C/C++ Programmierung – Aufgabenblatt „Kompiliervorgang“

Die **Abgabe** zu diesem Aufgabenblatt besteht aus den folgenden Dateien:

- Aufgabe 1: `bsp.s`
- Aufgabe 2: `bsp.c`, `bspcpp.cpp`, `func.h` und die unveränderte Datei `func.c`
- Aufgabe 3: Das Makefile
- Aufgabe 4: `macrobsp.c`
- Optionale Aufgabe 5: keine weitere Datei, sondern erweiterte Datei `bsp.c` aus Aufgabe 2.

1. Option `-S`

In dieser und der nächsten Teilaufgabe geht es nochmal um die Quellcode-Datei `bsp.c`, die Sie auch schon bei dem vorigen Aufgabenblatt verwendet haben.

Probieren Sie statt `-c` die Option `-S` (großes S). Es entsteht eine Datei mit Endung `.s`.
Kommentieren Sie diese Datei, indem Sie sie in einen Editor laden und mit **„//#“ markiert** Ihre Kommentare einfügen, in denen Sie die folgenden Fragen beantworten:

- Was enthält die Datei?
- Versuchen Sie die Zeilen mit der Initialisierung der Variablen `i,j,k` in dieser Datei wiederzufinden. Schreiben Sie an die entsprechende(n) Stelle(n) einen Kommentar.

Abgabe:

- Laden Sie die Datei `bsp.s` als Teil Ihrer Abgabe auf Moodle hoch.

2. Nutzung von Funktionen aus anderen Dateien/Bibliotheken

Um eine Funktion aus einer anderen Datei oder einer Bibliothek benutzen zu können muss:

- die Schnittstelle (Signatur) der Funktion bekannt sein (normalerweise geschieht das durch Einbinden einer Header-Datei mit `#include`)
- das Object-File oder die Bibliothek mit der Implementierung „gelinkt“ werden.

Die Datei `func.c` (siehe Moodle) enthält eine Funktion zur Berechnung der dritten Wurzel aus einer Zahl.

a) Erweitern Sie die Datei `bsp.c` um die Berechnung der dritten Wurzel aus 3.375 unter Nutzung der Funktion aus `func.c` und lassen Sie das Ergebnis mit `printf` ausgeben. Schreiben Sie dazu eine Header-Datei `func.h` und benutzen Sie `func.h` in `bsp.c` – also `func.c` ***nicht*** einbinden, `func.c` ***nicht*** komplett kopieren und auch ***nicht*** verändern. Linken Sie `bsp.o` und `func.o` sowie (falls nötig) die Mathematik-Bibliothek `libm`.

b) Kopieren Sie Ihre gerade erweiterte Datei `bsp.c` und nennen Sie sie **`bspcpp.cpp`**. Kompilieren und Linken Sie `bspcpp.cpp`, wobei die ausführbare Datei nun `meinecpp` heißen soll.

Abgabe:

- Fügen Sie die **Kommandozeilen** zum Kompilieren/Linken **als Kommentar ganz oben** in die Quellcode-Dateien `bsp.c` und `bspcpp.cpp` ein.
- Laden Sie die Dateien **`bsp.c`**, **`bspcpp.cpp`**, **`func.h`** und die *unveränderte* Datei **`func.c`** auf Moodle hoch.

3. Makefile

Erstellen Sie ein Makefile für die Erstellung des Programms der vorigen Aufgabe. Sie können wählen, ob Sie dies für `bsp.c` oder `bspcpp.cpp` machen möchten.
Unter Windows MSYS2 muss das Make-Programm evtl. zunächst installiert werden mit:
`pacman -S make`
Optional: Schreiben Sie *ein* Makefile für beides (hierfür müssen Sie recherchieren, es ist nicht in den Folien erklärt): Wenn nur `make` eingegeben wird, soll `bsp.c` kompiliert+gelinkt werden, wenn `make meinecpp` eingegeben wird, soll `bspcpp.cpp` kompiliert und als `meinecpp` gelinkt werden.

Abgabe:

- Laden Sie das Makefile auf Moodle hoch.

4. Präprozessor: Macros

Schreiben Sie ein ausführbares c-Programm **`macrobsp.c`** wie folgt:

a) Schreiben Sie ein Macro PYTHAGORAS(a,b), das die Länge der Hypotenuse eines rechtwinkligen Dreiecks gemäß der folgenden Formel berechnet[1]: $\sqrt{a^2 + b^2}$.
Verwenden Sie dabei *nicht* die Funktion `pow`, diese ist (evtl.) langsam und ungenau.

b) Lassen Sie PYTHAGORAS(3,4) und PYTHAGORAS(1+3,2+1) (exakt so hinschreiben!) berechnen und ausgeben.

c) Sind die Ergebnisse wie erwartet, nämlich jeweils 5? Wenn nicht, was ist zu korrigieren?
Beantworten Sie dies in der Quellcode-Datei in einem mit **„//#c)“** markierten Kommentar.

d) Probieren Sie statt `-c` die Kommandozeilenoption `-E` von `gcc`. Was passiert?
Beantworten Sie dies in der Quellcode-Datei ganz am Ende in einem mit **„//#d)“** markierten Kommentar.

Abgabe:

- Datei **`macrobsp.c`** mit den mit **„//#c)“** bzw. **„//#d)“** markierten Kommentaren als Antworten zur den Aufgabenteilen c) bzw. d).

5. Präprozessor: Bedingte Kompilierung

Ändern Sie die Datei `bsp.c` aus Aufgabe 2 wie folgt:

- Benutzen Sie einen `#ifdef` / `#endif` Block um (zur Kompilierzeit) mit Hilfe der Definition einer Makro-Konstanten `ZEIGEVARS` auswählen zu können, ob die ersten vier Ausgaben des Hauptprogramms angezeigt werden oder nicht.
- Mit Hilfe der zusätzlichen Option „`-D`“ von gcc kann von der Kommando-Zeile die Makro-Konstante definiert werden (ohne Leerzeichen zwischen `-D` und dem Namen der Konstanten, also `-DZEIGEVARS`). Probieren Sie auch nochmal „gcc -E".

Abgabe:

- Fügen Sie die **Kommandozeilen** zum Kompilieren/Linken **als Kommentar ganz oben** in die Quellcode-Datei `bsp.c` ein.
- Laden Sie die gemäß dieser Aufgabe geänderte Datei **`bsp.c`** statt der Datei `bsp.c` mit nur der Lösung aus Aufgabe 2 hoch.

[1] Also diese Formel als Macro implementieren – dabei `sqrt` aufrufen, aber *nicht* `hypot`.