Übung 5

Aufgabe 1: Sortieren durch Einfügen

In dieser Aufgabe geht es um Felder (Arrays), wie Sie sie in der Vorlesung kennengelernt haben. Um die Handhabung von eindimensionalen Arrays zu üben, werden Sie ein Programm schreiben, das ein Feld sortiert. Dieses Feld muss zuerst mit Zufallszahlen gefüllt werden.

Den Vorgang zum Sortieren erklären wir am Beispiel [2 4 3 1] wie folgt:

1. Wir betrachten das Feld nur bis zur ersten Zahl: [2]. Dieses Feld ist offenbar schon sortiert.
2. Wir nehmen die nächste Zahl im Feld hinzu: Wir haben Glück: Da $2 < 4$, ist das Feld [2 4] schon sortiert.
3. Wir nehmen die nächste Zahl im Feld hinzu: Wir haben ein sortiertes Feld [2 4 .] und die Zahl 3, sowie eine Lücke an der Stelle der Zahl 3. Solange die Zahl vor der Lücke grösser ist als die 3, schieben wir die Zahl vor der Lücke in die Lücke: [2 . 4]. Die Lücke ist an der richtigen Stelle: Hier setzen wir die 3 ein: [2 3 4].
4. Wir nehme die nächste Zahl im Feld hinzu: Wir haben ein sortiertes Feld [2 3 4 .] und die Zahl 1. Solange die Zahl vor der Lücke grösser ist als die 1 oder wir nicht die erste Stelle erreicht haben, wird die Zahl vor der Lücke zur Lücke verschoben: $1 < 4$: [2 3 . 4], $1 < 3$: [2 . 3 4], $1 < 2$: [. 2 3 4]. Nun ist die Lücke ganz vorne. Wir können sie nicht weiter verschieben. Hier setzen wir die 1 ein: [1 2 3 4].
5. Es gibt keine weiteren Zahlen im Feld. Das Feld ist sortiert.

Aufgabe: Schreiben Sie ein Programm, das folgende Aufgaben in der angegebenen Reihenfolge erledigt:

1. Für eine von Ihnen frei wählbare Konstante N füllt es das Feld
   ```
   int a[N];
   ```
   mit Zufallszahlen.
2. Es gibt daraufhin den Inhalt des Feldes aus.
3. Dann sortiert es dieses Feld. Dabei verwendet es den oben angegebenen Sortieralgorithmus, den Sie so verallgemeinert haben, dass er auf dieses Feld anwendbar ist.
4. Anschliessend soll wiederum der Inhalt des Feldes ausgegeben werden.

Aufgabe 2: Hornerschema

In dieser Aufgabe kommen wir auf Übung 4, Aufgabe 3 (Polynom auswerten) und Übung 3, Aufgabe 2 (Sinuskurve zeichen) zurück. Ging es in Übung 4 um Verwendung von Feldern, so soll es in dieser Aufgabe um die Programmierung von Prozeduren und Funktionen gehen. Damit die Aufgabe etwas spannender wird, sollen Sie das Polynom jedoch mit dem **Hornerschema** auswerten. Es ist die in der Informatik gängige Methode, um ein Polynom auszuwerten.
Das Hornerschema ist eine spezielle Reihenfolge zur effizienten Auswertung eines Polynoms $p(x)$. Es ist durch die folgende Klammerungsreihenfolge gegeben:

$$p(x) = (\ldots((p_n \cdot x + p_{n-1}) \cdot x + \ldots + p_1) \cdot x + p_0$$

Aufgabe:

a) Ändern Sie das Programm von Übung 4, Aufgabe 3 so ab, dass

- die Auswertung des Polynoms in einer Funktion `double eval(...)` erfolgt. Die Parameter der Funktion bestimmen Sie selbst.
- die Auswerung des Polynoms mit Hilfe des Hornerschenas erfolgt.

b) Erweitern Sie Ihr Programm um eine Prozedure

```
void polyplot(double xmin, double xmax, double p[], int grad)
```

die das Polynom `p` vom Grad `grad` für das Intervall `(xmin,xmax)` auf dem Bildschirm zeichnet. Verwenden Sie dabei die Funktion `eval` aus Aufgabe a). Orientieren Sie sich am Programm aus Übung 3, Aufgabe 2.

Aufgabe 3: Permutationen

Entwerfen Sie ein Programm, das für eine natürliche Zahl n alle Permutationen der Zahlen

$$0, 1, \ldots, n-1$$

auf dem Bildschirm ausgibt.

Ihre Vorgangsweise gliedern Sie in folgende Teilaufgaben:

a) **Aufgabe:** Überlegen Sie sich eine Vorgangsweise am Beispiel $n = 4$.

b) Sei $P_{0,\ldots,n-1}$ die Menge der Permutation der Zahlen $0, 1, \ldots, n-1$.

Aufgabe: Finden Sie eine rekursive Vorschrift, um $P_{0,\ldots,n-1}$ auf die Mengen $P_{1,\ldots,n-1}$, $P_{0,2,\ldots,n-1}$, …, $P_{0,\ldots,n-2}$ (jeweils 1 Element entfernt) zurückzuführen. Erweitere die Vorschrift um eine Abbruchbedingung. Wann ist man fertig?

c) Implementieren Sie Ihre Erkenntnisse aus den Aufgaben a) und b) in einem rekursiven `C++`-Programm.