Programmieren für Physiker

Interfakultatives Institut für Anwendungen der Informatik
Institut für Theoretische Teilchenphysik

Prof. Dr. M. Steinhauser, Dr. A. Mildenberger
`http://comp.physik.kit.edu`

SS 2017 – Blatt 11
Bearbeitungszeitraum: bis 12. Juli 2017

Klausurtermin: Dienstag, 25. Juli 2017, 17.30 Uhr.
Es ist erforderlich, sich bis zum 18.07.2017 im Studierendenportal zur Klausur anzumelden.
Erster Buchstabe Ihres Nachnamens: A-K $\widehat{=}$ Gottlieb-Daimler-HS (HMO) (10.21),
L-Z $\widehat{=}$ Carl-Benz-HS (HMU) (10.21).
Dauer: 90 Minuten. Es sind keine Hilfsmittel erlaubt. Bitte Studierendenausweis mitbringen.

Aufgabe 28: Nullstellenbestimmung — Pflichtaufgabe

Eine Methode, um Nullstellen einer (stetigen) Funktion numerisch zu bestimmen, ist das Bisektionsverfahren. Bei dem Verfahren startet man mit einem Intervall $[x_0, x_1]$, das eine Nullstelle der gegebenen Funktion $f(x)$ einschließen soll. Lassen Sie also die Werte x_0 und x_1 eingeben und überprüfen Sie anhand der Vorzeichen von $f(x_0)$ und $f(x_1)$, ob eine Nullstelle im Intervall vorhanden ist.
Als neue Teststützstelle x_{test} wird nun die Mitte des Intervalls herangezogen $x_{\text{test}} = (x_0 + x_1)/2$. An der neu gewonnen Stelle ist die Funktion auszuwerten und abhängig vom Vorzeichen wird entweder die obere oder die untere Intervallgrenze neu gesetzt, so dass die Nullstelle stets innerhalb des betrachteten Intervalls bleibt. Wiederholen Sie diese Schritte, bis entweder die Intervallgröße kleiner als $\varepsilon_x = 10^{-4}$ oder der Funktionswert $|f(x_{\text{test}})| < \varepsilon_f = 10^{-3}$ wird.
Ihr Programm soll folgendermaßen strukturiert sein: Definieren Sie für die zu untersuchende Funktion $f(x)$ eine C++-Funktion. Schreiben Sie für die Nullstellensuche eine Funktion, die als Übergabewerte die zu untersuchende Funktion per Funktionszeiger erhält und als weitere Argumente das Anfangsintervall und die gewünschten Genauigkeitswerte. Die Nullstellensuche soll ausschließlich die im Argument übergebene Funktion verwenden. Der Rückgabewert soll die ermittelte Nullstelle sein. Geben Sie während der Iteration bei jedem Schritt die Schrittzahl, das Nullstellen-Intervall und den Funktionswert $f(x_{\text{test}})$ aus.

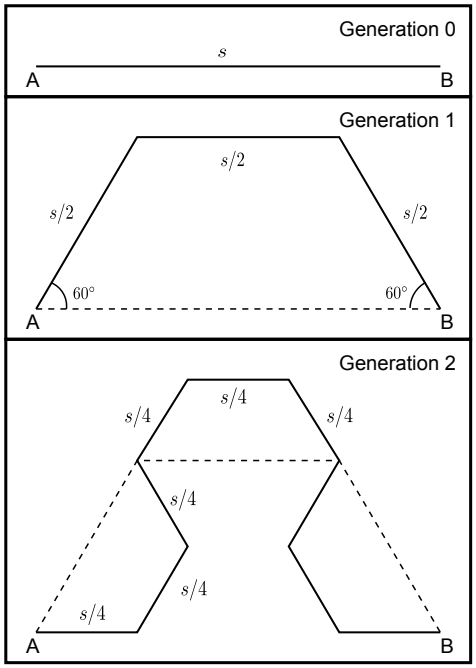
Testfunktionen: (i) $f(x) = \cos(x) - x$, $x_0 = -10$, $x_1 = 10$.
(ii) $f(x) = e^x - x^3$, $x_0 = 2$, $x_1 = 10$ (`double`-Arithmetik empfohlen).

Aufgabe 29: Sierpiński-Dreieckskurve — freiwilliges Zusatztestat[1]

In dieser Aufgabe soll mit C++ eine grafische Ausgabe der sogenannten Sierpiński-Dreieckskurve programmiert werden. Auf der WWW-Seite zur Vorlesung finden Sie eine Anleitung zur Verwendung der Grafikbibliothek `CImg` (`http://cimg.sourceforge.net/`) unter Linux und Windows.

[1] Sie können bei dieser freiwilligen Aufgabe ein Testat erhalten, welches gewertet wird; die Aufgabe zählt aber nicht als Pflichtaufgabe.

Die gesuchte Kurve ist ein fraktales Objekt, das durch folgende Vorschrift erzeugt wird: Man beginnt mit einer Strecke $\overline{AB}$, wie rechts im obersten Bild dargestellt.
In der nächsten Generation wird die Strecke durch einen Streckenzug aus drei Strecken der halben Länge ersetzt, dabei werden mit der Urpsrungsstrecke 60°-Winkel gebildet. Dies ist im mittleren Teilbild zu sehen, zur Illustration ist auch die Ursprungsstrecke gestrichelt eingezeichnet.
Allgemein werden beim Schritt von Generation $n-1$ zu Generation n alle Teilstrecken jeweils durch drei Teilstrecken mit 60° Grad-Abweichung ersetzt. Dabei ist allerdings zu beachten, dass diese 60°-Winkel abwechselnd von der Ausgangsstrecke nach links oder rechts vom bisherigen Streckenzug weisen und diese Abweichung auch alternierend mit dem Generationsindex beginnt. Im unteren Teilbild ist dies für die 2. Generation dargestellt.



Schreiben Sie ein Programm, dass für ein einzugebendes $0 \leq n \leq 12$ den Streckenzug dieser Generation grafisch auf dem Bildschirm ausgibt.

Aufgabe 30: Dekonstruktivismus **freiwillig**

Was gibt das untenstehende Programm auf dem Bildschirm aus und warum?

```
#include <iostream>
using namespace std ;

class sinnfrei {
private:
    int aha ;
public:
    sinnfrei (int a)  { aha = a ; }
    ~sinnfrei()
        {   if (aha>0)
            {  sinnfrei dummy(aha-1) ;
            }
            cout << aha << endl ;
        }
} ;

int main()
{   int a ;
    cout << "Bitte 'ne positive ganze Zahl: " ;
    cin >> a ;
    sinnfrei dummy(a) ;
}
```
