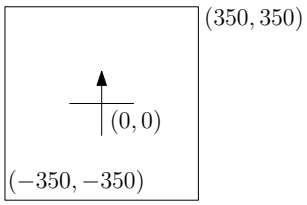
Želváci

Želvák Krunimír je velký myslitel. Zjistil, ze když si za krunýř přiváže trochu křídy, kreslí za sebou při svém plazení cestičku. I pojal plán nakreslit svoji podobiznu, samozřejmě včetně přesných detailů krunýře. Hned se dal pln nadšení do díla a práce mu šla pěkně od ... tlapy.

„Co to děláš, dědečku," zeptal se jednou Krunimíra jeho vnuk Krunoslav. „Krešlím tady švou poďobižnu," odpověděl Krunimír. „Žačal jsem š ní, když tvůj tatík ještě nebyl na švětě, a ještě nemám ani krunýř," dodal smutně. „To už ji aši dokrešlit neštihnu..." Vnuk Krunoslav, znalec moderní techniky, mu však poradil: „Tak si nech napsat program, který ji nakreslí za Tebe." Protože se ale s tlapami a zobákem moc dobře neprogramuje, najali si želváci vás.



Napište aplikaci, která bude simulovat Krunimírovo chování. Krunimír se nalézá v ohradě, což je bílý čtverec s rohy v souřadnicích $(-350, -350)$ a $(350, 350)$. Na začátku je Krunimír ve středu ohrady (tj. na souřadnicích $(0, 0)$) a kouká na sever (tj. v kladném směru osy y), jak je nakresleno na obrázku vpravo.

Vaše aplikace dostane *program* pro Krunimíra. Tento program je posloupnost následujících příkazů:

- `left(úhel)` – Krunimír se natočí o daný úhel proti směru hodinových ručiček, přičemž úhel musí být násobek 90 (Krunimír tedy kouká na sever, východ, jih nebo západ).
- `right(úhel)` – Stejné jako `left`, jenom se Krunimír točí po směru hodinových ručiček.
- `pen(hodnota)` – Pokud je $hodnota \leq 0$, Krunimír odloží křídu. Pokud je $hodnota > 0$, Krunimír si křídu nasadí. Na začátku má Krunimír křídu odloženou.
- `forward(kolik)` – Krunimír popoleze vpřed ve směru svého natočení o *kolik* jednotek. Hodnota *kolik* může být kladná, nulová, nebo záporná (Krunimír leze vzad).

Všechna čísla v programu jsou celočíselná. Mezerové znaky (mezera, tabulátor, znaky konce řádky CR a LF) se v programu mohou vyskytovat kdekoliv (kromě uvnitř názvů funkcí a uvnitř čísel), ignorujte je.

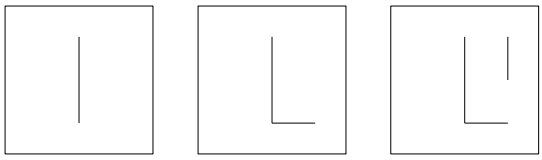
Po spuštění umožní vaše aplikace uživateli zvolit si soubor s programem pro Krunimíra a zadat počet tahů h. Můžete předpokládat, že program je bezchybný a že Krunimír nikdy nevyleze mimo svou ohradu. Vaším cílem je vytvořit obrázek, na kterém je černě nakresleno prvních h tahů, které Krunimír provedl. Tahem se rozumí každý příkaz `forward`, který Krunimír vykoná s nasazenou křídou. Pokud je zadaný počet tahů 0, vykreslete všechny Krunimírovy tahy.

Obrázek uložte v jednom z formátů PNG, BMP nebo GIF. To, že vykreslujete želví grafiku, neznamená, že váš program musí být stejně pomalý jako Krunimír :–)

Příklad: *Krunimírův program*

```
forward(200) pen ( 1 ) forward(
-400  )    right (90)
  forward(100)  left(90) pen (0)
forward(200)pen(1)forward(200)
```

Tahů: 1 *Tahů: 2* *Tahů: 3*



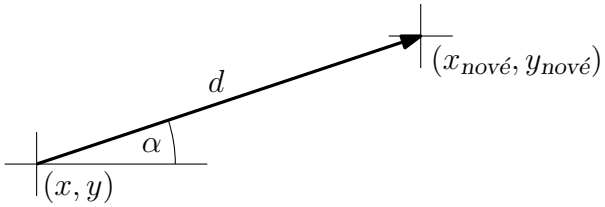
Za tuto základní část programu můžete získat 100 bodů. Kromě samotné funkčnosti se hodnotí i přehlednost zdrojového kódu a elegance řešení.

Dále můžete implementovat řadu rozšíření. Každé rozšíření se skládá z několika kroků, které musíte implementovat postupně jeden po druhém. Různá rozšíření můžete implementovat v libovolném pořadí. U každého rozšíření je uveden počet bodů, který získáte, pokud splníte všechny kroky v něm uvedené.

- **Lépe vybavený želvák [50 bodů]**
  1. Nový příkaz `color(č,z,m)` nastaví barvu křídy. Funkce má tři parametry, intenzitu červené, zelené a modré barvy v rozsahu 0 až 255. Hodnotu menší než nula považujte za nulu, hodnotu větší než 255 za 255.
  2. Rozšiřte příkaz `pen(šířka)` tak, že si po jeho provedení Krunimír nasadí křídu dané šířky. V případě, že je $šířka \leq 0$, dojde k odložení křídy.

3. Rozšiřte příkazy `left` a `right` tak, aby mohly jako parametr dostat libovolný celočíselný počet stupňů. Pokud stojí želvák na (x, y) a je otočený α stupňů od východního směru, tak pohybem vpřed o d jednotek se dostane na souřadnice

$$x_{nové} = x + d \cdot \cos(\alpha)$$
$$y_{nové} = y + d \cdot \sin(\alpha)$$



Pozor na to, že ve většině jazyků je třeba zadat úhel v radiánech. Přepočet je

$$radiány = stupně * \pi/180 \approx stupně/57.295779513082320877 21$$

Pozici želváka si musíte pamatovat jako desetinné číslo, ale při vykreslování můžete zaokrouhlovat.

- **Chytřejší želvák [100 bodů]**
  1. Program může obsahovat příkaz `repeat (počet) { nula a více příkazů }`. Při provedení příkazu `repeat` dojde k provedení příkazů v jeho těle *počet*-krát (pro počet ≤ 0 se příkaz neprovede vůbec).
  2. Program může obsahovat příkaz `if (číslo) { nula a více příkazů }`. Při provedení příkazu `if` dojde k provedení příkazů v jeho těle pouze v případě, že *číslo* je větší než nula.
  3. Program může obsahovat nové funkce. Ty se definují pomocí `define novafunkce() { nula a více příkazů }`. Jméno nové funkce se skládá jenom z malých písmen anglické abecedy. Novou funkci lze použít kdekoliv *po* její definici a funkce je možné volat rekurzivně, tj. funkce smí volat sama sebe. Definice funkcí nemohou být vnořené, tj. uvnitř definice funkce není možné definovat další funkci.
  4. Nově definované funkce mohou mít libovolný počet parametrů. Taková funkce se definuje jako `define f(parametry) { nula a více příkazů }`. Parametrů je libovolný pevný počet, jsou celočíselné, jejich jména se skládají z malých písmen anglické abecedy a jsou oddělená čárkou. Uvnitř funkce s parametry můžete jako argument libovolné funkce (a jako argument příkazů `if` a `repeat`) použít `a`, `a+b`, `a-b`, `a*b`, `a/b`, kde `a` a `b` jsou buď čísla nebo parametry aktuální funkce. Dělení je celočíselné.
     Příklad:
     ```
     define spirala(delka) {
       if (delka) { forward(delka) right(90) spirala(delka-4) }
     } pen(1) spirala(100)
     ```
     
  5. Uvnitř funkce s parametry můžete jako argument libovolné funkce (a jako argument příkazů `if` a `repeat`) použít jakýkoliv výraz, který obsahuje celá čísla, parametry aktuální funkce, kulaté závorky a operátory `+ - * /`. Priority operátorů jsou stejné jako v matematice.
- **Želváčí farma [50 bodů]**
  Toto rozšíření je možné implementovat až po splnění prvních 4 kroků z rozšíření Chytřejší želvák.
  1. Příkaz `split { nula a více příkazů }` vytvoří klon želváka na stejném místě a se stejnou orientací. Původní želvák pokračuje v provádění dalšího příkazu za příkazem `split`, klon provede příkazy uvedené v těle příkazu `split`.
     Pokud je želváků více, nijak se navzájem neovlivňují a provádí své tahy paralelně. Tedy všichni želváci provedou svůj první tah, teprve potom provedou všichni svůj druhý tah a tak dál. V rámci jednoho tahu není pořadí želváků definované.