Programmierpraktikum im SS

„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 immer mal Pausen.
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 aller 15 Minuten ab.
- Programmieren Sie alle Aufgaben weiter unten in diesem Dokument in einem offline-Compiler.
- **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, und noch anderen Testfällen.
- Beim Übersetzen des Programms dürfen keine Warnungen auftreten. Alle Warnungen müssen beseitigt werden.

- Halten Sie sich bitte nach Möglichkeit 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 oder meinem Kollegen im Praktikum oder schicken es mir per email an irene.rothe@h-brs.de, in dem Sie den *Code direkt in die email kopieren* und im Subjekt folgende Informationen schreiben: Lea-Kürzel und Gruppe.
  Zeigen Sie alle Programme nach Möglichkeit zur Praktikumszeit.
  Bemerkung: Wenn ich noch Dinge geändert haben möchte, kann es sein, dass ich dies sehr knapp und direkt formuliere, das ist nie persönlich gemeint. Mir geht es nur darum, dass Sie perfekte Programme am Ende des Semesters besitzen. Also bitte nicht sauer auf mich sein, wenn meine emails manchmal nicht sehr förmlich sind.
  Wie ein perfektes Programm aussieht: siehe Dokument
  ChecklisteWannIstMeinProgrammEinreichbar

Programmieren Sie offline, zum Beispiel:
https://www.youtube.com/watch?v=clx8HthehKs

Coco startet mit der C Programmierung Teil 1 - Der Compiler

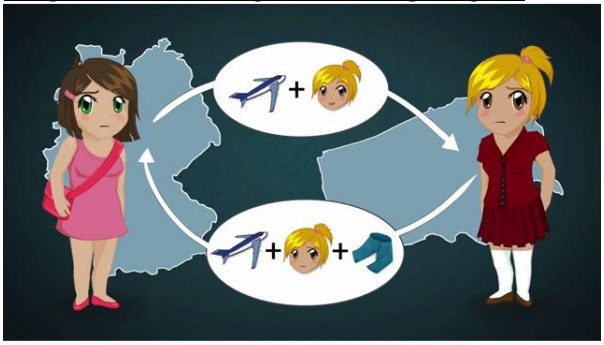
Bemerkung: Am Ende dieses Dokuments finden Sie Tipps zur Syntaxfehlersuche und zum Testen!

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 in Ihrem Hangmanprogramm am Anfang.

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 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 (siehe Informatik1, Aufgabe 6)
- Funktion für das Setzen eines Schiffes als ‘S‘. Die Eingabeparameter der Funktion sollen sein: Feld, x-Koordinate, y-Koordinate
- Hauptprogramm zum Testen der Funktionen

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.	

<u>Aufgabe 3 (Vorlesung FelderStringsZeiger):</u>
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));
```

Schicken das Bild als Foto oder png-Bilddatei an die email-Adresse Ihres Praktikumbetreuers, z.B. irene.rothe@h-brs.de

Aufgabe 4 (Vorlesung FelderStringsZeiger):
Lernziele: Call by Reference
Schreiben Sie *eine* Funktion, die aus einer quadratischen Gleichung Nullstellen berechnet. Nutzen Sie dabei Ihre Aufgabe 4 aus Informatik 1.
Achtung: Die Nullstellen sollen in der Funktion berechnet werden, aber erst im Hauptprogramm ausgegeben werden.

Beispieleingabe:
p=-4
q=3
Beispielausgabe:
x1=3
x2=1

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

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 aussehen: z=f(x,y), natürlich mit viel schöneren Variablen- und Funktionsnamen.

- Die erste komplexe Zahl soll wie folgt eingegeben werden:
  r1=…
  i1=…

- Die zweite komplexe Zahl soll wie folgt eingegeben werden:
  r2=…
  i2=…
- Das Ergebnis soll wie folgt ausgegeben werden:
  rErgebnis=
  iErgebnis=

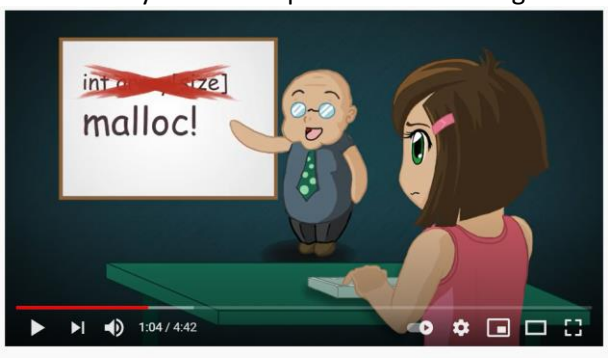
Beispieleingabe:
r1=1
i1=1
r2=2
i2=2
Beispielausgabe:
rErgebnis=3
iErgebnis=3

Aufgabe 6 (Vorlesung Strukturen):
Lernziele: Strukturen
Programmieren Sie das Spiel Schiffe versenken für 2 Spieler, die am gleichen Rechner sitzen, so einfach wie möglich, aber auch so verständlich wie möglich. Orientieren Sie sich beim Programmieren daran, wie Sie das Spiel als Kind gespielt haben.
Nutzen Sie dabei Strukturen.
Nutzen Sie die programmierten Funktionen aus Aufgabe 2.
Achtung: Zeichnen Sie unbedingt zuerst ein Flussdiagramm!
Tipp: Testen Sie Ihr Programm erst mal nur mit einem „Einer-Bötchen“.

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



Coco startet mit der C Programmierung Teil 7 Dynamische Speicherplatzreservierung
https://youtu.be/GoCROdDejeE
Ändern Sie Ihr Galgenmännchen-Spiel aus dem letzten Semester so ab, dass Sie das zu ratende Wort am Anfang bei Programmausführungsstart eingeben. Verwendung 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.

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

Aufgabe 8 (Vorlesung SpeicherListen):
Lernziele: Listen
Sehr empfehlenswert zur Wiederholung: https://www.youtube.com/watch?v=p8ZLiDG1WnY
Schreiben Sie eine *Funktion*, die das i-te Listen-Element auf dem Bildschirm ausgibt.
Nutzen Sie (durch Rauskopieren des C-Codes aus der Vorlesung, ohne diesen zu verändern) die Implementierung einer Liste und die Funktionen zur Veränderung dieser aus der Vorlesung.
Ihr Programm soll wie folgt ausgeführt werden können:

- Eingabe einer beliebigen Zahl i
- Ausgabe des Inhaltes des Elements an der Stelle i

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:

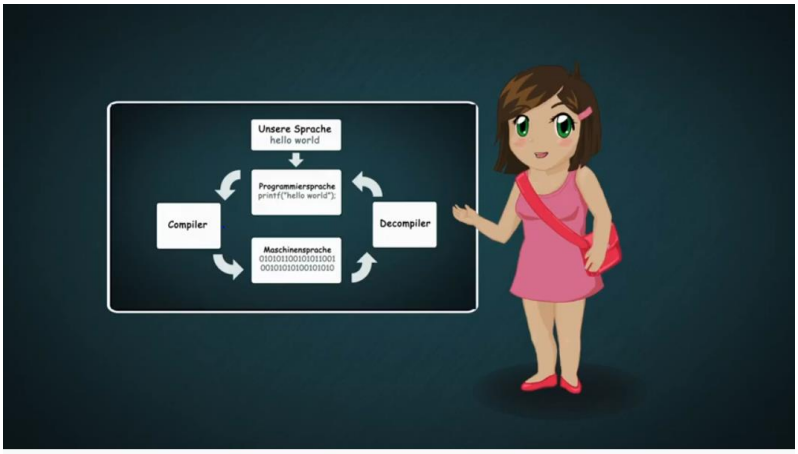
1. Wenn Sie einen Fehler in der Vorlesung finden, bekommen Sie Zusatzpunkte. Dies bestätige ich Ihnen per email.
2. Legen Sie eine Quizfrage im Digitalen Makerspace http://dms.ccmjs.eu an (Login: irothe_studi Passwort: inf-is-fun, als „nicht öffentlich“)) und schicken mir den Link.
3. Wenden Sie sich 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