Specyfikacje formalne i programy funkcyjne

Opis projektu

Julia Kłos, Jakub Wrzosek

Kod programu zawiera się w jednym pliku źródłowym, w którym znajdują się zarówno deklaracje wykorzystywanych typów danych oraz definicje wykorzystywanych funkcji.

1. **Deklaracje typów.**

Nie licząc standardowych typów danych dostępnych dla języka Haskell w programie wykorzystywane są dwa całkowicie nowe typy danych (***State*** oraz ***Cell***) oraz dwa synonimy (***Value*** oraz ***Table***). Typ ***State*** określa stan w jakim znajduje się pole z tablicy (mozaiki) oraz odpowiada za przechowanie koloru na jaki dane pole jest pokolorowane lub wartość *UNDECIDED* jeśli pole nie zostało jeszcze przeprocesowane. Typ ***Cell*** przechowuje niezbędne informacje na temat każdego pola. Są to: stan ***State***, w jakim znajduje się dane pole oraz wartość ***Value*** określającą ilość sąsiadujących pól, które muszą zostać pokolorowane. Pierwszym synonimem jest ***Value,*** przyjmujące wartość *Maybe Int* (dla ilości sąsiadujący pól do zamalowania) oraz ***Table,*** czyli tablicę będącą odpowiednim odwzorowaniem danych tekstowych przechowywanych w pliku, reprezentujących daną mozaikę, na typy wykorzystywane przez program.

1. **Wczytywanie danych oraz prezentacja wyników**

Wczytywanie mozaiki z pliku .txt odbywa się za pomocą funkcji ***readPuzzle****.* Funkcja ta wczytuje zawartość pliku, którego nazwa zostaje pobrana od użytkownika przez funkcję ***getFileName***. W następnej kolejności za pomocą funkcji ***convertToTable*** dane tekstowe konwertowane są do struktury danych ***Table,*** na której wykonywana jest część obliczeniowa programu. Za prezentacja wyników działania programu w konsoli (wypisanie stanów pól tablicy reprezentowanych przez znaki) odpowiada funkcja ***printMap***.

1. **Rozwiązanie mozaiki**

Rozwiązanie problemu kolorowania mozaiki zostało zrealizowane w sposób iteracyjny.   
W programie generowane są różne możliwości rozwiązań kolejno dla każdego z pól, których poprawność jest następnie sprawdzana. W każdej iteracji proces ten jest powtarzany, aż do uzyskania prawidłowego rozwiązania, które spełnia wymogi zadania. Po wygenerowaniu struktury ***Table*** za pomocą funkcji ***convertToTable,*** wywoływana jest funkcja ***solvePuzzle***, która odpowiada za rozwiązanie mozaiki. Wykorzystuje ona funkcje ***positionsList,*** która generuje listę komórek w tablicy ***Table*** oraz ***solveOnePass,*** która odpowiada za przeprowadzenie pojedynczej iteracji obliczeń. Funkcja ***solveOnePass*** wykorzystuje funkcje takie jak ***checkCorrectnes***, ***isSolved*** oraz***setTwoNextTables***. Dodatkowo wykorzystywane są inne (mniejsze) funkcje pomocnicze takie jak: ***stateList, isSolvedRow, replace\_elem, replace***czy ***byInd.*** Opisy tych funkcji zawarte zostały w formie komentarzy w kodzie źródłowym