Informe Trabajo Final

Escoba del 15 - Programación Declarativa

Docente: Maximiliano A. Eschoyez.

Alumnos: BOZZANO, Felipe Gabriel.
OLMOS BOSSA, Francisco Augusto.

Institución: Universidad Blas Pascal.

Carrera: Ingeniería Informática.

Asignatura: Programación Declarativa.

Fecha entrega: 17/12/2021

Problemática

Resumen

Realizamos un programa de software que corre, de manera interactiva con el usuario, el juego “La escoba del 15”, codificado en el paradigma de programación funcional en los lenguajes Haskell y Python.

Consigna

El objetivo de este Trabajo Final es desarrollar un programa que permita al usuario jugar un juego de cartas que implique cartas en mano, en mesa y en el mazo, tanto para naipes españoles como ingleses. Se deberán considerar los siguientes puntos:

- El programa debe ser interactivo con el usuario.
- La interfaz de usuario será en modo texto.
- Deberá estructurarse en módulos según las funcionalidades.
- Deberá contener tipos de datos definidos específicamente para resolver la problemática.

El juego deberá implementarse en los lenguajes de programación Haskell y Python. Si bien Python es multiparadigma, se deberá realizar dentro del paradigma de Programación Funcional. Al finalizar ambas versiones, se debe realizar una comparación de similitudes y diferencias entre ambos lenguajes.

Solución

Python

La primera implementación del juego fue en este lenguaje ya que estamos más familiarizados que con Haskell. Al ser multiparadigma, nos facilitó mucha veces el hecho de poder escribir código rápido para asegurarnos que la solución pensada funcionaba correctamente, y luego podíamos adaptarla al paradigma de programación funcional. Sin embargo, nos resultó más sencillo atacar directamente el problema desde este paradigma, ya que las soluciones que traen estos lenguajes son fáciles de usar.

Para resolver el problema, generamos funciones que nos devuelven los “actores” del juego (jugadores, mazo, mesa), y otras funciones que los usan para correr el juego. El jugador activo tiene la posibilidad de elegir si tiene escoba o no, y elegir las cartas que quiere levantar de la mesa o la carta que quiere tirar de su mano. Si el jugador se equivoca, no se le da otra chance de volver a jugar, sino que se lo obliga a tirar una carta de su mano, luego de habernos asegurado que la suma de las cartas elegidas no es igual a 15.

Al finalizar, se muestra el puntaje del ganador.

Haskell

Debido a que Haskell solo acepta programación funcional, encaramos la solución creando módulos para el Mazo y el Jugador definiendo estos nuevos tipos de datos con funciones necesarias para la resolución del problema, como ser mezclar las cartas, repartir cartas a los jugadores, crear el mazo, los jugadores. Contamos con una función principal encargada de comenzar el juego y permitirle a cada jugador realizar su jugada en su turno. Al terminarse todas las cartas del mazo y que todos los jugadores se queden sin cartas en la mano, es cuando damos por concluido el juego y calculamos quien fue el ganador.

Al modularizar las funciones, nos facilitó la implementación del programa, ya que redujimos la complejidad dividiendo el problema en partes mas pequeñas.

Trasladamos las funcionalidades y flujos ya establecidos en el diseño del juego en Python a Haskell.

Conclusión

El problema que podemos mencionar en Python es que al ser multiparadigma, muchas veces nos vimos tentados a dejar soluciones del paradigma procedural, que nos alejaba de nuestro objetivo principal. Si bien Python cuenta con muchas funciones que nos facilitaron algunas implementaciones como ser el método para mezclar las cartas, el cual ya esta resuelto en este lenguaje, notamos que la resolución en Haskell fue mucho mas funcional que la de Python.

Un inconveniente que tuvimos a la hora de desarrollar el juego en Haskell fue que no pudimos controlar las respuestas de los usuarios ante interacciones con el juego, como ser el indice de las cartas elegidas ya sea en la mesa como en la mano del jugador. Podemos concluir que la interacción con el usuario complejiza el programa en Haskell mientras que en Python es muy sencillo.

A su vez pensamos que hubiera sido mejor arrancar la solución del juego primero en Haskell y no en Python como lo hicimos ya que en este ultimo tuvimos mas libertad de salirnos del paradigma funcional.

Nos dimos cuenta de que la comunidad de Python en el paradigma funcional es mucho mas grande y por ende es mas sencillo encontrar informacion de utilidad que en Haskell.

Link Github: ESCOBA15-PD