Abteilung für
Angewandte Mathematik

SS 2024
Ludwig Striet

Einführung in die Programmierung für Studierende der Naturwissenschaften

Blatt 4 – 13.05.2024

Abgabe: *Bis 26.05.2024, 23:59 Uhr per E-Mail an Ihre/n Tutor/in*

Aufgabe 1 (5 Punkte)**.** Welche der folgenden Ausdrücke werden vom Compiler übersetzt und bei welchen gibt es Kompilierfehler? Welche erzeugen Fehler bei der Laufzeit und welche sind Korrekt? Beschreiben Sie kurz das Verhalten!

```
int a;
&a = 3;
```

```
int* a;
*a = 17;
```

```
double* c = new double[3];
c[0] = 0.0; c[1] = 1.0; c[2] = 2.0;
```

```
int a = 5;
int* b = &a;
(*b)++;
```

```
int a = 5;
int& b = 7;
delete a;
```

Aufgabe 2 (2 Punkte)**.** Schreiben Sie ein Programm, das eine Zahl `n` vom Terminal einliest. Anschließend soll ein Array mit `n` Gleitkommazahlen angelegt und `n` Zahlen von der Konsole eingelesen und im Array gespeichert werden. Danach soll das Programm den Durchschnitt, das Minimum und das Maximum der Zahlen ausrechnen, im Terminal ausgeben und das Array wieder löschen.

Aufgabe 3 (3 + 2 Punkte)**.** *(a)* Implementieren Sie eine Funktion `double sqrt(double x)`, in der Sie mit dem Heron-Verfahren die Wurzel einer nicht-negativen Zahl `x` approximieren.

(b) Implementieren Sie eine Funktion `double sqrt_abs(double x, bool* is_imaginary)`, in der Sie die Wurzel des *Betrags* einer Zahl `x` berechnen. Falls `is_imaginary` nicht `NULL` ist soll darin gespeichert werden, ob die Wurzel eine imaginäre Zahl ist, also ob `x` negativ ist. Sie können für diese Funktion Ihre Implementierung der `sqrt(...)` Funktion aus (a) verwenden oder die `std::sqrt` Funktion aus der Standardbibliothek.

Aufgabe 4 ((3 + 5) Punkte)**.** ,
(a) Implementieren Sie eine Funktion `forward_diff_quot` die als Argumente Gleitkommazahlen `x`, `h` sowie eine Funktion `f` (als Funktionenpointer) übergeben bekommt und die nach der Formel

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

angenäherte Ableitung von `f` an der Stelle `x` als Gleitkommazahl zurück gibt. Dabei ist h eine kleine positive Zahl, z.B. 10^{-10}.
Hinweis: Sie finden auf der Vorlesungswebseite das Programm `integral.cc` in welchem ein Beispiel für die Verwendung eines Funktionenpointers angegeben ist.

(b) Das Newton-Verfahren zum finden einer Nullstelle x^* einer gegebenen Funktion f geht so:
Eingabe: Reelle Zahl x_0 und Toleranz $\delta > 0$.

(1) Setze $x = x_0$
(2) Setze $x_{alt} = x_0$,
(3) Wenn $|f(x_{alt})| < \delta$: Stoppe
(4) Setze $x_{neu} = x_{alt} - f(x_{alt})/f'(x_{alt})$.
(5) Setze $x_{alt} = x_{neu}$
(6) Gehe zu Schritt (1).

Ausgabe: Approximation x_k von x^*.
Implementieren Sie das Verfahren und testen Sie es beispielsweise an einer quadratischen Funktion wie z.B. $f(x) = (x-2) * (x+3)$. Wählen Sie z.B. $\delta = 10^{-12}$ und $h = 10^{-10}$.
Hinweis: Sie dürfen das Verfahren mit einem `x0` in der Nähe der tatsächlichen Nullstellen initialisieren. Sie müssen nicht überprüfen, ob überhaupt Nullstellen existieren. Zum Berechnen der Ableitung können Sie die Approximation aus Teil *(a)* verwenden. Den Betrag $|f(\cdot)|$ können Sie ausrechnen oder `cmath` einbinden und die Funktion `std::abs` verwenden.
In Aufgabe (a) und (b) ist f eine gegebene Funktion, die als `C++`-Funktion definiert wird.