Programmierung in C

Übung 6 : Einführung in die Programmiersprache C

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Aufgaben zum Semesterausklang:

Liebe Studierende,
Sie haben in den vergangenen Wochen einige Programmieraufgaben zu konkreten Themen bearbeitet, bei denen Sie sehr fokussiert auf bestimmte Aspekte der Programmiersprache C sein konnten.
In diesem Übungsblatt ist es deutlich anders, zum Einen sind hier Aufgaben gestellt, bei denen Sie den gesamten Themenumfang der Vorlesung beachten und zum Anderen aber auch sicher nützliches Wissen aus Ihrem bisherigen Studium einschließlich Ihrer Kreativität einbringen können – und sollten.

Syntaktischer Hinweis *zu den im Folgenden zu bearbeitenden Aufgaben:*
An einigen Stellen werden Sie gebeten, dass Ihr C-Code *sowohl unter Linux als auch unter Windows compilier- und linkbar sein soll.* Diese Code-Portabilität erreichen Sie zum Einen dadurch, dass Sie Funktionen des C-Standards benutzen, oder zum Anderen, und dieser Fall ist in den Aufgabenstellungen gemeint, indem Sie die Fähigkeiten des C-Präprozessors zur bedingten Codeerzeugung, der dann vom Compiler compiliert wird, ausnutzen.

Hier ein Beispiel für ein kleines Programm, das unter Windows und Linux compilierbar und linkbar ist, aber jeweils auf beiden Systemen etwas anders auf den Bildschirm schreibt:

```
#include<stdlib.h>
#include<stdio.h>
void ausgabe()
{
#if defined(__APPLE__) && defined(__MACH__)
       printf("Ich laufe auf einem Apple-System.");
#elif defined(_WIN64)
       printf("Ich laufe auf Windows 64-bit.");
#elif defined(_WIN32)
       printf("Ich laufe auf Windows 32-bit.");
#elif defined(__linux__)
       printf("Ich laufe auf Linux.");
#elif defined(__unix__)
       printf("Ich laufe auf Unix.");
#elif defined(_POSIX_VERSION)
       printf("Ich laufe auf Posix-System.");
#else
       printf("Ich laufe unter unbekanntem System.");
#endif
}

int main()
{
       ausgabe();
       return 0;
}
```

Eine sehr ausführliche Übersicht von Symbolen, die zur Unterscheidung von diversen Systemen vom Präprozessor ausgewertet werden können, sind beispielsweise unter folgender URL im Webarchive einsehbar:
https://web.archive.org/web/20180221212835/www.nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system .

Auch für Apple IPHONE Simulatoren lässt sich via Präprozessor-Symbole-Analyse speziellen Code einleiten, vgl. https://stackoverflow.com/a/5920028 , der für andere Systeme nicht zu generieren ist.

Für die anstehenden Prüfungsvorbereitungen und natürlich Ihre Prüfungen wünschen wir Ihnen viel Erfolg.

Aufgabe 1: (Zugriff auf Umgebungsvariablen des Betriebssystems, Datei schreiben)

Vorbemerkung:
In den bisherigen Aufgaben hatte die `main()`-Funktion Ihres Programms keine oder zwei Parameter. Der C-Sprachstandard sieht die Möglichkeit einer drei-parametrigen `main()`-Funktion vor. Der erste Parameter ist die Anzahl der Kommandozeilenargumente. Der zweite Parameter ist ein Zeiger auf das Array der Kommandozeilenargumente.
Der dritte Parameter ist ein Zeiger auf einen Speicherbereich, der die Umgebungsvariablen des Betriebssystems in einem zusammenhängenden Block enthält. Dieser Speicherblock kann als Array angesehen werden.

```
int main(int argc, char *argv[], char *envp[])
```

Der C-Standard beschreibt diese Variante der Signatur der **`main(...)`** als „common alternative“. Allerdings wird diese Variante sowohl in C89 im Anhang G (Portability Issues) als auch unter C99 und C11 im Anhang J (Portability Issues) geführt.

Hingegen verbietet der POSIX.1 Standard das dritte Argument. POSIX.1 erlaubt den portablen Zugriff auf Umgebungsvariablen des Betriebssystems anhand der jeweiligen Namen über die C-Standard Funktion **`getenv()`** aus **stdlib.h** bzw. über die POSIX-eigene Variable **`extern char **environ`**, vgl.:

- http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap08.html

Dennoch unterstützen die meisten Unix-Varianten sowie MS-Windows diesen dritten main-Parameter, vgl.:

- https://docs.microsoft.com/de-de/cpp/c-language/arguments-to-main
- https://www.gnu.org/software/libc/manual/html_node/Program-Arguments.html

Ihre Aufgabe:

a) Schreiben Sie ein Programm mit einer drei-parametrigen `main()`-Funktion, die den Inhalt des aktuellen Speicherblockes der Umgebungsvariablen des Betriebssystems strukturiert in eine Datei schreibt.
Der Dateiname soll als Programmparameter / Kommandozeilenargument Ihrem Programm bei dessen Aufruf bekannt gemacht werden. (Hinweis: Verwenden Sie `argv`.)
Die Struktur der zu erzeugenden Datei soll dem folgenden Muster gehorchen, wobei die Einrückung der jeweiligen Value-Zeile ein Tabulator-Zeichen sein soll:

```
<Name der Umgebungsvariable1>
      <value> Wert der Umgebungsvariable1 </value>
</Name der Umgebungsvariable1>
<Name der Umgebungsvariable2>
      <value> Wert der Umgebungsvariable2 </value>
</Name der Umgebungsvariable2>
```

b) Schreiben Sie ein Programm, dass mit einer zwei-parametrigen `main`-Funktion implementiert ist, und das gleiche leistet und sowohl unter Linux als auch unter Windows compilier- und linkbar ist.

Hinweis: Bedingte Kompilierung und Verwendung von #ifdef vgl. Seite 1.

Windows-API:	extern char **_environ	(beachte Unterstrich!)
POSIX-API:	extern char **environ	(hier kein Unterstrich!)

Aufgabe 2: (Strukturen schreiben und ändern, Arrays, Bibliotheken nutzen)
Gruppenarbeit 4 Personen pro Gruppe. Nach notwendigen Gruppengesprächen und Vereinbarungen zu viert sollen Sie die folgend beschriebenen Funktionalitäten als getrennte Zweier-Teams implementieren und anschließend Ihr Compiler-Ergebnis jeweils mit dem anderen Zweier-Team Ihrer Vierer-Gruppe austauschen, damit diese dann jenes weiter nutzen können.

1.) Beschreibung der Miniwelt:
Das Studentensekretariat hat alle relevante Angaben über Studenten gespeichert.
Dazu gehören Name, Vorname, Matrikelnummer, Adresse und Telefonnummer.
Jede Adresse besteht aus Straßennamen, Hausnummer, Postleitzahl und Stadtnamen.
Um mit studentischen Daten arbeiten zu können braucht das Studentensekretariat Funktionen, mit denen die Komponenten von Studierenden Daten eingefügt, bearbeitet und anzeigt werden können.

Die Komponenten dieser Daten sollen in Strukturen abgelegt werden, die dann Grundlage für zwei Datentypen sein sollen, die Sie '`tAdresse`' und '`tStudent`' nennen sollen.

Deklarieren Sie die beiden Typen '`tAdresse`' und '`tStudent`' in einer Header-Datei '`student.h`'.

Damit Ihre Software die Anforderungen des Studierendensekretariats erfüllt, müssen Sie Funktionen zum Bearbeiten von Daten der Typen '`tAdresse`' und '`tStudent`' deklarieren und implementieren.

a) Für den Typ '`tAdresse`' sind sind vorzusehen:
- Eingabe/Änderung/Ausgabe der kompletten Adresse.
- Änderung/Ausgabe der Straßennamen, Hausnummer, des Stadtnamen

b) Auch für den Typ '`tStudent`' soll es ähnliche Funktionen geben, genauer:
- Eingabe/Änderung/Ausgabe der kompletten Information über einen Studenten.
- Änderung/Ausgabe der Namen, Nachnamen, Matrikelnummer, Telefonnummer
- Änderung/Ausgabe der gesamten Adresse
- Änderung/Ausgabe der Straßennamen, Hausnummer, Postleitzahl, Stadtnamen

Einigen Sie sich bitte innerhalb Ihrer Vierer-Arbeitsgruppe auf die genauen Funktionssignaturen, d.h. Funktionsnamen und Parameter, die nötig sein werden, um die Funktionen, die das Studierendensekretariat benötigt, zu programmieren.

Die Typ- und Funktionsdeklarationen sollen vollständig in der Datei '`student.h`' enthalten sein, und für jedes Zweier-Team Ihrer Vierer-Gruppe identisch sein.

2.) Jedes Zweier-Team implementiert die obigen Funktionen nun für sich in einer eigenen Datei mit dem Namen '`student.c`'.

a) Diese Datei soll dann die Definitionen aller in '`student.h`' deklarierten Funktionen enthalten. Vergessen Sie nicht, in die Datei '`student.c`' die Headerdatei '`student.h`' mit der Preprozessordirektive `#include "student.h"` einzubinden!
Die Datei '**`student.c`**' **darf** die Funktion **`main()`** **nicht** enthalten.

b) Übersetzen Sie die Datei '`student.c`' mit dem Compileraufruf :

```
gcc -c student.c
```

Dabei erzeugt der Compiler die Objektdatei mit dem Namen '`student.o`'.

Die beiden Dateien '`student.h`' und '`student.o`' können als (rein statisch bindbare) Bibliothek angesehen werden.
(Dynamisch bindbare Bibliotheken sind unter Windows „dynamic link libraries“(*.dll) und unter Linux „shared objects“ (*.so) sind hier nicht im Fokus der Aufgabenstellung.)

3.) Schreiben Sie nun ohne Partner ein individuelles Programm, das die Funktionen aus der Bibliothek Ihres Zweier-Teams testet, in dem Sie ein Array mit mindestens 2 höchstens 20 Datensätzen füllen. Der Quelltext des Programms soll den Namen '`student_test.c`' haben. Vergessen Sie nicht, die Headerdatei '`student.h`' einzubinden.
Übersetzt wird das Programm mit dem Compileraufruf:

```
gcc student_test.c -o student_test student.o
```

Die ausführbare Testanwendung ist dann `student_test`.

4.) Die beiden Zweier-Teams einjeder Vierer-Gruppe sollen jeweils untereinander die jeweilige 'student.o' und 'student.h' austauschen.
Tauschen Sie nicht den Quelltext *.c aus.
Führen Sie die gcc-Anweisung aus (3.) nochmal aus, wobei Sie Ihre 'student_test.c' mit der ausgetauschten 'student.o' zu einer Testanwendung verbinden. Starten Sie Ihre Testanwendung. Gelingt dies? Wenn nein, warum nicht? Wenn ja, warum?

5.) Tauschen Sie nun 'student.o' und 'student.h' mit jemandem, der nicht in Ihrer Vierer-Gruppe war.
Führen Sie die gcc-Anweisung aus (3.) nochmal aus, wobei Sie Ihre 'student_test.c' mit der ausgetauschten 'student.o' zu einer Testanwendung verbinden. Starten Sie Ihre Testanwendung. Gelingt dies? Wenn nein, warum nicht? Wenn ja, warum?

Aufgabe 3: (Strukturen in Dateien schreiben und auslesen, dynamische Listen)

a) Schreiben Sie folgendes Anwender-Programm unter der Benutzung der Bibliothek 'student.o' der vorangegangenen Aufgabe.

Das Programm liest eine Zahl **n** von der Tastatur ein.
Danach liest es ebenfalls von der Tastatur n Datensätze vom Typ 'tStudent' ein
und speichert diese in der binären Datei mit dem Namen 'student.db'.
Da der Wert von **n** erst zur Laufzeit feststeht, sollen Sie keine Arrays verwenden, sondern eine

dynamisch wachsende einfach verkettete Liste.

b) Schreiben Sie ein Programm, das die Datensätze in der Datei 'student.db' nach Matrikelnummern in absteigender Reihenfolge sortiert. Nutzen Sie dabei die Möglichkeiten, die Ihnen durch Verwendung von Pointern gegeben werden.

c) Schreiben Sie ein Programm, das die Datensätze aus der binären Datei 'student.db' einliest und daraus einen Report als Textdatei "student.txt" erstellt.
Der Report soll enthalten wie viele Studierende erfasst sind. Für jeden erfassten Studierenden soll im Report eine Teilnahmebestätigung, adressiert an die betreffende Person, geschrieben werden.

Hinweise:
Dateien, Binärdateien, Textdateien, Dateizugriffe (fopen, fclose, fread, fwrite, fprintf, fseek, fflush, fsetpos, rewind), sizeof, Sortieralgorithmen, einfach verkettete Liste.

Herausforderung :
(Algorithmische Ideen, Dateioperationen, Arrays, Listen, modulares Programmieren)

Aufgabe 4 : (sogenanntes Postamt-Problem (Post-Office-Problem[1]))
In einer etwas größeren Stadt existieren meistens mehrere Briefkästen.
Stellen Sie sich nun vor, dass Sie an einer beliebigen Position in der Stadt sind und einen Brief in den nächsten Kasten einwerfen möchten. Sie sind fremd in der Stadt, aber Ihr mobiles Hilfsgerät ist in der Lage, Ihnen einen annehmbaren Weg vorzuschlagen.

Perspektivenwechsel: Sie sind im Bereich der Programmierung an der Entwicklung eines solchen mobilen rechnergestützten Hilfsmittels beteiligt.
Nun stehen Sie vor dem Problem, dass Ihr Softwaremodul den kürzesten Weg suchen soll.
Sie haben einen Stadtplan mit eingezeichneten Briefkästen und auch den Standort des Anwenders können Sie zur Laufzeit erfragen.

Zur Vereinfachung des Problem abstrahieren Sie von physikalischen und örtlichen Gegebenheiten, in dem Sie annehmen, dass Sie sich auf der Luftlinie bewegen können und keinen Umweg um Häuser machen müssen bei welchem zur Abstandsbestimmung die Manhattan-Metrik, auch City-Block-Metrik genannt, notwendig zu beachten wäre. Entfernungen in Luftlinie werden mit der EUKLIDischen Metrik bestimmt.

Schreiben Sie dazu ein Programm, das zu gegebenen Positionen von Briefkästen und beliebigem Standort, den nächsten Kasten ermittelt und die Länge des Weges ausgibt.

Die Angaben der Orte seien dabei zwei-dimensional in der Form `(X,Y)`, X und Y aus {1,...,20}.
Diese Angaben werden beim Programmstart per Kommandozeilenparameter übergeben.
Ebenso soll es möglich sein, dass das Programm die Koordinaten-Paare aus einer Textdatei einließt: Wenn der erste Parameter **`-f`** ist, dann wird der zweite Parameter als Dateiname genutzt. Alle weiteren Parameter werden dann ignoriert. In der Koordinatendatei steht jedes Koordinatenpaar in einer Zeile.

Ihr Programm soll den ermittelten Weg und dessen Länge direkt am Bildschirm ausgeben.
Damit Sie überprüfen können, ob Sie Ihre algorithmische Idee tatsächlich richtig umgesetzt haben, schreibt Ihr Programm Zwischenergebnisse, d.h. gefundene Wege bzw. deren Längen zu allen Briefkästen, zur Laufzeit als Protokoll in eine Textdatei. In diesem Protokoll soll kenntlich gemacht sein, welche Lösung am Bildschirm ausgegeben wurde.

Beispielpositionen:

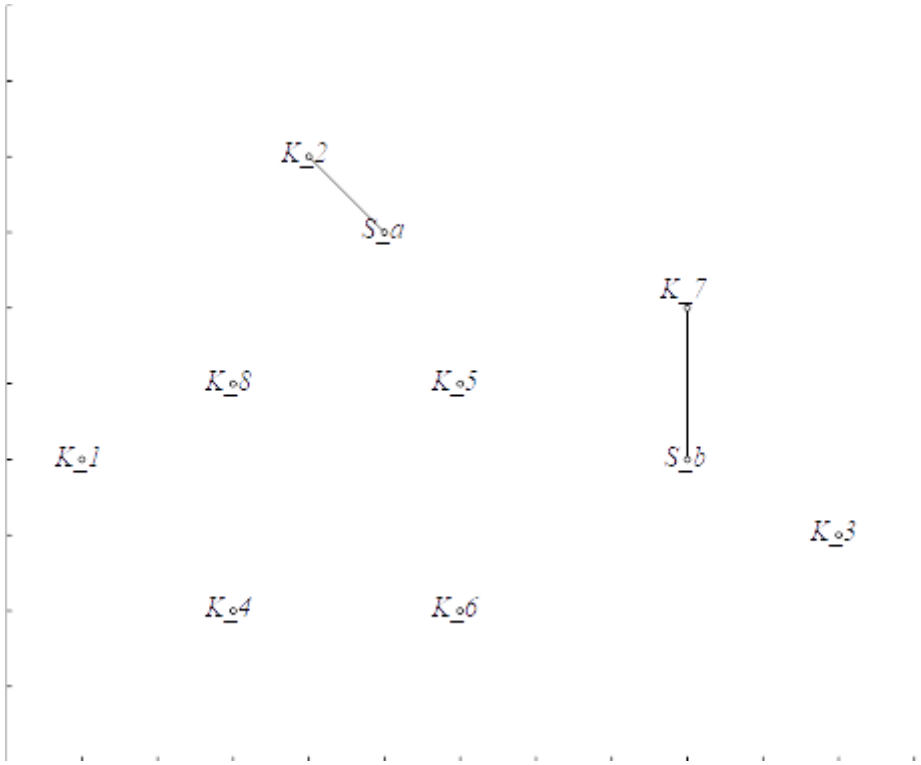
1.Kasten (1,4)	2.Kasten (4,8)	3.Kasten (11,3)	4.Kasten (3,2)
5.Kasten (6,5)	6.Kasten (6,2)	7.Kasten (9,6)	8.Kasten (3,5)

Zwei Beispiellösungen mit verschiedenen Standorten für die Bildschirmausgabe bei gleichen Kästen:

(a) Standort (5,7)	-> Kasten 2	=> Weglaenge: 1,41.. (Hinweis: Wurzel 2)
(b) Standort (9,4)	-> Kasten 7	=> Weglaenge: 2,0

Eine zeichnerische Darstellung einer Lösung bei gegebenen Beispielpositionen und Standorten finden Sie auf der folgenden Seite. Ihr Programm braucht keine Zeichnung erzeugen.

1 Ursprüngliche Formulierung und Untersuchung des Post-Office-Problem in
D. E. Knuth. The Art of Computer Programming, Volume III: Sorting and Searching, Addison-Wesley, 1973.
Inzwischen auch allgemeiner bekannt als Suche des nächsten Nachbarn (nearest neighbor search) beliebig dimensioniert.



Post-Office-Problem - für zwei Standorte bei 8 Kästen

Hinweis: Zur Berechnung werden Sie den EUKLIDischen Abstand auf einer Ebene nutzen.
Beispiele für Programmaufrufe in der Konsole:

```
findPost (1,4) (4,8) (11,3) (3,2) (6,5) (6,2) (9,6) (3,5)
findPost -f koord.txt
```

Aufgabe 5: (Simulation von Personenbewegungen und Auswertung)
Die Marine der Republik Inebria besitzt ein einziges Schiff. Jedes Jahr, wenn die Marine eine Übung abhält, sticht dieses für einige Tage in See und macht dabei regelmäßig einen Besuch im Hafen von Firewater auf den Booze-Inseln. Hier haben die Matrosen Landgang, bei dem sie die diversen örtlichen Kneipen aufsuchen, und am späten Abend entsprechend betrunken, auf ihr Schiff zurückkehren.

Um auf das Schiff zu gelangen, müssen sie ein Dock überqueren, an dessen Ende ihr Schiff liegt. Normalerweise wäre dies kein Problem, jedoch im betrunkenen Zustand haben inebrianische Seeleute kein besonderes Orientierungsvermögen mehr und auch keine große Lust, das Schiff wieder zu betreten. Daher hat es sich bei ihnen eingebürgert, einen Würfel zur Orientierung zu benutzen. Vor jedem Schritt wird dieser geworfen, und das Ergebnis des Wurfes bestimmt die Richtung des nächsten Schrittes.

Zeigt der Würfel eine 1 oder 4, so geht der Matrose einen Schritt vorwärts, zeigt er eine 2, einen Schritt rückwärts. Ist es eine 3, so geht der Matrose nach rechts, bei einer 5 geht er nach links. Fällt eine 6, so wirft der Matrose noch einmal.

Nun ist das zu überquerende Dock 20 Meter lang und 4 Meter breit.
Die Länge eines Seemannsschrittes beträgt in der Regel 1 Meter.

Die Seeleute betreten das Dock gerade noch in der Mitte. Weichen sie also um mehr als zwei Schritte nach links oder rechts ab, fallen sie ins Wasser. Sie werden nach einiger Zeit von der Hafenwacht an Land gezogen und nüchtern bei der Polizei aus.
Verlassen sie dagegen das Dock wieder durch Rückwärtsschritte, so werden sie von Feldjägern eingefangen und müssen die Nacht an Land in einer Ausnüchterungszelle verbringen.

Es hat sich zudem gezeigt, dass der Würfel - falls er überhaupt so häufig geworfen werden muss - mit dem (`USHRT_MAX - 1`)-ten Wurf ins Wasser fällt, so dass die Matrosen nun dort einschlafen, wo sie gerade stehen, da sie nun keinen Würfel mehr haben, um entsprechend Ihrer Regeln zum Schiff zu gelangen.

a) Man ermittle durch Simulation der Bewegung einzelner Matrosen, wie viele von den 66 Mann Besatzung mit Landgang noch in dieser Nacht vor Mitternacht das Schiff erreichen. Für jeden Matrosen sind dabei die einzelnen Schritte auszugeben, falls sich der Matrose bei einem Wurf von der Stelle bewegt. Außerdem ist bei jedem Matrosen auszugeben, ob er das Schiff erreicht, ins Wasser fällt oder die Nacht in der Zelle verbringen muss.

b) Schreiben Sie ein Auswertungsprogramm, dass die Simulationsroutine einstellbar oft laufen lässt. Dieses Auswertungsprogramm soll die Durchlaufergebnisse der Simulationen, sowie daraus berechnete Werte für Median, Durchschnitt, Varianz und Standardabweichung sowie den Erwartungswert bezogen auf das Ereignis „Matrose erreicht Schiff“ für jeden Matrosen in eine gemeinsame Textdatei schreiben.

c) Wie ändern sich die obigen statistischen Werte, wenn sich die Anzahl der maximal möglichen Würfe des Würfels ändert? Was passiert also, wenn der Würfel (`USHRT_MAX - 1`), (`UINT_MAX - 1`), (`ULONG_MAX - 1`), (`ULLONG_MAX - 1`) mal geworfen werden kann?

Hinweise: Zufallsgenerator, random(), malloc(...), free(…), verkettete Liste, Dateioperationen

Aufgabe 6: (Client-Server Echo)
Ihre Aufgabe ist es zwei Programme zuschreiben mit Namen eServerZ und eClientZ.
eServerZ öffnet auf localhost einen Port und wartet auf Verbindungen.
eClientZ soll sich mit diesem Port verbinden.
Alle Zeichen, die der eClientZ zur Laufzeit von der Tastatur empfängt, sendet er an den eServerZ.
Diese soll der eServerZ zurück an den eClientZ senden. Der eClientZ gibt die Antwort des eServersZ auf der Konsole aus.
Der eClientZ beendet die Verbindung zum eServerZ beim Lesen des EOF-Zeichens.
Hat der eClientZ die Verbindung zum eServerZ unterbrochen soll sich auch der eServerZ beenden.

Betriebsystemkompatiblität Linux/Windows sollen Sie durch bedingte Kompilierung erreichen.

Hinweis: socket(…), connect(…), send(…), recv(...)
Literaturempfehlung: http://openbook.rheinwerk-verlag.de/c_von_a_bis_z/025_c_netzwerkprogrammierung_004.htm