LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

Grundlagenpraktikum: Rechnerarchitektur

Z-Kurven (A216)
Projektaufgabe – Aufgabenbereich Bildverarbeitung

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 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 16. Juli 2023, 23:59 Uhr (CEST)**. Die Abgabe erfolgt per Git in das für Ihre Gruppe eingerichtete Projektrepository. Bitte beachten Sie die in der `README.md` angegebene Liste von abzugebenden Dateien.

Die **Abschlusspräsentationen** finden in der Zeit vom **21.08.2023 – 01.09.2023** 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 Ihren Tutor.

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

Mit freundlichen Grüßen
Die Praktikumsleitung

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

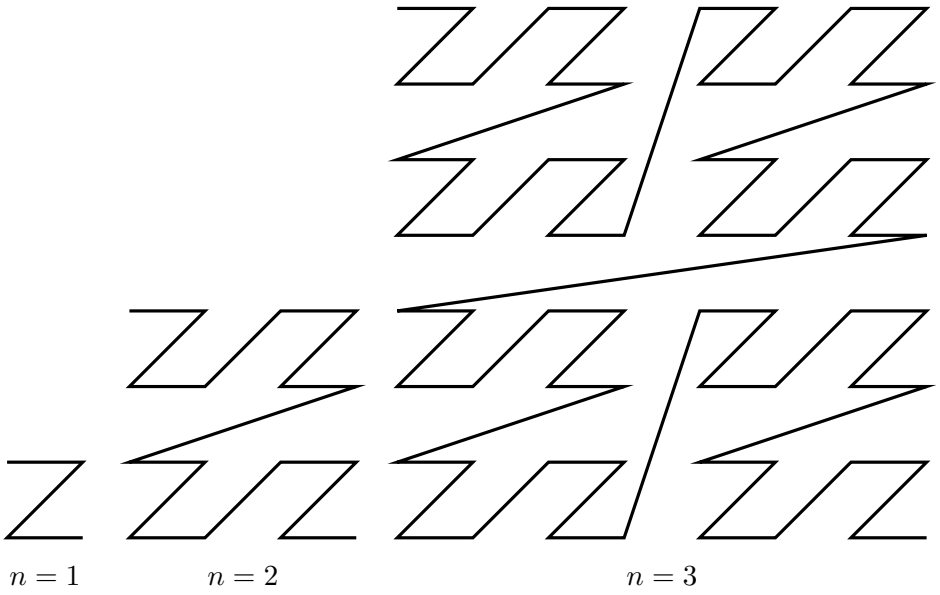
2 Z-Kurven

2.1 Überblick

Im Zuge Ihrer Projektaufgabe werden Sie theoretisches Wissen aus der Mathematik im Anwendungszusammenhang verwenden, um einen Algorithmus in C zu implementieren. Sie konzentrieren sich dabei auf das Feld des *Image Processing*, in welchem Pixelbilder, wie sie typischerweise Digitalkameras produzieren, als Eingabe für bestimmte Algorithmen verwendet werden und mathematische Überlegungen dadurch sichtbar gemacht werden.

2.2 Funktionsweise

In Ihrer Projektaufgabe befassen Sie sich mit sogenannten Z-Kurven. Dieser Text gibt Ihnen eine kurze Einführung, jedoch ist zum genaueren Verständnis ein Nachlesen in der Literatur erforderlich.
Die Z-Kurve ist eine planare Kurve die im einfachsten Fall ($n = 0$) einem Einheitsquadrat einbeschrieben wird. Man nennt diese Kurve "raumfüllend", da sie jede Ecke des Einheitsquadrats passiert. Durch einen rekursiven Prozess lassen sich aus der Basiskurve die Z-Kurven höheren Grades konstruieren. (Im Bild sind die Kurven für $n = 2$ und $n = 3$ dargestellt.)



Die Konstruktion der Z-Kurve für $n = 1, 2, \cdots$ ist Thema Ihrer Projektaufgabe.

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.

2.3.1 Theoretischer Teil

- Entwickeln Sie einen Algorithmus, welcher für eine Eingabe n eine geordnete Liste an (ganzzahligen) Koordinaten ausgibt, welche die Abfolge der Knotenpunkte der Z-Kurve n-ten Grades repräsentiert. Bevorzugen Sie iterative Lösungsansätze und vermeiden Sie Rekursion.
- Beschreiben Sie ein Beispiel, wofür die Z-Kurve eingesetzt wird.

2.3.2 Praktischer Teil

- Definieren Sie den Datentyp `coord_t` als einen Integer-Datentypen, der geeignet ist, Koordinaten auf der Kurve darzustellen.
- Implementieren Sie in der Datei mit Ihrem C-Code die Funktion:

  ```
  void z_curve(unsigned degree, coord_t* x, coord_t* y)
  ```

  Die Funktion bekommt den Grad der zu konstruierenden Z-Kurve `degree` übergeben und schreibt eine geordnete Liste von x- und y-Koordinaten in die Speicherbereiche `x` und `y`. Allokieren Sie hierzu in Ihrem Rahmenprogramm Buffer passender Größe.
- Implementieren Sie in der Datei mit Ihrem C-Code die Funktion:

  ```
  void z_curve_at(unsigned degree, size_t idx, coord_t* x, coord_t* y)
  ```

  Die Funktion bekommt den Grad der zu konstruierenden Z-Kurve `degree` und einen Index auf dieser Kurve übergeben und schreibt die zu diesem Index gehörende von x- und y-Koordinate in die Speicherbereiche `x` und `y`.
- Implementieren Sie in der Datei mit Ihrem C-Code die Funktion:

  ```
  size_t z_curve_pos(unsigned degree, coord_t x, coord_t y)
  ```

  Die Funktion bekommt den Grad der zu konstruierenden Z-Kurve `degree` und eine Position auf der Kurve übergeben und gibt den zu dieser Position gehörenden Index auf der Kurve zurück.

- Wir schreiben Ihnen kein Ausgabeformat vor, legen Ihnen aber nahe, die von der `z_curve`-Funktion zurückgegebenen, mit Linien verbundenen Koordinaten in Form einer SVG-Datei auszugeben. (SVG-Bild-Dateien werden durch Text beschrieben; ziehen Sie, wenn nötig, geeignete Literatur heran, um sich über das SVG-Format zu informieren.)

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.

- `-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.
- `-d<Zahl>` — Grad der zu konstruierenden Z-Kurve
- `<Zahl>` — Positional Argument: x
- `<Zahl>` — Positional Argument: y
- `-p` — Falls gesetzt, soll `z_curve_pos` aufgerufen werden
- `-i<Zahl>` — idx
- `-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 aussagekräftigen Fehlermeldung auf `stderr` und einer kurzen Erläuterung zur Benutzung terminieren sollte.

2.4 Allgemeine Bewertungshinweise

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

- 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 (nur dann) ändern, wenn Sie dies (in Ihrer Ausarbeitung) begründen.
- Verwenden Sie die angegebenen Funktionsnamen für Ihre Hauptimplementierung. Falls Sie mehrere Implementierungen schreiben, legen wir Ihnen nahe, für die Benennung der alternativen Implementierungen mit dem Suffix „`_V1`“, „`_V2`“ etc. zu arbeiten.
- Denken Sie daran, das Laufzeitverhalten Ihres Codes zu testen (Sichere Programmierung, 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.