Prozedurale Programmierung – Übung 8

WS 2022/23 *Johannes Jurgovsky*

Hochschule **Rosenheim**
University of Applied Sciences

Aufgabe 1

In der Community wird ein (fast leeres) Projekt in der Datei „primzahlen.zip“ bereitgestellt.

Schreiben Sie ein C-Programm, das alle Primzahlen zwischen 2 und einer zur Übersetzungszeit festzulegenden Obergrenze `OG` (einschließlich) berechnet und auf dem Bildschirm ausgibt. Die Berechnung soll durch das „Sieb des Eratosthenes“ erfolgen. Dieser Algorithmus funktioniert wie folgt:

- schreibe alle natürlichen Zahlen von 2 bis zur Obergrenze OG auf; zunächst sind alle diese Zahlen potentiell Primzahlen und werden als „prime“ markiert
- gehe nun die noch als prim markierten Zahlen durch (es genügt bis zur Wurzel aus OG) und markiere alle ihre Vielfachen als „not prime“
- alle am Ende noch als „prime“ markierte Zahlen sind tatsächlich Primzahlen

Anmerkungen:

- Für die Berechnung müssen die Zahlen von 2 bis OG nicht gespeichert werden; es ist ausreichend ein Array mit Markierungen `PRIME` oder `NOT_PRIME` zu speichern. Der Arrayindex steht dabei stellvertretend für die betrachtete Zahl.
- Das Array zur Speicherung der Markierungen ist in der `main()` bereits definiert. Es hat die Länge N = OG + 1. Damit entspricht der letzte Index im Array der Obergrenze OG.
- Verwenden Sie zum Markieren der Arrayindizes die vordefinierten Konstanten `PRIME` und `NOT_PRIME`.
- Implementieren Sie die Funktion `sieve(int markers[], int N)`, die die tatsächliche Berechnung nach dem oben beschriebenen Algorithmus durchführt. Diese soll als Parameter das Markerfeld sowie dessen Größe übergeben bekommen.
- Implementieren Sie die Funktion `show(int markers[], int N)`, die die markierten Primzahlen auf dem Bildschirm durch Komma getrennt ausgibt. Diese soll als Parameter ebenfalls das Markerfeld sowie dessen Größe übergeben bekommen. Zur Vereinfachung darf am Ende der ausgegebenen Primzahlen noch ein letztes Komma stehen, also z.B.: 2, 3, 5, 7,

Ausgabe:

```
--------------------------------------------------
Sieb des Eratosthenes
--------------------------------------------------
Primzahlen zwischen 1 und 1000:
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73,
79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163,
 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251,
 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349,
 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443,
 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557,
 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647,
 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757,
 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863,
 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983,
 991, 997,
```

Aufgabe 2

Diese Aufgabe kann unabhängig von Aufgabe 1 bearbeitet werden.

Implementieren Sie die Funktion `void selectionSort(struct Vector2_s data[], int n)`, die Vektoren vom Typ `struct Vector2_s` in einem Array aufsteigend nach deren Länge (L2Norm) mittels „Selectionsort“ sortiert. Die Prototypen der benötigten Hilfsfunktionen sind in der `vectors.h` bereits

angelegt.

- Informieren Sie sich über die Funktionsweise des SelectionSort-Verfahrens
- Lagern Sie die Berechnung der Vektornorm in die Funktion `normL2` aus.
- Lagern Sie die Vertauschung zweier Vektoren in die Funktion `swap` aus.
- Lagern Sie die Suche nach der Arrayposition, die den kürzesten Vektor enthält, in die Funktion `argMin` aus.