Universität des Saarlandes
Fachbereich 6.1 Mathematik
Dr. R. Grzibovskis

Übungen zur Vorlesung Modellierung und Programmierung WS 2015–2016

Blatt 3

Aufgabe 1 (3+1+3+1 Punkte)

Eine Permutation $\sigma \in P_n$ ist eine bijektive Abbildung

$$\sigma : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}.$$

Da die Urbildmenge klar ist, identifiziert man oft die Permutation mit ihem Bild und schreibt kurz

$$\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n)).$$

(a) Implementieren Sie eine C-Funktion
`void lies_perm(int feld[], int n);`
die eine Permutation del Länge n einliest und in einem entsprechenden Feld ablegt. Dabei sollen zu große, zu kleine sowie bereits eingegebene Werte abgewiesen werden und eine neue Eingabeaufforderung nach sich ziehen.

Um doppelt eingegebene Werte zu verhindern, implementieren Sie eine Hilfsfunktion
`int ist_in_feld(int feld[], int zahl, int anzahl);`
die überprüft, ob sich der Wert von `zahl` unter den ersten `anzahl` Einträgen in `feld` befindet.

(b) Implementieren Sie eine Funktion, die eine Permutation am Bildschirm anzeigt.

(c) Eine *Inversion* ist ein Paar (i, j) von Indizes, für die gilt

$$i < j \text{ und } \sigma(i) > \sigma(j).$$

Es sei i_σ die Anzahl der Inversionen, die in der Permutation auftreten, dann definiert man das *Signum* von σ durch

$$\operatorname{sgn}(\sigma) = (-1)^{i_\sigma},$$

d.h. das Signum ist 1, wenn die Anzahl der Inversionen gerade ist und -1 andernfalls.

Implementieren Sie eine C-Funktion
`int perm_signum(int feld[], int n);`
die das Signum einer Permutation zurückliefert.

(d) Schreiben Sie eine Funktion
`void transpos(int feld[], int index1, int index2);`
die eine Transposition durchführt, d.h. die Einträge an den Stellen `index1` und `index2` miteinander vertauscht.

Testen Sie Ihre Funktionen in einem Hauptprogramm für $n = 4$, indem Sie eine Permutation einlesen, zur Kontrolle ausgeben, ihr Signum bestimmen und auch dieses ausgeben. Anschließend führen Sie an der Permutation eine Transposition durch, geben die dadurch entstandene Permutation aus, und bestimmen schließlich (mit Ausgabe) deren Signum.

Aufgabe 2 (3+2+2+3 Punkte)

Sei $f : [a, b] \mapsto \mathbb{R}$ eine stetige Funktion mit $f(a)f(b) < 0$. Dann existiert mindestens eine Nullstelle $x \in [a, b]$, d.h. $f(x) = 0$. Wir nehmen an, dass es nur eine Nullstelle gibt. Mit dem folgenden Algorithmus findet man die Nullstelle mit gewünschter Genauigkeit ε, d.h. man findet ein $\tilde{x}$, sodass die Nullstelle x liegt in $[\tilde{x} - \varepsilon, \tilde{x} + \varepsilon]$.

1. Setze $a_1 = a$, $b_1 = b$, $i = 1$.
2. Berechne $x_i = (b_i + a_i)/2$.
3. Wenn $f(a_i)f(x_i) > 0$, setze $a_{i+1} = x_i$, $b_{i+1} = b_i$. Sonst setze $b_{i+1} = x_i$, $a_{i+1} = a_i$.
4. Wenn $b_{i+1} - a_{i+1} < \varepsilon$ oder $f(x_i) = 0$, ist das Ergebnis $\tilde{x} = x_i$. Sonst setze $i := i + 1$ und wiederhole Schritte 2, 3 und 4.

(a) Implementieren Sie eine C-Funktion

```
double finde_nullstelle(double a, double b, double eps);
```

die für gegebene `a`, `b` eine approximative Nullstelle mit Genauigkeit `eps` für die Funktion

```
double func(double x);
```

zurückliefert.

(b) Testen Sie Ihre Funktion in einem Hauptprogramm für folgenden Funktionen und Intervallen

1. $f(x) = x^2 - 1$, $a = 0.5$, $b = 1.5$;
2. $f(x) = 2\sin(x) - 1$, $a = 0$, $b = \pi/2$;
3. $f(x) = e^{x^2} - 1000$, $a = 0$, $b = 3$.

wobei $\varepsilon = 10^{-10}$.

(c) Implementieren Sie eine Ausgabe für die Anzahl Iterationen, die der Algorithmus braucht, um die approximative Nullstelle zu finden. Wie variiert diese Anzahl für die Fälle 2 und 3 aus (b) mit Genauigkeiten 10^{-6}, 10^{-8}, 10^{-10}, 10^{-12}.

(d) Implementieren Sie eine C-Funktion

```
double finde_nullstelle_neu(double a, double b, double eps);
```

analog zu (a), wobei der Algorithmus die folgende Änderung hat: Die Formel in Schritt 2 wird durch

$$x_i = a_i - f(a_i)(b_i - a_i)/(f(b_i) - f(a_i))$$

ersetzt. Führen Sie Teil (c) mit dem neuen Algorithmus durch.

Abgabe der Lösungsvorschläge und Vorführung der praktischen Aufgabe vor dem 15.12.2015.