Übungen zu Computergrundlagen WS 2017/2018

Übungsblatt 12: Programmieren in C

26. Januar 2018

Allgemeine Hinweise

- Abgabetermin für die Lösungen ist **Freitag, 02.02.2018, 11:00 Uhr**
- Schickt die Lösungen bitte per Email an Euren Tutor:
  - Montag 11:30 – 13:00: Julian Zeller (`julian.zeller@icp.uni-stuttgart.de`)
  - Montag 14:00 – 15:30: Miriam Kohagen (`mkohagen@icp.uni-stuttgart.de`)
  - Dienstag 14:00 – 15:30: Ingo Tischler (`itischler@icp.uni-stuttgart.de`)
  - Dienstag 15:45 – 17:15: Konrad Breitsprecher (`konrad@icp.uni-stuttgart.de`)
  - Donnerstag 09:45 – 11:15: Ashreya Jayaram (`ashreyaj@icp.uni-stuttgart.de`)

Ziel dieses Aufgabenblattes ist es, π numerisch mit C-Programmen zu approximieren. Dabei sollen dieselben Algorithmen wie auf Blatt 11 verwendet werden und dieselbe Anzahl an Berechnungen durchgeführt werden (1.000.000 Monte-Carlo Versuche/Stützstellen).

Aufgabe 12.1: Berechnung von π mit C (7 Punkte)

- **12.1.1** (3 Punkte) Schreibt ein C-Programm, das π mit der Monte-Carlo Methode aus Blatt 11 abschätzt. Dazu sollen gleichverteilte Zufallszahlen aus den Einheitsquadrat gezogen werden. Das Verhältnis von Punkten, die innerhalb des Einheitskreises liegen, zu der Gesamtzahl an gezogenen Punkten geht für hinreichend viele Versuche gegen $\pi/4$.

  Hinweise:

  - Bedingte Ausführung in C ermöglicht ähnlich wie in Python der `if (...) {...}`-Befehl. Dabei wird der Code in den geschweiften Klammern nur ausgeführt, wenn die Bedingung in der runden Klammer erfüllt ist.
  - Benutzt Fließkommazahlen vom Typ `double`.
  - Wie in Python 2.x gilt auch in C: Wenn Eure Näherung für π Null ist, dann habt Ihr wahrscheinlich nicht darauf geachtet, das Verhältnis der Punktanzahlen als Fließkommazahl zu berechnen. Eine einfache Möglichkeit, um eine Ganzzahl `N` in eine Fließkommazahl zu wandeln, ist etwa `1.0*N` oder `(double)N`.
  - Fließkomma-Zufallszahlen zwischen 0 und 1 erzeugt in C die Funktion `drand48()`. Um diese benutzen zu können, müsst Ihr mit Hilfe des Befehls

    ```
    #include<stdlib.h>
    ```

    die Header der Standard-Bibliothek am Anfang des Programms einbinden. Hilfe zu dieser Funktion findet Ihr auf ihrer `man`-Seite.
  - Zum Kompilieren Eures C-Programms verwendet am besten den Befehl

```
gcc -std=gnu99 -O3 -Wall -o compute_pi_1 compute_pi_1.c -lm
```

Beachtet, dass anstelle von "-std=c99" hier "-std=gnu99" gewählt ist. Das ist nötig, damit gcc auch Funktionen des POSIX-Standards wie `drand48()` in der Standardbibliothek freischaltet.

- **12.1.2** (2 Punkte) Schreibt nun ein C-Programm, das π mit Hilfe der Monte-Carlo Integration des Einheitskreises approximiert, also die Fläche unter der Funktion $f(x) = \sqrt{1 - x^2}$ nähert.

  Hinweis: Die Wurzel berechnet die Funktion `sqrt(x)`. Um diese benutzen zu können, müsst Ihr zusätzlich zur `stdlib.h` auch den Header `math.h` einbinden.

- **12.1.3** (2 Punkte) Schreibt schließlich ein C-Programm, das wie auf Blatt 11 die Integration des Einheitskreises mit gleichmäßig verteilten Stützstellen durchführt.

  Hinweis: Auch hier müsst Ihr darauf achten, rechtzeitig auf Fließkommazahlen zu wechseln.

Aufgabe 12.2: Laufzeitvergleich von Python und C (3 Punkte)

Da C-Programme verhältnismäßig kompliziert und länger sind als vergleichbare Programme in Python, muss es etwas geben, das diesen Mehraufwand rechtfertigt. Der große Vorteil von C ist die Geschwindigkeit, die die kompilierten Programme erreichen. Dies wollen wir nun durch einen Vergleich der Laufzeiten Eurer C-Implementationen mit entsprechenden Pythonskripten zeigen.

- **12.2.1** (1 Punkt) Passt Eure Programme so an, dass π jeweils 100 mal berechnet wird und messt die Laufzeit für alle 3 Berechnungsmethoden.

  Hinweis: Um die Laufzeit eines Programms zu bestimmen, könnt Ihr den Shell-Befehl `time` benutzen.

- **12.2.2** (1 Punkt) Messt nun ebenfalls die Laufzeit der 4 Pythonskripte in `/group/cgl/2017/12` mit derselben Anzahl an Stützstellen und Wiederholungen.

- **12.2.3** (1 Punkt) Vergleicht die Laufzeiten der verschiedenen Implementierungen. Warum verhält sich `compute_pi_4.py` etwas anders als die anderen Pythonversionen?