Пермский филиал федерального государственного автономного образовательного учреждения высшего

 “Национальный исследовательский университет

“Высшая школа экономики”

*Факультет экономики, менеджмента и бизнес-информатики*

Дмитриев Арсений Алексеевич

Лабораторная работа №8. «Работа с файлами»

*Отчет по практической работе*

студента образовательной программы «Программная инженерия»

по направлению подготовки 09.03.04 Программная инженерия

руководитель

к. тех. н., доцент каф.

инф. техн. в биз.

Викентьева О.Л.

Пермь, 2020 г.

Постановка задачи

Общая

Написать Windows-приложение для работы с простой базой данных, хранящей информацию об объекте на внешнем носителе.  Приложение должно выполнять следующие функции:

1. Создание базы данных, содержащей записи указанного формата.
2. Просмотр базы данных.
3. Удаление элементов из базы данных (по ключу/ по номеру).
4. Корректировка элементов в базе данных (по ключу / по номеру).
5. Добавление элементов в базу данных (в начало / в конец/ с заданным номером).
6. Выполнение задания, указанного в варианте.

Вариант №7

Типизированный файл содержит данные о ежемесячных доходах подразделений фирмы «Феникс» за пять лет. Программа должна

1. Добавлять, удалять, корректировать, позволять просматривать записи файла.
2. Определять:
3. для каждого подразделения самые прибыльные годы;
4. наиболее длинный период каждого подразделения с доходом ниже среднего по всей фирме.

Анализ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Функция | Входные данные | Классы входных данных | Выходные данные / Результат |
| 1 | Добаление нового подразделения | * Название * Позиция добавления | Название - одна из литеральных констант или уже повтор уже существующего подразделения | Сообщение об ошибке |
| Позиция - целое число, больше чем число существующих записей |
| Название - корректная строка, Позиция - попадает в существующий промежуток | Запись нового подразделения |
| 2 | Удаление подразделения | Название | Название - подразделения с таким названием не существует | Сообщение об ошибке |
| Существует подразделение с таким названием | Удаление подразделения |
| Ключ (Номер записи) | Ключ - целое число, больше чем количество существующих записей | Сообщение об ошибке |
| Ключ - целое число, попадающее в диапазон существующих записей | Удаление подразделения |
| 3 | Изменение подразделения (названия) | Название | Название - одна из литеральных констант | Сообщение об ошибке |
| Название - подразделения с таким названием не существует |
| Существует подразделение с таким названием | Измененное подразделение |
| Ключ (Номер записи) | Ключ - целое число, больше чем количество существующих записей | Сообщение об ошибке |
| Ключ - целое число, попадающее в диапазон существующих записей | Удаление подразделения |
| 4 | Добавление значения месячного дохода | * Название * Индекс * Значение дохода * Позиция добавления | Название - одна из литеральных констант | Сообщение об ошибке |
| Название - подразделения с таким названием не существует |
| Индекс - целое число не попадающее в промежуток [0; 59] |
| Значение дохода - отрицательное вещественное (или целое) число или слишком большое число |
| Позиция - целое число, больше чем число существующих записей |
| * Существует подразделение с таким названием * Индекс - целое число из промежутка [0; 59] * Значение дохода - положительное вещественное или целое число попадающее в рамки типа double | Добавление значения дохода |
| 5 | Удаление значения месячного дохода | * Название * Индекс | Название - одна из литеральных констант | Сообщение об ошибке |
| Название - подразделения с таким названием не существует |
| Существует подразделение с таким названием | Удаление значения дохода |
| Индекс - целое число не попадающее в промежуток [0; 59] | Сообщение об ошибке |
| Индекс - целое число из промежутка [0; 59] | Удаление значения дохода |
| Ключ (Номер записи) | Ключ - целое число, больше чем количество существующих записей | Сообщение об ошибке |
| Ключ - целое число, попадающее в диапазон существующих записей | Удаление значения дохода |
| 6 | Изменение дохода | * Название * Индекс * Значение дохода | Название - одна из литеральных констант | Сообщение об ошибке |
| Название - подразделения с таким названием не существует |
| Существует подразделение с таким названием | Измененное значение дохода |
| Индекс - целое число не попадающее в промежуток [0; 59] | Сообщение об ошибке |
| Индекс - целое число из промежутка [0; 59] | Измененное значение дохода |
| Значение дохода - отрицательное вещественное (или целое) число или слишком большое число | Сообщение об ошибке |
| Значение дохода - положительное вещественное или целое число попадающее в рамки типа double | Измененное значение дохода |
| Ключ (Номер записи) | Ключ - целое число, больше чем количество существующих записей | Сообщение об ошибке |
| Ключ - целое число, попадающее в диапазон существующих записей | Измененное значение дохода |
| 7 | Подсчет общегодовых доходов | Доходы за 60 месяцев | Значение доходов - отрицательное вещественное (или целое) число или слишком большое число | Сообщение об ошибке |
| Значение дохода - положительное вещественное или целое число попадающее в рамки типа double | Значение суммарного дохода за каждый из 5 лет |
| Значение дохода - литеральная константа (т.е. обзначение удаленного или неприсвоенного значения) | Заначение неполного дохода за каждый год |
| 8 | Вычисление среднего по фирме дохода | Значения доходов для всех подразделений | Значение доходов - отрицательное вещественное (или целое) число или слишком большое число | Сообщение об ошибке |
| Значение дохода - положительное вещественное или целое число попадающее в рамки типа double | Значение среднего дохода по фирме |
| Значение дохода - литеральная константа (т.е. обзначение удаленного или неприсвоенного значения) | Значение среднего дохода по фирме без некоторых месяцев |

Проектирование

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Входные данные | Выходные данные/Результат |
| 1 | AddDepartment  Добавление подразделения в фирму | string name - название  int pos – позиция, на которую добавляется подразделение | Создается и добавляется новое подразделение с названием name |
| 2 | DeleteDepartment  Удаление подразделения из фирмы по названию | string name - название | Удаляется подразделение |
| 3 | DeleteDepartment  Удаление подразделения из фирмы по номеру | int key – номер записи | Удаляется подразделение |
| 4 | ChangeDepartmentName  Меняет название подразделения | int key – номер записи  string rename – новое название | Меняется название подразделения |
| 5 | ChangeDepartmentName  Меняет название подразделения | string name – прежнее название  string rename – новое название | Меняется название подразделения |
| 6 | SetIncome  Устанавливает значение дохода для определенного месяца | string name – название подразделения  int index – номер месяца  double income – значение дохода  int pos – позиция, на которую добавляется подразделение | Задается новое значение дохода в классе Department |
| 7 | DeleteIncome  Удаление значения дохода для определенного месяца | string name – прежнее название  int index – номер месяца | Удаляется запись о доходе за месяц |
| 8 | DeleteIncome  Удаление значения дохода для определенного месяца | Int key - номер записи | Удаляется запись о доходе за месяц |
| 9 | ChangeIncome  Изменение значения дохода за определенный месяц | string name – название подразделения  int index – номер месяца  double income – значение дохода | Корректируется информация о доходе за определенный месяц |
| 10 | ChangeIncome  Изменение значения дохода за определенный месяц | int key – номер записи  double income – значение дохода | Корректируется информация о доходе за определенный месяц |
| 11 | SumYearsIncomes  Подсчет общегодовых значений дохода для всего подразделения | (косвенно) double[] inc[60] – значения дохода за 60 месяцев | (косвенно) double[] yearsIncome – общегодовые доходы |
| 12 | Task2Prepare()  Вычисление среднемесячного дохода по всей фирме |  | double averageIncome – среднемесячный доход по всей фирме |

Тестирование функций

1. Добавление подразделения

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Критерии\тесты | Т1 | Т2 | Т3 | Т4 | Т5 | Т6 | Т7 | Т8 |
| Критерии входных данных: название | | | | | | | | |
| Пустая строка | + |  |  |  |  |  |  |  |
| Литеральная константа программы |  | + |  |  |  |  |  |  |
| Название уже существ. подразд. |  |  | + |  |  |  |  |  |
| Корректное название |  |  |  | + | + | + | + | + |
| Критерии входных данных: значения дохода | | | | | | | | |
| Вещественное отрицательное |  |  |  | + |  |  |  |  |
| Вещественное неотрицательное |  |  |  | + | + | + | + | + |
| Нечисло |  |  |  |  | + |  |  |  |
| Критерии входных данных: позиция | | | | | | | | |
| В начало |  |  |  |  |  | + |  |  |
| В конец |  |  |  |  |  |  | + |  |
| По значению |  |  |  |  |  |  |  | + |
| Критерии выходных данных | | | | | | | | |
| Сообщение об ошибке | + | + | + | + | + |  |  |  |
| Добавление |  |  |  |  |  | + | + | + |

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ввод | Ожидаемый результат | Результат работы |
| Т1 | Название: «» | Сообщение об ошибке | Сообщение об ошибке |
| Т2 | Название «deletedDepartment» | Сообщение об ошибке | Сообщение об ошибке |
| Т3 | (До этого создается подр. «Пермь»)  Название: «Пермь» | Сообщение об ошибке | Сообщение об ошибке |
| Т4 | Название: «Москва»  Янв 2015: -15530  Февр 2016: 133485  Май 2019: 20154 | Сообщение об ошибке | Сообщение об ошибке |
| Т5 | Название: «Питер»  Янв 2015: 3359  Апр 2017: дофига | Сообщение об ошибке | Сообщение об ошибке |
| Т6 | Название: «Екат»  Янв 2015: 37594  Февр 2016: 341112  Май 2019: 20154  Добавляется: в начало | Добавление | Добавление |
| Т7 | Название: «Пермь»  Февр 2016: 15545  Добавляется: в конец | Добавление | Добавление |
| Т8 | (До этого создается 5 подразделений)  Название: «Казань»  Янв 2015: 37594  Февр 2016: 341112  Май 2019: 20154  Добавляется: на поз. 3 | Добавление | Добавление |

1. Удаление подразделения

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Критерии\тесты | Т1 | Т2 | Т3 | Т4 | Т5 | Т6 |
| Критерии входных данных: название | | | | | | |
| Пустая строка | + |  |  |  |  |  |
| Литеральная константа программы |  | + |  |  |  |  |
| Название несуществующего подразделения |  |  | + |  |  |  |
| Корректное, существующее название |  |  |  | + |  |  |
| Критерии входных данных: ключ | | | | | | |
| Целое, в диапазоне [1; номер послед. записи] |  |  |  |  |  | + |
| Целое, вне диапазона |  |  |  |  | + |  |
| Критерии выходных данных | | | | | | |
| Сообщение об ошибке | + | + | + |  | + |  |
| Удаление подразделения |  |  |  | + |  | + |

Перед каждым тестом создаются 5 подразделений (название-номер), с любым числом записей о доходах:

Пермь-1, Москва-2, Казань-3, Екат-4, Питер-5

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ввод | Ожидаемый результат | Результат работы |
| Т1 | Название: «» | Сообщение об ошибке | Сообщение об ошибке |
| Т2 | Название: «deletedDepartment» | Сообщение об ошибке | Сообщение об ошибке |
| Т3 | Название: «Екатеринбург» | Сообщение об ошибке | Сообщение об ошибке |
| Т4 | Название: «Казань» | Удаление подр. «Казань», с ключом 3 | Удаление подр. «Казань», с ключом 3 |
| Т5 | Ключ: 7 | Сообщение об ошибке | Сообщение об ошибке |
| Т6 | Ключ: 4 | Удаление подр. с ключом 3, с названием «Екат» | Удаление подр. с ключом 3, с названием «Екат» |

1. Удаление значений дохода

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Критерии\тесты | Т1 | Т2 | Т3 | Т4 | Т5 | Т6 |
| Критерии входных данных: название подразделения | | | | | | |
| Пустая строка | + |  |  |  |  |  |
| Литеральная константа программы |  | + |  |  |  |  |
| Название несуществующего подразделения |  |  | + |  |  |  |
| Корректное, существующее название |  |  |  | + | + | + |
| Критерии входных данных: ключ | | | | | | |
| Целое, в диапазоне [1; номер послед. записи] |  |  |  |  |  | + |
| Целое, вне диапазона |  |  |  |  | + |  |
| Критерии выходных данных | | | | | | |
| Сообщение об ошибке | + | + | + |  | + |  |
| Удаление подразделения |  |  |  | + |  | + |

Перед каждым тестом создаются 3 подразделения (название:<существующие записи дохода>)

Пермь: <все 60 месяцев случайными числами> (номера от 1 до 60 подряд)

Москва- <янв 2015 – фев 2017 и апр 2019 – дек 2019> (номера от 61 до 95 подряд)

Казань: <янв 2016 – май 2017> (номера от 96 до 112)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ввод | Ожидаемый результат | Результат работы |
| Т1 | Название: «» | Сообщение об ошибке | Сообщение об ошибке |
| Т2 | Название: «deletedDepartment» | Сообщение об ошибке | Сообщение об ошибке |
| Т3 | Название: «Екатеринбург» | Сообщение об ошибке | Сообщение об ошибке |
| Т4 | Название: «Пермь»  Янв 2018 | Удаление записи о доходе подр. «Пермь» за Янв 2018 | Удаление записи о доходе подр. «Пермь» за Янв 2018 |
| Т5 | Ключ: 0 | Сообщение об ошибке | Сообщение об ошибке |
| Т6 | Ключ: 33 | Удаление записи о доходе подр. «Пермь» за Сен 2017 | Удаление записи о доходе подр. «Пермь» за Сен 2017 |

1. Корректирование записей о подразделениях

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Критерии\тесты | Т1 | Т2 | Т3 | Т4 | Т5 | Т6 | Т7 |
| Критерии входных данных: название | | | | | | | |
| Пустая строка | + |  |  |  |  |  |  |
| Название несуществующего подразделения |  | + |  |  |  |  |  |
| Корректное, существующее название |  |  | + | + | + |  |  |
| Критерии входных данных: ключ | | | | | | | |
| Целое, в диапазоне [0; номер послед. записи] |  |  |  |  |  |  | + |
| Целое, вне диапазона |  |  |  |  |  | + |  |
| Критерии входных данных: новое название | | | | | | | |
| Пустая строка |  |  | + |  |  |  |  |
| Название уже существующего подразделения |  |  |  | + |  |  |  |
| Корректное название |  |  |  |  | + | + | + |
| Критерии выходных данных | | | | | | | |
| Сообщение об ошибке | + | + | + | + |  | + |  |
| Изменение названия подразделения |  |  |  |  | + |  | + |

Перед каждым тестом создаются 5 подразделений (название-номер), с любым числом записей о доходах:

Пермь-1, Москва-2, Казань-3, Екат-4, Питер-5

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ввод | Ожидаемый результат | Результат работы |
| Т1 | Название: «» | Сообщение об ошибке | Сообщение об ошибке |
| Т2 | Название: «Владивосток» | Сообщение об ошибке | Сообщение об ошибке |
| Т3 | Название: «Москва»  Новое название: «» | Сообщение об ошибке | Сообщение об ошибке |
| Т4 | Название: «Казань»  Новое название: «Екат» | Сообщение об ошибке | Сообщение об ошибке |
| Т5 | Название: «Питер»  Новое название: «Петербург» | Подразделение «Питер» переименовано в «Петербург» | Подразделение «Питер» переименовано в «Петербург» |
| Т6 | Ключ: 9  Новое название: «Вологда» | Сообщение об ошибке | Сообщение об ошибке |
| Т7 | Ключ: 5  Новое название: «Петербург» | Подразделение «Питер» переименовано в «Петербург» | Подразделение «Питер» переименовано в «Петербург» |

1. Корректирование записей о доходах

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Критерии\тесты | Т1 | Т2 | Т3 | Т4 | Т5 | Т6 | Т7 |
| Критерии входных данных: название | | | | | | | |
| Пустая строка | + |  |  |  |  |  |  |
| Название несуществующего подразделения |  | + |  |  |  |  |  |
| Корректное, существующее название |  |  | + | + | + |  |  |
| Критерии входных данных: ключ | | | | | | | |
| Целое, в диапазоне [1; номер послед. записи] |  |  |  |  |  |  | + |
| Целое, вне диапазона |  |  |  |  |  | + |  |
| Критерии входных данных: значения дохода | | | | | | | |
| Вещественное отрицательное |  |  | + |  |  |  |  |
| Вещественное неотрицательное |  |  |  |  | + | + | + |
| Нечисло |  |  |  | + |  |  |  |
| Критерии выходных данных | | | | | | | |
| Сообщение об ошибке | + | + | + | + |  | + |  |
| Удаление подразделения |  |  |  |  | + |  | + |

Перед каждым тестом создаются 3 подразделения (название:<существующие записи дохода>)

Пермь: <все 60 месяцев случайными числами> (номера от 1 до 60 подряд)

Москва- <янв 2015 – фев 2017 и апр 2019 – дек 2019> (номера от 61 до 95 подряд)

Казань: <янв 2016 – май 2017> (номера от 96 до 112)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ввод | Ожидаемый результат | Результат работы |
| Т1 | Название: «» | Сообщение об ошибке | Сообщение об ошибке |
| Т2 | Название: «Волоколамск» | Сообщение об ошибке | Сообщение об ошибке |
| Т3 | Название: «Москва»  Янв 2016  Доход: -15,4 | Сообщение об ошибке | Сообщение об ошибке |
| Т4 | Название: «Пермь»  Сен 2018  Доход: “$” | Сообщение об ошибке | Сообщение об ошибке |
| Т5 | Название: «Казань»  Май 2016  Доход: 7954 | Изменение дохода за Май 2016 в подр. Казань на 7954 | Изменение дохода за Май 2016 в подр. Казань на 7954 |
| Т6 | Ключ: 143  Доход: 15463 | Сообщение об ошибке | Сообщение об ошибке |
| Т7 | Ключ: 107  Доход: 6694 | Изменение дохода за Дек 2016 в подр. Казань на 6694 | Изменение дохода за Дек 2016 в подр. Казань на 6694 |

1. Определение самого прибыльного года

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерии\тесты | Т1 | Т2 | Т3 | Т4 |
| Критерии входных данных: наличие хотя бы 1 записи о доходе | | | | |
| 1 и больше |  | + | + | + |
| Ни одной | + |  |  |  |
| Критерии поиска: позиция самого прибыльного года | | | | |
| первый год |  | + |  |  |
| последний год |  |  | + |  |
| 2-4 год |  |  |  | + |
| Критерии выходных данных | | | | |
| Первый год и доход = 0 | + |  |  |  |
| Номер года и доход |  | + | + | + |

Перед каждым тестом создается подразделение

Янв 2015 – Дек 2015 : каждый месяц доход {X1}

Янв 2016 – Дек 2016 : каждый месяц доход {X2}

Янв 2017 – Дек 2017 : каждый месяц доход {X3}

Янв 2018 – Дек 2018 : каждый месяц доход {X4}

Янв 2019 – Дек 2019 : каждый месяц доход {X5}

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ввод | Ожидаемый результат | Результат работы |
| Т1 | X1, X2, X3, X4, X5 – не установлены | 2015 год, доход 0 | 2015 год, доход 0 |
| Т2 | X1 = 100, X4 = 99, X5 = 39  X2, X3 – не установлены | 2015 год, доход: 1200 | 2015 год, доход: 1200 |
| Т3 | X1 = 95, X2 = 110, X3 = 45, X4 = 120, X5 – не установлено | 2018 год, доход: 1440 | 2018 год, доход: 1440 |
| Т4 | X1 = 80, X2 = 100, X3 = 100, X4 = 110, X5 = 120 | 2019 год, доход: 1440 | 2019 год, доход: 1440 |

Тестирование общее

1. Сценарий 1

|  |  |  |
| --- | --- | --- |
| № | Действие | Ожидаемое поведение |
| 1 | Создаем 2 подразделения с неполным заполнением доходов | Создалось 2 подразделения |
| 2 | Просматриваем базу | Демонстрация базы |
| 3 | Изменяем (по факту устанавливаем) доход, который ранее не был задан, запись помещаем в начало | Проиходит запись дохода |
| 4 | Изменяем (по факту устанавливаем) доход, который ранее не был задан, запись помещаем в конец | Проиходит запись дохода |
| 5 | Удаляем запись о доходе №2 | Удаляется запись о доходе из 1 созданного подразделения |
| 6 | Корректируем запись о доходе №2 | Корректируется запись о доходе из 1 созданного подразделения, следующая за удаленной |

1. Сценарий 2

|  |  |  |
| --- | --- | --- |
| № | Действие | Ожидаемое поведение |
| 1 | Создаем 2 подразделения с неполным заполнением доходов | Создалось 3 подразделения |
| 2 | Проверяем самый прибыльный год в 1 подр. | Показывается самый прибыльный год в 1 подр. |
| 3 | Проверяем периоды спада для обоих подразделений | Показываются периоды спада в подразделениях |
| 4 | Корректируем запись в подр. 1 об известном месяце в году, отличающемся от самого прибыльного, и записываем новое значение дохода равное общегодовому значению самого прибыльного года | Проиходит перезапись дохода |
| 5 | Удаляем запись о доходе в 1 подр. | Удаление записи дохода |
| 6 | Удаляем запись о доходе во 2 подр. | Удаление записи дохода |
| 7 | Проверяем самые прибыльные годы в 1 подр. | Показывается самый прибыльный год в 1 подр., он отличен от предыдущего значения из п. 2 |
| 8 | Проверяем периоды спада для обоих подразделений | Показываются периоды спада в подразделениях (они изменились) |

1. Сценарий 3

|  |  |  |
| --- | --- | --- |
| № | Действие | Ожидаемое поведение |
| 1 | Загружаем из файла данные о 4 подразделениях | Должна загрузиться информация о подразделениях |
| 2 | Корректируем названия 1 и 3 | Переименовываются 1 и 3 подр. |
| 3 | Удаляем 2 подразделение | Удаляется 2 подр. и все записи о доходе, связанные с ним |
| 4 | Удаляем большую часть записей в 4 подразделении (по ключам) | Удаляются записи о доходах у 4 подразделения |
| 5 | Проверяем базу | Должна появиться база |
| 6 | Корректируем записи во 2 подразделении (по номеру) | Должны изменится номера (т.к. до этого произошла "чистка") и изменится записи дохода |

Листинг

1. Class Firm

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab8

{

[Serializable]

public struct IncomeId

{

public string name { get; set; }

public int index { get; set; }

public IncomeId(string name, int ind)

{

this.name = name;

this.index = ind;

}

public static bool operator ==(IncomeId id1, IncomeId id2)

{

return (id1.name == id2.name && id1.index == id2.index) ? true : false;

}

public static bool operator !=(IncomeId id1, IncomeId id2)

{

return (id1.name == id2.name && id1.index == id2.index) ? false : true;

}

}

[Serializable]

public class Firm

{

public const string depToRemove = "deletedDepartment"; // Идентификатор удаленного подразделения

public const double incomeToRemove = -0.001; // Идентификатор удаленного дохода

private const int incomeIndexToRemove = -1; // Идентификатор удаленного индексатора дохода (индекс)

private const string incomeIdToRemove = "deletedIncome"; // Идентификатор удаленного индексатора дохода (название)

public string filePath = null;

#region dateVariables

public int startYear { get; private set; }

public int endYear { get; private set; }

public int startMonth { get; private set; }

public int endMonth { get; private set; }

public int yearSpan { get; private set; }

#endregion

public double averageIncome { get; private set; }

public string name { get; private set; } // Название фирмы

public string[] depNames;

private Dictionary<string, Department> deps; // Словарь отделений

private IncomeId[] incomeRecords;

private string[] departmentRecords;

public int depCount { get; private set; }

public int recDepId { get; private set; } // Нумерация записей для подразделений

public int recIncId { get; private set; } // Нумерация записей для месячных доходов

private int toBeDeletedInc; // Счетчик удаленных записей для доходов

private int toBeDeletedDep; // Счетчик удаленных подразделений

public Firm() { }

public Firm(string name, DateTime date)

{

this.name = name;

startMonth = date.Month;

endMonth = (date.Month == 1) ? 12 : date.Month - 1;

startYear = date.Year - 5;

this.endYear = (date.Month == 1) ? date.Year - 1 : date.Year;

yearSpan = endYear - startYear + 1;

recDepId = 0;

recIncId = 0;

deps = new Dictionary<string, Department>();

incomeRecords = new IncomeId[0];

departmentRecords = new string[0];

depNames = new string[0];

depCount = 0;

averageIncome = 0;

toBeDeletedInc = 0;

toBeDeletedDep = 0;

}

#region AdjustmentsToDeps

public void AddDepartment(string name, int pos)

{

Department dep = new Department(name, yearSpan, startMonth, endMonth);

deps.Add(dep.name, dep);

recDepId++;

string[] tmp = new string[recDepId];

for(int i = 0; i<pos; i++) tmp[i] = departmentRecords[i];

tmp[pos] = name;

for (int i = pos + 1; i < recDepId; i++) tmp[i] = departmentRecords[i - 1];

departmentRecords = tmp;

depNames = departmentRecords;

depCount++;

}

public void DeleteDepartment(string name)

{

for (int i = 0; i < depNames.Length; i++)

{

if (depNames[i] == name)

{

depNames[i] = depToRemove;

break;

}

}

for (int i = 0; i < recDepId; i++)

{

if (departmentRecords[i] == name)

{

departmentRecords[i] = depToRemove;

break;

}

}

for (int i = 0; i<recIncId; i++)

{

if(incomeRecords[i].name == name)

{

IncomeId id = new IncomeId(depToRemove, incomeIndexToRemove);

incomeRecords[i] = id;

toBeDeletedInc ++;

}

}

toBeDeletedDep++;

deps.Remove(name);

}

public void DeleteDepartment(int key)

{

DeleteDepartment(departmentRecords[key]);

}

public string GetDepartmentNameByKey(int key)

{

return departmentRecords[key];

}

public void ChangeDepartmentName(int key, string rename)

{

ChangeDepartmentName(departmentRecords[key], rename);

}

public void ChangeDepartmentName(string name, string rename)

{

for (int i = 0; i < depNames.Length; i++)

{

if (depNames[i] == name)

{

depNames[i] = rename;

break;

}

}

for (int i = 0; i < recDepId; i++)

{

if (departmentRecords[i] == name)

{

departmentRecords[i] = rename;

break;

}

}

deps[name].name = rename;

deps.Add(rename, deps[name]);

deps.Remove(name);

}

#endregion

#region AdjustmentsToIncome

public void SetIncome(string name, int index, double income, int pos)

{

IncomeId id = new IncomeId(name, index);

recIncId++;

IncomeId[] tmp = new IncomeId[recIncId];

for (int i = 0; i < pos; i++) tmp[i] = incomeRecords[i];

tmp[pos] = id;

for (int i = pos + 1; i < recIncId; i++) tmp[i] = incomeRecords[i - 1];

incomeRecords = tmp;

deps[name].SetIncome(income, index);

}

public void ChangeIncome(string name, int index, double income)

{

deps[name].SetIncome(income, index);

}

public void ChangeIncome(int key, double income)

{

ChangeIncome(incomeRecords[key].name, incomeRecords[key].index, income);

}

public void DeleteIncome(string name, int index)

{

IncomeId id = new IncomeId(name, index);

for(int i = 0; i<recIncId; i++)

{

if(incomeRecords[i] == id)

{

deps[name].SetIncome(incomeToRemove, index);

incomeRecords[i] = new IncomeId(incomeIdToRemove, -314);

break;

}

}

toBeDeletedInc++;

deps[name].SetIncome(double.MinValue, index);

if (toBeDeletedInc > 0.5 \* recDepId && toBeDeletedDep >= 2)

{

ClearBucket();

}

}

public void DeleteIncome(int key)

{

DeleteIncome(incomeRecords[key].name, incomeRecords[key].index);

}

public IncomeId GetInfoAboutIncome(int key)

{

return incomeRecords[key];

}

public bool WasIncomeSet(string name, int index)

{

if(deps[name].GetIncome(index) == Department.baseIncomeValue)

{

return false;

}

return true;

}

#endregion

public void Task1()

{

foreach(string name in depNames)

{

if(name != Firm.depToRemove)

{

deps[name].SumYearIncomes();

}

}

}

public void Task2Prepare()

{

double avgInc = 0;

int count = 0;

foreach (string name in depNames)

{

if (name != Firm.depToRemove)

{

count++;

deps[name].SumYearIncomes();

double inc = 0;

for(int i = 0; i<yearSpan; i++)

{

inc += deps[name].yearsIncome[i];

}

avgInc += inc / 60;

}

}

if(count > 0)

avgInc /= count;

this.averageIncome = avgInc;

}

public void Task2()

{

Task2Prepare();

foreach (string name in depNames)

{

if (name != Firm.depToRemove)

{

deps[name].avgIncome = averageIncome;

}

}

}

public Department GetDepartment(string name)

{

return deps[name];

}

private void ClearBucket()

{

int newSize = 0;

for(int i = 0; i<recIncId; i++)

{

if(incomeRecords[i].name != incomeIdToRemove)

{

newSize++;

}

}

IncomeId[] newIncId = new IncomeId[newSize];

int k = 0;

for (int i = 0; i < recIncId; i++)

{

if (incomeRecords[i].name != incomeIdToRemove)

{

newIncId[k++] = incomeRecords[i];

}

else

{

deps[incomeRecords[i].name].SetIncome(Department.baseIncomeValue, incomeRecords[i].index);

}

}

newSize = 0;

for(int i = 0; i<recDepId; i++)

{

if(departmentRecords[i] != depToRemove)

{

newSize++;

}

}

string[] newDepRecord = new string[newSize];

k = 0;

for (int i = 0; i < recDepId; i++)

{

if (departmentRecords[i] != depToRemove)

{

newDepRecord[k++] = departmentRecords[i];

}

}

Dictionary<string, Department> newDeps = new Dictionary<string, Department>();

foreach (string name in depNames)

{

if (name != depToRemove)

{

newDeps[name] = deps[name];

}

}

deps = newDeps;

departmentRecords = newDepRecord;

incomeRecords = newIncId;

depNames = newDepRecord;

recDepId = departmentRecords.Length;

recIncId = incomeRecords.Length;

depCount = recDepId;

toBeDeletedDep = 0;

toBeDeletedInc = 0;

}

}

}

1. Class Department

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab8

{

[Serializable]

public class Department

{

public string name;

public double[] inc;

public double[] yearsIncome;

public const double baseIncomeValue = double.MinValue;

private int yearSpan;

private int startMonth;

private int endMonth;

public double avgIncome { get; set; }

public Department() { }

public Department(string name, int yearSpan, int startMonth, int endMonth)

{

this.name = name;

inc = new double[60];

for (int i = 0; i < 60; i++) inc[i] = double.MinValue;

this.yearSpan = yearSpan;

this.startMonth = startMonth;

this.endMonth = endMonth;

yearsIncome = new double[yearSpan];

for (int i = 0; i < yearSpan; i++) yearsIncome[i] = 0;

}

public void SetIncome(double income, int index)

{

inc[index] = income;

}

public double GetIncome(int index)

{

return inc[index];

}

public void SumYearIncomes()

{

int year = 0;

int firstYear = 12 - startMonth + 1;

int lastYear = endMonth;

for(int i = 0; i<60; i++)

{

if(i < firstYear)

{

if(inc[i] != baseIncomeValue && inc[i] != Firm.incomeToRemove)

yearsIncome[year] += inc[i];

}

else

{

if ((i - firstYear) % 12 == 0) year++;

if (inc[i] != baseIncomeValue && inc[i] != Firm.incomeToRemove)

yearsIncome[year] += inc[i];

}

}

}

public int YearWithMaxIncome()

{

int year = 0;

double max = yearsIncome[year];

for(int i = 1; i<yearSpan; i++)

{

if(yearsIncome[i] > max)

{

max = yearsIncome[i];

year = i;

}

}

return year;

}

public double MaxIncomePerYear()

{

double max = yearsIncome[0];

for (int i = 1; i < yearSpan; i++)

{

if (yearsIncome[i] > max)

{

max = yearsIncome[i];

}

}

return max;

}

public int PeriodsWithLessIncome()

{

int max = 0;

int cur = 0;

for(int i = 0; i<60; i++)

{

if(inc[i] != Department.baseIncomeValue && inc[i] != Firm.incomeToRemove)

{

if(inc[i] < avgIncome)

{

cur++;

}

else

{

if (cur > max) max = cur;

cur = 0;

}

}

else

{

if (cur > max) max = cur;

cur = 0;

}

}

if (cur > max) max = cur;

return max;

}

}

}

1. Form MainForm

using System;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

using System.Windows.Forms;

namespace Lab8

{

public partial class MainForm : Form

{

private Firm phx;

public MainForm()

{

InitializeComponent();

DateTime date = DateTime.Today;

phx = new Firm("Феникс", date);

}

private void dataAccesButton\_Click(object sender, EventArgs e)

{

bdShow bds = new bdShow(phx);

bds.ShowDialog();

}

private void profitsYearsButton\_Click(object sender, EventArgs e)

{

TaskShow task = new TaskShow(phx);

task.ShowDialog();

}

private void downwardsYearsButton\_Click(object sender, EventArgs e)

{

Task2Form task = new Task2Form(phx);

task.ShowDialog();

}

private void addDepartamentButton\_Click(object sender, EventArgs e)

{

AddDepForm adf = new AddDepForm(phx);

adf.ShowDialog();

if (adf.DialogResult == DialogResult.OK)

{

phx = adf.GetUpdatedFirm();

}

adf.Dispose();

}

private void deletionButton\_Click(object sender, EventArgs e)

{

DeletionForm delF = new DeletionForm(phx);

delF.ShowDialog();

if(delF.DialogResult == DialogResult.OK)

{

phx = delF.GetUpdatedFirm();

}

}

private void changeButton\_Click(object sender, EventArgs e)

{

CorrectionForm crf = new CorrectionForm(phx);

crf.ShowDialog();

if(crf.DialogResult == DialogResult.OK)

{

phx = crf.GetUpdatedFirm();

}

}

private void saveButton\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "dat files (\*.dat)|\*.dat";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

string filename = saveFileDialog.FileName;

BinaryFormatter formatter = new BinaryFormatter();

using (FileStream fs = new FileStream(filename, FileMode.OpenOrCreate))

{

formatter.Serialize(fs, phx);

MessageBox.Show("Данные сериализованы", "", MessageBoxButtons.OK);

}

phx.filePath = filename;

}

}

private void loadButton\_Click(object sender, EventArgs e)

{

using (OpenFileDialog openFileDialog = new OpenFileDialog())

{

openFileDialog.InitialDirectory = "c:\\";

openFileDialog.Filter = "binary files (\*.dat)|\*.dat";

openFileDialog.FilterIndex = 1;

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

string filePath = openFileDialog.FileName;

BinaryFormatter formatter = new BinaryFormatter();

using (FileStream fs = new FileStream(filePath, FileMode.OpenOrCreate))

{

phx = (Firm)formatter.Deserialize(fs);

MessageBox.Show("Данные загружены", "", MessageBoxButtons.OK);

}

phx.filePath = filePath;

}

catch

{

MessageBox.Show("Ошибка чтения!", "", MessageBoxButtons.OK);

}

}

}

}

}

}

1. Form AddDepForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab8

{

public partial class AddDepForm : Form

{

private Firm phx;

public string[] monthsNames = { "", "Январь", "Февраль", "Март", "Апрель", "Май", "Июнь", "Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь", "Декабрь" };

private int positionToAdd;

public AddDepForm(Firm phx)

{

InitializeComponent();

this.phx = phx;

}

private void AddDepForm\_Load(object sender, EventArgs e)

{

InitIncomeTable();

endRadioButton.Checked = true;

}

public void InitIncomeTable()

{

int yearsAmount = phx.endYear - phx.startYear + 1;

// Оформление "Шапки"

string[] years = { "", "", "", "", "", "", "" };

for (int i = 1; i <= phx.endYear - phx.startYear + 1; i++)

{

years[i] = (phx.startYear + i - 1).ToString();

}

string[] months = { "", "Янв", "Февр", "Март", "Апр", "Май", "Июнь", "Июль", "Авг", "Сен", "Окт", "Ноя", "Дек" };

fontLabel.Hide();

// Годы

for (int i = 0; i < years.Length; i++)

{

Label lab = new Label();

lab.Text = years[i];

lab.Font = fontLabel.Font;

lab.Anchor = AnchorStyles.None;

profitShowTable.Controls.Add(lab, i, 0);

}

//Месяцы

for (int i = 1; i < months.Length; i++)

{

Label lab = new Label();

lab.Text = months[i];

lab.Font = fontLabel.Font;

lab.Anchor = AnchorStyles.Left;

profitShowTable.Controls.Add(lab, 0, i);

}

// Шапка оформлена

// Заполнение таблицы текстбоксами для ввода доходов

// Заполнение "неактивные" месяцы

for(int i = 1; i<phx.startMonth; i++)

{

TextBox tb = new TextBox();

tb.Anchor = AnchorStyles.None;

tb.ReadOnly = true;

tb.Text = "-";

profitShowTable.Controls.Add(tb, 1, i);

}

for(int i = phx.endMonth + 1; i<=12; i++)

{

TextBox tb = new TextBox();

tb.Anchor = AnchorStyles.None;

tb.ReadOnly = true;

tb.Text = "-";

profitShowTable.Controls.Add(tb, yearsAmount, i);

}

// Заполнение активных месяцев

int cur\_month = phx.startMonth;

int cur\_year = 1;

for(int i = 1; i<=60; i++)

{

TextBox tb = new TextBox();

tb.Anchor = AnchorStyles.None;

tb.Text = "-";

tb.Name = $"TB {cur\_month}-{cur\_year}";

profitShowTable.Controls.Add(tb, cur\_year, cur\_month);

cur\_month++;

if(cur\_month == 13)

{

cur\_month = 1;

cur\_year++;

}

}

}

public void InitNewAddition()

{

departName.Text = "";

endRadioButton.Checked = true;

int cur\_month = phx.startMonth;

int cur\_year = 1;

for (int i = 1; i <= 60; i++)

{

Control[] cntrs = profitShowTable.Controls.Find($"TB {cur\_month}-{cur\_year}", true);

if (cntrs.Length == 1)

{

TextBox tb = (TextBox)cntrs[0];

tb.Text = "-";

}

cur\_month++;

if (cur\_month == 13)

{

cur\_month = 1;

cur\_year++;

}

}

}

private bool CheckInputs()

{

string errorMsg = "";

bool ok = true;

// Проверка введенного названия отделения

if(departName.Text == "")

{

errorMsg = "Введите название нового отделения!";

ok = false;

}

else if(departName.Text == Firm.depToRemove)

{

errorMsg = "Такое название недопустимо!";

ok = false;

}

else

{

foreach (string name in phx.depNames)

{

if (name == departName.Text)

{

errorMsg = "Отделение уже существует!\nДля его редактирования используйте раздел \"Редактирование\"";

ok = false;

break;

}

}

}

if (!ok)

{

MessageBox.Show(errorMsg, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

// Проверка введенных значений дохода

int cur\_month = phx.startMonth;

int cur\_year = 1;

for (int i = 0; i < 60 && ok; i++)

{

double income = 0;

Control[] cntrs = profitShowTable.Controls.Find($"TB {cur\_month}-{cur\_year}", true);

if (cntrs.Length == 1)

{

TextBox tb = (TextBox)cntrs[0];

try

{

income = double.Parse(tb.Text);

}

catch

{

if (tb.Text != "-")

{

errorMsg = $"Данные за {monthsNames[cur\_month]} {cur\_year + phx.startYear - 1} введены не верно";

ok = false;

}

}

}

if(income < 0)

{

ok = false;

errorMsg = "Доход - величина положительная";

}

cur\_month++;

if (cur\_month == 13)

{

cur\_month = 1;

cur\_year++;

}

}

if (!ok)

{

MessageBox.Show(errorMsg, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

// Проверка введенной позиции

if (keyRadioButton.Checked)

{

int key = -1;

try

{

key = int.Parse(keyTextBox.Text);

}

catch

{

errorMsg = "Некорректно введен номер";

ok = false;

}

if(ok)

{

if(key > phx.recDepId + 1 || key == 0)

{

errorMsg = "Введенный номер не поддерживается";

ok = false;

}

else

{

positionToAdd = key;

}

}

}

if (!ok)

{

MessageBox.Show(errorMsg, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

return true;

}

private void okButton\_Click(object sender, EventArgs e)

{

if (CheckInputs())

{

string name = departName.Text;

phx.AddDepartment(name, positionToAdd - 1);

int cur\_month = phx.startMonth;

int cur\_year = 1;

for (int i = 0; i < 60; i++)

{

double income = 0;

bool ok = false;

Control[] cntrs = profitShowTable.Controls.Find($"TB {cur\_month}-{cur\_year}", true);

if(cntrs.Length == 1)

{

TextBox tb = (TextBox)cntrs[0];

if (tb.Text != "-")

{

ok = true;

income = double.Parse(tb.Text);

}

}

if (ok)

phx.SetIncome(name, i, income, phx.recIncId);

cur\_month++;

if (cur\_month == 13)

{

cur\_month = 1;

cur\_year++;

}

}

InitNewAddition();

MessageBox.Show($"Подразделение {name} успешно добавлено", "", MessageBoxButtons.OK);

DialogResult = DialogResult.None;

}

}

public Firm GetUpdatedFirm()

{

return phx;

}

private void cancelButton\_Click(object sender, EventArgs e)

{

DialogResult = DialogResult.OK;

}

private void beginRadioButton\_CheckedChanged(object sender, EventArgs e)

{

positionToAdd = 1;

keyTextBox.Enabled = false;

}

private void endRadioButton\_CheckedChanged(object sender, EventArgs e)

{

positionToAdd = phx.recDepId + 1;

keyTextBox.Enabled = false;

}

private void keyRadioButton\_CheckedChanged(object sender, EventArgs e)

{

keyTextBox.Enabled = true;

}

private void keyTextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!Regex.IsMatch(e.KeyChar.ToString(), @"\d") && e.KeyChar != 8)

{

e.Handled = true;

}

}

}

}

1. Form DeletionForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab8

{

public partial class DeletionForm : Form

{

private Firm phx;

private int delModeDepart;

private int delModeMonth;

public DeletionForm(Firm phx)

{

InitializeComponent();

this.phx = phx;

}

private void DeletionForm\_Load(object sender, EventArgs e)

{

nameRadioButton.Checked = true;

keyRadioButton.Checked = true;

yearUpDown.Maximum = phx.endYear;

yearUpDown.Minimum = phx.startYear;

}

private void cancelButton\_Click(object sender, EventArgs e)

{

DialogResult = DialogResult.OK;

}

public Firm GetUpdatedFirm()

{

return phx;

}

#region departmentPage

private void ClearDepsInputs()

{

nameTextBox.Text = "";

nameRadioButton.Checked = true;

}

private void deleteButton\_Click(object sender, EventArgs e)

{

if(delModeDepart == 0)

{

bool ok = false;

// Проверка введенного названия на корректность

foreach (string name in phx.depNames)

{

if (name == nameTextBox.Text)

{

ok = true;

break;

}

}

if (ok)

{

phx.DeleteDepartment(nameTextBox.Text);

MessageBox.Show($"Отделение {nameTextBox.Text} успешно удалено", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show($"Отделения {nameTextBox.Text} не найдено\nНичего не было удалено", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

bool ok = true;

string errorMsg = "";

int key = 0;

// Проверка введенного ключа на корректность

try

{

key = int.Parse(nameTextBox.Text);

}

catch

{

ok = false;

errorMsg = "Некорректно введен номер";

}

if (ok)

{

if(key > phx.recDepId || key == 0)

{

ok = false;

errorMsg = "Запись с таким номером не существует";

}

}

if (ok)

{

string name = phx.GetDepartmentNameByKey(key-1);

phx.DeleteDepartment(key-1);

MessageBox.Show($"Отделение {name} успешно удалено", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show(errorMsg, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

DialogResult = DialogResult.None;

ClearDepsInputs();

}

private void nameRadioButton\_CheckedChanged(object sender, EventArgs e)

{

delModeDepart = 0;

}

private void numberRadioButton\_CheckedChanged(object sender, EventArgs e)

{

delModeDepart = 1;

}

private void nameTextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (delModeDepart == 1)

{

if (!Regex.IsMatch(e.KeyChar.ToString(), @"\d") && e.KeyChar != 8)

{

e.Handled = true;

}

}

}

#endregion

#region monthPage

private void keyRadioButton\_CheckedChanged(object sender, EventArgs e)

{

delModeMonth = 0;

monthUpDown.Enabled = true;

yearUpDown.Enabled = true;

}

private void numRadioButton\_CheckedChanged(object sender, EventArgs e)

{

delModeMonth = 1;

monthUpDown.Enabled = false;

yearUpDown.Enabled = false;

}

private void yearUpDown\_ValueChanged(object sender, EventArgs e)

{

if (phx.yearSpan == 6) {

if (yearUpDown.Value == phx.startYear)

{

monthUpDown.Minimum = phx.startMonth;

monthUpDown.Maximum = 12;

monthUpDown.Value = monthUpDown.Minimum;

}

else if (yearUpDown.Value == phx.endYear)

{

monthUpDown.Minimum = 1;

monthUpDown.Maximum = phx.endMonth;

monthUpDown.Value = monthUpDown.Minimum;

} else

{

monthUpDown.Minimum = 1;

monthUpDown.Maximum = 12;

}

}

}

private void deletionButton\_Click(object sender, EventArgs e)

{

if(delModeMonth == 0)

{

// Проверка ввода названия на правильность

string errorMsg = "";

bool ok = false;

foreach (string name in phx.depNames)

{

if (name == keyTextBox.Text)

{

ok = true;

break;

}

}

if(!ok)

{

errorMsg = $"Отделения {nameTextBox.Text} не найдено\nНичего не было удалено";

MessageBox.Show(errorMsg, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Нахождение индекса месяца

int index = 0;

int month = Decimal.ToInt32(monthUpDown.Value);

int year = Decimal.ToInt32(yearUpDown.Value);

int cur\_month = phx.startMonth;

int cur\_year = phx.startYear;

for (int i = 0; i < 60; i++)

{

if(cur\_month == month && cur\_year == year)

{

index = i;

break;

}

cur\_month++;

if (cur\_month == 13)

{

cur\_month = 1;

cur\_year++;

}

}

if(phx.WasIncomeSet(keyTextBox.Text, index))

{

MessageBox.Show($"Значение дохода за {cur\_month} месяц {cur\_year} год удалено", "", MessageBoxButtons.OK);

phx.DeleteIncome(keyTextBox.Text, index);

} else

{

MessageBox.Show("Нельзя удалить запись, которой не существует", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

string errorMsg = "";

bool ok = true;

int key = 0;

// Проверка ввода ключа

try

{

key = int.Parse(keyTextBox.Text);

}

catch

{

ok = false;

errorMsg = "Номер введен некорректно";

}

if(ok)

{

if(key > phx.recIncId || key == 0)

{

ok = false;

errorMsg = "Записи с таким номером не существует";

}

}

if (!ok)

{

MessageBox.Show(errorMsg, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

IncomeId id = phx.GetInfoAboutIncome(key-1);

int cur\_month = phx.startMonth, cur\_year = phx.startYear;

for (int i = 0; i <= id.index; i++)

{

cur\_month++;

if (cur\_month == 13)

{

cur\_month = 1;

cur\_year++;

}

}

if (phx.WasIncomeSet(id.name, id.index))

{

MessageBox.Show($"Значение дохода за {cur\_month} месяц {cur\_year} года в подразделении {id.name} удалено", "", MessageBoxButtons.OK);

phx.DeleteIncome(key - 1);

} else

{

MessageBox.Show("Нельзя удалить запись, которой не существует", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

DialogResult = DialogResult.None;

ClearIncInputs();

}

public void ClearIncInputs()

{

keyRadioButton.Checked = true;

keyTextBox.Text = "";

}

private void keyTextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (delModeMonth == 1)

{

if (!Regex.IsMatch(e.KeyChar.ToString(), @"\d") && e.KeyChar != 8)

{

e.Handled = true;

}

}

}

#endregion

}

}

1. Form CorrectionForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab8

{

public partial class CorrectionForm : Form

{

private Firm phx;

private int chgModeDepart;

private int chgModeMonth;

private int positionChange;

public CorrectionForm(Firm phx)

{

InitializeComponent();

this.phx = phx;

positionChange = -1;

}

private void CorrectionForm\_Load(object sender, EventArgs e)

{

nameRadioButton.Checked = true;

keyRadioButton.Checked = true;

yearUpDown.Maximum = phx.endYear;

yearUpDown.Minimum = phx.startYear;

}

#region departmentPage

private void changeButton\_Click(object sender, EventArgs e)

{

if (chgModeDepart == 0)

{

string errorMsg = "";

bool ok = false;

bool fl = true;

foreach (string name in phx.depNames)

{

if (name == nameTextBox.Text)

{

ok = true;

}

if(name == newNameTextBox.Text)

{

fl = false;

errorMsg = $"Отделение с названием {newNameTextBox.Text} уже существует";

}

}

if (newNameTextBox.Text == "")

{

fl = false;

errorMsg = "Название не может быть пустой строкой";

} else if (newNameTextBox.Text == Firm.depToRemove)

{

fl = false;

errorMsg = "Некорректное новое название";

}

if (!ok || !fl)

{

if (!ok) errorMsg = $"Отделения {nameTextBox.Text} не найдено\nНичего не было удалено";

MessageBox.Show(errorMsg, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

phx.ChangeDepartmentName(nameTextBox.Text, newNameTextBox.Text);

MessageBox.Show($"Название подразделения {nameTextBox.Text} изменено на {newNameTextBox.Text}", "", MessageBoxButtons.OK);

}

else

{

bool ok = true;

string errorMsg = "";

int key = 0;

bool fl = true;

foreach (string name in phx.depNames)

{

if (name == newNameTextBox.Text)

{

fl = false;

errorMsg = $"Отделение с названием {newNameTextBox.Text} уже существует";

}

}

try

{

key = int.Parse(nameTextBox.Text);

}

catch

{

ok = false;

errorMsg = "Некорректно введен номер";

}

if (ok)

{

if (key > phx.depCount || key == 0)

{

ok = false;

errorMsg = "Записи с таким номером не существует";

}

}

if (!ok || !fl)

{

MessageBox.Show(errorMsg, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

phx.ChangeDepartmentName(key-1, newNameTextBox.Text);

MessageBox.Show($"Отделение успешно переименовано", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

DialogResult = DialogResult.None;

}

private void nameRadioButton\_CheckedChanged(object sender, EventArgs e)

{

chgModeDepart = 0;

}

private void numberRadioButton\_CheckedChanged(object sender, EventArgs e)

{

chgModeDepart = 1;

}

private void nameTextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (chgModeDepart == 1)

{

if (!Regex.IsMatch(e.KeyChar.ToString(), @"\d") && e.KeyChar != 8)

{

e.Handled = true;

}

}

}

#endregion

#region incomePage

private void correctionButton\_Click(object sender, EventArgs e)

{

if(chgModeMonth == 0)

{

string errorMsg = "";

bool ok = true;

foreach (string name in phx.depNames)

{

if (name == keyTextBox.Text)

{

ok = true;

break;

}

}

if (!ok)

{

errorMsg = $"Отделения {nameTextBox.Text} не найдено\nНичего не было изменено";

MessageBox.Show(errorMsg, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

double income = 0d;

try

{

income = double.Parse(incomeTextBox.Text);

}

catch

{

errorMsg = "Новое значение дохода введено некорректно";

ok = false;

}

if(!ok)

{

MessageBox.Show(errorMsg, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

int index = 0;

int month = Decimal.ToInt32(monthUpDown.Value);

int year = Decimal.ToInt32(yearUpDown.Value);

int cur\_month = phx.startMonth;

int cur\_year = phx.startYear;

for (int i = 0; i < 60; i++)

{

if (cur\_month == month && cur\_year == year)

{

index = i;

break;

}

cur\_month++;

if (cur\_month == 13)

{

cur\_month = 1;

cur\_year++;

}

}

if(phx.WasIncomeSet(keyTextBox.Text, index))

{

phx.ChangeIncome(keyTextBox.Text, index, income);

}

else

{

PositionForm pos = new PositionForm(phx);

pos.ShowDialog();

if(pos.DialogResult == DialogResult.OK)

{

positionChange = pos.positionChange;

}

phx.SetIncome(keyTextBox.Text, index, income, positionChange - 1);

}

}

else

{

string errorMsg = "";

bool ok = true;

int key = 0;

try

{

key = int.Parse(keyTextBox.Text);

}

catch

{

ok = false;

errorMsg = "Номер введен некорректно";

}

if (ok)

{

if (key > phx.recIncId || key == 0)

{

ok = false;

errorMsg = "Записи с таким номером не существует";

}

}

double income = 0d;

try

{

income = double.Parse(incomeTextBox.Text);

}

catch

{

errorMsg = "Новое значение дохода введено некорректно";

ok = false;

}

if (!ok)

{

MessageBox.Show(errorMsg, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

phx.ChangeIncome(key-1, income);

}

DialogResult = DialogResult.None;

}

private void keyRadioButton\_CheckedChanged(object sender, EventArgs e)

{

chgModeMonth = 0;

monthUpDown.Enabled = true;

yearUpDown.Enabled = true;

}

private void numRadioButton\_CheckedChanged(object sender, EventArgs e)

{

chgModeMonth = 1;

monthUpDown.Enabled = false;

yearUpDown.Enabled = false;

}

private void keyTextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (chgModeMonth == 1)

{

if (!Regex.IsMatch(e.KeyChar.ToString(), @"\d") && e.KeyChar != 8)

{

e.Handled = true;

}

}

}

private void yearUpDown\_ValueChanged(object sender, EventArgs e)

{

if (phx.yearSpan == 6)

{

if (yearUpDown.Value == phx.startYear)

{

monthUpDown.Minimum = phx.startMonth;

monthUpDown.Maximum = 12;

monthUpDown.Value = monthUpDown.Minimum;

}

else if (yearUpDown.Value == phx.endYear)

{

monthUpDown.Minimum = 1;

monthUpDown.Maximum = phx.endMonth;

monthUpDown.Value = monthUpDown.Minimum;

}

else

{

monthUpDown.Minimum = 1;

monthUpDown.Maximum = 12;

}

}

}

private void incomeTextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

}

#endregion

public Firm GetUpdatedFirm()

{

return phx;

}

private void cancelButton\_Click(object sender, EventArgs e)

{

DialogResult = DialogResult.OK;

}

}

}

1. Form ShowData

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab8

{

public partial class ShowData : Form

{

private string[] years = { "", "2015", "2016", "2017", "2018", "2019" };

private string[] months = { "", "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };

private Firm phx;

public ShowData(Firm phx)

{

InitializeComponent();

this.phx = phx;

}

private void bdShow\_Load(object sender, EventArgs e)

{

Label head = new Label();

head.Text = "Подразделение";

showTable.Controls.Add(head, 0, 0);

Label head1 = new Label();

head1.Text = "Доходы";

showTable.Controls.Add(head1, 1, 0);

int depCount = 0;

foreach (string dep in phx.depNames)

{

if (dep != Firm.depToRemove) depCount++;

}

RowStyle tmp = showTable.RowStyles[1];

showTable.RowCount = depCount + 1;

showTable.Size = new Size(910, 40 + depCount \* 510);

for (int i = 2; i < showTable.ColumnCount; i++)

{

showTable.RowStyles.Insert(i, new RowStyle(tmp.SizeType, tmp.Height));

}

int k = 1;

for(int i = 0; i < phx.depNames.Length; i++)

{

if(phx.depNames[i] != Firm.depToRemove)

{

FillRow(phx.depNames[i], k++, phx.GetDepartment(phx.depNames[i]));

}

}

}

public void FillRow(string name, int ind, Department dep)

{

Label label = new Label();

label.Text = $"{name}";

showTable.Controls.Add(label, 0, ind);

TableLayoutPanel table = SetSampleTableSettings(sampleTable);

table.Name = $"TLP {showTable.RowCount - 1}";

table.Visible = true;

Random rand = new Random();

int index = 0;

for (int j = 1; j < 13; j++)

{

if (j < phx.startMonth)

{

Label lab = new Label();

lab.Anchor = AnchorStyles.None;

lab.Text = "-";

table.Controls.Add(lab, 1, j);

}

else

{

Label lab = new Label();

lab.Anchor = AnchorStyles.None;

if (dep.GetIncome(index) == Department.baseIncomeValue || dep.GetIncome(index) == Firm.incomeToRemove)

lab.Text = "TBD";

else

lab.Text = string.Format("{0:f2}", dep.GetIncome(index));

table.Controls.Add(lab, 1, j);

index++;

}

}

for (int i = 2; i<phx.yearSpan; i++)

{

for(int j = 1; j<13; j++)

{

Label lab = new Label();

lab.Anchor = AnchorStyles.None;

if (dep.GetIncome(index) == Department.baseIncomeValue || dep.GetIncome(index) == Firm.incomeToRemove)

lab.Text = "TBD";

else

lab.Text = string.Format("{0:f2}", dep.GetIncome(index));

table.Controls.Add(lab, i, j);

index++;

}

}

for (int j = 1; j < 13; j++)

{

if (j <= phx.endMonth)

{

Label lab = new Label();

lab.Anchor = AnchorStyles.None;

if (dep.GetIncome(index) == Department.baseIncomeValue || dep.GetIncome(index) == Firm.incomeToRemove)

lab.Text = "TBD";

else

lab.Text = string.Format("{0:f2}", dep.GetIncome(index));

table.Controls.Add(lab, phx.yearSpan, j);

index++;

}

else

{

Label lab = new Label();

lab.Anchor = AnchorStyles.None;

lab.Text = "-";

table.Controls.Add(lab, phx.yearSpan, j);

}

}

showTable.Controls.Add(table, 1, ind);

}

private TableLayoutPanel SetSampleTableSettings(TableLayoutPanel copied)

{

TableLayoutPanel table = new TableLayoutPanel();

table.ColumnCount = copied.ColumnCount;

table.ColumnStyles.Clear();

for (int i = 0; i < table.ColumnCount; i++)

{

table.ColumnStyles.Insert(i, new ColumnStyle(copied.ColumnStyles[i].SizeType, copied.ColumnStyles[i].Width));

}

table.RowCount = copied.RowCount;

for (int i = 0; i < table.RowCount; i++)

{

table.RowStyles.Insert(i, new RowStyle(copied.RowStyles[i].SizeType, copied.RowStyles[i].Height));

}

table.Size = new Size(copied.Width, copied.Height);

table.Visible = true;

for (int i = 0; i < years.Length; i++)

{

Label lab = new Label();

lab.Text = years[i];

lab.Font = fontLabel.Font;

lab.Anchor = AnchorStyles.None;

table.Controls.Add(lab, i, 0);

}

for (int i = 1; i < months.Length; i++)

{

Label lab = new Label();

lab.Text = months[i];

lab.Font = fontLabel.Font;

lab.Anchor = AnchorStyles.Left;

table.Controls.Add(lab, 0, i);

}

return table;

}

}

}

1. Form Task1Show

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab8

{

public partial class TaskShow : Form

{

private Firm phx;

public TaskShow(Firm phx)

{

InitializeComponent();

this.phx = phx;

}

private void TaskShow\_Load(object sender, EventArgs e)

{

Label head = new Label();

head.Text = "Подразделение";

showTable.Controls.Add(head, 0, 0);

Label head1 = new Label();

head1.Text = "Доход";

showTable.Controls.Add(head1, 1, 0);

int depCount = 0;

foreach (string dep in phx.depNames)

{

if (dep != Firm.depToRemove) depCount++;

}

RowStyle tmp = showTable.RowStyles[1];

showTable.RowCount = depCount + 1;

showTable.Size = new Size(1200, 40 + depCount \* 40);

for (int i = 2; i < showTable.ColumnCount; i++)

{

showTable.RowStyles.Insert(i, new RowStyle(tmp.SizeType, tmp.Height));

}

phx.Task1();

int k = 1;

for (int i = 0; i < phx.depNames.Length; i++)

{

if (phx.depNames[i] != Firm.depToRemove)

{

FillRow(phx.depNames[i], k++, phx.GetDepartment(phx.depNames[i]));

}

}

}

public void FillRow(string name, int ind, Department dep)

{

Label label = new Label();

label.Text = $"{name}";

showTable.Controls.Add(label, 0, ind);

Label label1 = new Label();

label1.Anchor = (AnchorStyles.Left | AnchorStyles.Right | AnchorStyles.Top);

label1.Text = $"Самый прибыльный год: {dep.YearWithMaxIncome() + phx.startYear}, с доходом: {dep.MaxIncomePerYear()}";

showTable.Controls.Add(label1, 1, ind);

}

}

}

1. Form Task2Show

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab8

{

public partial class Task2Show : Form

{

private Firm phx;

public Task2Show(Firm phx)

{

InitializeComponent();

this.phx = phx;

}

private void Task2Form\_Load(object sender, EventArgs e)

{

Label head = new Label();

head.Text = "Подразделение";

showTable.Controls.Add(head, 0, 0);

Label head1 = new Label();

head1.Text = $"Периоды"; //; Средний доход по фирме: {phx.averageIncome}

head1.Anchor = (AnchorStyles.Left | AnchorStyles.Right);

showTable.Controls.Add(head1, 1, 0);

int depCount = 0;

foreach (string dep in phx.depNames)

{

if (dep != Firm.depToRemove) depCount++;

}

RowStyle tmp = showTable.RowStyles[1];

showTable.RowCount = depCount + 1;

showTable.Size = new Size(1200, 40 + depCount \* 40);

for (int i = 2; i < showTable.ColumnCount; i++)

{

showTable.RowStyles.Insert(i, new RowStyle(tmp.SizeType, tmp.Height));

}

phx.Task2();

int k = 1;

for (int i = 0; i < phx.depNames.Length; i++)

{

if (phx.depNames[i] != Firm.depToRemove)

{

FillRow(phx.depNames[i], k++, phx.GetDepartment(phx.depNames[i]));

}

}

}

public void FillRow(string name, int ind, Department dep)

{

Label label = new Label();

label.Text = $"{name}";

showTable.Controls.Add(label, 0, ind);

Label label1 = new Label();

label1.Anchor = (AnchorStyles.Left | AnchorStyles.Right | AnchorStyles.Top);

label1.Text = $"Самый большой период, с доходом меньше среднего (в месяцах): {dep.PeriodsWithLessIncome()}";

showTable.Controls.Add(label1, 1, ind);

}

}

}

1. Form PositionForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab8

{

public partial class PositionForm : Form

{

public int positionChange { get; private set; }

public Firm phx;

public PositionForm(Firm phx)

{

InitializeComponent();

this.phx = phx;

beginRadioButton.Checked = true;

}

private void beginRadioButton\_CheckedChanged(object sender, EventArgs e)

{

positionChange = 1;

keyTextBox.Enabled = false;

}

private void endRadioButton\_CheckedChanged(object sender, EventArgs e)

{

positionChange = phx.recIncId + 1;

keyTextBox.Enabled = false;

}

private void keyRadioButton\_CheckedChanged(object sender, EventArgs e)

{

keyTextBox.Enabled = true;

}

private void keyTextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!Regex.IsMatch(e.KeyChar.ToString(), @"\d") && e.KeyChar != 8)

{

e.Handled = true;

}

}

private void okButton\_Click(object sender, EventArgs e)

{

if (keyRadioButton.Checked)

{

string errorMsg = "";

bool ok = true;

int key = -1;

try

{

key = int.Parse(keyTextBox.Text);

}

catch

{

errorMsg = "Некорректно введен номер";

ok = false;

}

if (ok)

{

if (key > phx.recIncId || key == 0)

{

errorMsg = "Введенный номер не поддерживается";

ok = false;

}

else

{

positionChange = key;

}

}

if (ok)

{

DialogResult = DialogResult.OK;

}

else

{

DialogResult = DialogResult.None;

MessageBox.Show(errorMsg, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

DialogResult = DialogResult.OK;

}

}

}

}