

Übungsaufgabenserie 11

Grundlagen der Informatik und Numerik

Abgabe: 15. 01. 2014, 23:55 Uhr, elektronisch

1. *PolynomBerechner.java*

- (a) Schreiben Sie, ausgehend von einer Grobstruktur, ein Programm *PolynomBerechner.java*, welches unter Verwendung der Klasse `Polynom` aus der Vorlesung für beliebige Polynome den Wert und dessen Ableitung an einer beliebigen Stelle $x = x_0$ berechnet.
(Aufbau: Polynomeingabe; wiederholte Argumenteingabe mit Werteberechnung und Ergebnisausgabe)
- (b) Berechnen Sie *mit Ihrem* Programm die Funktionswerte und die Werte der ersten Ableitungen folgender Polynome und tragen Sie die Ergebnisse als Kommentar am Ende Ihres Programms ein:
- (i) das Vorlesungsbeispiel $f_1(x) = 2x^5 - x^4 - 5x^2 + 3x - 9$ an der Stelle $x_0 = 3$,
 - (ii) das Polynom $f_2(x) = x^6 - 2x^5 + 3x^3 + x^2 + x - 1$ an der Stelle $x_0 = -1$ und $x_0 = 1$
- (c) Überprüfen Sie die Ergebnisse aus (b) per Hand mittels Horner Schema (*.pdf*-Datei).

2. *RationaleFunktionenTabulator.java*

- (a) Schreiben Sie unter Verwendung der Klasse `RationaleFunktion` ein Programm zum Tabellieren dieser Funktionen im vom Nutzer vorgegebenen Wertebereich mit gewünschter Schrittweite. In der Tabelle soll zeilenweise jeweils das Argument und der dazugehörige Funktionswert ausgegeben werden.
(Aufbau: Funktionseingabe; Eingabe Startargument, Anzahl der Werte und Schrittweite; Erzeugen der Wertetabelle; Tabellenausgabe)
- (b) Tabellieren Sie *mit Ihrem* Programm folgende Polynome und tragen Sie die Ergebnisse als Kommentar am Ende Ihres Programms ein (Ausgabeumleitung):
- (i) das Polynom $f_3(x) = x^5 + 2x^4 - 5x^3 - 10x^2 + 4x + 8$ von -2.5 bis 2.5 mit der Schrittweite von 0.5 ,
 - (ii) die Funktion $f_4(x) = (x^2 - 1)/(x^2 + 1)$ von -4 bis 4 mit der Schrittweite von 0.5 und
 - (iii) die Funktion $f_5(x) = (3(x-2))/((x-1)^2(x+2))$ von -2.5 bis 2.5 mit der Schrittweite von 0.1 .
- (c) Fertigen Sie je eine Skizze zum Verlauf der Funktionen an (*.pdf*-Datei). Vergleichen Sie diese mit der grafischen Ausgabe Ihres Taschenrechners oder eines entsprechenden Computerprogramms.

3. *Summe.java*

Das Aufsummieren $s = \sum_{i=1}^n i$ der ersten n natürlichen Zahlen kann Rechenzeit optimal durch die Formel $s = \frac{n}{2}(n+1)$ von **Carl Friedrich Gauß (1777-1855)**, die er als Neunjähriger in der Anfängerschule dem erstaunten Lehrer Büttner lieferte, erfolgen.

- (a) Beweisen Sie die Gaußformel durch vollständige Induktion.
- (b) Erweitern Sie das Programm *Summe.java* um eine neue Methode *gauss*, welche s mittels der Gaußformel berechnet. Wählen Sie den Datentyp `long` für s und n . Starten Sie das Programm und Überprüfen Sie die Korrektheit Ihrer Methode anhand einiger Beispiele.
Hinweis: Vermeiden Sie unnötige Fehler bei der ganzzahligen Division durch 2 und beachten Sie Auswirkungen der Reihenfolge der Operationen in der Gaußformel.
- (c) Starten Sie das Programm für $n = 2^{32} - 1$ und für $n = 2^{32}$: Im Bereich $n \in [0, 2^{32} - 1]$ erhalten Sie mit beiden Methoden korrekte Lösungen für s , ab $n = 2^{32}$ sind die gelieferten Werte fehlerbehaftet, vgl. Abschnitt 3.4. der Vorlesung. Tragen Sie die Testergebnisse als Kommentar in das Programm ein. Beweisen Sie diese Beobachtung durch Abschätzung unter der Berücksichtigung, dass s als `long`-Zahl dargestellt wurde (*.pdf*-Datei).
Hinweis: Verwenden Sie bei der Abschätzung Zweierpotenzen, `Long.MAX_VALUE = 263 - 1`.
- (d) Weshalb kann man beim Aufsummieren keine Verbesserung der Berechnung durch die Verwendung von `double`-Zahlen erreichen (*.pdf*-Datei)?

Summe.java

```
import Tools.IO.*;                                     // Eingaben

/**
 * Programm berechnet die Summe der ersten
 * n natuerlichen Zahlen mittels verschiedener
 * Verfahren.
 */
```

```

public class Summe
{
/* ----- */
// service-Methoden

/**
 * Berechnen der Summe von 0 bis n.
 * @param n letzter Summand
 * @return Summe 0 bis n (0 fuer n < 1)
 */
public long summe( long n)
{
    long summe = 0;
    for( long i = 1; i < n + 1; i++) summe += i;
    return summe;
}

/**
 * Gauss-Formel, berechnet Summe von 0 bis n.
 * @param n letzter Summand
 * @return Summe 0 bis n (0 fuer n < 1)
 */
public long gauss( long n)
{
    long summe = 0;
// (b) Summenformel nach Gauss

    return summe;
}

/* ----- */

/**
 * Startet das Programm,
 * erzeugt Summe bis zu einer eingelesenen Zahl.
 */
public static void main( String[] args)
{
// Eingabe von n
    long n = IOTools.readLong( "Summe von 0 bis ");

// n im kritischen Bereich
//    long n = (long)Math.pow( 2.0, 32.0) - 1;
//    long n = (long)Math.pow( 2.0, 32.0);

// Objekterzeugung
    Summe s = new Summe();

// Summenberechnung und Ausgabe
    System.out.println( "Summe von 0 bis " + n);
    System.out.println
        ( "Berechnung kann einige Sekunden dauern!");

    System.out.println
        ( "Schleife : " + s.summe( n));
    System.out.println
        ( "Gauss    : " + s.gauss( n));
}
}

/* ----- */
// (c) Testergebnisse
// Summe von 0 bis 4294967295

// Summe von 0 bis 4294967296

```