Programmierpraktikum im SoSe

Aktualisiert am 23.6.23

„Kleine Bugs verursachen große Probleme. Große Bugs verursachen auch große Probleme.“

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 an folgenden Bearbeitungsplan:

Aufgaben	Abgabe nach Möglichkeit bis
1-3	Ende April, etwas in den Mai rein
4-6	Ende Mai, etwas in den Juni rein
7,8	Ende Juni, etwas in den Juli rein wäre okay

- Sind Sie zufrieden mit Ihrem Programm, zeigen Sie es mir im Praktikum. Ich führe eine Liste und hake das Programm dann ab.
- Wie ein perfektes Programm aussieht: siehe Link in Lea **Rothes Informatik 2** (https://ccmjs.github.io/akless-components/dms/app.html?app=app_collection,1646434079779X7569224634657625) ziemlich weit unten auf **Wann ist mein Programm einreichbar**
- 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. 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.

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

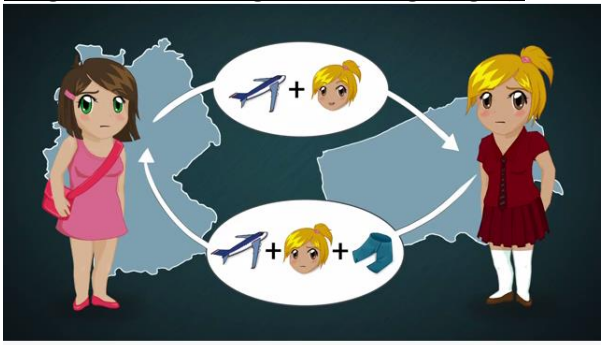
Aufgaben zur Veranstaltung Informatik 2:

Aufgabe 1 (Vorlesung FelderStringsZeiger):
Lernziele: Softwarereviewübung, Selbstreflexion
Sehen Sie sich Ihr Galgenmännchen vom letzten Semester noch mal an und beurteilen Sie es *kritisch*.

- Was gefällt Ihnen nicht so gut?
- Was würden Sie jetzt (ein paar Wochen später) anders machen?
- Was sind Schwachstellen beim Ausführen Ihres Spieles?

Beantworten Sie all diese Fragen als Kommentar am Anfang in Ihrem Hangmanprogramm.
Wenn Sie etwas kolossal stört, ändern Sie es gerne nun ab.

Aufgabe 2 (Vorlesung FelderStringsZeiger):



Coco startet mit der C Programmierung Teil 5 - Sinnvoller Einsatz von Zeigern
https://www.youtube.com/watch?v=laphh6vagb8&t=1s

Lernziele: zweidimensionale Felder mit Übergabe in Funktionen
Ein Ziel im Praktikum dieses Semester soll sein, das Spiel Schiffe versenken zu programmieren.

Quelle: Welt 2021

Im letzten Semester haben Sie dazu schon Vorbereitungen getroffen.
Formen Sie Ihr Programm aus Aufgabe 6 (Informatik 1) so um, dass die Punkte in einem Feld gespeichert werden. Implementieren Sie dies in einer *Funktion*, die als Eingabe ein zweidimensionales Feld übergeben bekommt.
Implementieren Sie Folgendes:

- **Funktion** zum Füllen eines zweidimensionalen Feldes mit Punkten
- **Funktion** für die Ausgabe eines zweidimensionalen Feldes mit Koordinaten für die Spalten als Buchstaben und für die Zeilen als Zahlen. Nutzen Sie dabei Ihre Lösung von Aufgabe 6 aus Informatik 1.
- **Funktion** für das Setzen eines Schiffes als 'S'. Die Eingabeparameter der Funktion sollen sein: Feld, x-Koordinate, y-Koordinate
- **Testprogramm** zum Testen der Funktionen

Wie immer gilt: Programmieren Sie alles so einfach wie möglich und so verständlich wie möglich.

Lerntagebuch:

Was habe ich gelernt bzw. ausgeführt?	Kreuz bei JA
Ich weiß, was meine selbstgeschriebene Software kann und nicht.	
Ich weiß, was ein 2-dimensionales Feld ist.	
Ich weiß, wie Felder in Funktionen übergeben werden.	

Aufgabe 3 (Vorlesung FelderStringsZeiger):
Lernziele: Zeiger auf Zeiger
Malen Sie ein Bild wie im youtube-Film https://www.youtube.com/watch?v=J7z4D1K3z5M&t=20s für folgenden Sachverhalt:

```
int *weltmeisterschaft[4];
int spielzeit;
int tore[]={1,7,1};
int weltmeister[]={1954,1974,1990,2014};
char gewinner[]={'D','E','U','T','S','C','H','L','A','N','D'};
int neuer[]={27,3,1986,52};


spielzeit = 120;
weltmeisterschaft[2]= &spielzeit;
*weltmeisterschaft = tore;
weltmeisterschaft[3]= weltmeister;
*(weltmeisterschaft+1)=&neuer[0];

printf ("%c\n",*(gewinner+2));
printf ("%i\n",*(weltmeisterschaft+1));
printf ("%i\n",**weltmeisterschaft);
printf ("%i\n",*(weltmeisterschaft[3]+1));
printf ("%c\n", gewinner[1]);
```

```
printf ("%i\n", weltmeisterschaft[2][1]);
printf ("%i\n",*(*(weltmeisterschaft+1)+1));
```

Zeigen Sie mir Ihr Bild im Praktikum

Aufgabe 4 (Vorlesung FelderStringsZeiger):

Lernziele: Call by Reference

Schreiben Sie eine **Funktion**, die

- von einer quadratischen Gleichung Nullstellen berechnet. Nutzen Sie dabei Ihre Aufgabe 4 aus Informatik 1.
- Die Anzahl der Nullstellen soll von der Funktion zurückgegeben werden.
- In der Funktion sollen die Nullstellen nicht ausgegeben werden, nur berechnet.

Schreiben Sie weiterhin ein **Testprogramm**, um Ihre Funktion zu testen

Vergessen Sie nicht, den Header auszufüllen (mit Testfällen für die verschiedenen Möglichkeiten).

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

Lerntagebuch:

Was habe ich gelernt bzw. ausgeführt?	Kreuz bei JA
Ich kenne mindestens 3 Bedeutungen eines Sternchens in einem C-Programm	
Ich weiß, was ** bedeutet	
Ich weiß, was das Zeichen & bedeutet	
Ich weiß, was eckige Klammern in der C-Programmierung bedeuten	

Aufgabe 5 (Vorlesung Strukturen):

Lernziele: Strukturen, Übergabe von Strukturen in Funktionen und Rückgabe

Strukturen können hier noch mal schön wiederholt werden:

https://www.youtube.com/watch?v=OrEFakXTD1E

Coco startet mit der C Programmierung Teil 6 Strukturen

Achtung: Leider fehlen im Filmchen die runden Klammern bei main.

Definieren Sie eine **Struktur** *Komplex*, die die komplexen Zahlen widerspiegelt, und implementieren Sie die Addition komplexer Zahlen als **Funktion**, die das Ergebnis der Addition als Struktur zurückgibt. Die Funktion soll dabei ungefähr wie folgt aufgerufen werden:

z=f(x,y)

natürlich mit viel schöneren Variablen- und Funktionsnamen.

Schreiben Sie ein **Testprogramm**, um Ihre Funktion zu testen.

Vergessen Sie nicht, den Header auszufüllen.

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

Aufgabe 6 (Vorlesung Strukturen):

Lernziele: Strukturen

Überlegen Sie sich, wie man sinnvoll Strukturen beim Spiel Schiffe versenken mit 2 Spielern und vielen Einer-Schiffen nutzen kann. Denken Sie dabei daran, wie Sie das Spiel eventuell in der Schule gespielt haben.

Implementieren Sie diese Struktur.

Programmieren Sie das Spiel Schiffeversenken für ein Einer-Schiff, implementieren Sie es aber so, dass man es einfach erweitern kann für viele Einer-Schiffe.

Nutzen Sie alle Ihre Funktion aus Aufgabe 2 ohne sie abzuändern!

Achtung: Zeichnen Sie zuerst ein Flussdiagramm!

Wie immer gilt: Programmieren Sie alles so einfach wie möglich und so verständlich wie möglich.

Lerntagebuch:

Was habe ich gelernt bzw. ausgeführt?	Kreuz bei JA
Ich weiß, wofür Strukturen nützlich sind und zwar für …	
Ich weiß, dass man auf Strukturen auf 2 Arten zugreifen kann.	

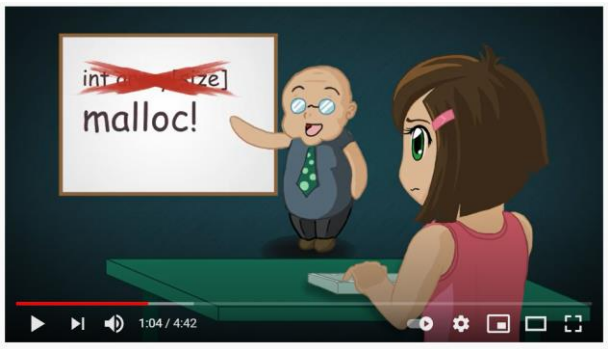
Aufgabe 7 (Vorlesung SpeicherListen):

Lernziele: dynamische Speicheranforderung

Ändern Sie Ihr Galgenmännchen-Spiel aus dem letzten Semester so ab, dass Sie das

- zu ratende Wort am Anfang nach Programmausführungsstart eingeben können.
- Verwenden Sie Speicheranforderungsfunktionen aus C (z.B. malloc und free), um flexibel auf die Länge des eingegebenen Wortes zu reagieren. Siehe hierzu auch die Folie „Flexible Eingabe“ aus der Vorlesung_L.
- Passen Sie alle statischen Speicheranforderungen (Frage: Woran erkennt man statische Speicheranforderungen?), wenn sinnvoll, entsprechend an.

Achtung: Ihr eigentliches Spielprogramm (also alles in der Spielschleife) lassen Sie bitte unverändert, das heißt, Variablennamensänderungen sind nicht nötig, wenn Sie oben Ihre Zeiger geschickt nennen!



Coco startet mit der C Programmierung Teil 7 Dynamische Speicherplatzreservierung

https://youtu.be/GoCROdDejeE

Aufgabe 8 (Vorlesung SpeicherListen):

Lernziele: Umgang mit Listen

Verwenden Sie folgenden C-Code aus der Vorlesung:

1. Definition einer Liste
2. Funktionen für Listen
3. Weiteren nützlichen Code, um Ihre neugeschriebene Funktion zu testen

Achtung: Ändern Sie die Funktionen aus der Vorlesung nicht ab. Wenn Sie Ihnen nicht gefallen, schreiben Sie zusätzlich neue Funktionen.

Aufgabenstellung:

Schreiben Sie eine **Funktion**, die

- das i-te Element einer Liste auf dem Bildschirm ausgibt. Zum Beispiel soll die Funktion das 42. Element der Liste ausgeben können.
- Wenn die Liste nicht groß genug ist, soll eine Informationstext ausgegeben werden.
- Die Position des auszugebenden Elements soll der Funktion als Parameter übergeben werden.

Schreiben Sie weiter ein **Testprogramm**, um Ihre Funktion gründlich zu testen.

Empfehlung zur Wiederholung: https://www.youtube.com/watch?v=p8ZLiDG1WnY

Lerntagebuch:

Was habe ich gelernt bzw. ausgeführt?	Kreuz bei JA
Ich weiß, wofür malloc da ist.	
Listen benutzen mindestens 3 C-Konstrukte, die ich aus früheren Praktika schon kannte.	
Ich kann gut C programmieren.	
Ich habe keine Angst vor der Klausur.	
Ich fühle mich beim Programmieren sicher.	

Aufgabe 9 (Kommentieren üben):

Wählen Sie eines Ihrer Programme aus und kommentieren Sie jede Zeile, um für die Klausur gut vorbereitet zu sein.

Aufgabe 10 :

Räumen Sie Ihren Ordner mit allen Programmen aus Semester 1 und 2 auf und führen Sie eventuell Verbesserungen bzgl. folgender Kriterien durch:

- Name der Datei sinnvoll? Weiß ich, was da drin ist?
- eventuell: was habe ich durch dieses Programm neu gelernt? Lernziel?

Aufgaben für Bonuspunkte (1-5 Punkte) für die Klausur:

1. Wenn Sie einen Fehler in der Vorlesung finden, bekommen Sie Zusatzpunkte. Dies bestätige ich Ihnen per email.
2. Legen Sie eine Quizfrage im Digitalen Makerspace http://dms.ccmjs.eu an (Login: irothe_studi Passwort: inf-is-fun, als „nicht öffentlich“)) und schicken mir den Link.
3. Wenden Sie sich an mich (irene.rothe@h-brs.de), um weitere Aufgaben zu erhalten für Bonuspunkte für die Klausur. Ich gebe Ihnen eine Auswahl von Aufgaben, wo Sie sich eine aussuchen können. Oder Sie programmieren das Spiel der Projektwoche aus dem ersten Semester fertig und schicken es mir. Ich sehe es mir dann an und entscheide, ob ich dafür Zusatzpunkte vergebe.

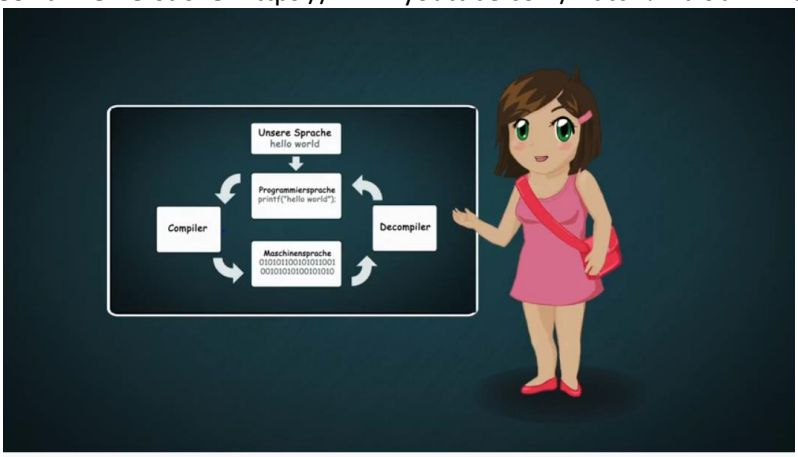
Achtung: Zusatzpunkte können Sie nur bis zum Vortag der Klausur erhalten. Heben Sie sich die emails auf, wo ich bestätigt habe, wie viele Zusatzpunkte Sie bekommen haben. Sie können sich durch

Zusatzpunkte nur um eine Note verbessern, Sie müssen dafür mindestens eine 4 vorher erreicht haben.

Wer gerne noch mehr Programmieren möchte, dem empfehle ich folgende Seite: https://www.codewars.com/

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

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*- 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
  - NICHT mit \n
  - **& vergessen**
  - Formatierer passt nicht zur Variable nach dem Komma

Tipps fürs Programmtesten:

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