Задания к курсу
“Разработка прикладных компьютерных систем”
3 курс, 2021-2022 уч. г. (Часть 2)

4. Реализуйте компилятор и оконное приложение редактора для написания программ для реализованного компилятора. Скомпилированные инструкции имеют вид 32-х разрядных целых чисел, в которых содержится код операции (младшие 5 бит) и индексы регистров, с которыми эта операция должна быть выполнена (регистры 32-битовые, каждый индекс регистра == 9 бит, от младших битов к старшим в инструкции расположены биты, соответствующие первому, второму, третьему индексам:

   |bbbbbbbbb|bbbbbbbbb|bbbbbbbbb|bbbbb|

   | 3 operand | 2 operand |1 operand | operation |

). Возможные значения кодов операций (в 10-й системе счисления):

   0 – вывести состояние всех регистров в системе счисления, которая записана в 1 операнде;

   1 – поразрядная инверсия над содержимом 1 операнда, результат сохраняется в 3 операнд;

   2 – дизъюнкция над 1 и 2 операндом, результат сохраняется в 3 операнд;

   3 – конъюнкция над 1 и 2 операндом, результат сохраняется в 3 операнд;

   4 – сложение по модулю 2 над 1 и 2 операндом, результат сохраняется в 3 операнд;

   5 – импликация над 1 и 2 операндом, результат сохраняется в 3 операнд;

   6 – коимпликация над 1 и 2 операндом, результат сохраняется в 3 операнд;

   7 – эквиваленция над 1 и 2 операндом, результат сохраняется в 3 операнд;

   8 – стрелка Пирса над 1 и 2 операндом, результат сохраняется в 3 операнд;

   9 – штрих Шеффера над 1 и 2 операндом, результат сохраняется в 3 операнд;

   10 – сложение 1 и 2 операнда, результат сохраняется в 3 операнд;

   11 – вычитание из 1 операнда 2 операнда, результат сохраняется в 3 операнд;

   12 – умножение 1 и 2 операнда, результат сохраняется в 3 операнд;

   13 – целочисленное деление 1 операнда на 2 операнд, результат сохраняется в 3 операнд;

   14 – остаток от деления 1 операнда на 2 операнд, результат сохраняется в 3 операнд;

   15 – обмен содержимого 1 и 2 операндов (операция swap);

16 – занести в 1 операнд в байт с номером, который находится во 2 операнде, байт, значение которого лежит на месте 3 операнда;

17 – вывести содержимое операнда 1 в системе счисления, основание которой записано во втором операнда;

18 - ввести в операнд 1 в системе счисления, основание которой записано во 2 операнде, значение с клавиатуры;

19 – найти максимальное значение 2^p, на которое делится 1 операнд, результат сохраняется в 3 операнд;

20 – сдвиг влево содержимого 1 операнда на количество бит, которое находится во 2-ом операнде, результат сохраняется в 3 операнд;

21 - сдвиг вправо содержимого 1 операнда на количество бит, которое находится во 2-ом операнде, результат сохраняется в 3 операнд;

22 – циклический сдвиг влево содержимого 1 операнда на количество бит, которое находится во 2-ом операнде, результат сохраняется в 3 операнд;

23 – циклический сдвиг вправо содержимого 1 операнда на количество бит, которое находится во 2-ом операнде, результат сохраняется в 3 операнд;

24 – скопировать значение 2 операнда в 1 операнд.

Поток команд поступает из двоичного потока (MemoryStream). Для запуска компиляции и исполнения пользователь может нажать на кнопку с картинкой Play (зелёный треугольник) (для компиляции и исполнения инструкций, указанных в редакторе), либо нажать на кнопку с картинкой папки для создания диалога открытия файла и выбора через него файла с инструкциями (для компиляции и исполнения инструкций, указанных в файле). Во время загрузки файла, процесса компиляции исходного кода и исполнения скомпилированного кода, поверх основного окна должен появиться диалог загрузки с меняющимся текстом: во время поиска файла - “Searching for file”, во время компиляции кода - “Compiling”, во время исполнения скомпилированного кода - “Executing”. При ошибке компиляции (проверку корректности инструкций реализуйте при помощи отдельного сервиса (класса) валидации) необходимо отобразить диалог с сообщением о количестве ошибок и информацией о каждой ошибке и кнопкой OK (диалог может быть закрыт как по кнопке OK, так и по нажатию на затемнённый фон). В случае успешной компиляции, в модальном окне, имитирующем консоль, должно быть произведено исполнение извлечённых из двоичного потока инструкций. При возникновении ошибки времени исполнения (деление на 0, невалидное основание системы счисления, некорректный пользовательский ввод (обработка ввода должна производиться посредством валидатора из задания 1)), исполнение инструкций должно быть прервано и должна быть сгенерирована исключительная ситуация собственного типа, содержащая порядковый номер (начиная с 1) инструкции и дополнительную информацию о произошедшей ошибке; исключительная ситуация должна быть

перехвачена и в диалоге MessageDialog должна быть выведена информация о произошедшей ошибке (диалог может быть закрыт как по кнопке OK, так и по нажатию на затемнённый фон). После исполнения инструкций окно консоли должно продолжать отображаться до момента закрытия его пользователем; попытки закрытия окна консоли во время исполнения кода должны быть перехвачены и проигнорированы. Для ввода информации в консоль необходимо использовать стилизованные экранные клавиатуры, реализованные в задании 2 (ввод в консоль с физической или стандартной экранной клавиатуры не допускается).

Оконное приложение редактора содержит две области, в которых содержатся инструкции в текстовом и в графическом формате. Графическая область представляет собой последовательность визуализированных инструкций и операндов: операции выбираются посредством элемента управления ComboBox, значения индексов операндов вводятся при помощи TextBox, в графической области имеются возможности добавления новых инструкций в произвольное место и удаления инструкций. Текстовый формат содержит строковое представление команд в формате

<команда>,<1 oprnd indx>,<2 oprnd indx>,<3 oprnd indx>

, по одной команде на каждой отдельной строке (пробельные символы между лексемами игнорируются). При внесении любых изменений в текстовый формат инструкций, эти изменения должны быть применены также к графическому формату и наоборот.

Приложение должно быть построено на базе архитектуры MVVM с использованием функционала, реализованного в заданиях 1 и 2.

5. Разработать интерактивную игру «Угадай шрифт». В оконном приложении выводится строка случайным шрифтом (все параметры шрифта определяются случайным образом). Целью пользователя за наименьшее число шагов определить все параметры шрифта: название шрифта, размер, тип начертания, цвет шрифта. При этом у пользователя есть подсказки: можно посмотреть случайную букву имени шрифта; посмотреть начертание своей фразы одним выбранным шрифтом; узнать больше, меньше или равен искомому предполагаемый размер шрифта; выполнить не более 4 проверок цвета по его четырём составляющим в формате CMYK. Ваша программа должна вести лог-файл действий пользователя, а также хранить информацию о 10 пользователях, которые выполнили задание лучше всех. Также необходимо разработать систему очков, которые будут начисляться и сниматься с пользователя за каждый его ход. Постарайтесь разработать как можно более удобный и приятный интерфейс для данной игры.

6. Реализуйте оконное приложение для визуализации структур данных типа “пирамида” (ключами узлов являются целые числа типа int).

   Для структур данных реализуйте 4 сборки (одна с интерфейсом функционала пирамиды (вставка элемента, возврат минимального элемента, удаление минимального элемента, decrease/increase key, слияние двух пирамид) и настройками (ориентация на min или на max); остальные три - с классом, реализующим интерфейс из вышеописанной сборки: бинарная пирамида, биномиальная пирамида, фибоначчиева пирамида. В реализованных пирамидах допускается существование равных по отношению порядка ключей.

   В оконном приложении реализуйте отображение коллекции пирамид заданного типа (тип задаётся именем сборки, в которой описан; имя сборки помещается в файл конфигураций appsettings формата json, находящийся в папке сборки приложения; реализовать с использованием механизма DI/IoC); каждая модель пирамиды отображается на своей вкладке элемента управления TabControl. Также на всех вкладках (кроме последней) в заголовке должна присутствовать кнопка закрытия вкладки (закрытие необходимо реализовать через пробрасывание события из модели модели отображения пирамиды к модели отображения всего окна); при нажатии на заголовок последней вкладки должна быть создана вкладка с пустой пирамидой; в рабочей части всех вкладок, кроме последней, визуализируется пирамида (способ отображения должен быть удобен и нагляден, отображение должно быть реализовано при помощи перерисовки собственного элемента управления с использованием механизма двойной диспетчеризации (паттерн “Посетитель”), отображение должно быть помещено в стилизованный на базе задания 2 элемент управления ScrollViewer с отображением элементов управления ScrollBar в случае, когда отображению не хватает места в разметке) и находятся элементы управления (TextBlock, TextBox, Button) для ввода данных, делегируемых интерфейсным методам (кроме метода слияния; вводимые данные должны быть подвергнуты валидации в командах, привязанных к элементам управления Button (CanExecute для RelayCommand или DelegateCommand). Состояние коллекции пирамид можно сериализовать в файл собственного формата (формат сериализации - бинарный) и десериализовать из файла (при этом существующая коллекция должна быть очищена перед десериализацией). Также через пункт меню можно запросить слияние текущей коллекции пирамид в одну пирамиду (последовательно к первой пирамиде подливаются остальные), которая останется единственной в коллекции.

   Предусмотрите обработку всевозможных исключительных ситуаций, которые могут произойти.

   Приложение должно быть построено на базе архитектуры MVVM с использованием функционала, реализованного в заданиях 1 и 2.

7. Разработать оконное приложение, позволяющее моделировать ход световых лучей. На форме находится источник света, из которого вертикально вниз идут лучи заданного цвета. Под источником находится рабочая область. Внутри рабочей области содержатся зеркала и призмы. Луч света, встретившись с зеркалом, отражается от него, при встрече с призмой луч преломляется и распадается на три составляющие: красную, синюю и зеленую. Внизу рабочей области находится приемник света: луч, попавший на приемник, оставляет на нем вертикальную полоску, которая по цвету совпадает с цветом луча. Все объекты на форме можно «модифицировать»: перетаскивать по форме, захватив мышкой; деформировать, растягивая за вершины призм или зеркал. У источника света можно изменять только длину и его можно включать и выключать. Все объекты не должны наезжать друг на друга и покидать границы рабочей области. При нажатии на объект правой кнопкой мыши на объекте, должен появиться диалог с настройками соответствующего объекта (при открытии диалогов в них уже должна содержаться информация о текущем состоянии выбранного объекта.):
   - Источник света:
     - компоненты цвета (RGB);
     - длина источника;
     - включен или выключен;
     - координаты левого верхнего угла;
     - кнопки «Accept» и «Decline».
   - Зеркало:
     - координаты левого и правого концов.
   - Призма:
     - коэффициенты преломления для RGB;
     - количество вершин их координаты;
     - окно предпросмотра (с возможностью деформации призмы).

   В программе обязательно должны присутствовать:
   - Меню с пунктами:
     - File:
       - New;
       - Save;
       - Load;
       - Quit (при выходе предложить сохранить изменения);
     - Objects:
       - Mirrors (появляется диалог, в котором находится список всех зеркал; при выборе определенного зеркала его состояние отображается в этом же диалоге);
       - Illuminant (появляется диалог с информацией об источнике света);

        - Призмы (появляется диалог, в котором находится список всех призм; при выборе определенной призмы, ее состоянии отображается в этом же диалоге);
    - About.
- Строка состояния, в которой отображается текущие координаты мыши, и информация, в каком объекте находится курсор мыши.

Сериализация и десериализация данных - бинарная.

Приложение должно быть построено на базе архитектуры MVVM (везде, где это представляется возможным) с использованием функционала, реализованного в заданиях 1 и 2.