Aufgaben

Modulare Programmierung und Bibliotheken

Aufgabe 1

a)

Schreibt zu folgendem Quelltext einen Header, der alle mit `extern` markierten Symbole exportiert. Achtet dabei darauf, Include-Guards zu benutzen!

```
1  extern int zahl = 0;
2  static int privat = 99;
3  extern int zaehler = 0;
4
5  static void benutzePrivat()
6  {
7      privat = privat + 1;
8  }
9
10 extern int getZaehler()
11 {
12     return ++zaehler;
13 }
14
15 extern void setZahl(int setTo)
16 {
17     zahl = setTo;
18 }
```

b)

Bindet den Header korrekt in diesen Quelltext ein, kompiliert und führt das Programm dann aus!

```
1 #include <stdio.h>
2
3 int main()
4 {
5     setZahl(42);
6     printf("Dies ist die %d. Funktion. %d kommt sehr oft
        ↪ als Beispielzahl vor...", getZaehler(), zahl);
7     getchar();
8     return 0;
9 }
```

Aufgabe 2

Versucht mit einem eigenen Beispiel zirkuläre Abhängigkeiten hervorzurufen! Benutzt dafür mindestens vier verschiedene Header und probiert das ganze aus, indem ihr versucht zu kompilieren!

Aufgabe 3

a)

Schreibt folgende Funktionen in jeweils eine Datei und kompiliert diese einzeln mit dem Compiler-Flag `-c`!

```
1  /** Funktion 1 */
2  long max (long a, long b)
3  {
4      return((a > b) ? a : b);
5  }
6
7  /** Funktion 2 */
8  long min (long a, long b)
9  {
10     return((a < b) ? a : b);
11 }
```

b)

Erzeugt aus beiden Objektdateien eine statische Bibliothek mit dem Befehl
`ar rs libminmax.a max.o min.o`

c)

Erstellt dann eine dynamische Bibliothek mit dem Befehl
`gcc -shared -o libminmax.so min.o max.o`

d)

Schreibt einen passenden Header und bindet ihn in folgenden Programmcode ein!

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("%d ist groesser als %d", max(2, 3), min(2, 3));
6     getchar();
7     return 0;
8 }
```

Benutzt nun erst die statische Bibliothek, indem ihr beim Übergeben an den Compiler explizit `-static` angibt, benutzt danach die dynamische Bibliothek, indem ihr den Befehl weglässt!