Informatik I
Übungen zum Kapitel 1: C - Eine Einführung in Beispielen

Aufgabe 1-1:

Geben Sie das 'hello world' - Programm ein und
bringen Sie es mit dem C-Compiler zum Laufen!

```
#include

main()
{
  printf("hello world\n");
}
```

Machen Sie sich dabei mit der Entwicklungsumgebung vertraut!
Geben Sie zusätzlich den Text "Hallo Welt !" aus.

Aufgabe 1-2:

Welche Ausgabe erwarten Sie von folgendem Programm?

```
#include
main()
{
printf("\nZeile 1");
printf(" Zeile 2 ");
printf(" Zeile 3 ");
printf(" Letzte Zeile\n");
}
```

Aufgabe 1-3:

Schreiben Sie ein Programm, welches folgendes Muster ausgibt:

```
  *
 * *
* * *
```

Aufgabe 1-4:

Schreiben Sie ein Programm, welches für die Zahlen 1 - 10 jeweils die Zahl
sowie das Quadrat dieser Zahl ausgibt, also

```
1    1
2    4
3    9
4    16
5    25
   .
   .
   .
10   100
```

Hinweis: Verwenden Sie bei der Ausgabe einen Tabulator, damit die Zahlen
geordnet in einer Spalte untereinander erscheinen !

Aufgabe 1-5:

Schreiben Sie ein Programm, welches die Eingabe auf die Ausgabe
kopiert und dabei Tabulatoren durch "\t", Zeilenenden durch "\n"
und mehrere aufeinanderfolgende Leerzeichen durch 1 Leerzeichen
ersetzt.

Aufgabe 1-6:

Schreiben Sie ein Programm, das überprüft ob alle geöffneten Klammern
(runde Klammern, geschweifte Klammern)
in einem Programmtext auch wieder geschlossen werden.
Falls dies nicht der Fall ist, soll angegeben werden, wie viele Klammern
für jede Klammerkategorie öffnend bzw. schließend sind.

Aufgabe 1-7:

Schreiben Sie ein Programm, welches die Längen der eingegebenen
Wörter zählt und am Ende (Ende der Eingabedatei oder Z) die
Wortlängen und die Anzahl der Wörter mit dieser Länge als
Histogramm ausgibt.
Skalieren Sie dabei die Ausgabe der "*" so, dass die maximale Anzahl
von Worten in der 30. Spalte der Ausgabezeile steht.

Die Ausgabe sollte etwa in folgender Form erfolgen
(hier wurde das Programm auf "C_01_ueb.txt" angewandt):

```
Histogramm
   1 (Anzahl=   81) :                    *
   2 (Anzahl=   39) :          *
   3 (Anzahl=   98) :                        *
   4 (Anzahl=   26) :       *
   5 (Anzahl=   27) :        *
   6 (Anzahl=   32) :         *
   7 (Anzahl=   42) :           *
   8 (Anzahl=   45) :            *
   9 (Anzahl=   24) :       *
  10 (Anzahl=   10) :    *
  11 (Anzahl=   13) :    *
  12 (Anzahl=    6) : *
  13 (Anzahl=    3) : *
  14 (Anzahl=    4) : *
  15 (Anzahl=   10) :    *
  16 (Anzahl=    1) : *
  17 (Anzahl=    0) :
  18 (Anzahl=    0) :
  19 (Anzahl=    1) : *
 Es gab 4 Wort(e) mit einer Länge >= 20
```

Übungsaufgaben zu "Programmieren in C"
Kapitel 2 "Datentypen, Operatoren und Ausdrücke"

Aufgabe 2-1:

Schreiben Sie ein Programm zur Bestimmung des Wertebereichs von Variablen vom Typ char, short, int und long, sowohl für signed als auch für unsigned, indem Sie die entsprechenden Werte aus den Standard-Headerdateien (limits.h) ausgeben.
(Erweitern Sie das Programm damit, dass Sie die Wertebereiche zusätzlich berechnen.)

Aufgabe 2-2:

Schreiben Sie ein Programm "HTOI", das eine Zeichenkette mit hexadezimalen Ziffern (einschließlich eines optionalen 0x oder 0X) einliest, in den entsprechenden ganzzahleigen Wert umwandelt und diesen als Dezimalzahl ausgibt.
Die zulässigen Ziffern sind 0..9, a..f und A..F.
Unzulässige Ziffern werden als Ende der Eingabe interpretiert, zuviel eingegebene Ziffern werden ignoriert.

Aufgabe 2-3:

Schreiben Sie ein Programm "SQUEEZE", das zwei Zeichenketten s1 und s2 einliest, jedes Zeichen aus der Zeichenkette s1 entfernt, das in der Zeichenkette s2 vorkommt, und s1 dann ausgibt.

Aufgabe 2-3a:

Schreiben Sie ein Programm "FINDC", das zwei Zeichenketten s1 und s2 einliest, die Position des ersten Zeichens in der Zeichenkette s1 bestimmt, das in der Zeichenkette s2 vorkommt, und diese Position ausgibt.
Enthält s1 kein Zeichen aus s2, so soll "-1" ausgegeben werden.

Aufgabe 2-4:

Schreiben Sie ein Programm "INVERT", das einen Hex-Wert x und zwei dezimale Werte n und p einliest und im Wert x genau die n Bits komplementiert (von 1 in 0 und umgekehrt verwandelt), die an Bit-Position p beginnen und bis p-n+1 reichen.
Der Wert p=0, n=1 komplementiert z.B. das niederfertigste Bit in x. Die anderen Bits sollen unverändert bleiben.
Der neue Wert von x soll wieder hexadezimal ausgegeben werden.
Unsinnige Werte für p und n dürfen für x einen undefinierten Wert ergeben.

Aufgabe 2-5:

Schreiben Sie ein Programm "LOWER", das eine Zeichenkette einliest, alle Grossbuchstaben in Kleinbuchstaben umwandelt und die Zeichenkette ausgibt.
Die Umwandlung darf sich auf die Zeichen A-Z beschränken.
Verwenden Sie für die Konvertierung einen bedingten Ausdruck

anstelle einer if-else-Anweisung und verwenden Sie auch nicht
die Konvertierungsfunktion der Standard-Bibliothek.
(Erweitern Sie das Programm, sodass auch die deutschen Umlaute
konvertiert werden.)

Übungsaufgaben zu "Programmieren in C"
Kapitel 3 "Kontrollstrukturen"

Aufgabe 3-1:

Schreiben Sie ein Programm "SUBSTITUTE", das eine Zeichenkette
einliest und darin die deutschen Sonderzeichen (Ž-š, „-□, á)
durch die jeweiligen Darstellungen Ae, Oe, Ue, ae, oe, ue, ss
ersetzt und dann ausgibt.

Aufgabe 3-2:

Schreiben Sie ein Programm "EXPAND", das eine Zeichenkette s1
einliest und diese nach s2 schreibt, wobei in der Zeichenkette
s1 vorkommende Abkürzungen wie a-z in der Zeichenkette s2
durch die äquivalente vollständige Liste abc...xyz
dargestellt werden sollen.
Berücksichtigen Sie Groß- und Kleinbuchstaben sowie
Ziffern, und akzeptieren Sie auch Angaben wie a-b-c oder
a-z0-9 und auch -a-z.
Am Anfang oder Ende einer Reihe sollte - übernommen werden.

Aufgabe 3-3:

Schreiben Sie ein Programm "NTOB", das eine ganze Zahl n und
eine Basis b einliest und die Zahl n mit der Basis b in einer
Zeichenkette s ablegt.
Zum Beispiel wird bei b=16 die Zahl n in Hexadezimal-
darstellung als Folge von einzelnen hexadezimalen Ziffern
in der Zeichenkette s abgelegt.

Aufgabe 3-4:

Schreiben Sie ein Programm "ITOA", das eine Dezimalzahl n
und eine Breite w einliest. Die Zahl n soll in eine Zeichenkette
ungewandelt werden und mit der minimalen Breite w ausgegeben werden.
Dazu soll die Zeichenkette nach der Umwandlung falls nötig links
mit Leerzeichen aufgefüllt werden, um sie für die geforderte
Breite genügend groß (breit) zu machen.

Übungsaufgaben zu "Programmieren in C"
Kapitel 4 "Funktionen und Programmstruktur"

Aufgabe 4-1:

Implementieren Sie die Funktion strrindex(char s[], char t[]), welche die Startposition des am weitesten rechts im String s vorkommenden Strings t als Ergebnis zurückliefert oder den Wert -1. falls t in s nicht vorkommt.

Aufgabe 4-2:

Erweitern Sie die Funktion atof so, daá sie auch die Konvertierung von Gleitpunktwerten mit Exponenten unterstützt. Einem Gleitpunktwert kann bei dieser Schreibweise optional ein e oder E angefügt werden, dem dann ein Wert mit oder ohne Vorzeichen für den Exponenten folgen muss, wie z.B. bei den folgenden drei Gleitpunktwerten: -47.23e+6, 81.49E-10, .127e3.

Aufgabe 4-3:

Erweitern Sie das Programm zur Auswertung von arithmetischen Ausdrücken in umgekehrter polnischer Notation um den Modulo-Operator % und die Unterstützung der Bearbeitung negativer Zahlen.

Aufgabe 4-4:

Erweitern Sie das Programm zur Auswertung von arithmetischen Ausdrücken in umgekehrter polnischer Notation um Kommandos zur Ausgabe und Duplizierung des obersten Stack-Elementes ohne es vom Stack zu löschen, zum Vertauschen der obersten beiden Stack-Elemente und zum Löschen des gesamten Stack-Inhalts.

Aufgabe 4-5:

Erweitern Sie das Programm zur Auswertung von arithmetischen Ausdrücken in umgekehrter polnischer Notation um die Funktionen sin, exp und pow.

Aufgabe 4-6:

Ändern Sie die Funktion getop so ab, dass ihre Realisierung ohne die Funktion ungetch auskommt. Hinweis: Verwenden Sie zur Lösung eine lokale Variable der Speicherklasse static.

Aufgabe 4-7:

Übertragen Sie die Ideen, die der rekursiven Realisierung der Funktion printd zugrunde liegen, auf eine rekursive Implementierung der Funktion itoa, d.h. ein Integerwert soll in einen String mit Hilfe einer rekursiven Funktion konvertiert werden.

Aufgabe 4-8:

Entwickeln Sie eine rekursive Version der Funktion reverse(char s[]) zum Spiegeln des Strings s.

Aufgabe 4-9:

Definieren Sie ein Makro swap(t, x, y), der die Werte der zwei an der Position x bzw. y übergebenen Parameter vom Typ t vertauscht. Hinweis: Verwenden Sie zur Lösung einen Block.

Übungsaufgaben zu "Programmieren in C"
Kapitel 5 "Zeiger und Vektoren" Teil 1: Abschnitt 1-6

Für die in den folgenden Aufgaben zu implementierenden Funktionen soll jeweils eine Funktion main zum Austesten der Funktionen geschrieben werden.
Bei der Formulierung der Funktionen selbst sollten Zeiger intensiv eingesetzt werden.

Aufgabe 5-1:

Implementieren Sie die Funktion getint, die beim Aufruf Zeichen einliest und daraus (bei geeigneter Eingabe) einen ganzzahligen Wert zusammensetzt. Diese Funktion sollte Ihnen aus der Vorlesung bekannt sein.

Aufgabe 5-2:

Implementieren Sie in Analogie zur Funktion getint die Funktion getfloat, die beim Aufruf Zeichen einliest und daraus (bei geeigneter Eingabe) eine Gleitpunktzahl (z.B. -99.765) zusammensetzt.

Aufgabe 5-3:

Erweitern Sie die Funktion getfloat derart, dass Gleitpunktzahlen Mit Angabe von ganzzahligen Exponenten (z.B. -99.876E-10) verarbeitet werden können.

Aufgabe 5-4:

Schreiben Sie eine Funktion strcat, die zwei Zeichenketten verkettet, d.h. die zweite Zeichenkette an das Ende der ersten Zeichenkette anfügt.

Aufgabe 5-5:

Schreiben Sie eine Funktion ausschnitt, die aus einer Zeichenkette einen Teilstring extrahiert und den ausgeschnittenen Teil in eine zweiten Zeichenkette kopiert. Der Ausschnitt wird durch Angabe des ersten zu kopierenden Zeichens und der Anzahl der zu kopierenden Zeichen definiert.
Lösen Sie diese Aufgabe zunächst unter ausschließlicher Benutzung von Vektoren (Feldern) und in einer zweiten Version unter Benutzung von Zeigern.

Programmierumgebungen

Für die Bearbeitung der Aufgaben können Sie folgende Programmierumgebungen nutzen:
- Borland Turbo-C (DOS)
- Borland C++ (Windows)
- Microsoft Visual C++ (Windows)

Die Umgebungen sind in der Bedienung ähnlich.

Für die Bearbeitung der Aufgaben sollten Sie die Programmierumgebung Borland Turbo-C (unter DOS) nutzen.

> Die Umgebung Microsoft Visual C++ 6.0 ist für den Anfang sehr schwierig zu bedienen. Daher ist von deren Nutzung abzuraten, außer Sie haben schon Erfahrung mit dieser Umgebung.

Anmerkung:
Da C eine echte Untermenge von C++ ist, kann jeder C++-Compiler auch "normale" C-Programme übersetzen.

Hinweis:
Um eine Kompatibilität zwischen verschiedenen Programmierumgebungen sicherzustellen (z.B. hat Borland spezifische Erweiterungen) sollten folgende Einstellungen gemacht werden:
- ANSI-C
- ".c" als Default-Namenserweiterung (statt .cpp)

Die Einstellung dieser Parameter ist jeweils unten beschrieben.

Turbo-C

Starten:
- Öffnen Sie ein MS-DOS-Fenster (Icon "MS-DOS-Eingabeaufforderung")
- Wechseln Sie in ein Verzeichnis, in welchem Sie Schreibrechte haben
  Im C212 ist dies üblicherweise Ihr Home-Verzeichnis, also Laufwerk O:
  oder die Verzeichnisse C:\user und C:\temp
  Empfehlenswert ist das Home-Verzeichnis O: , da Ihnen dann die dort gespeicherten Programme/Daten erhalten bleiben, da niemand anderes sie löschen kann.
- Aufruf im DOS-Fenster: tc

Sie sollten dann folgende Einstellungen machen:
- Als Sprachstandard ANSI-C einstellen:
  "options" -> "compiler" -> "source"
- Default-Namenserweiterung ".c" (statt ".cpp") einstellen:
  "options" -> "environment" -> "editor" -> "default-extension"
- Es sollten dann definierte Pfade eingestellt werden (z.B. das Diskettenlaufwerk), unter denen z.B. die Programmquellen und die übersetzten Programme abgelegt werden:
  Pfade können unter "options" -> "directories" eingestellt werden
  Sie sollten hier ihr Home-Verzeichnis O: oder ein Unterverzeichnis in O: einstellen.
  (Voreinstellung ist oft das Verzeichnis C:\user)

Erstellen und Testen von Programmen:
- Die Eingabe des Programmes erfolgt im Editor-Fenster
- Umgang mit dem Programmcode:
  "file" -> "new" Neues Editor-Fenster (z.B. für neues Programm)
  "file" -> "save" Programm speichern
  "file" -> "open" Gespeichertes Programm öffnen/laden
- Übersetzung starten (Compiler/Linker):

"compiler" -> "built all" Zumindest beim ersten Übersetzen muss
"built all" ausgeführt werden, später
ist auch "compile" ausreichend.
War das Programm fehlerfrei, so erfolgt die Meldung "success".
Lagen Fehler vor, so erfolgen Meldungen "error" (dann konnte kein ablauffähiges Programm erzeugt werden) oder "warning" (es konnte ein ablauffähiges Programm erzeugt werden, aber es wird auf mögliche Ursachen für Laufzeitfehler, auf "unsauberen" Programmcode usw. hingewiesen). Grundsätzlich sollte die Programmierung so erfolgen, daß auch keine "warnings" auftreten !

- Das erzeugte, ablauffähige Programm wird unter dem Namen abgelegt, welcher beim Speichern des Programmcodes gewählt wurde, allerdings mit der extension "exe" (für "executable" / "ausführbar"), also z.B. Programmcode test1.c -> test1.exe
  Wurde kein Name eingstellt, so ist die Voreinstellung "noname00.exe".
- Zum Ablauf des Programmes wählen sie
  - "file" -> "DOS Shell" sie befinden sich dann in der DOS-Umgebung
  - "test1" ruft ihr Programm auf (falls es so hiess)
- Rückkehr in Turbo-C erfolgt durch die Eingabe von "exit"