

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 6

Aufgabe 6.1. Schreiben Sie eine Funktion `int anagram(char* firstStr, char* secondStr)`, die überprüft, ob ein Wort das Anagramm eines zweiten Wortes ist. Ein Anagramm eines Wortes ist eine Buchstabenfolge, die allein durch Permutation der Buchstaben gebildet wird. Beispielsweise ist "Ampel" ein Anagramm von "Palme".

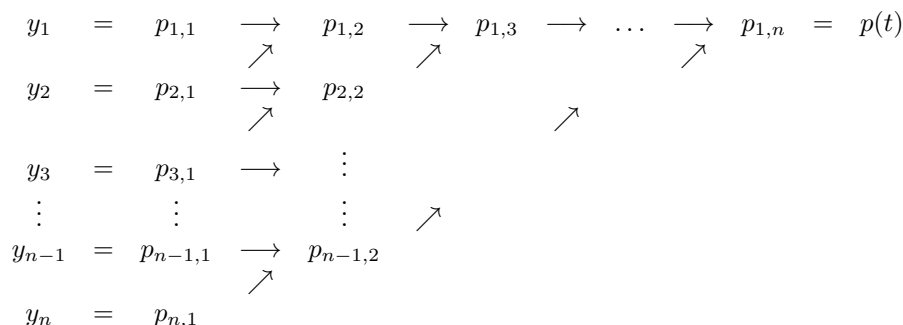
Schreiben Sie ferner ein aufrufendes Hauptprogramm, welches zwei Wörter einliest und überprüft, ob die Wörter Anagramme sind. Falls dies der Fall ist, soll von der Funktion der Wert 1 zurückgegeben werden, sonst der Wert 0. Wie haben Sie Ihren Code auf Korrektheit getestet? Speichern Sie den Source-Code unter `anagram.c` in das Verzeichnis `serie06`.

Aufgabe 6.2. Zu gegebenen reellen Stützstellen $x_1 < \dots < x_n$ und Funktionswerten $y_j \in \mathbb{R}$ garantiert die Lineare Algebra ein eindeutiges Polynom $p(t) = \sum_{j=1}^n a_j t^{j-1}$ vom Grad $n-1$ mit $p(x_j) = y_j$ für alle $j = 1, \dots, n$. Nun sei $t \in \mathbb{R}$ fixiert und $p(t)$ gesucht. Man kann $p(t)$ mit dem *Neville-Verfahren* berechnen, ohne zunächst den Koeffizientenvektor $a \in \mathbb{R}^n$ berechnen zu müssen: Dazu definiere man für $j, m \in \mathbb{N}$ mit $m \geq 2$ und $j + m \leq n + 1$ die Werte

$$p_{j,1} := y_j,$$

$$p_{j,m} := \frac{(t - x_j)p_{j+1,m-1} - (t - x_{j+m-1})p_{j,m-1}}{x_{j+m-1} - x_j}.$$

Es gilt dann $p(t) = p_{1,n}$. Schreiben Sie eine Funktion `neville`, die den Auswertungspunkt $t \in \mathbb{R}$ sowie die Vektoren $x, y \in \mathbb{R}^n$ übernimmt und $p(t)$ mittels Neville-Verfahren berechnet. Dazu berücksichtige man das folgende schematische Vorgehen



Der mathematische Beweis für diesen Algorithmus folgt in der Vorlesung zur Numerischen Mathematik. Zunächst schreibe man die Funktion so, dass die Matrix $(p_{j,m})_{j,m=1}^n$ vollständig aufgebaut wird. Speichern Sie den Source-Code unter `neville.c` in das Verzeichnis `serie06`. Sie können den Code testen, indem Sie für ein bekanntes Polynom p als Funktionswerte $y_j = p(x_j)$ wählen.

Aufgabe 6.3. Man kann das Neville-Verfahren aus Aufgabe 6.2 so programmieren, dass zur Speicherung der Werte *keine* Matrix $(p_{j,m})_{j,m=1}^n$ aufgebaut wird, sondern die gegebenen y_j -Werte geeignet überschrieben werden. Dadurch wird kein weiterer Speicher benötigt. Man realisiere dieses Vorgehen in einer Funktion `neville2`. Wie haben Sie Ihren Code auf Korrektheit getestet? Speichern Sie den Source-Code unter `neville2.c` in das Verzeichnis `serie06`.

Aufgabe 6.4. Schreiben Sie eine Bibliothek zur Verwaltung von *spaltenweise* gespeicherten $m \times n$ -Matrizen. Implementieren Sie die folgenden Funktionen

- `double* mallocmatrix(int m, int n)`
Allokieren von Speicher für eine spaltenweise gespeicherte $m \times n$ -Matrix.


```
    printPlayboard(playboard); // Gib die neue Spielsituation am Bildschirm aus.  
    player=changePlayer(player); // Wechsle den Spieler.  
}  
winner=getWinner(playboard);  
printWinnerMessage(winner);  
delPlayboard(playboard);
```

Die Funktion `makeMove` sollte dabei den Spieler solange nach einem Spielzug fragen, bis dieser gültig ist. (Felder dürfen nicht überschrieben werden!) Für die Funktion `getWinner` können Sie unter anderem Aufgabe 6.7 verwenden. Die Funktion `isGameFinished` kann wiederum die Funktion `getWinner` verwenden. Speichern Sie den Source-Code unter `tictactoe.c` in das Verzeichnis `serie06`.