vorlesung Felder und Zeiger):

Schreiben Sie ein Programm, das folgendes Feld:

```
char wort[]={ 'I', 'N', 'F', 'O', 'R', 'M', 'A', 'T', 'I', 'K'};
```

in einer Schleife ausgibt mit einem Zeilenumbruch am Ende.

Aufgabe 13 (Vorlesung Felder und Zeiger):

Programmieren Sie das Spiel Hangman. Folgende Vorgaben gibt es:

- Nutzen Sie Ihr Flussdiagramm aus Aufgabe 10 für das Hangman-Spiel für die Programmierung.
- Lassen Sie das zu erratene Wort vor Spielbeginn mit `scanf("%s",...);` eingeben und speichern Sie es in einem Feld.
- Vermeiden Sie im eigentlichen Spielprogrammcode Zahlen, wie z.B. die Länge des Wortes. Nutzen Sie dafür eine Variable.
- Nutzen Sie Programmcode aus früheren Aufgaben wie folgt:
 - Aufgabe 3 als Funktion
 - Aufgabe 5 als Funktion (**Achtung:** Hier sind kreative Anpassungen nötig!)
 - Aufgabe 8 für die Spielschleife
 - Aufgabe 12 für die Ausgabe des Ratefortschritts.

Lerntagebuch:

Was habe ich gelernt bzw. ausgeführt?	Kreuz bei JA
Ich weiß, dass Flussdiagramme sehr nützlich sind.	
Ich weiß, dass ich mit Rekursion meinen Rechner fertig machen kann.	
Ich weiß, wo ein Feld beginnt.	
Ich weiß, dass ein Feld unflexibel ist.	

Aufgabe 14 (Kommentieren üben):

Wählen Sie eines Ihrer Programme aus und kommentieren Sie jede Zeile, um für die Klausur gut vorbereitet zu sein. Wenn Sie möchten, können Sie mir das kommentierte Programm schicken: irene.rothe@h-brs.de.

Freiwillige Aufgabe:

Falls Sie noch nicht genug haben...programmieren Sie einen einarmigen Banditen:

- Ein Spieler hat ein **Anfangskapital** von N (Euro).
- In jeder Runde des Spiels setzt der Spieler einen Teil seines Geldes als **Einsatz** ein.
- Es werden drei Zufallszahlen zwischen 0 und 9 gewürfelt.
- Sind alle 3 Zahlen gleich, vervierfacht sich des Spielers Einsatz.
- Sind 2 Zahlen gleich, verdoppelt sich des Spielers Einsatz.
- Sind keine 2 Zahlen gleich, verliert der Spieler seinen Einsatz an den einarmigen Banditen.
- Das Spiel bricht ab, wenn der Spieler sein gesamtes Kapital verloren hat oder sich sein Kapital verdoppelt hat.

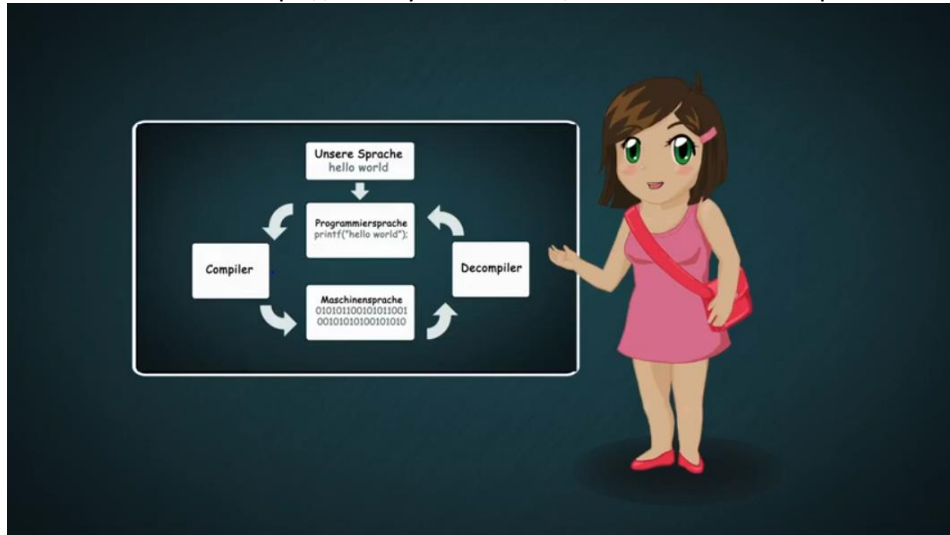
Vorschlag zur Erzeugung von Zufallszahlen:

```
#include<stdlib.h>
#include<time.h>

// Start Zufallszahlengenerator abhaengig von der Zeit
// wird nur EINMAL im Programm aufgerufen, am besten in main
srand (time(NULL));
// Erzeugung einer Zufallszahl zwischen 0 und max
int zufallszahl;
zufallszahl=rand()%max;
```

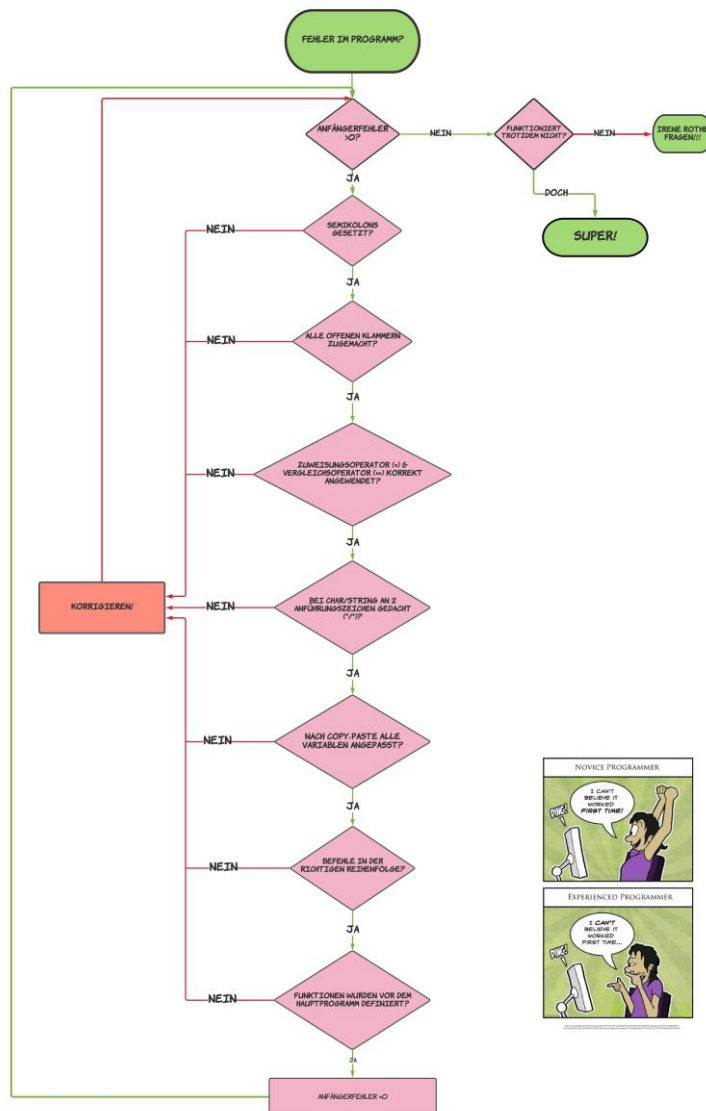
Fehlersuche = Lösen von Problemen!

Video zur Fehlersuche: <https://www.youtube.com/watch?v=Ji6bMBDuqKU>



Coco startet mit der C Programmierung Teil 2 - Fehler beim Programmieren

Flussdiagramm zur Fehlersuche:



Hilfe zur Fehlersuche in C-Programmen:

- 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 Abbruchkriterium 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 "..."
 - **& vergessen**
 - Formatierer passt nicht zur Variable nach dem Komma

Tipps fürs Programmtesten:

- Programmtests vor Beginn der Programmierung planen
- Sinnvolle Variablennamen, 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:

- Beim 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 ultimate 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