1. Übung

Ausgabe 25.10.05 — Abgabe 2.11/3.11/4.11.05

Bei Fragen und Problemen können Sie uns per E-mail unter den folgenden Adressen erreichen:
Christian Gollan, Richard Zens, Klaus Macherey {gollan, k.macherey, zens}@cs.rwth-aachen.de

Allgemeine Hinweise

Übungen werden grundsätzlich in den Übungsgruppen abgegeben. Zur Vorlesung gibt es eine Website, die immer die aktuellen Termine und ggf. wichtige Ankündigungen enthält. Dort werden auch die Übungszettel und sonstige Dokumente, die zur Lösung der Aufgaben benötigt werden, abgelegt. Sie erreichen die Website unter:

`http://www-i6.informatik.rwth-aachen.de/HTML/Lehre/SysPro`

In den ersten Übungen werden Sie mit wichtigen Konzepten der Programmiersprache C vertraut gemacht. Es wird davon ausgegangen, daß Sie bereits grundlegende Kenntnisse vom Aufbau eines Programms in einer modularen Programmiersprache besitzen. Machen Sie sich mit Unix vertraut. Alle Programmieraufgaben müssen normalerweise in C (oder C++) und unter Unix entwickelt werden. Mögliche Informationsquellen hierzu sind:

- `http://www.informatik.rwth-aachen.de/Pool/unix.html`
- Handbuch des RRZN: *Unix - Eine Einführung in die Benutzung*
- das Unix-Kommando `man`

Hinweise zu Programmieraufgaben

Im Rahmen der Übungen werden verschiedene Programmieraufgaben in der Programmiersprache C gestellt. Geben Sie bei solchen Aufgaben den Quelltext des Programms und ein Testprotokoll ab. Sie müssen den Quelltext ordentlich kommentieren und darauf achten, daß Ihr Quelltext strukturiert, modular und systemunabhängig ist. Die Verständlichkeit und Übersichtlichkeit des Quelltextes fließen mit in die Bewertung der Aufgabe ein. Schicken Sie bei Programmieraufgaben die Ergebnisse per e-Mail an den Betreuer Ihrer Übungsgruppe (`uebung01...uebung06@i6.informatik.rwth-aachen.de`).

Aufgabe 1 (3 Punkte): Einführung in C

Die erste Teilaufgabe befasst sich mit den verschiedenen Zahlentypen.

Die Programmiersprache C bietet Zahlentypen mit unterschiedlichen Größen, Wertebereichen und Präzisionen an, z.B.:

	Typ	Größe	Wertebereich	Beispiele
Ganzzahlig	(signed) char	8 Bit	-128 ... 127	39, -12, 'a'
	unsigned char	8 Bit	0 ... 255	
	(signed) short	16 Bit	-32768 ... 32767	
	unsigned short	16 Bit	0 ... 65535	
	(signed) int	32 Bit	-2147483648 ... 2147483647	
	unsigned int	32 Bit	0 ... 4294967295	
Fließkomma	float	32 Bit	-1×10^{38} ... 1×10^{38}	-2.0, -1.3e-12
	double	64 Bit	-1.8×10^{308} ... 1.8×10^{308}	

In C gibt es eine automatische (implizite) Typkonversion. Man kann also z.B. folgende Zuweisung machen:

```
int i=3;
float f=2.5;
...
f = i;
```

Implizite Typkonversionen stellen eine erhebliche Fehlerquelle dar. Man sollte daher die Konvertierung explizit durch sog. *casting* anweisen:

```
f = (float)i;
```

Zu beachten ist, dass innerhalb von arithmetischen Ausdrücken auf der rechten Zuweisungsseite ebenfalls Typkonversionen auftauchen können.

a) Übersetzen Sie das Programm mit den Namen `u1_1`. Der zugehörige Quelltext `u1_1.c` ist auf der i6-Webseite verfügbar. Überprüfen Sie die Umrechnung von Grad Celsius in Fahrenheit nach der genäherten Formel

$$t[F] = \frac{9}{5} \cdot t[C] + 32$$

und beseitigen Sie etwaige Fehler im Programm.

b) Erweitern Sie das Programm dahingehend, dass eine Celsius-Fahrenheit– Umrechnungstabelle ausgegeben wird. Es sollen die Temperaturen $-30, -20, -10, \ldots, 100$ Grad Celsius umgerechnet und tabellarisch ausgegeben werden.

Aufgabe 2 (6 Punkte): Einführung in C

Die zweite Aufgabe behandelt Felder. Felder sind Listen mit fester Länge, die Objekte eines bestimmten Typs beinhalten. So ist

```
int a[10];
```

ein Feld (Vektor, Liste, Array, ...) von 10 Elementen, die jeweils vom Typ `int` sind. Ein String ist ein Array mit Elementen vom Typ `char`. Der einzige Unterschied liegt darin, dass Strings mit dem Zeichen \0 enden (Strings sind *null-terminiert*). Ein String vom Typ

```
char s[10];
```

kann also neun Zeichen und das terminierende \0 aufnehmen. Strings können in der Variablendeklaration initialisiert werden, z.B.:

```
char s[10] = "hallowelt";
```

Man kann direkt auf die Elemente eines Arrays zugreifen, indem man ihren Index angibt. Zum Beispiel:

```
a[2] = 13;
```

Dabei ist zu beachten, dass das erste Element eines Feldes den Index 0 hat!

Ein klassischer Verschlüsselungsalgorithmus ist die *Vigenere-Chiffre*. Sie funktioniert wie folgt: Man nimmt zunächst den zu verschlüsselnden Klartext, wandelt alle Klein- in Großbuchstaben um und entfernt alle Leer- und Sonderzeichen. Den Buchstaben A bis Z werden die Indizes 0 bis 25 zugeordnet, die Ziffern 0 bis 9 erhalten die Indizes 26 bis 35. Dann schreibt man unter jeden Buchstaben des Textes einen Buchstaben vom Codewort und addiert die Indizes, z.B. B (1) + E (4) ergibt F (5) und 2 (28) + U (20) ergibt M (48 modulo 36 = 12). Ist der Text länger als das Codewort, wird das Codewort mehrfach wiederholt.

```
Beispiel: Codewort SYSPRO2005
Klartext : Dieser Text soll verschluesselt werden.

Codierung: DIESERTEXTSOLLVERSCHLUESSELTWERDEN
Codewort : SYSPRO2005SYSPRO2005SYSPRO2005SYSP
-------------------------------------------
Chiffre  : V6W7V5L4NOAC3OCSJI2C3IW79SDJM991W2
```

Der Empfänger der Nachricht kann den Text entschlüsseln, indem er unter den chiffrierten Text wieder das Codewort schreibt, diesmal die Indizes jedoch subtrahiert:

```
Chiffre  : V6W7V5L4NOAC3OCSJI2C3IW79SDJM991W2
Codewort : SYSPRO2005SYSPRO2005SYSPRO2005SYSP
-------------------------------------------
Codierung: DIESERTEXTSOLLVERSCHLUESSELTWERDEN

Klartext : Dieser Text soll verschluesselt werden.
```

a) Schreiben Sie ein Programm, das vom Benutzer zunächst ein Codewort (max. 100 Zeichen) und dann einen beliebigen Text (max. 100 Zeichen) erfragt. Fragen Sie ab, ob der Text ver- oder entschlüsselt werden soll. Führen Sie die Vorverarbeitung des Textes sowie die Ver-/Entschlüsselung durch und geben Sie das Ergebnis auf dem Bildschirm aus.

b) Worin könnten Schwächen des *Vigenere-Chiffres* bestehen? Begründen Sie Ihre Vermutung.

Aufgabe 3 (5 Punkte): Einführung in C

Ziel der 3. Aufgabe ist es, dass Sie sich mit den verschiedenen Arten von Variablenvereinbarungen, dem Gültigkeitsbereich von Variablen und der Parameterübergabe bei Funktionsaufrufen vertraut machen.

Grundsätzlich unterscheidet man zwischen globalen und lokalen Variablen. Globale Variablen sind im gesamten Programm gültig. Lokale Variablen können hingegen nur in der Funktion benutzt werden, in der sie deklariert wurden. Globale und lokale Variablen können gleiche Namen haben. Bei Namenskonflikten hat die lokale Variable Vorrang.

Die Parameterübergabe an Funktionen erfolgt nach dem *call-by-value*-Prinzip. Im Funktionskopf werden Variablen deklariert. Diese erhalten bei Ausführung jeweils den Wert zugewiesen, der beim Funktionsaufruf angegeben wurde. Die Variablen können dann zwar im Körper der Funktion modifiziert werden, jedoch wird der neue Wert nicht an die aufrufende Funktion zurückgegeben. Die Parameterrückgabe muss entweder explizit über den Ergebniswert der Funktion (mittels `return`) oder implizit über Speicheradressen erfolgen. Im zweiten Fall wird nicht der Wert einer Variable, sondern die Adresse ihrer Speicherstelle an die Funktion übergeben. Alle Änderungen, die an der adressierten Speicherstelle vorgenommen werden, sind dann automatisch auch in der aufrufenden Funktion sichtbar.

a) Erzeugen Sie das Programm `u1_3a`. Starten Sie es und erklären Sie die Ausgaben. Wieso haben die Variablen `a` und `b` zu unterschiedlichen Zeitpunkten verschiedene Werte? Welchen Rückgabewert hat die Funktion `diff()`?

b) Compilieren Sie nun den Quelltext zum Programm `u1_3b` und starten Sie es. Das Programm hat die Aufgabe, ein Array von Integerzahlen nach dem einfachen *bubble-sort*-Verfahren absteigend zu sortieren. Gleichzeitig soll die größte Zahl in der Variable `max` gespeichert werden. Finden und korrigieren Sie die Programmierfehler. Lassen Sie dabei die Funktionsvereinbarung von `exchange()` unverändert.

c) Globale Variablen sollten so sparsam wie möglich verwendet werden, da sie unerwünschte Nebeneffekte hervorrufen können, evtl. Speicherplatz verschwenden und häufig eine Fehlerquelle darstellen. Verändern Sie den Quelltext von `u1_3b` noch einmal und beseitigen Sie alle globalen Variablen. Sie können dazu auch die Funktionsvereinbarung von `exchange()` modifizieren.

Aufgabe 4 (6 Punkte): Einführung in UNIX

Machen sie sich mit den Möglichkeiten von UNIX vertraut. UNIX bietet eine Vielzahl kleiner Utility-Programme an, die zur Manipulation von Textfiles geeignet sind (z.B. `awk`, `grep`, `cut`, `paste`, `sort`, `uniq`, `diff`, `tr`, `wc`, `cat`, ...). Viele Aufgaben, zu denen man bei anderen Betriebssystemen komplette Programme schreiben müßte, lassen sich unter UNIX durch geschickte Verknüpfung dieser Tools über Pipes direkt auf der Kommandozeile einer Shell lösen.

Lösen Sie die Aufgaben unter Verwendung der genannten UNIX–Tools. Geben Sie neben der Lösung die verwendete Befehlsfolge an.

a) Auf der i6–Webseite finden Sie einen Ausschnitt aus 'In 80 Tagen um die Welt' (`80days.txt`). Laden Sie das File herunter. Entfernen Sie die Zeichen `" ? . ! : ; , + & ‘` ersetzen Sie die Zeichen Apostroph `’` und Bindestrich `-` durch Leerzeichen und entfernen Sie alle Leerzeilen.

b) Wieviele Zeilen und Wörter hat der Text nun? Wieviele Zeilen enthalten die Zeichenfolge “the” nicht (dabei nicht nach Groß– und Kleinschreibung unterscheiden)?

c) Formatieren sie den Text so um, daß jedes Wort getrennt in einer neuen Zeilen steht. Ermitteln Sie, wieviel unterschiedliche Wörter der so vorverarbeitete Text enthält und wie häufig diese sind. Welches sind die 10 häufigsten Wörter?

d) Wie groß ist die durchschnittliche Wortlänge?

e) Wie häufig kommen die einzelnen Buchstaben vor (auch hier nicht nach Groß– und Kleinschreibung unterscheiden)? Ermitteln sie die häufigsten Buchstabenpaare und –tripel (ohne Leerzeichen) im Text.