Programmierung, Algorithmen, Datenstrukturen 2

a.o. Univ.-Prof. Dr. Martin Welk — UMIT
Dipl.-Ing. Elias Tappeiner — SS 2022
Institut für Biomedizinische Bildanalyse — Mech-BSc/ET-BSc

Hausübung 2

Abgabe: 4. April 2022, 23.55 Uhr in Moodle

Es wird empfohlen, die Übungen in Gruppen (max. 3 Studierende) zu bearbeiten und gemeinsam abzugeben.

Abzugeben ist von diesem Blatt die Aufgabe 2.1, die übrigen Aufgaben sind zur selbstständigen Bearbeitung sowie zur Vorbereitung der abgabepflichtigen Aufgaben des nächsten Blattes.

Hinweise zur Abgabe

- Gruppenarbeiten werden von **einer/m** Studierenden in Moodle eingereicht; die Namen **aller** Beteiligten sind bei der Abgabe zu vermerken! (Empfehlung: Kommentar im Quelltextfile)

Aufgabe 2.1 *(ohne Abgabe)*

Berechnen Sie das Produkt $141151 \cdot 34567$ mit der Karatsuba-Methode aus der Vorlesung (T2.7). Geben Sie dazu alle Zwischenschritte an. Für die Multiplikation bis zu dreistelliger Zahlen darf der Taschenrechner verwendet werden.

Hinweis: Achten Sie darauf, dass der Algorithmus gleiche Länge der Faktoren voraussetzt!

Aufgabe 2.2 ***(Abgabe)***

In Moodle finden Sie ein Quellcodearchiv, in dem ein C++-Programmrahmen für die Implementation der Ganzzahlarithmetik-Algorithmen aus Kapitel T2 der Vorlesung vorbereitet ist. Dabei ist

- `main.cpp` ein Hauptprogramm, das von der Kommandozeile die Abkürzung einer Operation (`a` für Addition, `m` für (Volksschul-) Multiplikation, `k` für Karatsuba-Multiplikation) gefolgt von zwei ganzzahligen Operanden einliest;

- `int_basic.h|.cpp` ein Modul mit einem Datentyp (`Digit`) und elementaren Operationen (`elem_add3` für Volladdierer, `elem_mult` für elementare Multiplikation) sowie Ein- und Ausgabe-Hilfsroutinen
- `int_algos.h|.cpp` ein Modul mit den zusammengesetzten Algorithmen für die Addition und Multiplikation ganzer Zahlen.

Zahlen werden in diesem Programm durchgängig als `vector <Digit>` dargestellt, mit der Vereinbarung, dass die niederwertigste (!) Ziffer im `vector`-Element mit Index 0 steht und die höchstwertige Ziffer im letzten `vector`-Element. Das entspricht der Nummerierung in den Algorithmen auf den Vorlesungsfolien aus Block 1/Kapitel T2.

Bitte lassen Sie die Module `main` und `int_basic` unverändert und ergänzen Sie die zwei Funktionen in `int_algos.cpp`, in denen noch Code fehlt (suchen Sie nach der Zeichenkette `HIER ERGAENZEN`).

(a) In der Funktion `additionc` müssen Sie den Algorithmus zur Addition aus Block 1, Folie 29 vervollständigen.

Hierbei soll die elementare Operation (Volladdierer) ausschließlich durch Aufruf der Funktion `elem_add3` (siehe Vorgabe) realisiert werden.

Achten Sie darauf, welche Teile schon dastehen! – Abweichend vom Algorithmus auf der Folie wird hier u_0 nicht mit 0, sondern mit einem Funktionsparameter `carry` initialisiert; damit kann anschließend diese Funktion für Addition (Aufruf aus der Funktion `addition` weiter unten) und Subtraktion (Aufruf aus der Funktion `subtraktion` weiter unten, die zuvor den Subtrahenden in sein Neunerkomplement umwandelt und dann noch einen Übertrag 1 mit in die Addition einspeist) verwendet werden.

Nach Ergänzung dieses Teils sollte Addition und Multiplikation korrekt funktionieren, Aufruf z. B.

```
test a 114115116117118119 234567891234
```

für Addition (wenn `test` der Programmname Ihres ausführbaren Programms ist) mit dem Ergebnis

```
0114115350685009353
```

sowie

```
test m 114115116 234567
```

für Volksschulmultiplikation mit dem Ergebnis

```
000026767640414772
```

Lesen Sie auch die (bereits fertig implementierten) Funktionen `multn1` (Algorithmus von Block T21, Folie 33) und `multiplikation` (Algorithmus in Anlehnung an Block T21, Folie 34) aufmerksam durch, um die Umsetzung nachzuvollziehen.

(b) In der Funktion `karatsuba` müssen Sie den Algorithmus zur Karatsuba-Multiplikation aus Block T21, Folie 45 vervollständigen.

Hierbei sollen die noch fehlenden Ziffernberechnungen ausschließlich durch Aufruf der Funktionen `addition` (Addition zweier mehrstelliger Zahlen), `subtraktion` (Subtraktion zweier mehrstelliger Zahlen), `shiftleft` (Linksverschiebung, siehe im Programmtext) und rekursive Aufrufe der Funktion `karatsuba` realisiert werden.

Achten Sie darauf, welche Teile schon dastehen: Der Anfangsfall $n \leq 3$ und die Zerlegung der Ziffernfolgen der Faktoren a und b in a', a'' bzw. b', b'' sind bereits implementiert!

Wenn dieser Algorithmus vervollständigt ist, muss auch die Karatsuba-Multiplikation korrekt funktionieren, z. B.

```
test k 114115116 234567
000026767640414772
```

Aufgabe 2.3 *(Abgabe)*

Erstellen Sie ein C++-Programm zur Berechnung von Nullstellen einer Funktion f mittels des Newton-Verfahrens.

Die Funktion f sowie ihre Ableitung f' sollen dabei in einem eigenen Modul `thefunction` als `f()` und `df()` implementiert sein (Beispielmodul in Moodle bereit gestellt), das von dem Modul mit dem Newton-Verfahren aufgerufen wird.

Nach Eingabe von Startnäherung x_0 und Abbruchschranke ε sollen fortlaufend die Werte $x_1, x_2, \ldots$ gemäß

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

berechnet werden, bis $|x_{k+1} - x_k| < \varepsilon$ gilt. Bei $f'(x_k) = 0$ sollte vorzeitig abgebrochen werden, ebenso wenn nach 100 Iterationen die Abbruchbedingung nicht erfüllt ist.