Technische Universität München
Fakultät für Informatik
Rechnerarchitektur-Praktikum
SS 2018

ASSEMBLER POLYNOME

Anwenderdokumentation

Bearbeitet von:
Oleg Patrascu
Matthias Unterfrauner

07.07.2018

INHALT

1. Aufgabe des Programms

Im Rahmen des Rechnerarchitektur Praktikums an der Technischen Universität München,
wurde ein Programm zu Berechnung der Polynomkoeffizienten und Normierungskonstante erstellt.

Laut unserer Formel ein Polynom kann auf zwei Arten dargestellt werden :

$$f(x) = \sum_{i=0}^{N} a_i x^i = c(1 * x^N + \sum_{i=0}^{N-1} b_i x^i)$$

Bsp:

Als **Eingabe** kriegt unser Programm ein Polynom:

*f(x) = 1 + 4*x + 2x^2*

Als **Ausgabe** bekommen wir wieder ein Polynom mit einer Normierungskonstante zurück:

*f(x) = C * (0.5 + 2*x + x^2), mit C = 2, die Normierungskonstante*

2. Systemanforderungen

Das Programm wurde auf 64-Bit Linux Machine getestet aber mit einer 32-Bit Machine wird auch funktionieren weil wir ein 64-Bit kompiliertes Programm zwingen.

Unseres Programm hat folgende Abhängigkeiten:

```
nasm (version >= 2.13.02)
gcc    (version >= 7.3.0)
make
```

Auf **Ubuntu** man kann mit dem Befehl **sudo apt-get install** die benötigte Abhängigkeiten installieren.

3. Erzeugen und Testen des Programms

Zuerst betrachten wir die Projektstruktur, um ins richtige Verzeichnis zu navigieren:

```
readme

root
    makefile

    headers
      polynom.h
      utils.h
    src
      main.c
      norm.asm
      utils.c
    test
      test.c
      polynom.txt
```

Für uns, der Anwender wäre interessant nur die **readme** Datei, dort steht die Beschreibung jeder Datei des Projekts aber technische Kenntnisse sind notwendig.

Zuerst, müssen wir in **root** Verzeichnis des Projekts navigieren, dafür kann man den Befehl **cd** sowohl auf Linux als auch auf Windows verwenden, nachdem wir zum **root** Verzeichnis navigiert haben, können wir unseren richtigen Pfad mit dem Befehl **pwd** (oder auf Windows einfach **cd**) bestätigen.

Es sind jetzt 3 make Ziele möglich: **all**, **test** und **clean**.

Mit dem Befehl

```
make clean
```

werden alle erzeugte Binär- und Objektdateien gelöscht.

Mit dem Kommando

```
make all
```

wird aus dem Quellcode ein ausführbares Programm an dem Pfad `./main.out` erstellt. Im nachfolgenden ist die zu erwartende Ausgabe bei einer erfolgreichen Kompilierung zu finden.

Mit dem Befehl `./main.out` führt man das Programm aus, das für die Berechnung der Polynome notwendig ist.

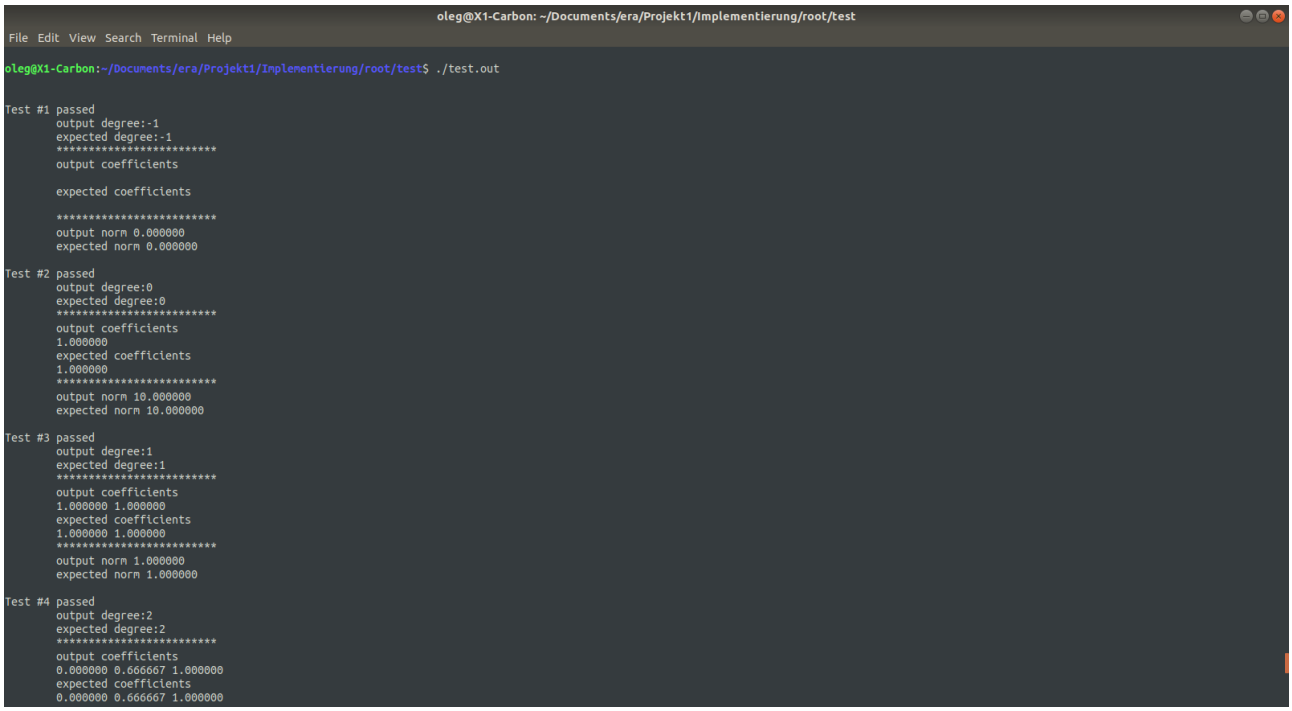
Mit dem Befehl

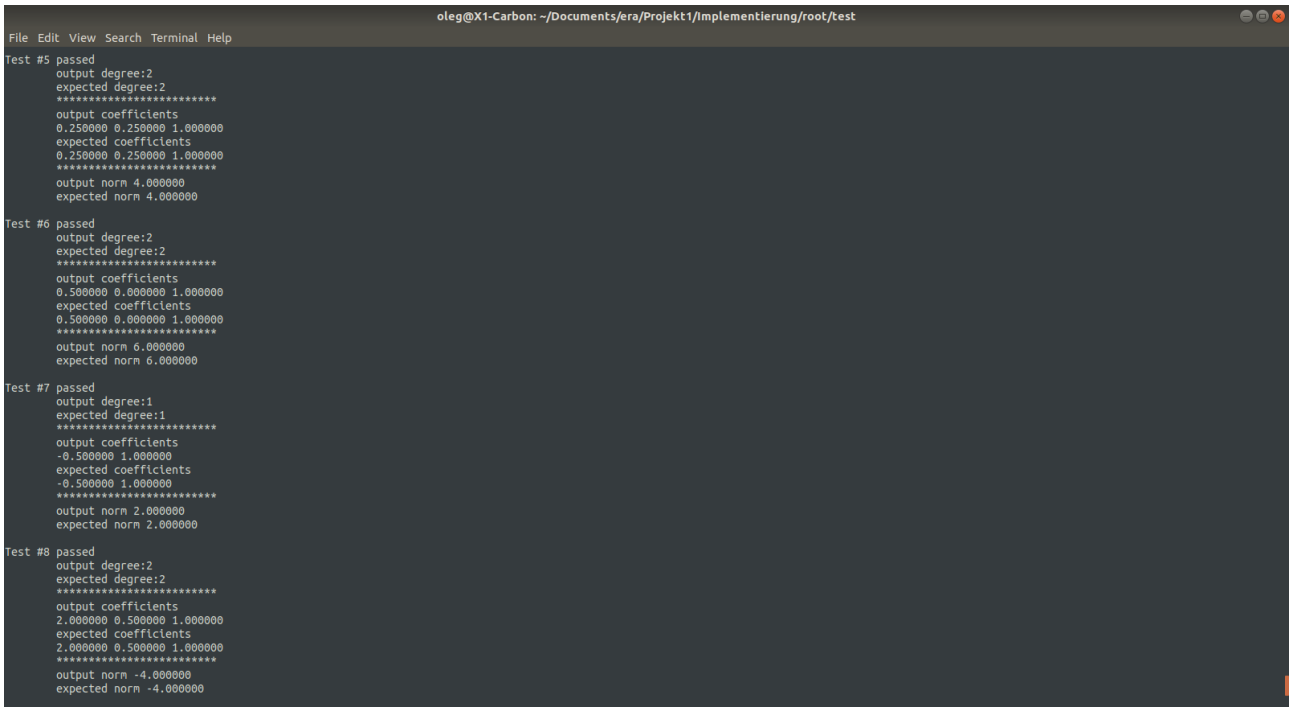
```
make test
```

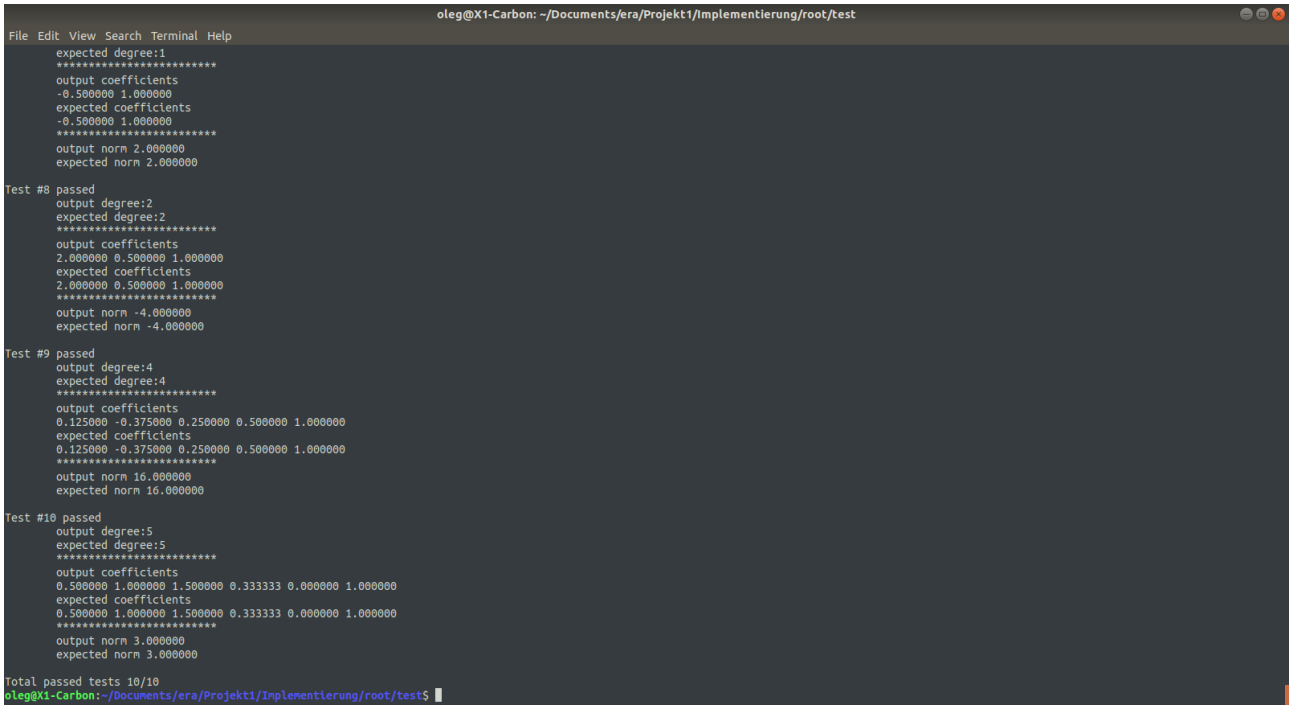
wird dem Anwender ermöglicht, einen ausführlichen Testlauf zu betrachten.
Die ausführbare Datei findet man in `test/test.out`

Die Testfälle dienen als einen guten Einblick für was man eingeben kann.
Sie beinhalten auch Fälle wie negative Grade der Polynome die nicht erlaubt sind aber auch Polynome mit allen Koeffizienten auf 0 gesetzt.

Mögliche Tests :




oleg@X1-Carbon: ~/Documents/era/Projekt1/Implementierung/root/test
File Edit View Search Terminal Help
Test #5 passed
output degree:2
expected degree:2
****
output coefficients
0.250000 0.250000 1.000000
expected coefficients
0.250000 0.250000 1.000000
****
output norm 4.000000
expected norm 4.000000
Test #6 passed
output degree:2
expected degree:2
****
output coefficients
0.500000 0.000000 1.000000
expected coefficients
0.500000 0.000000 1.000000
****
output norm 6.000000
expected norm 6.000000
Test #7 passed
output degree:1
expected degree:1
****
output coefficients
-0.500000 1.000000
expected coefficients
-0.500000 1.000000
****
output norm 2.000000
expected norm 2.000000
Test #8 passed
output degree:2
expected degree:2
****
output coefficients
2.000000 0.500000 1.000000
expected coefficients
2.000000 0.500000 1.000000
****
output norm -4.000000
expected norm -4.000000


oleg@X1-Carbon: ~/Documents/era/Projekt1/Implementierung/root/test
File Edit View Search Terminal Help
expected degree:1
****
output coefficients
-0.500000 1.000000
expected coefficients
-0.500000 1.000000
****
output norm 2.000000
expected norm 2.000000
Test #8 passed
output degree:2
expected degree:2
****
output coefficients
2.000000 0.500000 1.000000
expected coefficients
2.000000 0.500000 1.000000
****
output norm -4.000000
expected norm -4.000000
Test #9 passed
output degree:4
expected degree:4
****
output coefficients
0.125000 -0.375000 0.250000 0.500000 1.000000
expected coefficients
0.125000 -0.375000 0.250000 0.500000 1.000000
****
output norm 16.000000
expected norm 16.000000
Test #10 passed
output degree:5
expected degree:5
****
output coefficients
0.500000 1.000000 1.500000 0.333333 0.000000 1.000000
expected coefficients
0.500000 1.000000 1.500000 0.333333 0.000000 1.000000
****
output norm 3.000000
expected norm 3.000000
Total passed tests 10/10
oleg@X1-Carbon:~/Documents/era/Projekt1/Implementierung/root/test$

4. Ausführung des Programms

Mit dem Befehl

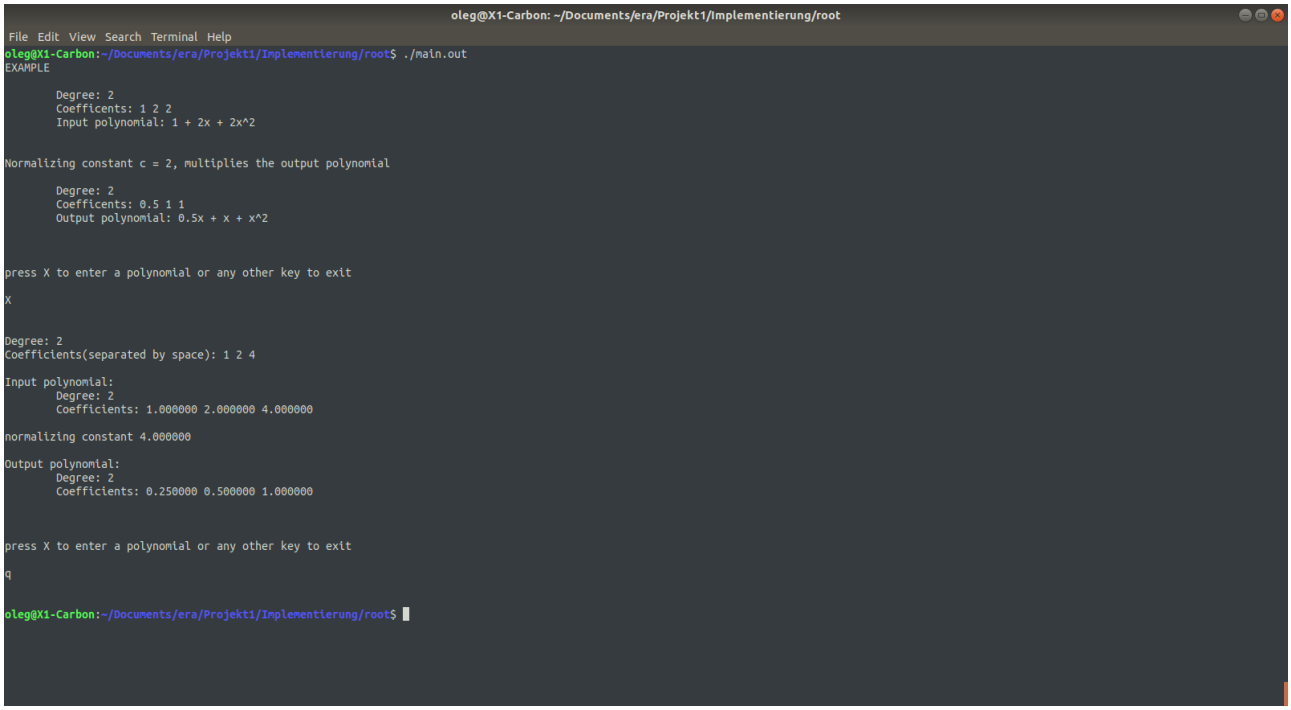
```
./main.out
```

Führen wir das Main Programm aus. Der Anwender wird dabei vorerst in einen Auswahlbildschirm geleitet. Dort hat er die Auswahl beliebig viele Polynome einzugeben und sieht gleich welche Koeffizienten und Normierungskonstante berechnet wurden. Der Benutzer muss **die Taste „X" oder „x"** drücken wenn er ein neues Polynom eingeben will, für Abbruch drückt man eine andere beliebige Taste.

4.1. Richtige Eingabe

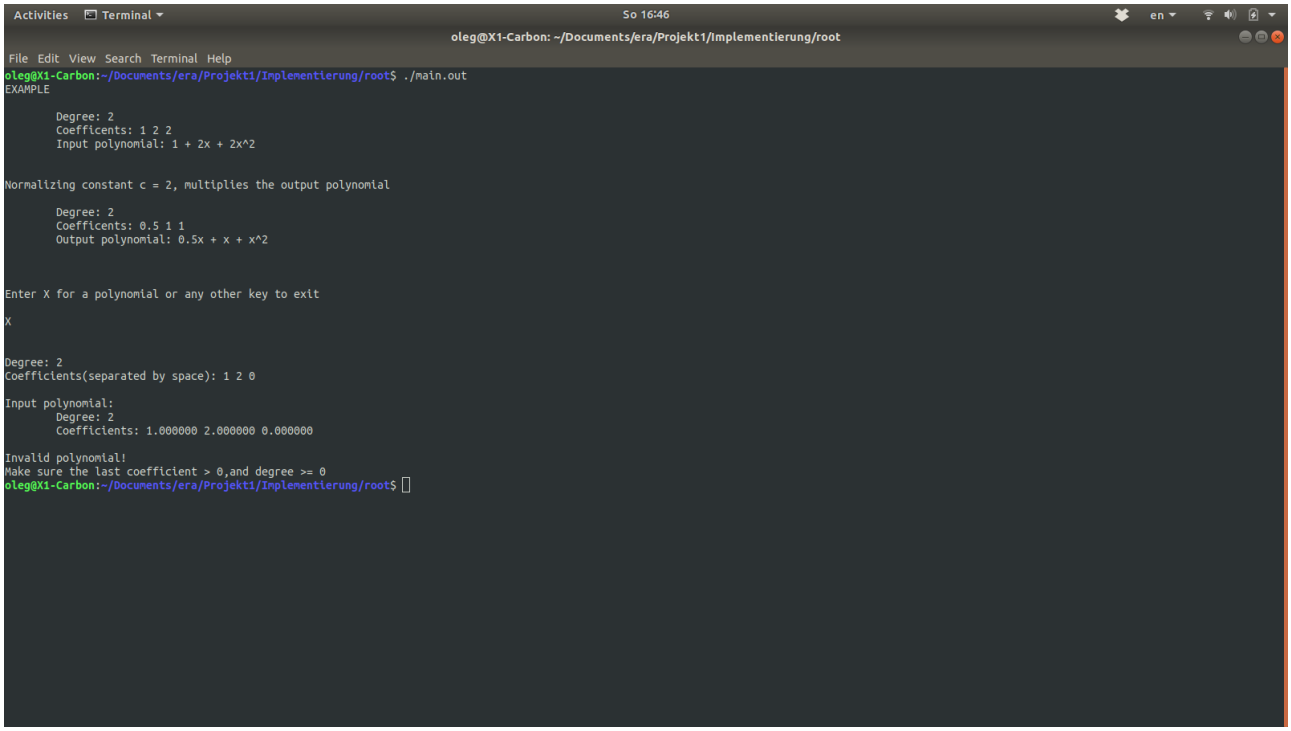
Im folgenden Screenshot betrachten wir den Fall für eine richtige Eingabe, nämlich :
x + 2x + 4x^2

Wir geben zuerst den Grad des Polynoms ein : *2*
dann die Koeffizienten : *1 2 4* (von Entertaste oder Leertaste getrennt)



4.2. Falsche Eingabe

In diesem Fall betrachten wir eine Falsche Eingabe die zur Programmhalt führt.
In diesem Beispiel, wurde der letzte Koeffizient des Polynoms auf 0 gesetzt aber der Grad stimmt dann nicht und der Benutzer muss ein neues Polynom eingeben, der Benutzer wird auch informiert, dass seint Polynom falsch ist.



Es kann auch sein, dass der Benutzer einen negativen (nicht erlaubter) Grad eingibt dann muss er wieder ein neues Polynom eingeben.