Zadanie 12

(Grafika żółwiowa / Turtle graphics) Żółw potrafi wykonywać trzy rozkazy: Idź prosto 1m (`"F"`), obróć się o 60° w lewo (`"L"`) oraz obróć się o 60° w prawo (`"R"`). Proszę napisać funkcję `TurtleGo`, która dla ciągu rozkazów (np. "FRR-FLFFLFLRF") zwróci kolejne położenia żółwia, który startuje w punkcie $(0, 0)$.

Proszę tak zmodyfikować funkcję, by wartość kąta obrotu można było podać jako dodatkowy argument funkcji.

Proszę wykorzystać program `TurtleGo` do narysowania śnieżynki Kocha, dla której listę rozkazów można generować w następujący sposób.

1. Zacznij od listy `"FRFRFRFRFRF"`.
2. Każdy rozkaz `"F"` zamień na `"FLFRRFLF"` (funkcja `StringReplace`).
3. Powtórz punkty 1. i 2. ok. sześć razy (funkcja `Nest` lub `Do`).
4. Wykonaj program `TurtleGo` na ciągu rozkazów.
5. Wykorzystaj funkcje `Line` i `Graphics` by narysować trasę przejścia żółwia.

Proszę użyć funkcji `Manipulate`, by zobrazować wpływ zmiany kąta na kształt śnieżynki.

Proszę poeksperymentować z różnymi listami rozkazów (t.j. z różnymi regułami zamiany `"F"` na inne rozkazy) oraz rozszerzyć zakres rozkazów tak, by można skorzystać z recepty na trójkąt Sierpińskiego. Inne ciekawe przykłady znajdują się w rozdziale 1.3 książki "The Algorithmic Beauty of Plants"

Najłatwiej podejść do implementacji przez podzielenie programu na proste podprogramy, w szczególności na program do generowania listy rozkazów, programy do wydawania rozkazów żółwiowi i program do generowania listy położeń. Stan żółwia można reprezentować np. jako wyrażenie `Turtle[{x,y},angle]`, na które działać będziemy funkcjami typu `TurtleMove[Turtle[{x,y},angle],"F"]`, które zwracać będą odpowiednio zmieniony stan żółwia.

Proszę policzyć obwód i pole powierzchni śnieżynki Kocha w zależności od liczby iteracji.

(nieobowiązkowe) Proszę pokolorować trasę, którą przebył żółw funkcją `Hue`. Proszę tak rozszerzyć program, by było możliwe podawanie kilku możliwych reguł dla punktu 2. wraz z prawdopodobieństwami ich wylosowania (najłatwiej skorzystać w funkcji `StringReplace` z `RuleDelayed` czyli `:>`).

1 Zadanie 13

(nieobowiązkowe) Proszę napisać program `TurtleGoExtra` j.w., ale o liście rozkazów rozszerzonej o znaczniki `"["`, który oznacza zapisanie położenia i kierunku żółwia, i `"]"` przywracającego zapisane położenie (np. `"F[F]RF"`, mówi idź prosto, zapisz stan, idź prosto, przywróć zapisany stan, skręć w prawo, idź prosto). Dopuszczalne są zagnieżdżone znaczniki `"["` i `"]"`. Można skorzystać z faktu, że funkcja `StringReplace[#,{"["-> ",{", "]" -> "},"}]&` generuje z listy rozkazów coś, co wygląda prawie jak wyrażenie w Mathematice (zagnieżdżona lista), proszę skorzystać z tego faktu i tak przetworzyć ciąg wejściowy by na wyjściu otrzymać zagnieżdżoną listę z ciągami rozkazów i listami jako elementami (funkcje `ToExpression`, `StringTrim`, `StringJoin`, `DeleteCases`, `Characters`,`ToString` mogą okazać się pomocne). Mając zagnieżdżoną listę rozkazów można działać rekurencyjnie.

Proszę wykorzystać tak napisany program aby narysować dwuwymiarowe (a później trójwymiarowe) drzewo. Recepta na dwuwymiarowe drzewo jest opisana na Wikipedii, trójwymiarowe drzewo trzeba stworzyć samemu.

(dla bardzo znudzonych) Narysowane drzewo proszę pokolorować, a na końcach łodyg umieścić zielone listki.