LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

Grundlagenpraktikum: Rechnerarchitektur

Potenzreihen (A321)

Projektaufgabe – Aufgabenbereich Algorithmik

1 Organisatorisches

Auf den folgenden Seiten finden Sie die Aufgabenstellung zu Ihrer Projektaufgabe für das Praktikum. Die Rahmenbedingungen für die Bearbeitung werden in der Praktikumsordnung festgesetzt, die Sie über die Praktikumshomepage[1] aufrufen können.

Wie in der Praktikumsordnung beschrieben, sind die Aufgaben teils relativ offen gestellt. Besprechen Sie diese innerhalb Ihrer Gruppe und konkretisieren Sie die Aufgabenstellung. Die Teile der Aufgabe, in denen C-Code anzufertigen ist, sind in C nach dem C17-Standard zu schreiben.

Der **Abgabetermin** ist **Sonntag 21. JUli 2024, 23:59 Uhr (CET)**. Die Abgabe erfolgt per Git in das für Ihre Gruppe eingerichtete Projektrepository auf dem `master` Branch. Bitte beachten Sie die in der `README.md` angegebene Liste von abzugebenden Dateien.

Die **Abschlusspräsentationen** finden in der Zeit vom **19.08.2024 – 30.08.2024** statt. Weitere Informationen werden noch bekannt gegeben. Beachten Sie, dass die Folien für die Präsentation am obigen Abgabetermin im PDF-Format abzugeben sind und keine nachträglichen Änderungen akzeptiert werden können.

Bei Fragen/Unklarheiten in Bezug auf den Ablauf und die Aufgabenstellung wenden Sie sich bitte an Ihre:n Tutor:in.

Wir wünschen Ihnen viel Erfolg und Freude bei der Bearbeitung Ihrer Aufgabe!

Mit freundlichen Grüßen
Die Praktikumsleitung

[1] `https://gra.caps.in.tum.de`

2 Potenzreihen

2.1 Überblick

Die Algorithmik ist ein Teilgebiet der Theoretischen Informatik, welches wir hier unter verschiedenen praktischen Aspekten beleuchten: Meist geht es um eine konkrete Frage- oder Problemstellung, welche durch mathematische Methoden beantwortet oder gelöst werden kann. Sie werden im Zuge Ihrer Projektaufgabe ein Problem lösen und die Güte Ihrer Lösung wissenschaftlich bewerten.

2.2 Funktionsweise

Eine Potenzreihe ist eine Reihe der Form

$$\sum_{i=0}^{\infty} a_i (z - z_0)^i$$

mit $a_i, z, z_0 \in \mathbb{C}$ und $i \in \mathbb{N}$. Den konstanten Wert z_0 nennt man dabei Entwicklungspunkt. Die einfachste Form der Potenzreihe mit reellen a_i, z, z_0, deren Folgenglieder ab einem bestimmten a_k alle gleich Null sind sind Ihnen bereits seit vielen Jahren vertraut: Polynome. Ihr Programm soll beliebige Potenzreihen für gegebene Koeffizienten a_i und z_0 an beliebigen komplexen Stellen z auswerten können.

2.3 Aufgabenstellungen

Ihre Aufgaben lassen sich in die Bereiche Konzeption (theoretisch) und Implementierung (praktisch) aufteilen. Sie können (müssen aber nicht) dies bei der Verteilung der Aufgaben innerhalb Ihrer Arbeitsgruppe ausnutzen. Antworten auf konzeptionelle Fragen sollten an den passenden Stellen in Ihrer Ausarbeitung in angemessenem Umfang erscheinen. Entscheiden Sie nach eigenem Ermessen, ob Sie im Rahmen Ihres Abschlussvortrags auch auf konzeptionelle Fragen eingehen. Die Antworten auf die Implementierungsaufgaben werden durch Ihren Code reflektiert.

Wichtig: Mindestens eine Ihrer Implementierungen muss mit mathematischen Operationen auskommen, die ausschließlich grundlegende Berechnungen durchführen (im Zweifel: die vier Grundrechenarten, Shifts und logische Operationen), nicht jedoch Instruktionen, die komplexere Berechnungen durchführen (z.B. Wurzel, Logarithmus, Exponentiation, etc.).

2.3.1 Theoretischer Teil

- Entwickeln Sie einen Algorithmus, welcher möglichst effizient die Potenzierung einer komplexen Zahl vornimmt.

2.3.2 Praktischer Teil

- Implementieren Sie in der Datei mit Ihrem C-Code die Funktion:

```
float complex p(size_t n, float complex a[n], float complex z_0,
                float complex z)
```

  Die Funktion bekommt `n` vom Benutzer spezifizierte komplexe Koeffizienten im Array `a`, sowie einen komplexen Entwicklungspunkt `z_0` und eine auszuwertende Stelle `z` übergeben. Die Funktion soll die oben definierte Formel für die Potenzreihe auswerten und das Ergebnis zurückgeben.

- Verwenden Sie die Funktion p, um Potenzreihen in Form beliebiger Eingaben in Diagramme zu plotten. Sie können Real- und Imaginärteil in getrennte Diagramme plotten, oder auf eine alternative Darstellungsform zurückgreifen. Wir empfehlen Ihnen, der Einfachheit halber als Ausgabeformat SVG zu verwenden, Sie können aber auch eine beliebige andere Methode zur Darstellung der Ergebnisse wählen.

2.3.3 Rahmenprogramm

Ihr Rahmenprogramm muss bei einem Aufruf die folgenden Optionen entgegennehmen und verarbeiten können. Wenn möglich soll das Programm sinnvolle Standardwerte definieren, sodass nicht immer alle Optionen gesetzt werden müssen. Wird eine Option mit Argument gesetzt, so ist das entsprechende Argument zu benutzen. Die Reihenfolge der Optionen bei einem gültigen Aufruf muss irrelevant sein. Wir empfehlen Ihnen, die Kommandozeilenparameter mittels `getopt_long`[2] zu parsen.

- `-V` *`<Zahl>`* — Die Implementierung, die verwendet werden soll. Hierbei soll mit `-V 0` Ihre Hauptimplementierung verwendet werden. Wenn diese Option nicht gesetzt wird, soll ebenfalls die Hauptimplementierung ausgeführt werden.
- `-B`*`<Zahl>`* — Falls gesetzt, wird die Laufzeit der angegebenen Implementierung gemessen und ausgegeben. Das *optionale* Argument dieser Option gibt die Anzahl an Wiederholungen des Funktionsaufrufs an.
- *`<Dateiname>`* — Positional Argument: Eingabedatei, welche die vom Benutzer festgelegten Werte des Arrays a beinhaltet
- *`<Realteil>,<Imaginärteil>`* — Positional Argument: Beliebige Anzahl an Werten des Arrays a
- `-z` *`<Realteil>,<Imaginärteil>`* — Auswertungspunkt z
- `-e` *`<Realteil>,<Imaginärteil>`* — Entwicklungspunkt z_0
- `-o` *`<Dateiname>`* — Ausgabedatei

[2] Siehe `man 3 getopt_long` für Details

- `-h` — Eine Beschreibung aller Optionen des Programms und Verwendungsbeispiele werden ausgegeben und das Programm danach beendet.
- `--help` — Eine Beschreibung aller Optionen des Programms und Verwendungsbeispiele werden ausgegeben und das Programm danach beendet.

Sie dürfen weitere Optionen implementieren, beispielsweise um vordefinierte Testfälle zu verwenden. Ihr Programm muss jedoch nur unter Verwendung der oben genannten Optionen verwendbar sein. Beachten Sie ebenfalls, dass Ihr Rahmenprogramm etwaige Randfälle korrekt abfangen muss und im Falle eines Fehlers mit einer Fehlermeldung, die auf den konkreten Fehler hinweist und einer kurzen Erläuterung zur Benutzung **terminieren** sollte. Fehlermeldungen jeglicher Art sollen auf `stderr` ausgegeben werden.

2.4 Allgemeine Bewertungshinweise

Beachten Sie grundsätzlich alle in der Praktikumsordnung angegebenen Hinweise. Die folgende Liste konkretisiert einige der Bewertungspunkte:

- Diese PDF-Datei darf nicht verändert, gelöscht oder umbenannt werden.
- Die in `README.md` geforderte Struktur des Projektrepositorys muss eingehalten werden.
- Der Exit-Code Ihrer Implementierung muss die erfolgreiche Ausführung oder das Auftreten eines Fehlers widerspiegeln. Benutzen Sie dafür die Makros aus `stdlib.h`.
- Stellen Sie unbedingt sicher, dass *sowohl* Ihre Implementierung *als auch* Ihre Ausarbeitung auf der Referenzplattform des Praktikums (`lxhalle`) kompilieren und vollständig korrekt bzw. funktionsfähig sind.
- Die Implementierung soll mit GCC/GNU as kompilieren. Verwenden Sie keinen Inline-Assembler. Achten Sie darauf, dass Ihr Programm keine x87-FPU- oder MMX-Instruktionen und SSE-Erweiterungen nur bis SSE4.2 verwendet. Andere ISA-Erweiterungen (z.B. AVX, BMI1) dürfen Sie nur benutzen, sofern Ihre Implementierung auch auf Prozessoren ohne derartige Erweiterungen lauffähig ist.
- Sie dürfen die angegebenen Funktionssignaturen, bis auf die Änderung des Funktionsnamens für die Benennung der unterschiedlichen Implementierungen (siehe nächster Punkt) *nicht* ändern.
- Verwenden Sie die angegebenen Funktionsnamen für Ihre Hauptimplementierung. Für alle weiteren Implementierungen gilt folgendes Schema: Hinter den Funktionsnamen wird das Suffix „`_V1`“, „`_V2`“, usw. angehängt.
- Denken Sie daran, das Laufzeitverhalten Ihres Codes zu testen (Sichere Programmierung, Korrektheit/Genauigkeit, Performanz) und behandeln Sie *alle möglichen*

Eingaben, auch Randfälle. Ziehen Sie ggf. alternative Implementierungen als Vergleich heran.

- Eingabedateien, welche Sie generieren, um Ihre Implementierungen zu testen, sollten mit abgegeben werden; größere Eingaben sollten stattdessen stark komprimiert oder (bevorzugt) über ein abgegebenes Skript generierbar sein.
- Stellen Sie Performanz-Ergebnisse nach Möglichkeit grafisch dar.
- Vermeiden Sie unscharfe Grafiken und Screenshots von Code.
- Geben Sie die Folien für Ihre Abschlusspräsentation im PDF-Format ab. Achten Sie auf hinreichenden Kontrast (schwarzer Text auf weißem Grund!) und eine angemessene Schriftgröße. Verwenden Sie 16:9 als Folien-Format und fügen Sie Foliennummern hinzu.