Einführung in die Programmierung
WS 2015/2016
Blatt 12

Simon Hangl, Sebastian Stabinger, Benedikt Hupfauf, Johannes Kessler

2015–02–27

- Abgabe bis spätestens Dienstag 21:59:59 über OLAT (`https://lms.uibk.ac.at/olat/dmz/`).
- Bereiten Sie jede Aufgabe so vor, dass Sie Ihre Lösung im Proseminar präsentieren können!
- Benennen Sie Ihre Abgabe nach folgendem Schema: *Gruppennummer-Nachname*`-blatt`*Übungsblattnummer*`.tar.gz` Wenn Sie also Max Mustermann heißen und Gruppe 1 besuchen, heißt die Datei von Übung 12: `1-mustermann-blatt12.tar.gz`
- Compilieren Sie alle Programme mit den Optionen `-Wall -Werror -std=c99`

Feedback

Nutzen Sie die angebotenen Möglichkeiten, uns Feedback zu geben (eMail, Tutorium, Proseminar). Hier können Sie uns auf Probleme, notwendige Stoffwiederholungen, Unklarheiten, aber auch positive Dinge, die beibehalten werden sollten, hinweisen.

Testen und Dokumentation

Stellen Sie sicher, dass alle Lösungen fehlerfrei kompilieren. Testen Sie Ihre Lösungen ausführlich (z.B. auf falsche Eingaben, falsche Berechnungen, Sonderfälle) und dokumentieren Sie sie. Dies hilft Ihnen bei der Präsentation und uns beim Nachvollziehen Ihrer Entscheidungen. Lesen Sie die Aufgaben *vollständig* durch.

Aufgabe 1 (7 Punkte)

In dieser Aufgabe soll eine kleine Produktdatenbank implementiert werden. Ein Datensatz enthält **Name, Länge und Anzahl** von Produkten in einem Lager. Ein Datensatz soll jeweils in einem `struct` gespeichert werden.

Implementieren Sie ein Programm in welchem Sie dem Benutzer die Möglichkeit geben in einem Menü einen *neuen Datensatz* hinzuzufügen, alle vorhandenen *Datensätze auszugeben* und das Programm zu *beenden*. Sie können für die Implementierung ein Array fixer Größe verwenden.

Sie wollen diese Datenbank sowohl in der EU als auch den Vereinigten Staaten verkaufen. Aus diesem Grund müssen Sie in der Lage sein die Länge entweder als `int` in Millimeter oder als `double` in Zoll zu speichern.

Verwenden Sie Präprozessordirektiven um zwei Versionen des Programms compilieren zu können (einmal mit der Länge in Millimeter und einmal in Zoll). Es soll sich sowohl der Typ der Länge in der Struktur ändern als auch die korrekte Längenbezeichnung bei der Ausgabe verwendet werden.

Mittels der Compileroption `-D` können während des Compilierens Präprozessormakros definiert werden. Z.B. führt der Aufruf `gcc -D BLA test.c` dazu, dass in der Datei `test.c` das Präprozessormakro `BLA` definiert ist, so als würde `#define BLA` am Anfang des Source Codes stehen.

Sorgen Sie dafür, dass mit der Option `-D INCH` die amerikanische Version des Programms compiliert wird. Falls diese Option fehlt soll die europäische Version erzeugt werden.

Hinweis: *Sie können große Teile der ersten Aufgabe des letzten Übungszettels wiederverwenden.*

Hinweis: *Abgabe: 1-mustermann-a1.c*

Aufgabe 2 (3 Punkte)

Definieren Sie ein Makro namens `DEBUG` welches einen Parameter `msg` übernimmt. Das Makro soll dazu dienen Debugnachrichten auszugeben.

Um leichter navollziehen zu können woher die Nachricht im Quellcode kommt soll vor der Nachricht automatisch sowohl der Dateiname, die Codezeile, als auch die Funktion ausgegeben werden in der `DEBUG` aufgerufen wurde.

Wir rufen z.B. `DEBUG("Hilfe")` in der Datei `foobar.c` in Zeile 9 in Funktion `test` auf, dann soll die Ausgabe folgendermaßen aussehen:
`foobar.c:test():9 --> Hilfe`.

Frage: Warum müssen wir `DEBUG` als Präprozessormakro definieren und können keine Funktion dafür schreiben?

Hinweis: *Abgabe: 1-mustermann-a2.c*