Zadání příkladu pro cvičení z předmětu Programování pro fyziky

Z mechaniky znáte dobře zákony rovnoměrného přímočarého pohybu i zákony zachování hybnosti a energie. Budete mít za úkol doplnit několik řádek kódu, který s použitím těchto zákonů simuluje dokonale pružné srážky sady dvourozměrných hmotných disků.

Na přednášce jste se dozvěděli, že kromě polí existují i záznamy, tedy datové struktury, kde k jejich položkám přistupujeme za použití tečky a identifikátoru položky. V této domácí úloze si to máte za úkol vyzkoušet. Zadání úlohy je připraveno tak, aby stačilo doplnit pár řádek. Tyto řádky jsou označeny voláním funkce `zdeChybi`.

1. Vyjděte z kódu dostupného na stránce pro zadávání a odevzdávání úloh. Klíčovou součástí je deklarace datového typu `tPuntik`, který obsahuje následující informace o každém hmotném kruhovém disku:

   - Vektor polohy $\mathbf{r}$,
   - Vektor rychlosti $\mathbf{v}$,
   - Poloměr a,
   - Hmotnost m.

   Všechny disky (puntíky) jsou uloženy v jednom poli, které představuje globální proměnnou modifikovanou v místech kódu, kde se řeší jejich rozmístění, volný přímočarý pohyb a jejich srážky.

2. Po nastudování kódu odpovězte na 4 otázky v Kvizu na jeho konci. Odpovědi uveďte tamtéž.

3. Doplňte kód v proceduře `Odraz` podle tam zmíněné webové stránky. Všimněte si, že v okamžiku srážky dvou dotýkajících se disků o polohách středů $\mathbf{r}_i$, rychlostech $\mathbf{v}_i$, poloměrech a_i a hmotnostech m_i (kde $i = 1, 2$) se rychlosti změní na $\mathbf{v}'_i$ podle předpisu

   $$\mathbf{v}'_1 = \mathbf{v}_1 - m_2(\mathbf{r}_1 - \mathbf{r}_2)\,Q, \tag{1}$$
   $$\mathbf{v}'_2 = \mathbf{v}_2 + m_1(\mathbf{r}_1 - \mathbf{r}_2)\,Q, \tag{2}$$

   kde

   $$Q = \frac{2}{m_1 + m_2}\frac{(\mathbf{v}_1 - \mathbf{v}_2)\cdot(\mathbf{r}_1 - \mathbf{r}_2)}{(a_1 + a_2)^2}. \tag{3}$$

4. Doplňte kód ve funkci `cas_do_srazky` spočítaný pro dva disky o souřadnicích středů $\mathbf{r}_i$, rychlostech $\mathbf{v}_i$ a poloměrech a_i podle vztahu

   $$t = \frac{-(\mathbf{v}_1 - \mathbf{v}_2)\cdot(\mathbf{r}_1 - \mathbf{r}_2) - \sqrt{D}}{|\mathbf{v}_1 - \mathbf{v}_2|^2} \tag{4}$$

   kde

   $$D = \big[(\mathbf{v}_1 - \mathbf{v}_2)\cdot(\mathbf{r}_1 - \mathbf{r}_2)\big]^2 - |\mathbf{v}_1 - \mathbf{v}_2|^2\big[|\mathbf{r}_1 - \mathbf{r}_2|^2 - (a_1 + a_2)^2\big]. \tag{5}$$

   Záporné časy t znamenají, že srážka (možná) nastala v minulosti.

5. Pozorujte animaci pohybu puntíků za pomoci příkazů `gnuplot`. Zabudovaný `testSrazeni` by měl garantovat, že program bude vytvářet správný soubor `puntiky.txt`.

6. Změňte y-složku počáteční rychlosti malého disku o 10^{-15}. Pozorujte důsledky a své pozorování krátce popište v rámci odpovědi na čtvrtý bod kvízu.

Odevzdání řešení: Funkční kód s vyplněným kvizem a animovaný obrázek ve formátu `gif` vyrobený podle instrukcí uvedených na konci kódu odevzdejte na stránce pro odevzdávání domácích úloh.

Doplňující komentář: Vzorečky výše ukazují, že běžně potkáme fyzikální objekty s více vlastnostmi, které (ačkoli mají povahu čísel) přirozeně netvoří složky jednoho vektoru (např. zde hmotnost a poloha). Proto je často vhodné použít místo polí právě záznamy. Máme-li tedy

```
type  tPuntik = record
       x,y   : real; // 2D poloha
       vx,vy : real; // 2D rychlost
       r     : real; // polomer kruznice
       m     : real; // hmotnost (dulezita pri srazce)
end;
```

pak kinetickou energii disku uloženého v proměnné `a` spočteme

```
var a:tPuntik;
...
Ek := 0.5* a.m * ( sqr(a.vx) + sqr(a.vy) );
```

Takový zápis je bezpečnější a hezčí než pokud bychom použili pole

```
type tPuntikPole = array[1..6] of real;
```

a poté např. použili nepřehledný a neudržovatelný zápis

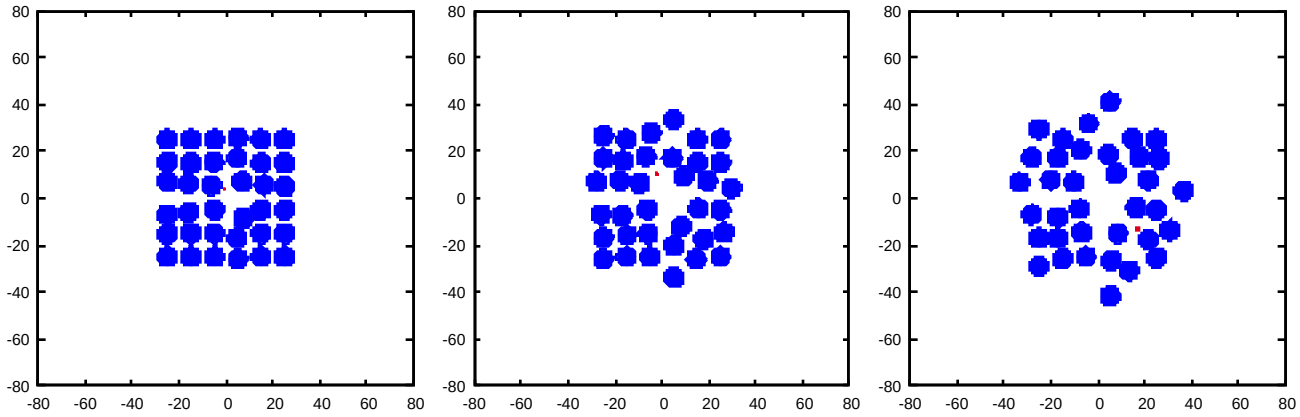
```
var a:tPuntikPole;
...
Ek := 0.5* a[6] * ( sqr(a[3]) + sqr(a[4]) );
```

případně i ten o dost správnější

```
const ivx = 3; ivy = 4; im = 6;
...
Ek := 0.5* a[im] * ( sqr(a[ivx]) + sqr(a[ivy]) );
```

Shrnuto, úloha má za úkol

- Seznámit se na jednoduchém problému s použitím záznamů.
- Naučit se upravovat cizí kód.
- Vyzkoušet nejprimitivnější metodu vytvoření animovaného obrázku.



Vysvětlení animace v programu `gnuplot`. Animace je prováděna dlouhým příkazem uvedeným na konci vzorového programu. Předpokládá se, že gnuplotu tyto příkazy „vnutíte“ pomocí přístupu copy/paste. Pokud by vás zajímalo, jak to ale funguje, následuje kratší vysvětlení. Dlouhý řádek tam uvedený je pro naše potřeby rozebrán na několik řádků za pomoci pokračovacího znaku „\“:

```
stats 'puntiky.txt' nooutput
do for [k=0:STATS_blocks-2] { \
  plot [-80:80][-80:80] 'puntiky.txt' \
  index k \
  using 1:2:3:3 \
  with circles palette fs solid; \
  pause 0.1 \
}
```

1. Nejprve se za pomoci příkazu `stats` zjistí, jaká data soubor „`puntiky.txt`“ obsahuje. Pokud byste vynechali slovo `nooutput`, zjistíte, co vše příkaz zjišťuje. I bez zobrazní statistiky je ale po skončení příkazu nastavena mj. proměnná `STATS_blocks` podle počtu bloků dat navzájem oddělených dvojicí prázdných řádků, které lze v souboru najít. (V našem programu funkce `vypisPolohyPuntiků` nezapomene na konci výpisu připojit tyto dva prázdné řádky, takže každý blok dat v soboru představuje jeden časový okamžik.)

2. Animace spočívá v postupném vykreslování jednotlivých snímků. Příkaz cyklu gnuplotu `do for` se o to postará.

3. Při animaci je většinou potřeba, aby se rozsah os mezi jednotlivými snímky neměnil. Proto je zobrazovaný interval na obou osách fixován. Pokud Vás zajímá detail, lze číslo 80 změnit. Vykreslujeme obsah souboru `puntiky.txt`.

4. V k-tém kroku animace vykreslujeme jen data z k-tého bloku odpovídající poloze puntíků v čase $t_k = k\Delta t$.

5. Vykreslované puntíky mají
   polohu-X : polohu-Y : velikost : barvu
   které se berou z prvního, druhého, třetího a opět třetího sloupečku dat

6. a malují se jako vybarvená kolečka, kde velikost se převádí na barvu za pomoci tzv. palety. K tomu jsme použili srozumitelné příkazy

   ```
   set palette model RGB defined ( 0 'red', 4 'blue' )
   set cbrange [0:4]
   ```

   Druhý příkaz je důležitý v okamžicích, kdy malý puntík zmizí z obrázku.

7. Krátkým pozastavením každého snímku určujeme rychlost animace.