Imperative Programmierung
Aufgabenblatt VII

1. Das rekursive Programm zur Bestimmung der Fibonacci-Zahlen wurde in der Funktion „fib1“ in der Datei fib.c implementiert. Das Programm benötigt allerdings 4 min und 27 s, da bei einer rekursiven Umsetzung viele Zahlen doppelt berechnet werden. Dies wird in Aufgabe 2 verbessert.
2. Zur Verbesserung des Programmes kann die dynamische Programmierung verwendet werden und wurde in der Funktion „fib2“ in der Datei fib.c implementiert. Das Programm definiert zuerst die erste und zweite Fibonacci-Zahl als 1. Danach werden sich immer die Letzen beiden Fibonacci-Zahlen in den Variablen a und b gespeichert. Somit müssen keine Zahlen doppelt berechnet werden und das Programm läuft in 1 s und 89 ms.
3. Die Funktion „fnord“ nimmt als Input eine 0 vom Typ „double“. Dann wird pro Aufruf der Funktion eine Ziffer eingelesen, diese Zahl mit 10 multipliziert und die Ziffer zur Zahl addiert. Am Ende gibt die Funktion die eingegebene Zahl als double zurück.
4. Eine iterative Implementierung von „fnord“ wurde in der Funktion „fnord2“ in der Datei fnord.c umgesetzt.
5. Das Programm zur Berechnung einer „beliebigen“ Fakultät wurde in der Funktion „fak“ in der Datei fak.c implementiert. Das Programm kann zwar keine beliebige Fakultät berechnen, allerdings bis zu n = 100.000 (also 100000!). Diese Zahl ist allerdings auch schon größer als 10^{456573}.
   Zur Berechnung wird ein Array der Länge 50000 genutzt. In dem jeweils 10 Ziffern vom Ergebnis gespeichert werden. Man könnte die Länge vom Array auch noch erweitern, allerdings würde die Berechnung dann auch mehr Zeit in Anspruch nehmen.
   Man muss nur bedenken, dass die Multiplikation der größten 10-stelligen Zahl mit n, kein größeres Ergebnis als der Maximale Wert des Typs „unsigned long long“ liefert, da sonst falsche Ergebnisse entstehen könnten.