Human-Computer Interaction Group

Leibniz Universität Hannover

Programmieren 1 – WS 2016/17

Prof. Dr. Michael Rohs, Oliver Beren Kaul, M.Sc.

Übungsblatt 2

Dieses Übungsblatt sollte in Zweiergruppen bearbeitet werden. Beide Gruppenmitglieder müssen die Lösung der Zweiergruppe einzeln abgeben. Die Namen beider Gruppenmitglieder müssen in der Abgabe genannt werden. Plagiate führen zum Ausschluss von der Veranstaltung.

Abgabe bis Donnerstag den 03.11. um 23:59 Uhr über https://assignments.hci.uni-hannover.de. Erstellen Sie alle zur Abgabe gehörigen Dateien im Verzeichnis assignment02 (s.u.). Entfernen Sie alle vom Compiler generierten Dateien (z.B. .exe-Dateien) aus dem Verzeichnis und komprimieren Sie es zu assignment02.zip. Laden Sie diese Datei dann bitte hoch.

Aufgabe 1: Formatierung von Quelltext

a) Für die Lesbarkeit von Quelltext ist die Formatierung sehr wichtig. Quelltext wird häufiger gelesen, als geschrieben. Gegeben sei folgender unformatierter Quelltext.

```
int f ( int i ) { printf ( "called f\n" ) ; if( i < 0 )
{return -i;} else { return 2*i;} }
```

Formatieren Sie diesen Quelltext nach folgenden Regeln:

- `{` ist das letzte Zeichen einer Zeile und steht nicht alleine in einer Zeile
- `}` ist das erste Zeichen einer Zeile, evtl. nach Leerzeichen
- `;` ist das letzte Zeichen einer Zeile und steht nicht alleine in einer Zeile
- kein Leerzeichen vor `;`
- Zeilen in einem `{...}`-Block werden vier Leerzeichen tiefer eingerückt, als der Block selbst
- kein Leerzeichen zwischen Funktionsname und `(`
- ein Leerzeichen nach `if`
- kein Leerzeichen nach `(`
- kein Leerzeichen vor `)`
- ein Leerzeichen vor und ein Leerzeichen nach einem binären Operator (wie `*`)

b) Erklären Sie das Verhalten der Funktion `f` möglichst kurz und prägnant.

Aufgabe 2: Vorbereitung und Überstunden

Hier wird zunächst die Vorbereitung Ihrer „Entwicklungsumgebung" beschrieben. Diese Schritte werden für die weiteren Aufgaben entsprechend sein und werden nur hier ausführlich beschrieben. Das Herunterladen und Kompilieren der Bibliothek ist nur einmalig erforderlich.

Detaillierte Schritte zur Installation der Programming I C Library finden sich unter http://hci.uni-hannover.de/files/prog1lib/index.html. **Führen Sie zunächst diese Schritte aus**, inklusive dem Kompilieren und Ausführen eines Beispiels in `prog1lib/script_examples`.

Laden Sie assignment02.zip mit den Template-Dateien für dieses Übungsblatt aus Stud.IP herunter und speichern Sie assignment02.zip im MinGW-Home-Verzeichnis. Öffnen Sie die MinGW-Shell. Unter Mac OS X und Linux verwenden Sie das Terminal statt MinGW. Die Kommandos sind identisch. Verwenden Sie das Template dieser Aufgabe wie folgt:

`pwd` ← gibt den Pfad des aktuellen Verzeichnisses aus (sollte /home/MyName sein)

`ls` ← listet den Inhalt des aktuellen Verzeichnisses

(prog1lib und assignment02.zip müssen im aktuellen Verzeichnis liegen)

`unzip assignment02.zip` ← Template-Dateien dieser Übung entpacken

`cd assignment02` ← change directory to assignment02

(`overtime.c` mit gutem Texteditor (z.B. Notepad++) editieren, speichern)

`make overtime` ← ausführbares Programm (overtime bzw. overtime.exe) erstellen

`./overtime` ← Programm starten (bzw. `overtime.exe`)

Wichtig: `prog1lib` und `assignment02` müssen im gleichen Verzeichnis liegen (z.B. in `/home/MyName`), da sonst die Bibliothek nicht gefunden wird.

Die letzten drei Schritte (editieren/speichern, `make overtime` und `./overtime`) führen Sie nun wiederholt aus, bis das Programm fertig ist. Die ↑-Taste stellt die vorherige Zeile wieder her. Kompilieren und Ausführen lassen sich kombinieren: `make overtime && ./overtime`

a) Entwickeln Sie eine Funktion, `overtime`, die für eine tatsächliche Wochenarbeitszeit (in Stunden) die darin enthaltenen Überstunden berechnet. Nehmen Sie eine reguläre Wochenarbeitszeit von 40 Stunden an. Verwenden Sie keine Fließkommazahlen. Verwenden Sie die im Skript unter Recipe for Atomic Data beschriebenen Schritte. Verwenden Sie die Template-Datei `overtime.c`. Passen Sie den Inhalt der Datei an die Aufgabe an (Funktionsnamen, Parameternamen, etc.), aber behalten Sie das generelle Layout bei. Geben Sie in der Testfunktion ein Beispiel (Eingabe und erwartetes Resultat) für eine Arbeitszeit unter 40h, ein Beispiel für 40h und ein Beispiel größer 40h an.

b) Generalisieren Sie die in (a) erstellte Funktion so, dass die Wochenarbeitszeit als Konstante definiert wird.

Hinweise:

- Windows und MinGW bei Benutzung von cmd.exe: Tragen Sie, wie in den Folien zur Hörsaalübung beschrieben, die Pfade zu den Verzeichnissen `C:\MinGW\bin` und `C:\MinGW\msys\1.0\bin` in die `PATH`-Umgebungsvariable ein. Dies ist notwendig, damit die Tools (`make`, `gcc`, etc.) gefunden werden.
- Windows und MinGW: Damit das unzip Kommando funktioniert muss der „bin"-Eintrag von msys-unzip im MinGW Installation Manager angewählt und installiert sein.
- Mit den Pfeiltasten (hoch, runter) lassen sich frühere Kommandos wieder abrufen.
- Wenn Sie beim Kompilieren eine Fehlermeldung bekommen, dass „base.h" nicht gefunden wurde, dann weicht Ihre Verzeichnisstruktur von der geforderten ab. Überprüfen Sie erneut die eingangs angegebenen Schritte.

- Die letzten beiden Schritte lassen sich auf der Kommandozeile kombinieren zu: `make overtime && ./overtime`

Aufgabe 3: Zeichenketten abkürzen

Verwenden Sie für diese Aufgabe die String-Funktionen der Bibliothek. Um die String-Funktionen der Bibliothek verwenden zu können, muss am Anfang Ihrer .c-Datei stehen:

`#include "string.h"` (in den Template-Dateien bereits vorhanden)

In der Online-Dokumentation finden Sie diese Funktionen unter Files → File List → string.h. Sie benötigen die Funktionen s_length, s_sub und s_concat. Die Lösung darf keine Schleifen verwenden.

a) Entwickeln Sie eine Funktion, die eine gegebene Zeichenkette (Typ: `String`) auf 8 Zeichen abschneidet, wenn diese länger als 8 Zeichen ist. Verwenden Sie die im Skript unter *Recipe for Atomic Data* beschriebenen Schritte. Verwenden Sie die Template-Datei `truncate_to_8.c`. Passen Sie den Inhalt der Datei an die Aufgabe an, aber behalten Sie das generelle Layout bei. Geben Sie ein Beispiel (Eingabe und erwartetes Resultat) für einen String der Länge kleiner 8, ein Beispiel für einen String der Länge 8 und ein Beispiel für einen String der Länge größer als 8.

b) Generalisieren Sie die in (a) erstellte Funktion so, dass die Länge nicht auf 8 Zeichen fixiert ist, sondern ein Parameter n der Funktion ist. Verwenden Sie die im Skript unter *Recipe for Atomic Data* beschriebenen Schritte. Verwenden Sie die Template-Datei `truncate_to_n.c`. Passen Sie den Inhalt der Datei an die Aufgabe an, aber behalten Sie das generelle Layout bei. Geben Sie ein Beispiel (Eingabe und erwartetes Resultat) für einen String der Länge kleiner n, ein Beispiel für einen String der Länge n und ein Beispiel für einen String der Länge größer als n. Verwenden Sie in den Beispielen zwei verschiedene Werte für n. Insgesamt sind also 6 Beispiele zu erstellen.

c) Verändern Sie die in (b) erstellte Funktion so, dass falls $n \geq 3$ die letzten drei Buchstaben der abgekürzten Zeichenkette durch `...` ersetzt werden. Beispiel: Für n = 6 würde `programming` zu `pro...` abgekürzt werden. Verwenden Sie die im Skript unter *Recipe for Atomic Data* beschriebenen Schritte. Verwenden Sie die Template-Datei `truncate_to_n_dots.c`. Passen Sie den Inhalt der Datei an die Aufgabe an, aber behalten Sie das generelle Layout bei. Geben Sie ein Beispiel (Eingabe und erwartetes Resultat) für einen String der Länge kleiner n, ein Beispiel für einen String der Länge n und ein Beispiel für einen String der Länge größer als n. Verwenden Sie in den Beispielen zwei verschiedene Werte für n. Insgesamt sind also 6 Beispiele zu erstellen.

Hinweis:

- Verwenden Sie in der Testfunktion die Funktion `check_expect_s` für `String` (nicht `check_expect_i`).

Aufgabe 4: Skript

Unter http://hci.uni-hannover.de/files/prog1script/script.html steht ein Skript zur Verfügung, das im Wesentlichen die in der Vorlesung vorgestellte Vorgehensweise bei der Lösung von Programmieraufgaben beschreibt.

a) Lesen Sie Kapitel 3 (Evaluating Functions Without Side-Effects) und beantworten Sie folgende Frage: Warum können Funktionen mit Seiteneffekt so nicht dargestellt werden?

b) Lesen Sie Kapitel 4 (Recipe for Enumerations) und beantworten Sie folgende Frage: Welche Problematik ergibt sich bei der `switch`-Anweisung?

c) Lesen Sie Kapitel 5 (Recipe for Intervals) und beantworten Sie folgende Frage: Welche Rolle spielen Aufzählungen (`enumerations`) und Konstanten (`const`) bei Intervallen?

d) Was war Ihnen beim Lesen der Kapitel 3-5 unklar? Wenn nichts unklar war, welcher Aspekt war für Sie am interessantesten?