Praktikum Objektorientierte Programmierung in C++ (WS 2018/2019)

Kettenbrüche/Continued Fraction

Ein **regulärer Kettenbruc**h für eine beliebige reelle Zahl x ist definiert als

$$x = f_0 + \cfrac{1}{f_1 + \cfrac{1}{f_2 + \cfrac{1}{f_3 + \ddots}}}, f_0 \in \mathrm{Z}, f_k \in \mathrm{N}, k > 0,$$

kurz geschrieben als $x = [f_0; f_1, f_2, f_3, \ldots]$.

Jede rationale Zahl kann eindeutig durch einen endlichen regulären Kettenbruch dargestellt werden, irrationale Zahlen durch einen unendlichen./
A **regular continued fraction** for any real valued number x is defined by the formula above and can be written in a short form with squared brackets $x = [f_0; f_1, f_2, f_3, \ldots]$.
Each rational number is uniquely represented by a finite regular continued fraction, each irrational number by an infinite one.

Beispiele/Examples:

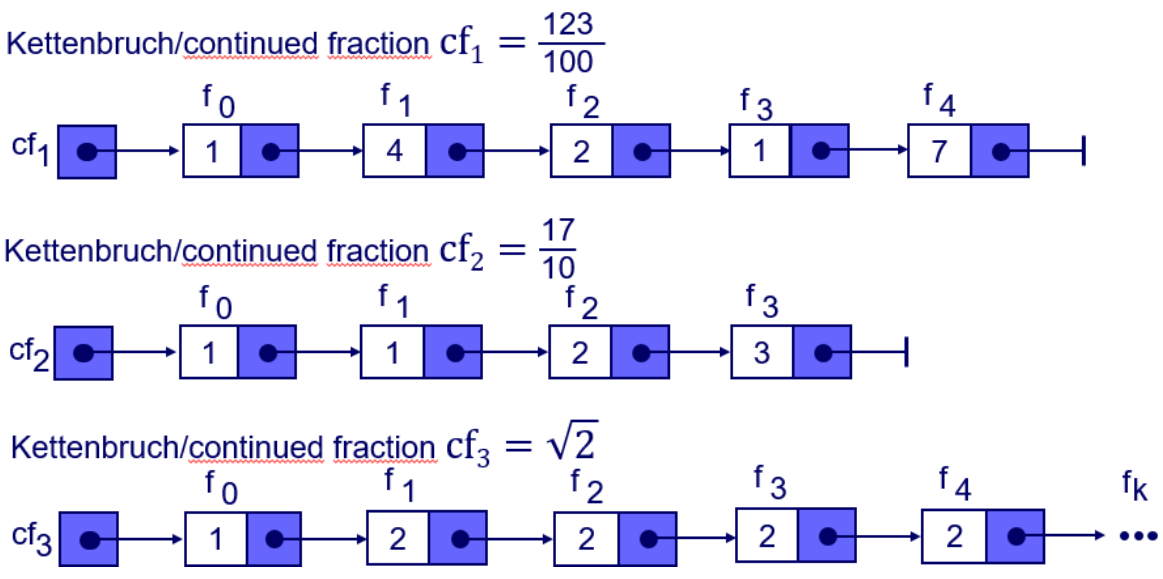
$$\frac{123}{100} = 1 + \cfrac{1}{4 + \cfrac{1}{2 + \cfrac{1}{1 + \frac{1}{7}}}} = 1.23, \text{ kurz/short } \frac{123}{100} = [1; 4, 2, 1, 7]$$

$$\frac{17}{10} = 1 + \cfrac{1}{1 + \cfrac{1}{2 + \frac{1}{3}}} = 1.7, \text{ kurz/short } \frac{17}{10} = [1; 1, 2, 3]$$

$$\sqrt{2} = 1 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \ddots}}} = 1.4142135\ldots, \text{ kurz/short } \sqrt{2} = [1; 2, 2, 2, 2, \ldots] = [1; \overline{2}]$$

Aufgabe ist, diese Kurzformen für reguläre Kettenbrüche im Computer durch Listen ganzer Zahlen zu repräsentieren, diese einzugeben, ausuzrechnen und wieder zu löschen; es sollen dabei **keine globalen Variablen** verwendet werden./
Task is to represent this short form for continued fractions inside a computer by lists of integer numbers, to input, calculate and also delete them; **no global variables** shall be defined and used.

Beispiele/Examples:



Aufgabe 1/Task 1

Definieren Sie eine Struktur für obige Listenelemente zur Implementierung von Kettenbrüchen als Listen mit diesem Elementen./
Define a structure for above list elements to implement continued fraction by lists with these elements.

Aufgabe 2/Task 2

Definieren Sie eine Funktion mit einem Kettenbruch $cf = [f_0;f_1,f_2,f_3, \ldots,f_k]$ als Zeiger/Liste im ersten Parameter und einer natürlichen/ganzen Zahl f_{k+1} als zweitem Parameter, die den Kettenbruch entsprechend erweitert zu $cf = [f_0;f_1,f_2,f_3, \ldots,f_k,f_{k+1}]$, also an die Liste im ersten Parameter ein weiteres Element mit Wert f_{k+1} hinten anhängt. Geben Sie den erweiterten Kettenbruch/die erweiterte Liste als Funktionswert zurück!/
Define a function with a continued fraction $cf = [f_0;f_1,f_2,f_3, \ldots,f_k]$ as pointer/list as first and a natural/integer number f_{k+1} as second parameter extending the continuous fraction to $cf = [f_0;f_1,f_2,f_3, \ldots,f_k,f_{k+1}]$, i.e. appending a further element with value f_{k+1} at the end of the list. Return the extended continued fraction/the resulting list as function value.

Aufgabe 3/Task 3

Definieren Sie eine Funktion, die einen als Zeiger-Parameter übergebenen Kettenbruch komplettt löscht, also den Speicherplatz aller Elemente der Liste auf dem Heap wieder frei gibt. Geben Sie wie in den Beispielen unten alle gelöschten Werte f_k zur Kontrolle aus./
Define a function with a continued fraction as pointer parameter deleting the complete list, freeing the memory of all list elements on the heap. Output like in the examples below all deleted values f_k to check your function.

Aufgabe 4/Task 4

Definieren Sie eine Funktion, die den Wert eines als Zeiger-Parameter übergebenen Kettenbruchs als Gleitpunktzahl vom Typ **`double`** berechnet und zurück liefert.
Hinweis: besonders einfach und kurz ist eine rekursive Lösung./
Define a function with a continued fraction as pointer parameter returning its value as a **`double`** coded floating point value.
Hint: a recursive solution is especially appropriate and short in this case.

Aufgabe 5/Task 5

Schreiben Sie eine Funktion ohne Parameter, in der Sie zuerst $f_0 \in \mathrm{Z}$ einlesen und danach in einer Schleife die Werte $f_k \in \mathrm{N}$, $k = 1, 2, \ldots$, also den Kettenbruch dabei sukzessive als Liste aufbauen und zurück geben./
Write a function without parameter, first reading $f_0 \in \mathrm{Z}$, then in a loop the values $f_k \in \mathrm{N}$, $k = 1, 2, \ldots$, successively constructing the continued fraction as list and return it.

Aufgabe 6/Task 6

Schreiben Sie eine **`main`**-Funktion, in der Sie nacheinander drei Zeigervariable für einen Kettenbruch definieren. Initialisieren Sie die erste Variable über geeignet verschachtelte Funktionsaufrufe direkt im Kode mit dem Kettenbruch für das obige Beispiel $cf_1 = \frac{123}{100} = [1; 4, 2, 1, 7]$ und die zweite mit $cf_2 = \frac{17}{10} = [1; 1, 2, 3]$. Lesen Sie den dritten Kettenbruch cf_3 über einen Funktionsaufruf der obigen Funktion aus Aufgabe 5 ein.
Geben Sie jeweils die über Funktionsaufrufe (Aufgabe 4) berechneten Gleitpunktzahlen für die Werte der drei Kettenbrüche mit mindestens 15 Nachkommastellen aus und löschen anschließend die Kettenbrüche/Listen komplett auf dem Heap ebenfalls über entsprechende Funktionsaufrufe (Aufgabe 3; siehe Beispiele unten).
Testen Sie Ihr Programm mindestens für die Eingaben $1 = [1]$, $2 = [1; 1]$, $0.5 = [0; 2]$, $\sqrt{2} \approx [1; 2, 2, 2, 2, \ldots]$ und $\pi \approx [3; 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, \ldots]$, weitere Beispiel finden Sie hier./
Write a **`main`** function and define three pointer variables for continued fractions. Initialise the first above example $cf_1 = \frac{123}{100} = [1; 4, 2, 1, 7]$ and the second $cf_2 = \frac{17}{10} = [1; 1, 2, 3]$ by appropriate nested functions calls directly in code. Input the third continued fraction cf_3 by giving a function call to above defined function (task 5).
For all three continued fractions output its floating point values calculated by respectively given function calls (task 4) with at least 15 digits after the decimal point and then delete the memory in heap for the continued fractions/lists also by appropriate function calls (task 3; see also examples below).
Test your program at least for inputs $1 = [1]$, $2 = [1; 1]$, $0.5 = [0; 2]$, $\sqrt{2} \approx [1; 2, 2, 2, 2, \ldots]$ and $\pi \approx [3; 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, \ldots]$, find further examples here.

Wichtig zu beachten/Important to Regard

Verwenden Sie ausschließlich Ein- und Ausgaben über C++, **keine Aufrufe** von **`scanf`**, **`printf`** oder von **`malloc`**, **`calloc`**, ... oder **`free`**! Ihre Quellkode-Datei muss **`h2_<IhreMatrikelnummer>.cpp`** heissen, die Endung muss für den Plagiatschecker sowie für unsere Prüfprogramme **`.cpp`** sein, darf also **keine Textdatei** mit Endung **`.txt`** sein, **keine Projektdatei** **`.cbp`** oder ähnlich und auch **keine `.rar`, `.zip`-, ... Datei** sein (kann der Plagiatschecker nicht verarbeiten, Sie würden also 0 Punkte bekommen), Ihr Programm darf auch **nur ASCII-Zeichen** enthalten, **nur die Standard-C++-Bibliotheken** einbinden (also keine mit Endung **`.h`** wie **`conio.h`**, **`stdio.h`**, **`windows.h`**, ... oder solche mit Endung **`.hpp`**) und soll dem Standard **C++11** (oder neuer) folgen./
Only use C++ input and output, **no calls** of **`scanf`** or **`printf`** function or **`malloc`**, **`calloc`**, ... or **`free`**! Your source code file has to have name **`h2_<your matriculation number>.cpp`**, the ending **`.cpp`** is essential for the plagiarism checker as well as for our check programs, **`no text file`** with ending **`.txt`**, **no project file** **`.cbp`** or similar and **no `.rar`, `.zip`, ... file** (the plagiarism checker does not understand all these formats, therefore you would get 0 points for it), also your program is **only allowed containing ASCII characters**, **only includes of standard C++ libraries** (i.e. no libraries with ending **`.h`** like **`conio.h`**, **`stdio.h`**, **`windows.h`**, ... or **`.hpp`**) and shall follow standard **C++11** (or newer).

Beispiele Programmläufe/Examples Program Runs

```
cf1 = 123/100 = 1.23
delete: 1 4 2 1 7
cf2 = 17/10 = 1.7
delete: 1 1 2 3
continued fraction, enter first value f_0: 1
enter next part denominator f_1 (<= 0 for end): 2
enter next part denominator f_2 (<= 0 for end): 2
enter next part denominator f_3 (<= 0 for end): 2
enter next part denominator f_4 (<= 0 for end): 2
enter next part denominator f_5 (<= 0 for end): 2
enter next part denominator f_6 (<= 0 for end): 2
enter next part denominator f_7 (<= 0 for end): 2
enter next part denominator f_8 (<= 0 for end): 2
enter next part denominator f_9 (<= 0 for end): 2
enter next part denominator f_10 (<= 0 for end): 0
value inputted continued fraction cf3 = 1.41421362489487
delete: 1 2 2 2 2 2 2 2 2 2

cf1 = 123/100 = 1.23
delete: 1 4 2 1 7
cf2 = 17/10 = 1.7
delete: 1 1 2 3
continued fraction, enter first value f_0: 3
enter next part denominator f_1 (<= 0 for end): 7
enter next part denominator f_2 (<= 0 for end): 15
enter next part denominator f_3 (<= 0 for end): 1
enter next part denominator f_4 (<= 0 for end): 292
enter next part denominator f_5 (<= 0 for end): 1
enter next part denominator f_6 (<= 0 for end): 1
enter next part denominator f_7 (<= 0 for end): 1
enter next part denominator f_8 (<= 0 for end): 2
enter next part denominator f_9 (<= 0 for end): 1
enter next part denominator f_10 (<= 0 for end): 3
enter next part denominator f_11 (<= 0 for end): 1
enter next part denominator f_12 (<= 0 for end): 0
value inputted continued fraction cf3 = 3.141592653589815
delete: 3 7 15 1 292 1 1 1 2 1 3 1
```

Zuletzt geändert: Montag, 29. Oktober 2018, 18:12

◄ P3 Kettenbruch (Wikipedia; also read the english version or the one in your mother language) ►