Praktikum zur Vorlesung
„Grundlagen der Informatik 1“
Studiengang E-Technik

WS 2001/02

AG Betriebssysteme, FB 3
Jan Bredereke

Datum: 24.1.2002

Vorbemerkung

Dieser Text kann auf dem Rechner lablin mit dem Kommando
`xdvi /home/jbredere/pub/aufgaben/praktikum5.dvi &`
angesehen werden. Bei Bedarf kann er gedruckt werden wie in der Beschreibung von LaTeX angegeben. Der Text ist auch im WWW verfügbar.

Polymorphe Funktionen in C

Falls ihr einige der folgenden Aufgaben nicht versteht, oder falls ihr einige der verwendeten Begriffe nicht (mehr) kennt, dann fragt bitte den Praktikumsleiter, damit wir das für euch oder für alle Praktikumsteilnehmer erklären und besprechen können!

Aufgabe 1: Ein ganz einfaches Additionsprogramm

Schreibt ein Programm `myAdd(.c)`, das vom Benutzer zwei Integer-Zahlen per `scanf()` erfragt, sie addiert und die Summe ausgibt. Probiert das Programm aus.

Aufgabe 2: Eine Bibliothek mit einer Additionsfunktion

Schreibt eine Bibliotheksdatei `addfunc.c`, die eine Integer-Funktion `int add(int a, int b)` zur Addition zur Verfügung stellt, und schreibt eine dazugehörige Header-Datei `addfunc.h`. Ändert `myAdd` so, daß es `addfunc` benutzt. Übersetzt und bindet die Dateien und führt sie aus.

Falls Euch z.B. die Kommandos zum Übersetzen und Binden unklar sind, fragt bitte nach!

Hinweis: Im folgenden wird es nützlich sein, beim Übersetzen möglichst strenge Typprüfungen durchführen zu lassen. Übersetzt daher am besten mit `cc -Wall ...`

Aufgabe 3: Addition für einen weiteren Datentyp

Erweitert die Bibliothek `addfunc` um eine zweite Funktion `double add_double(double a, double b)`, die Fließkommazahlen addiert. Erweitert `myAdd` so, daß es erst zwei Integer-Zahlen addiert wie bisher, und dann zwei Fließkommazahlen. Probiert das Programm aus.

Hinweis: Zum Ausgeben von `double`-Zahlen benötigt ihr `printf("%g", ...)`, und zum Einlesen von `double`-Zahlen benötigt ihr `scanf("%lg", &...)`, mit „`l`“.

Aufgabe 4: Parameterübergabe als Referenz und nicht als Wert

Ändert die Funktion `add()` so, daß nicht zwei Integer-Zahlen, sondern zwei Zeiger auf Integer-Zahlen übergeben werden. Auch als Rückgabewert soll ein Zeiger auf das Integer-Ergebnis übergeben werden. Damit die zugehörige Variable nach dem Ende des Aufrufs von `add()` noch existiert, muß sie außerhalb dieser Funktion definiert werden, z.B. direkt davor. Macht das entsprechende für die Funktion `add_double()`. Paßt `myAdd` entsprechend an. Probiert das Programm aus.

Anmerkung: Wird die Funktion `add()` ein zweites Mal aufgerufen, dann wird dabei das Ergebnis des ersten Aufrufs überschrieben, und man sollte den Zeiger auf das erste Ergebnis nicht mehr verwenden.

Dies ist in unserem Falle kein Problem, da wir `add()` in `myAdd` nur einmal zur Addition von Integer-Zahlen aufrufen.

Aufgabe 5: Einführung einer verallgemeinerten, polymorphen Funktion

Ändert die Bibliothek `addfunc` so, daß es nur noch eine Funktion
`void *add(char typ, void *a, void *b)`
gibt, die sowohl Integer- als auch Fließkommazahlen addieren kann. Der neue erste Parameter `typ` gibt an, was zu tun ist. Enthält er `'i'`, dann sind Integer-Zahlen zu addieren, enthält er `'f'`, dann sind Fließkommazahlen zu addieren. Innerhalb der Funktion `add()` unterscheidet ihr die Fälle mit einem `switch`. Paßt `myAdd` entsprechend an.

Damit die Datentypen als korrekt erkannt werden, benötigt ihr Typ-Casts. Falls euch das nicht klar ist, fragt bitte nach!

Aufgabe 6: Weitere Datentypen für die polymorphe Funktion

Erweitert die Funktion `add()` so, daß sie für den Wert `'s'` im ersten Parameter zwei Strings aneinanderhängt und einen Zeiger auf eine Kopie der „Stringsumme“ zurückgibt.

Verwendet die Funktion `strlen()` zur Bestimmung der Längen der beiden Strings, die Funktion `malloc()`, um Speicherplatz für das Ergebnis (einschließlich des abschließenden Null-Bytes!) zu reservieren, und die Funktionen `strcpy()` und `strcat()` zum Kopieren bzw. Anhängen von Strings. Erweitert `myAdd` entsprechend und probiert auch diese neue Funktionalität aus.

Falls ihr Genaueres zur Verwendung den obigen String-Funktionen herausfinden möchtet, kann euch das `man`-Kommando weiterhelfen, z.B. `man strlen`.

Aufgabe 7: Zusatzaufgabe

Denkt euch weitere Datentypen (oder Paarungen von Datentypen) aus, die man „addieren“ kann, und erweitert die Funktion `add()` entsprechend.

Vergleicht euer `add()` mit der Funktion `printf()`. (Siehe `man 3 printf` oder die Ausgabe davon auf den Web-Seiten der Vorlesung unter „Hintergrundinformationen zu Session 2“.)