Übungen zu Computergrundlagen WS 2012/2013

Übungsblatt 12: Einführung in C

18. 1. 2013

Allgemeine Hinweise

- Abgabetermin für die Lösungen ist **Freitag, 25. 1. 2013, 13:30**.
- Abgabe wie immer als Email an Deinen Tutor.
- Bitte nur C-Programmcode und eine Textdatei mit den Antworten auf die Verständnisfragen schicken, *keine Binärprogramme*. Vergiss nicht, wie immer Deinen Namen als Kommentar an den Anfang der Dateien zu schreiben.

Aufgabe 12.1: Berechnung von π (8 Punkte)

12.1.1: Im Verzeichnis `~arnolda/computergrundlagen/12` befinden sich die Python-Skripte `pi1.py`, `pi2.py` und `pi3.py`, die auf drei verschiedene Weisen π annähern (vergleiche Übungsblatt 10).

Implementiere die drei Programme nun analog in der Programmiersprache C. Gib diese drei C-Programmquelltexte ab. (je Programm 2 Punkte)

Hinweise:

- `pi4.py` soll *nicht* in C übersetzt werden, da es kein einfaches Äquivalent zur komplexen Funktion `trapz` gibt.
- Lass die C-Programme zehnmal so viele Punkte berechnen wie das jeweilige Python-Programm.
- Verwende den Typ `double` für Fließkommazahlen.
- Zum Erzeugen einer Zufallszahl existiert die Funktion `rand()`. Diese erzeugt allerdings keine Fließkommazahl zwischen 0 und 1, sondern eine ganze Zahl zwischen 0 und `RAND_MAX`. Um daraus eine Fließkommazufallszahl zwischen 0 und 1 zu machen, verwende folgenden Befehl:

  ```
  double r = (double)rand() / RAND_MAX;
  ```

  Um die Funktion `rand()` verwenden zu können, muss die Headerdatei der Standardbibliothek eingebunden werden, vergleichbar mit **`import`** `random` in Python. Dazu füge folgende Zeile am Anfang der C-Datei ein:

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

- Zum Berechnen der Wurzel einer Zahl existiert die Funktion `sqrt()`. Um sie (und andere mathematische Funktionen) verwenden zu können, füge analog die folgende Zeile am Anfang der C-Datei ein:

  ```
  #include <math.h>
  ```

  Außerdem muss der Befehl zum Kompilieren die Option `-lm` verwenden, die die mathematische Bibliothek selber einbindet.

- Als Vorlage für die C-Programme kannst Du die Datei `~arnolda/computergrundlagen/12/template.c` verwenden.

- Als weitere Option beim Kompilieren solltest Du die Option `-O3` angeben (das erste Zeichen ist der Grossbuchstabe `O`, nicht die Zahl `0`!). Diese sorgt dafür, dass der Compiler den Code optimiert, also so kompiliert, dass er möglichst schnell ausgeführt werden kann.
- Der Gesamtbefehl zum Kompilieren von `pi1.c` sieht also zum Beispiel so aus:

```
gcc -O3 -lm --std=c99 -o pi1 pi1.c
```

- Nochmals die Bitte: *keine* Binärdateien, also das Ergebnis des `gcc`-Befehls, abgeben, nur die .c-Dateien, die Du schreibst.

12.1.2: Vergleiche die Geschwindigkeiten der C- und der Python-Programme. Wie viel mal schneller sind die C-Programme im Vergleich zu den entsprechenden Python-Programmen? Berücksichtige dabei die unterschiedliche Anzahl von Punkten (`MAX_STEPS`).

Betrachte zusätzlich das NumPy-Skript `pi4.py`, das wie `pi3.py` π mit Hilfe der Trapezregel berechnet. Wie ist das Verhältnis zwischen NumPy und C? (2 Punkte)

Hinweis: Zur Laufzeitmessung kannst Du den Unixbefehl `time` wie folgt verwenden:

```
> time python pi1.py
Abschaetzung fuer pi nach 1000000 Schritten: 3.14164

real    0m0.684s
user    0m0.672s
sys     0m0.008s
```

Die wesentliche Zeit dabei ist die erste, "echte" Zeit (real). Das ist die Zeit, die man auch auf einer Stopuhr messen würde.

Aufgabe 12.2: Verständnis (2 Punkte)

12.2.1: Warum wurde in dieser Vorlesung das Programmieren wohl mit Hilfe von Python eingeführt und nicht gleich in C? (1 Punkt)

12.2.2: Warum werden hingegen aktuelle, graphisch aufwändige Spieletitel meist in kompilierten Sprachen wie C oder C++ geschrieben? (1 Punkt)