Allgemeines

Siehe Testat 0

Aufgabe 4: Funktionen

Fassen Sie Ihr in Aufgabe 3 entwickeltes Programm in eine Funktion mit der Schnittstelle

```
int primfaktor(int);
```

Die Funktion gibt bei jedem Funktionsaufruf mit einem Parameter n den nächsten Primfaktor von n zurück, so wie Testat 3 beschrieben (also numerisch aufsteigend, mehrfach vorkommende Faktoren werden auch mehrfach zurückgegeben). Sobald sich der Wert der übergebenen Zahl n ändert beginnt die Funktion wieder von vorne mit der Primfaktorzerlegung. Hat die Funktion alle Primfaktoren für ein n errechnet und zurückgegegeben so gibt sie (bis zum Aufruf mit einem neuen Wert n) immer -1 zurück.

Im Fehlerfall gibt die Funktion immer den Wert -2 zurück.

- Die gesamte Kommunikation der Funktion muss über die **Schnittstelle** erfolgen.
- Innerhalb der Funktion sind **keine Ein- und Ausgaben** erlaubt.
- Es sind **keine globalen Variablen** erlaubt.
- Ein Prototyp der Funktion ist Pflicht.
- Die in Testat 3 notwendige Fallunterscheidung für Primzahlen ist nicht notwendig. Wenn eine Primzahl zu zerlegen ist, dann wird diese eben als einziger Primfaktor zurückgegeben.
- Die main-Funktion ist **nicht Bestandteil des Testats**, sollte aber in der Datei vorhanden sein, damit der **Compilertest** bestanden wird (testen Sie hierfür Ihr Programm mit dem Online-Compiler!).
- Das Programm ist mit dem Dateinamen `<matrikel-nr>-testat-4.c` (also bspw. 12345-testat-4.c) bei moodle hochzuspielen, Termin ist 16.12.2019 10:00 Uhr

Eine Main-Funktion, welche die zu entwickelnde Funktion testet, könnte z.B. folgende sein:

```
int main() {
        int pf;

        while ((pf = primfaktor(12)) > 1)
                printf("%d\n", pf);
        while ((pf = primfaktor(7)) > 1)
                printf("%d\n", pf);
        printf("%d\n", primfaktor(1));
        return 0;
}
```

Die Ausgabe dieses Hauptprogramms ist dann gemäß der Funktionsbeschreibung

```
2
2
3
7
-2
```