Prozedurale Programmierung – Übung 8

WS 2019/20 *Prof. Dr. F.J. Schmitt*

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 den „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 „prim“ markiert
- gehe nun die noch als prim markierten Zahlen durch (es genügt bis zur Wurzel aus OG) und markiere alle ihre Vielfachen als „nicht prim“
- alle am Ende noch als „prim“ markierte Zahlen sind tatsächlich Primzahlen

Anmerkungen:

- definieren Sie in Ihrem Programm eine Obergrenze OG so, dass alle Primzahlen zwischen 2 und 1000 berechnet werden
- verwenden Sie zum Markieren der Zahlen als „prim“ bzw. „nicht prim“ ein Feld vom Typ int. Als Marker sollen die vordefinierten Konstanten PRIM und NICHT_PRIM dienen. Das Feld soll in main() definiert werden.
- Schreiben Sie eine Funktion sieb(), die die tatsächliche Berechnung nach dem oben beschriebenen Algorithmus durchführt. Diese soll als Parameter das Zahlenfeld sowie die Feldobergrenze übergeben bekommen.
- Schreiben Sie eine Funktion ausgabe(), die die berechneten Primzahlen auf dem Bildschirm durch Komma getrennt ausgibt. Diese soll als Parameter ebenfalls das Zahlenfeld sowie die Feldobergrenze 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,
Servus!
```

Aufgabe 2

Erweitern sie Aufgabe 1 mit dynamischer Speicherverwaltung so, dass die Obergrenze des Feldes zur Laufzeit eingelesen wird und der Speicher im Heap verwaltetb wird.

Aufgabe 2

Das folgende Programm implementiert eine Sortierfunktion (Bubblesort) auf einem Feld ganzer Zahlen und ruft diese Funktion in main() mit Beispieldaten auf. Es wird ein unsortiertes Feld an die Funktion übergeben, und nach dem Aufruf gibt main() das sortierte Feld aus. Die Ausgabe des Codes soll also sein: 1, 2, 2, 3, 5, 8, 10

Der unten stehende Code enthält Fehler, sowohl syntaktische (d.h., der Compiler zeigt beim Übersetzen eine Fehlermeldung an) als auch semantische (d.h., das Programm lässt sich übersetzen, läuft aber falsch).
Suchen Sie die Fehler und tragen Sie die Korrekturen in den unten stehenden Code ein!

```
#include<stdio.h>;
#include<stdlib.h>
#include<math.h>

void bubblesort(int *array, int length);

int main()
{
	int feld[] = {1, 2, 10, 5, 3, 2, 8};
	int anzEl = sizeof(feld);

	bubblesort(&feld, anzEl);

	int i;

	for(i = 0; i < anzEl - 1; i++)
		printf("%d, ", feld[i]);
	printf("%d\n", feld[anzEl - 1]);

	return 0;
}

void bubblesort(int *array, int length)
{
	int i, j;

	for (i = 0; i < length - 1; --i)
	{
		for (j = 0, j < length - i - 1; ++j)
		{
			if (array[j] > array[j + 1])
			{
				int tmp = array[j];
				array[j] = array[j + 1];
				array[j + 1] = tmp;
			}
		}
	}
}
```