Prof. Dr. Alexander Voß

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Selbstlernkontrolle

Lösen Sie folgende Aufgaben ohne Hilfsmittel (!):

- Lesen Sie Aufgabe komplett durch!
- Implementieren Sie nicht alles auf einmal. Gönnen Sie sich Zwischenschritte.
- Implementieren und Testen Sie ein Funktion `IsPrime`, die testet, ob es sich bei einer übergebenen ganzen Zahl `n` um eine Primzahl handelt oder nicht.
  Geben Sie diese Information zurück (prim oder nicht) und falls nicht, den ersten Teiler, den Sie gefunden haben.
  Geben Sie das Ergebnis aus.

  Testbeispiele: n=23 und n=42

  Zusatzaufgabe 1: `n` wird von der Tastatur eingelesen.
  Zusatzaufgabe 2: Schreiben Sie eine Menge von Tests mit Hilfe der Funktion `assert`.

  Tipp: Implementieren Sie zunächst ein leeres Programm mit einer einfachen main-Funktion, welches kompiliert und läuft. Danach überlegen Sie mit Ihrem Nachbarn, wie die Signatur der Funktion am besten aussieht, so dass Sie zwei Informationen zurückgeben können. Implementieren Sie zunächst die angegebenen Testbeispiele.

Selbstlernkontrolle

Lösen Sie folgende Aufgaben:

- Definieren Sie einen struct Polynom mit einem Pointer coeffs auf doubles und einem int grad.
- Definieren Sie einen typedef Polynom_t vom Type struct Polynom. Alternativ arbeiten Sie mit struct Polynom.

```
typedef struct Polynom Polynom_t;
```

- Implementieren Sie eine Funktion Init2 mit folgender Signatur, die ein Polynom 2. Grades dynamisch anlegt und diesen Zeiger zurückgibt. Die drei Koeffizienten sind ebenfalls dynamisch angelegt und mit den übergebenen Werten gefüllt.

```
Polynom_t* Init2(double x0, double x1, double x2)
```

- Das folgende Programm sollte laufen:

```
int main()
{
    Polynom_t* p = Init2(1.0,2.0,3.0);
    printf("(%f)x^2+(%f)x^1+(%f)x^0,grad=%d \n",
    p->coeffs[2],p->coeffs[1],p->coeffs[0],
        p->grad);
    return EXIT_SUCCESS;
}
```

Selbstlernkontrolle

Ich habe folgende Themen in C verstanden:

- Eigene Programme erstellen, compilieren und linken.
- Strings, Zahlen und Zeichen verwenden und via printf formatiert ausgeben und einlesen.
- Statische Variablen verwenden.
- Schleifen aller Art sowie if und switch benutzen.
- Funktionen deklarieren, definieren und aufrufen. Gleiches mit einer variablen Anzahl von Argumenten.
- assert und casts verwenden.
- Pointer/Zeiger sowie Arrays inkl. Zeigerarithmetik verwenden. Darunter fallen auch Zeiger auf Zeiger, konstante Zeiger, Zeiger auf Konstanten und Zeiger auf Funktionen
- Speicher dynamisch anfordern und freigeben.
- Stringfunktionen wie strcpy, strlen, strcmp einsetzen.
- Dateien lesen und schreiben.
- Datenstrukturen vom Typ struct, union und enum sowie typedef einsetzen.
- Makros und #defines.