Fakultät IV Elektrotechnik und Informatik
Einführung in die Programmierung WS 2022/2023
Manfred Hauswirth
Damien Foucard / Uwe Kuehn / Aljoscha Meyer

Aufgabenblatt 6

Ausgabe: *25.10.2022 14:00*
Abgabe: *27.10.2022 22:00*

Thema: *Pointer Syntax & Speicher*

Abgabemodalitäten

1. Die Aufgaben des C-Kurses bauen aufeinander auf. Versuche daher die Aufgaben zeitnah zu bearbeiten.

2. Alle abzugebenden Quelltexte müssen ohne Warnungen und Fehler auf Deinem Rechner mit dem Befehl `clang -std=c11 -Wall -g` kompilieren.

3. Die Abgabe für den Quellcode erfolgt ausschließlich über unser Git im entsprechenden Branch. Nur wenn ein Ergebnis im ISIS-Kurs angezeigt wird, ist sichergestellt, dass die Abgabe erfolgt ist. Die Abgabe ist bestanden, wenn Du an Deinem Test einen grünen Haken siehst.

4. Du kannst bis zur Abgabefrist beliebig oft neue Versionen abgeben. Lies Dir die Hinweise der Tests genau durch, denn diese helfen Dir, Deine Abgabe zu korrigieren.
Bitte beachte, dass ausschließlich die letzte Abgabe gewertet wird.

5. Die Abgabe erfolgt, sofern nicht anders angegeben, in folgendem Branch: `ckurs-b<xx>-a<yy>`, wobei `<xx>` durch die zweistellige Nummer des Aufgabenblattes und `<yy>` durch die entsprechende Nummer der Aufgabe zu ersetzen sind.

6. Gib für jede Aufgabe die Quellcodedatei(en) gemäß der Vorgabe ab. Im ISIS-Kurs werden zum Teil Vorgabedateien bereitgestellt. Nutze diese zur Lösung der Aufgaben.

7. Die Abgabefristen werden vom Server überwacht. Versuche, Deine Abgabe so früh wie möglich zu bearbeiten. Du minimierst so auch das Risiko, die Abgabefrist auf Grund von „technischen Schwierigkeiten“ zu versäumen. Eine Programmieraufgabe gilt als bestanden, wenn alle bewerteten Teilaufgaben bestanden sind.

Sieb des Eratosthenes

Primzahlen zwischen 2 und n können mit dem nach *Eratosthenes von Kyrene* bezeichneten Verfahren berechnet werden.

Datenstruktur
Das Verfahren benötigt ein Array Z, in dem für alle Zahlen z_i zwischen 2 und n die Information gespeichert ist, ob es sich um Primzahlen (ausgedrückt als 1) oder Nicht-Primzahlen (ausgedrückt als 0) handelt. Zunächst wird davon ausgegangen, dass alle Zahlen Primzahlen sind. Im Verlauf des Programms wird das nach und nach korrigiert, indem alle Vielfache von Primzahlen als Nicht-Primzahlen markiert werden.
Achtung: Da in `C` die Indizierung eines Arrays mit 0 beginnt, laufen die Indizes von 0 bis $n-2$, während die repräsentierten Zahlen von 2 bis n laufen. Anders ausgedrückt: Der Index i des Arrays repräsentiert die Zahl z_i, also $z_i = i + 2$.

Verfahren
Das Verfahren ist im Folgenden Schritt für Schritt erklärt:

1. Deklariere ein Array der geeigneten Länge für jede Zahl zwischen 2 und inklusive n. (Also der Größe $n-1$)
2. Initialisiere jedes Element der Liste mit 1. (Wir nehmen an, dass es eine Primzahl ist.)
3. Für jedes i-te Element zwischen 2 und n mache Folgendes:
   - Wenn das Element am Index i den Wert von 1 hat (d.h. es ist eine Primzahl) :
     - Setze alle Vielfache dieser Zahl z_i auf 0 (d.h. es ist keine Primzahl). Dies wird erreicht, indem alle Vielfache x der Indizes $i+x*z_i = i+x*(i+2)$ auf 0 gesetzt werden. (Begründung: Jedes Vielfache einer Primzahl ist keine Primzahl.)
4. Jetzt gilt: Alle Zahlen z_i, für die `array[i] == 1` gesetzt ist, sind Primzahlen.

Anders als im Original wird in dieser Variante keine Quadratwurzel gezogen.

Beispiel für $n = 9$

1. Initialisiere das Array.

```
Index (i): 0, 1, 2, 3, 4, 5, 6, 7
Zahl (z_i): 2, 3, 4, 5, 6, 7, 8, 9
array[i]: 1, 1, 1, 1, 1, 1, 1, 1
```

2. Für Zahl $z_i = 2$ ($i = 0$), werden alle Vielfache von 2 auf 0 gesetzt

```
Index (i): 0, 1, 2, 3, 4, 5, 6, 7
Zahl (z_i): 2, 3, 4, 5, 6, 7, 8, 9
array[i]: 1, 1, 0, 1, 0, 1, 0, 1
```

3. Für Zahl $z_i = 3$ ($i = 1$), werden alle Vielfache von 3 auf 0 gesetzt

```
Index (i): 0, 1, 2, 3, 4, 5, 6, 7
Zahl (z_i): 2, 3, 4, 5, 6, 7, 8, 9
array[i]: 1, 1, 0, 1, 0, 1, 0, 0
```

4. Die Zahlen $4, 6, 8, 9$ werden übersprungen, da sie schon auf 0 gesetzt sind.
5. Die Zahlen 5 und 7 bewirken keine Veränderungen am Zustand des Arrays, da ihr Vielfaches ($10, 15, \ldots$ und $14, 21, \ldots$) kleiner als n ist.
6. Primzahlen sind alle Zahlen, die einen Wert von 1 besitzen. Also $2, 3, 5, 7$.

Aufgabe 1 Primzahlensieb

Schreibe ein Programm, das alle Primzahlen zwischen 2 und einer eingelesenen Zahl n, einschließlich n, ausgibt. Zu diesem Zweck soll das Sieb des Eratosthenes (siehe oben) verwendet werden. (Die Zahl 1 ist keine Primzahl.)

Damit das Programm mit beliebig großen Zahlen umgehen kann, soll dynamisch Speicher reserviert werden. Reservierter Speicher muss wieder freigegeben werden. Da bei dynamischer Speicherreservierung häufig die Menge des Speicher von Benutzereingaben abhängt, ist ein sparsamer Umgang mir Speicher unabdinglich. Versuche den Speichergebrauch deines Programms gering zu halten.

Ein beispielhafter Aufruf inkl. Ausgabe des Programms wird in Listing 1 gezeigt.

Listing 1: Programmbeispiel

```
> clang -std=c11 -Wall -g ckurs_blatt06_aufgabe01.c input2.c
                -o ckurs_blatt06_aufgabe01_loesung
> ./ckurs_blatt06_aufgabe01
Bitte gebe eine Nummer ein: 10
2, 3, 5, 7,
```

Wie Du in Listing 1 sehen kannst, kompilieren wir die Aufgabe wieder mit einer Bibliothek, `input2.c`. Diese Bibliothek stellt die folgenden Funktionen zur Verfügung:

- `int lese_int()`
  Diese Funktion gibt eine eingelesene Zahl zurück.
- `void print_prim(int *array, int laenge)`
  Diese Funktion gibt die zuvor berechneten Primzahlen im Array `array` der Länge `laenge` aus.

Stelle sicher, dass die Dateien `input2.c` und `input2.h` im selben Ordner liegen.

Um die Hausaufgabe zu vereinfachen, bitten wir Dich, die vorgegebene Programmstruktur zu verwenden (siehe Listing 2). Den Fall, dass $n = 0$ musst Du nicht gesondert behandeln. Die Abgabe muss folgenden Kriterien entsprechen:

- Die Zahl n wird mithilfe von `lese_int` eingelesen.
- Die Primzahlen zwischen 2 und n, einschließlich n, werden ausgegeben. Dafür muss die Funktion `print_prim` verwendet werden.
- Die Datei `input2.c` darf nicht verändert werden.
- Der Speicher für das Array mit $n-1$ Feldern des Typs `int` wird mithilfe von `malloc` reserviert.
- Dynamisch reservierter Speicher wird wieder freigegeben.
- Es werden keine zusätzlichen Bibliotheken verwendet.

Listing 2: Mögliche Programmstruktur

```c
#include <stdio.h>
#include <stdlib.h>
#include "input2.h"

int main() {
    int n = lese_int();
    int laenge = n-1;

    // Hier implementieren

    // Mit print_prim Primzahlen ausgeben
    //print_prim(array, laenge);

    return 0;
}
```

Nutze zur Lösung der Aufgabe die Vorgaben aus unserem ISIS-Kurs. Füge Deine Lösung als Datei `ckurs_blatt06_aufgabe01.c` im entsprechenden Abgabebranch in Dein persönliches Repository ein und übertrage die Lösung an die Abgabeplattform.