The sun, with all those planets revolving around it and dependent on it, can still ripen a bunch of grapes as if it had nothing else in the universe to do.
Galileo Galilei

UNIVERSITÄT DES SAARLANDES

FR Mathematik
Andreas Buchheit

8. Übung zur Vorlesung
Programmierung
im Sommersemester 2019

Abgabe: Mittwoch, den 12.06.2019 bis spätestens 12 Uhr.

Aufgabe 8.1. (15 Punkte) N-Körperproblem
Im Folgenden sollen Sie zur Einübung von Strukturen die Trajektorien der Planeten des Sonnensystems bestimmen. Die Bewegung von Sonne und Planeten wird in allen drei Ortsdimensionen berechnet.

(a) **(3 Punkte)** Definieren Sie einen neuen Datentyp `solar_system` der durch eine Struktur gegeben ist, die

- die Anzahl an Körpern
- einen Pointer auf ein `double` array von Teilchenpositionen (hierbei werden die x, y, und z Positionen der einzelnen Teilchen nacheinander in einem einzigen zusammenhängendes Array abgespeichert)
- einen Pointer auf ein `double` array von Teilchengeschwindigkeiten
- einen Pointer auf ein `double` array von Teilchenmassen

enthält. Schreiben Sie eine Funktion

```
solar_system* new_solar_system(size_t num_bodies);
```

welche eine Struktur für das Sonnensystem anlegt und in der Struktur die Anzahl an Teilchen festhält. Die Stuktur soll, wie auch die Arrays die in ihr enthalten ist, dynamisch alloziert werden. Ein Pointer auf die Struktur wird zurückgeben. Schreiben Sie eine zugehörige Prozedur

```
void free_solar_system(solar_system*);
```

die den Speicher der Arrays in der Struktur, sowie die Struktur selbst wieder freigibt.

(b) **(2 Punkte)** Schreiben Sie eine Prozedur

```
void set_values_solar_system(
  solar_system*,
  double* positions,
  double* velocities,
  double* masses);
);
```

die die Positionen, Geschwindigkeiten und Massen in der Stuktur setzt. Die anfänglichen Positionen, Geschwindigkeiten und Massen finden Sie im Header `solar_system.h` auf der Website. Sie sind als globale Arrays im Row Major Format für die 5 massereichsten Objekte im Sonnensystem gespeichert (Massen sind absteigend geordnet). Hierbei wurden bereits geeignete dimensionlose Einheiten gewählt (Positionen in AU (mittlerer Abstand Erde-Sonne), Zeit in Tagen, Geschwindigkeit in AU/Tag, Massen in Sonnenmassen), weitere Umskalierungen sind nicht erforderlich. Die dimensionslose Gravitationskonstante G (in passenden dimensionlosen Einheiten) ist durch das Makro `GRAV_CONST` gegeben.

(c) **(2 Punkte)** Schreiben Sie eine Prozedur, welche als einzigen Parameter eine `solar_system` variable erhält und die momentanen Teilchenpositionen ausgibt. Schreiben Sie eine weitere Prozedur für die Teilchengeschwindigkeiten und die Massen.

(d) **(3 Punkte)** Schreiben Sie eine Prozedur

```
void acceleration_solar_system(solar_system*, double* acc_array);
```

welche die Beschleunigungen berechnet, die auf die Teilchen wirken. Die Beschleunigung $\boldsymbol{a}_i(t)$ auf Teilchen i zum Zeitpunkt t durch die gravitative Wechselwirkung mit allen anderen Teilchen ist gegeben durch

$$\boldsymbol{a}_i(t) = -\sum_{\substack{j=1 \\ j\neq i}}^{N} G m_j \frac{\boldsymbol{r}_i(t) - \boldsymbol{r}_j(t)}{||\boldsymbol{r}_i(t) - \boldsymbol{r}_j(t)||^3}.$$

Hierbei sind $\boldsymbol{r}_i(t) \in \mathbb{R}^3$ die Teilchenpositionen, m_i, $i = 1, \ldots, N$ die Teilchenmassen, $||\ ||$ ist die euklidische Norm in 3D und G ist die Gravitationskonstante, welche in dimensionlosen Einheiten bereits als Makro `GRAV_CONST` im Header bereitgestellt wird. Speichern Sie die Beschleunigungen für alle Teilchen im Row Major Format im Array `acc_array` ab.

(e) **(3 Punkte)** Schreiben Sie eine Prozedur

```
void propagate_solar_system(solar_system*, double delta_t);
```

welche die Teilchenpositionen und Geschwindigkeiten nach einem Zeitschritt `delta_t` bestimmt und in der Struktur des Sonnensystems aktualisiert. Ein einfaches aber effektives Verfahren zur Berechnung der neuen Positionen und Geschwindigkeiten ist die Leapfrog Integration, die Positionen und Geschwindigkeiten wie folgt propagiert

$$\begin{aligned}
\boldsymbol{r}_i(t + \Delta t) &= \boldsymbol{r}_i(t) + \boldsymbol{v}_i(t)\Delta t + \frac{1}{2}\boldsymbol{a}_i(t)\Delta t^2, \\
\boldsymbol{v}_i(t + \Delta t) &= \boldsymbol{v}_i(t) + \frac{1}{2}\Big(\boldsymbol{a}_i(t) + \boldsymbol{a}_i(t + \Delta t)\Big)\Delta t.
\end{aligned}$$

Berechnen Sie zunächst mithilfe der Prozedur aus (c) die Beschleunigung zum Zeitpunkt t und speichern Sie diese. Aktualisieren sie dann die Positionen in der Struktur und berechnen Sie erneut die Beschleunigungen und sichern Sie diese in einem zweiten Arrray. Daraufhin können Sie die neuen Geschwindigkeiten bestimmen. Schreiben Sie die aktualisieren Geschwindigkeiten zurück in die Struktur.

(f) **(2 Punkte)** Durch wiederholte Anwendung des Leapfrog Verfahrens können die Trajektorien der Planeten bestimmt werden. Da es sich bei dem Leapfrog Verfahren um ein symplektisches Verfahren handelt, liefert es auch nach langen Integrationszeiten verlässliche Ergebnisse.

Verwenden Sie nun einen Tag als Zeitschritt. Bestimmen Sie mithilfe ihres Algorithmus, wie viele Tage ein Jahr auf dem Jupiter dauert, d.h. wie lange der Jupiter benötigt um den Nullpunkt des Koordinatensystems einmal zu umrunden und zu seiner ursprünglichen Position zurückzukehren. Wenn Ihr Code keine Bugs enthält, sollten Sie eine Zeit von circa 12 (Erden-)Jahren erhalten.

Bonus: Sie erhalten zwei Bonuspunkte, wenn Sie die berechneten Planetentrajektorien visualisieren (z.B. als pgm). Drei Bonuspunkte erhalten Sie für eine animierte Darstellung.