1 Testataufgaben

1.1 Sudoku-Löser **(4 Punkte)**

Erweitern Sie Ihr Sudoku-Programm aus dem Übungsblatt 05 (oder die Musterlösung zu Übungsblatt 05 → StudIP) um eine rekursive Funktion zur Lösung eines Sudokus, die von dem Benutzer aufgerufen werden kann. Ein mögliches rekursives Lösungsverfahren kann wie folgt aussehen:

1. Ist ein freies Feld verfügbar, gehen Sie zu Schritt 2, sonst zu Schritt 3.
2. Setzen Sie im freien Feld nacheinander die Werte 1 bis 9 ein und rufe die Funktion rekursiv auf. Meldet ein rekursiver Funktionsaufruf „Lösung gefunden“, geben Sie diese Erfolgsmeldung sofort zurück.

   Meldet keiner der neun rekursiven Funktionsaufrufe „Lösung gefunden“, geben Sie das Feld wieder frei und „keine Lösung“ zurück.
3. Prüfen Sie, ob alle 81 Felder alle drei Regeln erfüllen. Falls erfüllt, geben Sie „Lösung gefunden“ zurück, sonst „keine Lösung“.

1.2 Die Mandelbrot-Menge **(6 Punkte)**

Eines der bekanntesten Fraktale ist die Mandelbrot-Menge[1]. Ihre Visualisierung in Abbildung 1 ist auch als “Apfelmännchen” bekannt.

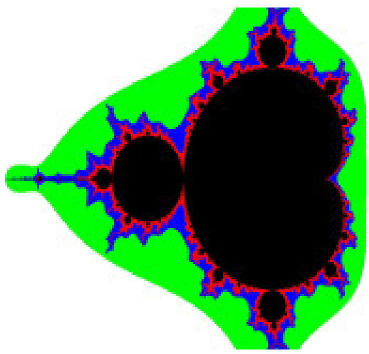

Abbildung 1: “Apfelmännchen”.

Die Mandelbrot-Menge $\mathbb{M}$ besteht aus allen komplexen Zahlen $c = x + iy \in \mathbb{C}$, für welche die rekursiv definierte Folge

$$z_{n+1} := z_n^2 + c, \quad \text{mit} \quad z_0 := 0 \tag{1}$$

beschränkt bleibt. Die grafische Darstellung der Mandelbrot-Menge erfolgt in der komplexen Zahlenebene. Dabei werden die Punkte der Menge schwarz und der Rest farbig dargestellt.
Wie findet man heraus, ob eine komplexe Zahl $c = x + iy$ in $\mathbb{M}$ liegt oder nicht?
Zuerst werden zu einer Zahl $c \in \mathbb{C}$ rekursiv die Folgenglieder z_n, $n \leq$ `n_max` bestimmt. Die Rekursion wird abgebrochen, wenn entweder die maximale Anzahl `n_max` rekursiver Aufrufe erreicht ist oder der Betrag von z_n größer oder gleich 2 ist. Wird die maximale Rekursionstiefe

[1] `https://de.wikipedia.org/wiki/Mandelbrot-Menge`

`n_max` erreicht, so nimmt man an, dass die Zahl c innerhalb der Mandelbrot-Menge liegt. Der entsprechende Punkt (x, y) zu c wird dann in der graphischen Darstellung schwarz eingefärbt. Wird die Rekursion vorzeitig wegen $|z_n| \geq 2$ abgebrochen, so bekommt der entsprechende Punkt eine Farbe, die der Iterationszahl entspricht.

1. Schreiben Sie eine rekursive Funktion, welche für die übergebenen Punkte $c \in \mathbb{C}$ und $z_0 = 0$ die Folgenglieder der komplexen Folge (1) bestimmt und die maximal erreichte Rekursionstiefe n zurückgibt. Als Abbruchbedingung soll die beschriebene Bedingung $|z_n| \geq 2$ oder $n >$ `n_max` dienen.

2. Nutzen Sie Funktionen aus dem Übungsblatt 06! Die Bildpunkte der komplexen Zahlenebene im Bereich $\{-2 \leq x \leq 1,\ -1 \leq iy \leq 1\}$ sollen auf ein BMP abgebildet werden. Ermitteln Sie für jeden Bildpunkt mittels der geschriebenen Funktion eine Rekursionstiefe n zu jedem Bildpunkt. Abhängig von n soll dann eine Farbe gewählt werden.

2 Präsenzaufgaben

2.1 Rekursion

1. Schreiben Sie eine rekursive Funktion `fakultaet`, welche die Fakultät einer ganzen Zahl $n \geq 0$ berechnet.

$$n! = \begin{cases} 1 & \text{, für} \quad n = 0, \\ n \cdot (n-1)! & \text{, sonst.} \end{cases}$$

2. Schreiben Sie eine rekursive Funktion `binom`, welche die Binomialkoeffizienten für zwei ganze Zahlen $n \geq 0$ und $k \geq 0$ berechnet.

$$\binom{n}{k} = \begin{cases} 1 & \text{, für } k = 0 \text{ oder } k >= n, \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{, sonst.} \end{cases}$$

Veranschaulichen Sie sich diese rekursive Definition am Pascal'schen Dreieck:

n													
0							1						
1						1		1					
2					1		2		1				
3				1		3		3		1			
4			1		4		6		4		1		
5		1		5		10		10		5		1	
6	1		6		15		20		15		6		1
...													

3. Schreiben Sie eine rekursive Funktion `fib`, welche zu einer ganzen Zahl $n \geq 0$ die n-te Fibonacci-Zahl berechnet.

$$fib(n) = \begin{cases} n & \text{, für} \quad n < 2, \\ fib(n-1) + fib(n-2) & \text{, sonst.} \end{cases}$$

4. Schreiben Sie eine rekursive Funktion `cheb`, welche für eine ganze Zahl $n \geq 0$ das n-te Chebyshev-Polynom an der Stelle $x \in \mathbb{R}$ auswertet.

$$cheb(n,x) = \begin{cases} 1 & \text{, für } \quad n = 0, \\ x & \text{, für } \quad n = 1, \\ 2x \cdot cheb(n-1,x) - cheb(n-2,x) & \text{, sonst.} \end{cases}$$

Überprüfen Sie ihre Funktion an den Beispielen

$$cheb(5,x) = 16x^5 - 20x^3 + 5x \qquad cheb(6,x) = 32x^6 - 48x^4 + 18x^2 - 1.$$

5. Schreiben Sie eine rekursive Funktion `ggt`, welche den größten gemeinsamen Teiler zweier natürlicher Zahlen a und b nach dem Euklidischen Algorithmus bestimmt.

$$ggt(a,b) = \begin{cases} a & \text{, für } \quad b = 0, \\ ggt(b, a \mod b) & \text{, sonst.} \end{cases}$$

6. Schreiben Sie eine rekursive Funktion `acker`, welche die Ackermannfunktion für zwei natürliche Zahlen $m \geq 0$ und $n \geq 0$ bestimmt.

$$acker(m,n) = \begin{cases} n+1 & \text{, falls } \quad m = 0, \\ acker(m-1,1) & \text{, falls } \quad m > 0 \text{ und } n = 0, \\ acker(m-1, acker(m, n-1)) & \text{, sonst.} \end{cases}$$

Beachten Sie bei der Wahl des Datentyps für den Rückgabeparameter, dass die Funktion sehr schnell große Werte annehmen kann.

7. Schreiben Sie zu einer der folgenden Rekursionsvorschriften $rek(n)$ für ganze Zahlen $n \geq 0$ eine gleichnamige Funktion und testen Sie ihre Funktion in einem Hauptprogramm mit nebenstehender Formel auf Richtigkeit.

a) $rek(n) = \begin{cases} 2 & \text{, für } \quad n = 0, \\ 2 - \frac{1}{rek(n-1)} & \text{, sonst.} \end{cases}$ Test: $rek(n) = \dfrac{n+2}{n+1}$

b) $rek(n) = \begin{cases} n & \text{, für } \quad n < 2, \\ \frac{rek(n-1)+rek(n-2)}{2} & \text{, sonst.} \end{cases}$ Test: $rek(n \geq 2) = \dfrac{2}{3}\left(1 - \dfrac{(-1)^n}{2^n}\right)$

c) $rek(n) = \begin{cases} 0 & \text{, für } \quad n = 0, \\ \frac{1+rek(n-1)}{3} & \text{, sonst.} \end{cases}$ Test: $rek(n \geq 1) = \dfrac{1}{2}\left(1 - \dfrac{1}{3^n}\right)$

d) $rek(n) = \begin{cases} 5 & \text{, für } \quad n = 0, \\ 3 + \frac{2}{7-rek(n-1)} & \text{, sonst.} \end{cases}$ Test: $rek(n \to \infty) = 5 - \sqrt{2}$

e) $rek(n) = \begin{cases} 1 & \text{, für } \quad n = 0, \\ 1 + \frac{1}{rek(n-1)} & \text{, sonst.} \end{cases}$ Test: $rek(n \to \infty) = \dfrac{1+\sqrt{5}}{2}$

f) $rek(n) = \begin{cases} 1 & \text{, für } \quad n = 0, \\ \frac{1+rek(n-1)^2}{2+rek(n-1)} & \text{, sonst.} \end{cases}$ Test: $rek(n \to \infty) = \dfrac{1}{2}$

8. Schreiben Sie ein Programm, das von der Tastatur einen Text beliebiger Länge einliest und diesen gespiegelt ausgibt. Dies soll mithilfe einer rekursiven Funktion `umdrehen` geschehen, die mit `getchar()` jeweils nur ein Zeichen `c` von der Tastatur einliest und testet, ob es sich um `ENTER` (ASCII-Code 10) handelt. Wenn `c == 10`, dann `return;`, sonst `c != 10`, dann rufe die Funktion `umdrehen` rekursiv für das nächste Zeichen auf und gebe `c` aus, sobald der rekursive Unteraufruf beendet ist.

   Beispiel: Die Eingabe `So’n Quatsch` wird zu `hcstauQ n’oS`.

2.2 char-Datentyp

1. Schreibt eine Funktion `reverse10`, die zehn Zeichen von der Tastatur in ein `char`-Array einliest und anschließend in umgekehrter Reihenfolge ausgibt.

   Beispiel: `Otto______` wird zu `______ottO`.

2. Schreiben Sie ein Programm, das ein Zeichen von der Tastatur einliest und den zugehörigen dezimalen ASCII-Wert ausgibt. Bestimmen Sie die Werte für die Eingaben „8“, „*“, „A“, „a“, „Z“ und „z“.

3. Schreiben Sie eine Funktion `UpperCase`, die Klein-Buchstaben in Großbuchstaben umwandelt. Die Funktion besitze einen `char`-Eingabeparameter, der den zu konvertierenden Buchstaben enthält. Das Funktionsergebnis soll wiederum vom Typ `char` sein und den entsprechenden Großbuchstaben zurückgeben.

4. Schreibt ein Programm, das die ASCII-Zeichen zu den ASCII-Werten zwischen 32 bis 127 übersichtlich ausgibt.

5. Die ASCII-Werte von 0 bis 31 sind für Steuerzeichen reserviert. Einige dieser sind mit ihrer C-Escape-Sequenz in nachfolgender Tabelle angegeben.

ASCII	**englische Bezeichnung**	**deutsche Bezeichnung**	**C-Escape-Sequenz**
000	null character	Ende eines Strings	`\0`
008	backspace	Rückschritt-Taste	`\b`
009	horizontal tabulation	horizontaler Tabulator	`\t`
010	line feed	Zeilenvorschub	`\n`
011	vertical tabulation	vertikaler Tabulator	`\v`

   Ändern Sie die Anweisung

   ```
   printf("\n\n Das gibt's doch gar nicht oder doch ?\n\n");
   ```

   ausschließlich durch Einfügen von oben angegebenen Steuerzeichen (auch keine Leerzeichen einfügen!) so ab, dass folgende Ausgabe erscheint:

   ```
   D      a      s gibt's
   do ga ni
            oder doch ?
   ```