Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ОТЧЕТ

по лабораторной работе № 11

Тема работы: Записи. Типизированные файлы

Выполнил

студент: гр.551004 Довыдёнок М.А.

Проверил: Фадеева Е.П.

Минск 2016

Содержание

[1 Постановка задачи 4](#_Toc450689235)

[2 Структура модулей 5](#_Toc450689236)

[3 Описание алгоритмов 6](#_Toc450689237)

[3.1 Описание алгоритмов модуля UMain 6](#_Toc450689238)

[3.2 Описание алгоритмов модуля UBookList 6](#_Toc450689239)

[3.3 Описание алгоритмов модуля UVisitorList 7](#_Toc450689240)

[3.4 Описание алгоритмов модуля UBorrowList 8](#_Toc450689241)

[3.5 Описание алгоритмов модуля UFilesIntegration 9](#_Toc450689242)

[4 Структура типов 11](#_Toc450689243)

[4.1 Структура типов модуля UMain 11](#_Toc450689244)

[4.2 Структура типов модуля UBookList 11](#_Toc450689245)

[4.3 Структура типов модуля UFilesIntegration 11](#_Toc450689246)

[4.4 Структура типов модуля UVisitorList 12](#_Toc450689247)

[4.5 Структура типов модуля UBorrowList 12](#_Toc450689248)

[5 Структура данных 13](#_Toc450689249)

[5.1 Структура данных модуля UMain 13](#_Toc450689250)

[5.2 Структура данных модуля UFilesIntegration 13](#_Toc450689251)

[5.3 Структура данных модуля UBookList, UVisitorList, UBorrowList 14](#_Toc450689252)

[6 Схема алгоритма решения задачи по ГОСТ 19.701-90 15](#_Toc450689253)

[6.1 Схема алгоритма showBooksList 15](#_Toc450689254)

[6.2 Схема алгоритма showList 15](#_Toc450689255)

[6.3 Схема алгоритма addBook(book) 16](#_Toc450689256)

[6.4 Схема алгоритма deleteByCode(code) 17](#_Toc450689257)

[7 Результаты расчетов и тестирование программы 18](#_Toc450689258)

[7.1 Группа тестов вывода списков 18](#_Toc450689259)

[7.1.1 Тест 1.1 18](#_Toc450689260)

[7.1.1 Тест 1.2 18](#_Toc450689261)

[7.1.1 Тест 1.3 19](#_Toc450689262)

[7.1.1 Тест 1.4 20](#_Toc450689263)

[7.1.1 Тест 1.5 20](#_Toc450689264)

[7.1.1 Тест 1.6 21](#_Toc450689265)

[7.1.1 Тест 1.7 22](#_Toc450689266)

[7.1.1 Тест 1.8 22](#_Toc450689267)

[7.1.1 Тест 1.9 23](#_Toc450689268)

[7.1.1 Тест 1.10 24](#_Toc450689269)

[7.1.1 Тест 1.11 24](#_Toc450689270)

[7.1.1 Тест 1.12 25](#_Toc450689271)

[7.2 Группа тестов изменения данных в файлах 26](#_Toc450689272)

[7.2.1 Тест 2.1 26](#_Toc450689273)

[7.2.1 Тест 2.2 26](#_Toc450689274)

[7.2.1 Тест 2.3 27](#_Toc450689275)

[Приложение А 28](#_Toc450689276)

# Постановка задачи

В библиотеке имеется список книг. Каждая запись списка содержит: код книги, фамилию автора, название книги, год издания, язык издания. Также имеется список читателей, каждая запись которого содержит: код читателя, ФИО читателя, домашний адрес, контактный телефон. Книги, взятые каждым читателем, заносятся в отдельный список, содержащий код читателя, код книги, дату выдачи, требуемый срок возврата, и реальную дату возврата книги.

Требуется создать ПО, которое может добавлять, удалять и изменять записи в списках, а также:

- создавать список всех книг библиотеки

- создавать список всех читателей библиотеки

- создавать список книг, находящихся у читателей

- создавать список книг, находящихся в библиотеке

- создавать список должников (книгу не вернули в течении 10 дней после срока возврата)

- создавать список не доверенных читателей (должники и те, кто были должниками)

- осуществлять поиск книг по имени автора либо по названию

- осуществлять поиск читателя по имени

# Структура модулей

Таблица – Описание модулей программы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование  модуля | Назначение модуля | Подключаемые модули |
|  | UMain | Модуль главной формы, реагирует на действия пользователя и выводит результаты | UBookList, UAddBook, UVisitorList, UAddVisitor,  UBorrowList, UAddBorrow |
|  | UBookList | Модуль, реализующий список книг и алгоритмы управления им | UMain |
|  | UVisitorList | Модуль, реализующий список посетителей и алгоритмы управления им | UMain |
|  | UBorrowList | Модуль, реализующий список записей о займе и алгоритмы управления им | UMain |
|  | UFilesIntegration | Модуль, управляющий файлами. При запуске считывает данные, при отключении записывает | UBookList, UVisitorList, UBorrowList |
|  | UAddBook | Модуль диалога, запрашивает информацию о книге | UMain, UBookList, UBorrowList, USelectCode |
|  | UAddVisitor | Модуль диалога, запрашивает информацию о посетителе | UMain, UVisitorList, UBorrowList, USelectCode |
|  | UAddBorrow | Модуль диалога, запрашивает информацию о записи | UMain, UBookList, UVisitorList, UBorrowList, USelectCode |
|  | USelectCode | Модуль диалога, запрашивает код книги, записи или посетителя | UBookList, UVisitorList, UBorrowList, UMain, UAddBook, UAddVisitor, UAddBorrow |

# Описание алгоритмов

## Описание алгоритмов модуля UMain

Таблица 2 – Описание алгоритмов модуля UMain

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование  алгоритма | Назначение  алгоритма | Формальные  параметры | Рекомендуемый тип |
|  | setCaptions | Устанавливает заголовки компонентов |  | Процедура |
|  | setColumnsBooks, setColumnsVisitors, setColumnsBorrows | Устанавливает заголовки столбцов таблицы |  | Процедура |
|  | showBooksList,  showVisitorsList,  showBorrowsList | Заполняет таблицу данными |  | Процедура |
|  | showList | Вызывает один из алгоритмов заполнения таблицы данными |  | Процедура |

## Описание алгоритмов модуля UBookList

Таблица 3 – Описание алгоритмов модуля UBookList

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование  алгоритма | Назначение  алгоритма | Формальные  параметры | Рекомендуемый тип |
|  | toBook(result, code,  authorSurname, bookTitle, publishYear, publishLang) | Объединяет пять переменных в запись из 5 полей | result, code,  authorSurname, bookTitle, publishYear, publishLang | Функция.  Возвращаемый параметр: result |
|  | addBook(book) | Добавляет запись book в список | book | Процедура |
|  | deleteByCode(code) | Удаляет запись из списка по ключу code | code | Процедура |

Продолжение таблицы 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | findByAuthor  (result, name) | Возвращает записи из списка по имени автора name | result, name  Возвращаемый параметр: result | Функция.  Возвращаемый параметр: result |
|  | findByTitle(result, title) | Возвращает записи из списка по названию книги title | result, title  Возвращаемый параметр: result | Функция.  Возвращаемый параметр: result |
|  | findByCode(result, code) | Возвращает запись из списка по номеру книги code | result, code  Возвращаемый параметр: result | Функция.  Возвращаемый параметр: result |
|  | changeByCode  (code, newBook) | Изменяет запись в списке на newBook по номеру книги code | code, newBook  Возвращаемый параметр: result | Процедура |
|  | returnAllBooks  (result) | Возвращает все записи из списка | result  Возвращаемый параметр: result | Функция.  Возвращаемый параметр: result |
|  | bookExist(result, code) | Проверяет существует ли книга с номером code в списке | result, code  Возвращаемый параметр: result | Функция.  Возвращаемый параметр: result |

## Описание алгоритмов модуля UVisitorList

Таблица 4 – Описание алгоритмов модуля UVisitorList

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование  алгоритма | Назначение  алгоритма | Формальные  параметры | Рекомендуемый тип |
|  | addVisitor(visitor) | Добавляет запись visitor в список | visitor | Процедура |
|  | findByName(result,visitorName) | Возвращает записи из списка по имени посетителя visitorName | result, visitorName | Функция.  Возвращаемый параметр: result |

Продолжение таблицы 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | findByID(result, ID) | Возвращает запись из списка по номеру посетителя ID | result, ID  Возвращаемый параметр: result | Функция.  Возвращаемый параметр: result |
|  | deleteByID(ID) | Удаляет запись из списка по ключу ID | ID | Процедура |
|  | changeByID(ID, newVisitor) | Изменяет запись в списке на newVisitor по номеру посетителя ID | code, newVisitor | Процедура |
|  | returnAllVisitors  (result) | Возвращает все записи из списка | result  Возвращаемый параметр: result | Функция.  Возвращаемый параметр: result |
|  | visitorExist(result, code) | Проверяет существует ли посетитель с номером ID в списке | result, code  Возвращаемый параметр: result | Функция.  Возвращаемый параметр: result |

## Описание алгоритмов модуля UBorrowList

Таблица 5 – Описание алгоритмов модуля UBorrowList

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование  алгоритма | Назначение  алгоритма | Формальные  параметры | Рекомендуемый тип |
|  | addBorrow(borrow) | Добавляет запись visitor в список | visitor | Процедура |
|  | returnAllBorrow  (result) | Возвращает все записи из списка | result Возвращаемый параметр: result | Функция.  Возвращаемый параметр: result |
|  | returnOpenBorrows  (result) | Возвращает открытые записи из списка | result Возвращаемый параметр: result | Функция.  Возвращаемый параметр: result |
|  | returnClosedBorrows  (result) | Возвращает закрытые записи из списка | result Возвращаемый параметр: result | Функция.  Возвращаемый параметр: result |

Продолжение таблицы 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | changeByBorrowCode  (code, newBorrow) | Изменяет запись в списке на newBorrow по номеру записи code | code, newBorrow | Процедура |
|  | findByBorrowCode  (result, code) | Возвращает запись из списка по номеру записи code | result, code  Возвращаемый параметр: result | Функция.  Возвращаемый параметр: result |
|  | deleteByVisitorCode  (code) | Удаляет записи из списка по номеру посетителя code | code | Процедура |
|  | deleteByBookCode  (code) | Удаляет записи из списка по номеру книги code | code | Процедура |
|  | deleteByBorrowCode  (code) | Удаляет запись из списка по номеру записи code | code | Процедура |
|  | borrowExist(result, code) | Проверяет существует ли запись с номером code в списке | result, code Возвращаемый параметр: result | Функция.  Возвращаемый параметр: result |

## Описание алгоритмов модуля UFilesIntegration

Таблица 6 – Описание алгоритмов модуля UFilesIntegration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование  алгоритма | Назначение  алгоритма | Формальные  параметры | Рекомендуемый тип |
|  | checkFiles | Если файлы с данными не существуют, то создаёт их |  | Процедура |
|  | readFiles | Считывает данные из файлов в списки |  | Процедура |

Продолжение таблицы 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | saveFiles | Сохраняет данные из списков в файлы |  | Процедура |

# Структура типов

## Структура типов модуля UMain

Таблица 7 – Структура типов модуля UMain

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| TListType | (cBOOKS, cVISITORS, cBORROWS) | Определяет, какой список выводится в данных момент |
| TListAttribute | (cFULL, cOPEN, cCLOSE, cTITLE, cNAME) | Определяет какая часть списка выводится в данный момент |
| TDialogType | (cADD, cCHANGE, cDELETE) | Определяет какую операцию надо выполнить |

## Структура типов модуля UBookList

Таблица 8 – Структура типов модуля UBookList

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| TLang | (RU, EN, CH, BY, JP, SP) | Языки издания книги |
| TBook | code: Integer  authorSurname: String[30]  bookTitle: String[20]  publishYear: Integer  publishLang: TLang | Запись с информацией о книге |
| TBooks | array of TBook | Массив записей о книгах |
| TBookList | ^TBookElem | Список книг |
| TBookElem | book: TBook  next: TBookList | Элемент списка книг |

## Структура типов модуля UFilesIntegration

Таблица 11 – Структура типов модуля UFilesIntegration

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| TBookFile | file of TBook | Файл записей о книгах |
| TVisitorFile | file of TVisitor | Файл записей о посетителях |
| TBorrowFile | file of TBorrow | Файл записей о взятых книгах |

## Структура типов модуля UVisitorList

Таблица 9 – Структура типов модуля UVisitorList

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| TAddress | city: String[10];  street: String[20];  houseNumber: Integer;  case block: Boolean of  true: (flatNumber: Integer);  false: (); | Адрес посетителя |
| TName | firstName: String[20]  middleName: String[20]  lastName: String[20] | Имя посетителя |
| TVisitor | ID: Integer  name: TName  address: TAddress;  phoneNumber: String[20] | Запись с информацией о посетителе |
| TVisitors | array of TVisitor | Массив записей о посетителях |
| TVisitorList | ^TVisitorElem | Список посетителей |
| TVisitorElem = | record  visitor: TVisitor  next: TVisitorList | Элемент списка посетителей |

## Структура типов модуля UBorrowList

Таблица 10 – Структура типов модуля UBorrowList

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| TBorrow | code: Integer;  visitorID: Integer  bookCode: Integer  borrowDate: TDateTime  expReturnDate: TDateTime  realReturnDate: TDateTime | Запись о взятой книге |
| TBorrows | array of TBorrow | Массив записей о взятых книгах |
| TBorrowList | ^TBorrowElem | Список записей |
| TBorrowElem = | borrow: TBorrow  next: TBorrowList | Элемент списка записей |

# Структура данных

## Структура данных модуля UMain

Таблица 12 – Структура данных алгоритмов showBooksList, showVisitorsList, showBorrowsList

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| currentListType | TListType | Тип выведенного в listView списка |
| currentListAttribute | TListAttribute | Определяет, какие записи должны быть выведены |
| book, visitor, borrow | TBooks, TVisitors, TBorrows | Массивы записей для вывода в listView |
| i, j | Integer | Счетчики циклов |
| error | Boolean | Флаг исключения |

## Структура данных модуля UFilesIntegration

Таблица 13 – Структура данных основного алгоритма

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| BOOK\_PATH | 'books.dat' | Путь к файлу книг |
| VISITOR\_PATH | 'visitors.dat' | Путь к файлу посетителей |
| BORROW\_PATH | 'borrows.dat' | Путь к файлу записей |

Таблица 14 – Структура данных алгоритмов checkFiles, readFiles, saveFiles

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| bookFile | TBookFile | Файл книг |
| visitorFile | TVisitorFile | Файл посетителей |
| borrowFile | TBorrowFile | Файл записей о взятых книгах |

## Структура данных модуля UBookList, UVisitorList, UBorrowList

Таблица 15 – Структура данных основного алгоритма

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| books, visitors, borrows | TBookList, TVisitorList, TBorrowList | Списки для хранения информации во время работы программы |
| list | TBookList, TVisitorList, TBorrowList | Указатель на список, с которым выполняется действие |

# Схема алгоритма решения задачи по ГОСТ 19.701-90

## Схема алгоритма showBooksList



Рисунок 1 – Схема алгоритма showBooksList

## Схема алгоритма showList



Рисунок 2 – Схема алгоритма showList

## Схема алгоритма addBook(book)



Рисунок 3 – Схема алгоритма addBook(book)

## Схема алгоритма deleteByCode(code)



Рисунок 4 – Схема алгоритма deleteByCode (code)

# Результаты расчетов и тестирование программы

## Группа тестов вывода списков

### Тест 1.1

Таблица 16 – Тест 1.1

|  |  |
| --- | --- |
| Тестовая ситуация | Вывод полного списка книг |
| Исходные данные | В файле содержится информация о следующих книгах:  Война миров, Я – Легенда, Another Fine Myth, Myth Conceptions, Три товарища, Записки охотника, Eve, Mallory |
| Ожидаемый результат | При нажатии клавиши «Показать все книги», вы получите таблицу содержащую информацию об этих книгах |
| Полученный результат |  |

### Тест 1.2

Таблица 17 – Тест 1.2

|  |  |
| --- | --- |
| Тестовая ситуация | Вывод книг, которые взяты посетителями |
| Исходные данные | См. исходные данные к Тест 1.1  В файле содержится информация о посетителях: Довыдёнок М.А., Иванов И.И., Александров А.А  А также информация о взятых книгах: Eve Александровым, Mallory Ивановым (Вернул 12.04) |
| Ожидаемый результат | При нажатии клавиши «Показать взятые книги», вы получите таблицу содержащую информацию о книге Eve |

Продолжение таблицы 17

|  |  |
| --- | --- |
| Полученный результат |  |

### Тест 1.3

Таблица 18 – Тест 1.3

|  |  |
| --- | --- |
| Тестовая ситуация | Вывод книг, которые не взяты посетителями |
| Исходные данные | См. исходные данные к Тест 1.2 |
| Ожидаемый результат | При нажатии клавиши «Показать невзятые книги», вы получите таблицу содержащую информацию о всех книгах, кроме «Eve» |
| Полученный результат |  |

### Тест 1.4

Таблица 19 – Тест 1.4

|  |  |
| --- | --- |
| Тестовая ситуация | Вывод книг по фамилии автора |
| Исходные данные | См. исходные данные к Тест 1.2  Введите в поле «Введите фамилию автора» фамилию Асприн. |
| Ожидаемый результат | При нажатии клавиши «Найти книги по автору» вы получите информацию о книгах Another Fine Myth и Myth Conceptions |
| Полученный результат |  |

### Тест 1.5

Таблица 20 – Тест 1.5

|  |  |
| --- | --- |
| Тестовая ситуация | Вывод книг по названию |
| Исходные данные | См. исходные данные к Тест 1.2  Введите в поле «Введите название книги» названи Eve |
| Ожидаемый результат | При нажатии клавиши «Найти книги по названию» вы получите информацию о книге Eve |

Продолжение таблицы 20

|  |  |
| --- | --- |
| Полученный результат |  |

### Тест 1.6

Таблица 21 – Тест 1.6

|  |  |
| --- | --- |
| Тестовая ситуация | Вывод списка всех посетителей |
| Исходные данные | См. исходные данные к Тест 1.2 |
| Ожидаемый результат | При нажатии клавиши «Показать всех посетителей» вы получите информацию о всех посетителях из исходных данных |
| Полученный результат |  |

### Тест 1.7

Таблица 22 – Тест 1.7

|  |  |
| --- | --- |
| Тестовая ситуация | Вывод списка должников |
| Исходные данные | См. исходные данные к Тест 1.2 |
| Ожидаемый результат | При нажатии клавиши «Показать должников» вы получите информацию о Александрове А.А. т.к. он не вернул книгу в течении 10 дней |
| Полученный результат |  |

### Тест 1.8

Таблица 23 – Тест 1.8

|  |  |
| --- | --- |
| Тестовая ситуация | Вывод списка недоверенных посетителей |
| Исходные данные | См. исходные данные к Тест 1.2 |
| Ожидаемый результат | При нажатии клавиши «Показать недоверенных» вы получите информацию о Александрове А.А. и Иванове И.И. т.к. они не вернул книгу в течении 10 дней хотя бы один раз |

Продолжение таблицы 23

|  |  |
| --- | --- |
| Полученный результат |  |

### Тест 1.9

Таблица 24 – Тест 1.9

|  |  |
| --- | --- |
| Тестовая ситуация | Вывод информации о посетителях по фамилии |
| Исходные данные | См. исходные данные к Тест 1.2  Введите в поле «Введите имя посетителя» фамилию Довыдёнок. |
| Ожидаемый результат | При нажатии клавиши «Найти посетителей по имени» вы получите информацию о Довыдёнке М. А. |
| Полученный результат |  |

### Тест 1.10

Таблица 25 – Тест 1.10

|  |  |
| --- | --- |
| Тестовая ситуация | Вывод списка записей о взятых книгах |
| Исходные данные | См. исходные данные к Тест 1.2 |
| Ожидаемый результат | При нажатии клавиши «Показать все записи» вы получите информацию о всех двух записях из исходных данных |
| Полученный результат |  |

### Тест 1.11

Таблица 26 – Тест 1.11

|  |  |
| --- | --- |
| Тестовая ситуация | Вывод списка открытых записей о взятых книгах |
| Исходные данные | См. исходные данные к Тест 1.2 |
| Ожидаемый результат | При нажатии клавиши «Показать открытые записи» вы получите информацию о записи книги Eve на Александрова А.А. |

Продолжение таблицы 26

|  |  |
| --- | --- |
| Полученный результат |  |

### Тест 1.12

Таблица 27 – Тест 1.12

|  |  |
| --- | --- |
| Тестовая ситуация | Вывод списка закрытых записей о взятых книгах |
| Исходные данные | См. исходные данные к Тест 1.2 |
| Ожидаемый результат | При нажатии клавиши «Показать закрытые записи» вы получите информацию о записи книги Mallory на Иванова И.И. |
| Полученный результат |  |

## Группа тестов изменения данных в файлах

### Тест 2.1

Таблица 28 – Тест 2.1

|  |  |
| --- | --- |
| Тестовая ситуация | Добавление книги |
| Исходные данные | См. исходные данные к Тест 1.2  Название книги: TEST.  Автор: TEST.  Год издания: 1.  Язык публикации: Русский |
| Ожидаемый результат | При нажатии клавиши «Добавить книгу» появится окно «Добавить книгу», введя там любые данные и нажав клавишу «Добавить книу» вы увидите добавленную книгу в списке всех книг |
| Полученный результат |  |

### Тест 2.2

Таблица 29 – Тест 2.2

|  |  |
| --- | --- |
| Тестовая ситуация | Изменение информации о книге |
| Исходные данные | См. исходные данные к Тест 1.2  Номер книги: 9  Новый язык публикации: Английский |
| Ожидаемый результат | При нажатии клавиши «Изменить книгу» появится окно «Введите код», введя там код книги вы увидите окно «Изменить книу», где сможете изменить информацию о выбранной книге. Измения отобразяться в списке книг |

Продолжение таблицы 29

|  |  |
| --- | --- |
| Полученный результат |  |

### Тест 2.3

Таблица 28 – Тест 2.3

|  |  |
| --- | --- |
| Тестовая ситуация | Удаление книги |
| Исходные данные | См. исходные данные к Тест 1.2  Номер книги: 9 |
| Ожидаемый результат | При нажатии клавиши «Удалить книгу» появится окно «Удалить книгу», введя там номер книги и нажав удалить книгу, книга с данным номером исчезнет из списка книг |
| Полученный результат |  |

Приложение А

(обязательное)

Исходный код программы

unit UMain;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, UBookList, UVisitorList, UBorrowList, Menus, ActnList, ComCtrls,

ExtCtrls, StdCtrls, Buttons;

resourcestring

sCODE = 'Код';

sTITLE = 'Название';

sAUTHOR = 'Автор';

sPUBLISH\_YEAR = 'Год издания';

sPUBLISH\_LANG = 'Язык публикации';

sNAME = 'ФИО';

sADDRESS = 'Адрес проживания';

sPHONE = 'Телефон';

sBOOK\_TITLE = 'Название книги';

sBORROW\_DATE = 'Дата выдачи';

sEXP\_RETURN\_DATE = 'Дата возврата';

sREAL\_RETURN\_DATE = 'Возврат';

sALL\_BOOKS = 'Все книги';

sNOT\_BORROWED\_BOOKS = 'Невзятые книги';

sBORROWED\_BOOKS = 'Взятые книги';

sADD\_BOOK = 'Добавить книгу';

sDELETE\_BOOK = 'Удалить книгу';

sCHANGE\_BOOK = 'Изменить информацию о книге';

sSHOW\_ALL\_BOOKS = 'Показать все книги';

sSHOW\_BORROWED\_BOOKS = 'Показать взятые книги';

sSHOW\_NOT\_BORROWED\_BOOKS = 'Показать невзятые книги';

sALL\_VISITORS = 'Все посетители';

sDEBTS = 'Должники';

sBAD\_VISITORS = 'Недоверенные';

sADD\_VISITOR = 'Добавить посетителя';

sDELETE\_VISITOR = 'Удалить посетителя';

sCHANGE\_VISITOR = 'Изменить информацию о посетителе';

sSHOW\_ALL\_VISITORS = 'Показать всех посетителей';

sSHOW\_DEBTS = 'Показать должников';

sSHOW\_BAD\_VISITORS = 'Показать недоверенных';

sALL\_BORROWS = 'Все записи';

sOPEN\_BORROWS = 'Открытые записи';

sCLOSED\_BORROWS = 'Закрытые записи';

sADD\_BORROW = 'Записать книгу на посетителя';

sCHANGE\_BORROW = 'Изменить запись';

sCLOSE\_BORROW = 'Закрыть запись';

sSHOW\_ALL\_BORROWS = 'Показать все записи';

sSHOW\_OPEN\_BORROWS = 'Показать открытые записи';

sSHOW\_CLOSED\_BORROWS = 'Показать закрытые записи';

sINCORRECT\_VISITOR\_ID =

'Вы выбрали несуществующего посетителя';

sINCORRECT\_BOOK\_CODE = 'Вы выбрали несуществующую книгу';

sINCORRECT\_BORROW\_CODE =

'Вы выбрали несуществующую запись';

sFIELD\_MUST\_BE\_FILLED = 'Поле должно быть заполнено';

sENTER\_CORRECT\_DATA = 'Введите корректные данные';

sENTER\_HOUSE\_NUMBER = 'Введите номер дома';

sENTER\_PUBLISH\_YEAR = 'Введите год публикации';

sENTER\_BOOK\_TITLE = 'Введите название книги';

sENTER\_BOOK\_AUTHOR = 'Введите фамилию автора';

sENTER\_VISITOR\_NAME = 'Введите имя посетителя';

sENTER\_CODE = 'Введите код';

sTITLE\_BOOK\_SEARCH = 'Поиск книг по названию';

sAUTHOR\_BOOK\_SEARCH = 'Поиск книг по автору';

sNAME\_VISITOR\_SEARCH = 'Поиск посетителей по фамилии';

sFIND\_BOOKS\_BY\_AUTHOR = 'Найти книги по автору';

sFIND\_BOOKS\_BY\_TITLE = 'Найти книги по названию';

sFIND\_VISITORS\_BY\_NAME = 'Найти посетителей по имени';

sERROR = 'Ошибка';

sSELECT = 'Выбрать';

sLANG\_RU = 'Русский';

sLANG\_EN = 'Английский';

sLANG\_CH = 'Китайский';

sLANG\_BY = 'Белорусский';

sLANG\_JP = 'Японский';

sLANG\_SP = 'Испанский';

sCITY\_SHORT = 'г.';

sHOUSE\_SHORT = 'д.';

sFLAT\_SHORT = 'кв.';

sBOOK\_CODE = 'Код книги';

sVISITOR\_ID = 'Код посетителя';

sRETURN\_PERIOD = 'Срок возврата (дней)';

sFIRST\_NAME = 'Имя';

sMIDDLE\_NAME = 'Отчество';

sLAST\_NAME = 'Фамилия';

sSTREET = 'Улица';

sCITY = 'Город';

sHOUSE = 'Дом';

sFLAT = 'Квартира';

type

TListType = (cBOOKS, cVISITORS, cBORROWS);

TListAttribute = (cFULL, cOPEN, cCLOSE, cTITLE, cNAME);

TDialogType = (cADD, cCHANGE, cDELETE);

TfrmMain = class(TForm)

alMain: TActionList;

aAddBook: TAction;

aDeleteBook: TAction;

aChangeBook: TAction;

aAddVisitor: TAction;

aDeleteVisitor: TAction;

aChangeVisitor: TAction;

aAddBorrow: TAction;

aChangeBorrow: TAction;

menuMain: TMainMenu;

miBooks: TMenuItem;

miVisitors: TMenuItem;

miBorrows: TMenuItem;

msiAddBook: TMenuItem;

msiAddVisitor: TMenuItem;

msiAddBorrow: TMenuItem;

pnlControls: TPanel;

btnShowBadVisitors: TBitBtn;

aShowDebts: TAction;

aShowBadVisitors: TAction;

aShowBorrowedBooks: TAction;

aFindBooksByAuthor: TAction;

aFindBooksByTitle: TAction;

btnShowDebts: TButton;

btnShowBorrowedBooks: TButton;

btnFindBooksByAuthor: TButton;

editBooksAuthor: TEdit;

editBooksTitle: TEdit;

btnFindBooksByTitle: TButton;

btnShowAllBooks: TButton;

aShowAllBooks: TAction;

btnShowAllVisitors: TButton;

aShowAllVisitors: TAction;

editVisitorsName: TEdit;

btnFindVisitorsByName: TButton;

aFindVisitorsByName: TAction;

bevelLineTop: TBevel;

btnShowNotBorrowedBooks: TButton;

aShowNotBorrowedBooks: TAction;

bevelLineBottom: TBevel;

btnShowAllBorrows: TButton;

btnShowOpenBorrows: TButton;

btnShowClosedBorrows: TButton;

aShowAllBorrows: TAction;

aShowOpenBorrows: TAction;

aShowClosedBorrows: TAction;

pnlLists: TPanel;

labelCurrentList: TLabel;

lvCurrentList: TListView;

msiChangeBook: TMenuItem;

msiChangeVisitor: TMenuItem;

msiDeleteVisitor: TMenuItem;

msiDeleteBook: TMenuItem;

msiChangeBorrow: TMenuItem;

aCloseBorrow: TAction;

msiCloseBorrow: TMenuItem;

msiBookSplitter: TMenuItem;

msiShowAllBooks: TMenuItem;

msiShowBorrowedBooks: TMenuItem;

msiShowNotBorrowedBooks: TMenuItem;

msiVisitorSplitter: TMenuItem;

msiShowAllVisitors: TMenuItem;

msiShowDebts: TMenuItem;

msiShowBadVisitors: TMenuItem;

msiBorrowSplitter: TMenuItem;

msiShowAllBorrows: TMenuItem;

msiShowOpenBorrows: TMenuItem;

msiShowClosedBorrows: TMenuItem;

procedure FormCreate(Sender: TObject);

procedure aAddBookExecute(Sender: TObject);

procedure aShowAllVisitorsExecute(Sender: TObject);

procedure aShowAllBooksExecute(Sender: TObject);

procedure aShowAllBorrowsExecute(Sender: TObject);

procedure aAddVisitorExecute(Sender: TObject);

procedure aAddBorrowExecute(Sender: TObject);

procedure aFindBooksByTitleExecute(Sender: TObject);

procedure aFindBooksByAuthorExecute(Sender: TObject);

procedure aFindVisitorsByNameExecute(Sender: TObject);

procedure aShowOpenBorrowsExecute(Sender: TObject);

procedure aShowClosedBorrowsExecute(Sender: TObject);

procedure aShowNotBorrowedBooksExecute(Sender: TObject);

procedure aShowBorrowedBooksExecute(Sender: TObject);

procedure aShowDebtsExecute(Sender: TObject);

procedure aShowBadVisitorsExecute(Sender: TObject);

procedure aChangeBookExecute(Sender: TObject);

procedure aDeleteBookExecute(Sender: TObject);

procedure aDeleteVisitorExecute(Sender: TObject);

procedure aChangeVisitorExecute(Sender: TObject);

procedure aChangeBorrowExecute(Sender: TObject);

procedure aCloseBorrowExecute(Sender: TObject);

private

// Тип текущего выведенного списка

currentListType: TListType;

// Определяет какие записи требуется вывести

currentListAttribute: TListAttribute;

// Устанавливает заголовки компонентов

procedure setCaptions;

// Заполняет столбцы listView

procedure setColumnsBooks;

procedure setColumnsVisitors;

procedure setColumnsBorrows;

// Выводит список

procedure showBooksList;

procedure showVisitorsList;

procedure showBorrowsList;

public

// Тип вызываемого диалога

dialogType: TDialogType;

//Определяет, какой список необходимо вывести

procedure showList;

end;

var

frmMain: TfrmMain;

implementation

uses DateUtils, UAddBook, UAddVisitor, UAddBorrow;

{$R \*.dfm}

procedure TfrmMain.setCaptions;

begin

editBooksAuthor.Text := sENTER\_BOOK\_AUTHOR;

editBooksTitle.Text := sENTER\_BOOK\_TITLE;

editVisitorsName.Text := sENTER\_VISITOR\_NAME;

aAddBook.Caption := sADD\_BOOK;

aDeleteBook.Caption := sDELETE\_BOOK;

aChangeBook.Caption := sCHANGE\_BOOK;

aAddVisitor.Caption := sADD\_VISITOR;

aDeleteVisitor.Caption := sDELETE\_VISITOR;

aChangeVisitor.Caption := sCHANGE\_VISITOR;

aAddBorrow.Caption := sADD\_BORROW;

aChangeBorrow.Caption := sCHANGE\_BORROW;

aCloseBorrow.Caption := sCLOSE\_BORROW;

aShowAllVisitors.Caption := sSHOW\_ALL\_VISITORS;

aShowDebts.Caption := sSHOW\_DEBTS;

aShowBadVisitors.Caption := sSHOW\_BAD\_VISITORS;

aShowAllBooks.Caption := sSHOW\_ALL\_BOOKS;

aShowBorrowedBooks.Caption := sSHOW\_BORROWED\_BOOKS;

aShowNotBorrowedBooks.Caption := sSHOW\_NOT\_BORROWED\_BOOKS;

aShowAllBorrows.Caption := sSHOW\_ALL\_BORROWS;

aShowOpenBorrows.Caption := sSHOW\_OPEN\_BORROWS;

aShowClosedBorrows.Caption := sSHOW\_CLOSED\_BORROWS;

aFindBooksByAuthor.Caption := sFIND\_BOOKS\_BY\_AUTHOR;

aFindBooksByTitle.Caption := sFIND\_BOOKS\_BY\_TITLE;

aFindVisitorsByName.Caption := sFIND\_VISITORS\_BY\_NAME;

end;

// Заполняет столбцы listView

procedure TfrmMain.setColumnsBooks;

var

listColumn: TListColumn;

begin

listColumn := lvCurrentList.Columns.Add;

listColumn.Caption := sCODE;

listColumn.Width := 50;

listColumn := lvCurrentList.Columns.Add;

listColumn.Caption := sTITLE;

listColumn.Width := 150;

listColumn := lvCurrentList.Columns.Add;

listColumn.Caption := sAUTHOR;

listColumn.Width := 150;

listColumn := lvCurrentList.Columns.Add;

listColumn.Caption := sPUBLISH\_YEAR;

listColumn.Width := 125;

listColumn := lvCurrentList.Columns.Add;

listColumn.Caption := sPUBLISH\_LANG;

listColumn.Width := 125;

end;

// Заполняет столбцы listView

procedure TfrmMain.setColumnsVisitors;

var

listColumn: TListColumn;

begin

listColumn := lvCurrentList.Columns.Add;

listColumn.Caption := sCODE;

listColumn.Width := 50;

listColumn := lvCurrentList.Columns.Add;

listColumn.Caption := sNAME;

listColumn.Width := 200;

listColumn := lvCurrentList.Columns.Add;

listColumn.Caption := sADDRESS;

listColumn.Width := 200;

listColumn := lvCurrentList.Columns.Add;

listColumn.Caption := sPHONE;

listColumn.Width := 125;

end;

// Заполняет столбцы listView

procedure TfrmMain.setColumnsBorrows;

var

listColumn: TListColumn;

begin

listColumn := lvCurrentList.Columns.Add;

listColumn.Caption := sCODE;

listColumn.Width := 40;

listColumn := lvCurrentList.Columns.Add;

listColumn.Caption := sBOOK\_TITLE;

listColumn.Width := 150;

listColumn := lvCurrentList.Columns.Add;

listColumn.Caption := sNAME;

listColumn.Width := 200;

listColumn := lvCurrentList.Columns.Add;

listColumn.Caption := sBORROW\_DATE;

listColumn.Width := 100;

listColumn := lvCurrentList.Columns.Add;

listColumn.Caption := sEXP\_RETURN\_DATE;

listColumn.Width := 100;

listColumn := lvCurrentList.Columns.Add;

listColumn.Caption := sREAL\_RETURN\_DATE;

listColumn.Width := 100;

end;

// Выводит список книг

procedure TfrmMain.showBooksList;

var

book: TBooks;

borrow: TBorrows;

i, j: Integer;

listItem: TListItem;

error: Boolean;

begin

SetLength(book, 0);

SetLength(borrow, 0);

error := false;

case currentListAttribute of

// Вывод всех книг

cFULL:

begin

labelCurrentList.Caption := sALL\_BOOKS;

book := returnAllBooks;

end;

// Вывод невзятых книг

cOPEN:

begin

labelCurrentList.Caption := sNOT\_BORROWED\_BOOKS;

book := returnAllBooks;

borrow := returnOpenBorrows;

for i := 0 to length(book) - 1 do

begin

error := false;

for j := 0 to length(borrow) - 1 do

begin

if book[i].code = borrow[j].bookCode then

error := true;

end;

if not error then

book[i].code := -1;

end;

error := false;

end;

// Вывод взятых книг

cCLOSE:

begin

labelCurrentList.Caption := sBORROWED\_BOOKS;

book := returnAllBooks;

borrow := returnOpenBorrows;

for i := 0 to length(book) - 1 do

begin

error := false;

for j := 0 to length(borrow) - 1 do

begin

if book[i].code = borrow[j].bookCode then

error := true;

end;

if error then

book[i].code := -1;

end;

error := false;

end;

// Поиск по названию

cTITLE:

begin

labelCurrentList.Caption := sTITLE\_BOOK\_SEARCH;

if editBooksTitle.Text <> sENTER\_BOOK\_TITLE then

begin

book := findByTitle(editBooksTitle.Text);

end

else

begin

error := true;

MessageBox(Handle, PChar(sENTER\_BOOK\_TITLE),

PChar(sERROR), MB\_OK or MB\_ICONWARNING);

end;

end;

// Поиск по автору

cNAME:

begin

labelCurrentList.Caption := sAUTHOR\_BOOK\_SEARCH;

if editBooksAuthor.Text <> sENTER\_BOOK\_AUTHOR then

begin

book := findByAuthor(editBooksAuthor.Text);

end

else

begin

error := true;

MessageBox(Handle, PChar(sENTER\_BOOK\_AUTHOR),

PChar(sERROR), MB\_OK or MB\_ICONWARNING);

end;

end;

end;

if not error then

begin

// Очистка listView

lvCurrentList.Clear;

lvCurrentList.Columns.Clear;

setColumnsBooks;

// Вывод элементов

for i := 0 to length(book) - 1 do

begin

if book[i].code <> -1 then

begin

listItem := lvCurrentList.Items.Add;

listItem.Caption := IntToStr(book[i].code);

listItem.SubItems.Add(book[i].bookTitle);

listItem.SubItems.Add(book[i].authorSurname);

listItem.SubItems.Add

(IntToStr(book[i].publishYear));

listItem.SubItems.Add

(langToStr(book[i].publishLang));

end;

end;

end;

end;

// Выводит список посетителей

procedure TfrmMain.showVisitorsList;

var

visitor: TVisitors;

borrow: TBorrows;

i, j: Integer;

listItem: TListItem;

error: Boolean;

today: TDateTime;

begin

error := false;

SetLength(visitor, 0);

SetLength(borrow, 0);

case currentListAttribute of

// Вывод всех посетителей

cFULL:

begin

labelCurrentList.Caption := sALL\_VISITORS;

visitor := returnAllVisitors;

end;

// Вывод должников

cOPEN:

begin

labelCurrentList.Caption := sDEBTS;

today := date;

visitor := returnAllVisitors;

borrow := returnOpenBorrows;

for i := 0 to length(visitor) - 1 do

begin

error := false;

for j := 0 to length(borrow) - 1 do

if borrow[j].visitorID = visitor[i].ID then

if borrow[j].expReturnDate + 10 < today then

error := true;

if not error then

visitor[i].ID := -1;

end;

error := false;

end;

cCLOSE:

// Вывод недоверенных

begin

labelCurrentList.Caption := sBAD\_VISITORS;

today := date;

visitor := returnAllVisitors;

borrow := returnAllBorrows;

for i := 0 to length(visitor) - 1 do

begin

error := false;

for j := 0 to length(borrow) - 1 do

begin

if borrow[j].visitorID = visitor[i].ID then

begin

if borrow[j].realReturnDate = -1 then

begin

if borrow[j].expReturnDate + 10 < today then

error := true;

end

else

begin

if borrow[j].expReturnDate + 10 <

borrow[j].realReturnDate then

error := true;

end;

end;

end;

if not error then

visitor[i].ID := -1;

end;

error := false;

end;

// Поиск по фамилии

cNAME:

begin

labelCurrentList.Caption := sNAME\_VISITOR\_SEARCH;

if editVisitorsName.Text <> sENTER\_VISITOR\_NAME then

begin

visitor := findByName(toName('', '',

editVisitorsName.Text));

end

else

begin

error := true;

MessageBox(Handle, PChar(sENTER\_VISITOR\_NAME),

PChar(sERROR), MB\_OK or MB\_ICONWARNING);

end;

end;

end;

if not error then

begin

// Очистка listView

lvCurrentList.Clear;

lvCurrentList.Columns.Clear;

setColumnsVisitors;

// Вывод элементов

for i := 0 to length(visitor) - 1 do

begin

if visitor[i].ID <> -1 then

begin

listItem := lvCurrentList.Items.Add;

listItem.Caption := IntToStr(visitor[i].ID);

listItem.SubItems.Add(nameToStr(visitor[i].name));

listItem.SubItems.Add

(addressToStr(visitor[i].address));

listItem.SubItems.Add(visitor[i].phoneNumber);

end;

end;

end;

end;

// Выводит список записей о взятии книг

procedure TfrmMain.showBorrowsList;

var

borrow: TBorrows;

i: Integer;

listItem: TListItem;

begin

SetLength(borrow, 0);

case currentListAttribute of

cFULL:

begin

labelCurrentList.Caption := sALL\_BORROWS;

borrow := returnAllBorrows;

end;

cOPEN:

begin

labelCurrentList.Caption := sOPEN\_BORROWS;

borrow := returnOpenBorrows;

end;

cCLOSE:

begin

labelCurrentList.Caption := sCLOSED\_BORROWS;

borrow := returnClosedBorrows;

end;

end;

// Очистка listView

lvCurrentList.Clear;

lvCurrentList.Columns.Clear;

setColumnsBorrows;

// Вывод элементов

for i := 0 to length(borrow) - 1 do

begin

listItem := lvCurrentList.Items.Add;

listItem.Caption := IntToStr(borrow[i].code);

listItem.SubItems.Add

(findByCode(borrow[i].bookCode).bookTitle);

listItem.SubItems.Add

(nameToStr(findByID(borrow[i].visitorID).name));

listItem.SubItems.Add(DateToStr(borrow[i].borrowDate));

listItem.SubItems.Add

(DateToStr(borrow[i].expReturnDate));

if borrow[i].realReturnDate = -1 then

listItem.SubItems.Add('')

else

listItem.SubItems.Add

(DateToStr(borrow[i].realReturnDate));

end;

end;

// Определяет, какой список необходимо вывести

procedure TfrmMain.showList;

begin

case currentListType of

cBOOKS:

showBooksList;

cVISITORS:

showVisitorsList;

cBORROWS:

showBorrowsList;

end;

end;

// При создании формы выводит полный список книг

procedure TfrmMain.FormCreate(Sender: TObject);

begin

setCaptions;

Caption := Application.Title;

currentListType := cBOOKS;

currentListAttribute := cFULL;

showList;

end;

// Вывод полного списка книг

procedure TfrmMain.aShowAllBooksExecute(Sender: TObject);

begin

currentListType := cBOOKS;

currentListAttribute := cFULL;

showList;

end;

// Вывод невзятых книг

procedure TfrmMain.aShowNotBorrowedBooksExecute

(Sender: TObject);

begin

currentListType := cBOOKS;

currentListAttribute := cCLOSE;

showList;

end;

// Вывод взятых книг

procedure TfrmMain.aShowBorrowedBooksExecute

(Sender: TObject);

begin

currentListType := cBOOKS;

currentListAttribute := cOPEN;

showList;

end;

// Вывод всех посетителей

procedure TfrmMain.aShowAllVisitorsExecute(Sender: TObject);

begin

currentListType := cVISITORS;

currentListAttribute := cFULL;

showList;

end;

// Вывод списка должников

procedure TfrmMain.aShowDebtsExecute(Sender: TObject);

begin

currentListType := cVISITORS;

currentListAttribute := cOPEN;

showList;

end;

// Вывод списка недоверенных

procedure TfrmMain.aShowBadVisitorsExecute(Sender: TObject);

begin

currentListType := cVISITORS;

currentListAttribute := cCLOSE;

showList;

end;

// Вывод всех записей

procedure TfrmMain.aShowAllBorrowsExecute(Sender: TObject);

begin

currentListType := cBORROWS;

currentListAttribute := cFULL;

showList;

end;

// Вывод открытых записей

procedure TfrmMain.aShowOpenBorrowsExecute(Sender: TObject);

begin

currentListType := cBORROWS;

currentListAttribute := cOPEN;

showList;

end;

// Вывод закрытых записей

procedure TfrmMain.aShowClosedBorrowsExecute

(Sender: TObject);

begin

currentListType := cBORROWS;

currentListAttribute := cCLOSE;

showList;

end;

// Поиск книги по заголовку

procedure TfrmMain.aFindBooksByTitleExecute

(Sender: TObject);

begin

currentListType := cBOOKS;

currentListAttribute := cTITLE;

showList;

end;

// Поиск книги по автору

procedure TfrmMain.aFindBooksByAuthorExecute

(Sender: TObject);

begin

currentListType := cBOOKS;

currentListAttribute := cNAME;

showList;

end;

// Поиск посетителя по фамилии

procedure TfrmMain.aFindVisitorsByNameExecute

(Sender: TObject);

begin

currentListType := cVISITORS;

currentListAttribute := cNAME;

showList;

end;

// Добавление книги

procedure TfrmMain.aAddBookExecute(Sender: TObject);

begin

dialogType := cADD;

if (Assigned(frmAddBook)) then

frmAddBook.Close;

frmAddBook:=TfrmAddBook.Create(Self);

end;

// Изменение книги

procedure TfrmMain.aChangeBookExecute(Sender: TObject);

begin

dialogType := cCHANGE;

if (Assigned(frmAddBook)) then

frmAddBook.Close;

frmAddBook:=TfrmAddBook.Create(Self);

end;

// Удаление книги

procedure TfrmMain.aDeleteBookExecute(Sender: TObject);

begin

dialogType := cDELETE;

if (Assigned(frmAddBook)) then

frmAddBook.Close;

frmAddBook:=TfrmAddBook.Create(Self);

end;

// Добавление посетителя

procedure TfrmMain.aAddVisitorExecute(Sender: TObject);

begin

dialogType := cADD;

if (Assigned(frmAddVisitor)) then

frmAddVisitor.Close;

frmAddVisitor:=TfrmAddVisitor.Create(Self);

end;

// Изменение посетителя

procedure TfrmMain.aChangeVisitorExecute(Sender: TObject);

begin

dialogType := cCHANGE;

if (Assigned(frmAddVisitor)) then

frmAddVisitor.Close;

frmAddVisitor:=TfrmAddVisitor.Create(Self);

end;

// Удаление посетителя

procedure TfrmMain.aDeleteVisitorExecute(Sender: TObject);

begin

dialogType := cDELETE;

if (Assigned(frmAddVisitor)) then

frmAddVisitor.Close;

frmAddVisitor:=TfrmAddVisitor.Create(Self);

end;

// Добавление записи

procedure TfrmMain.aAddBorrowExecute(Sender: TObject);

begin

dialogType := cADD;

if (Assigned(frmAddBorrow)) then

frmAddBorrow.Close;

frmAddBorrow:=TfrmAddBorrow.Create(Self);

end;

// Изменение записи

procedure TfrmMain.aChangeBorrowExecute(Sender: TObject);

begin

dialogType := cCHANGE;

if (Assigned(frmAddBorrow)) then

frmAddBorrow.Close;

frmAddBorrow:=TfrmAddBorrow.Create(Self);

end;

// Закрытие записи

procedure TfrmMain.aCloseBorrowExecute(Sender: TObject);

begin

dialogType := cDELETE;

if (Assigned(frmAddBorrow)) then

frmAddBorrow.Close;

frmAddBorrow:=TfrmAddBorrow.Create(Self);

end;

end.

unit UBookList;

interface

var

booksLen: Integer = 0;

type

// Язык издания книги

TLang = (RU, EN, CH, BY, JP, SP);

// Запись о книге

TBook = packed record

code: Integer;

authorSurname: String[30];

bookTitle: String[20];

publishYear: Integer;

publishLang: TLang;

end;

TBooks = array of TBook;

// Конструктор TBook

function toBook(const code: Integer;

const authorSurname, bookTitle: String;

const publishYear: Integer;

const publishLang: TLang):TBook;

// Добавление книги

procedure addBook(book: TBook);

// Удаление книги по коду

procedure deleteByCode(code: Integer);

// Поиск книги по имени автора

function findByAuthor(name: String):TBooks;

// Поиск книги по названию

function findByTitle(title: String):TBooks;

// Поиск книги по коду

function findByCode(code: Integer):TBook;

// Изменение книги по коду

procedure changeByCode(code: Integer; newBook: TBook);

// Возвращает все книги библиотеки

function returnAllBooks:TBooks;

// Переводит TLang в строку

function langToStr(lang: TLang):String;

// Проверяет, существует ли книга с кодом

function bookExist(code: Integer):Boolean;

implementation

uses

UMain;

type

// Список книг в библиотеке

TBookList = ^TBookElem;

TBookElem = packed record

book: TBook;

next: TBookList;

end;

var

books: TBookList;

// Переводит TLang в строку

function langToStr(lang: TLang):String;

begin

case lang of

RU:

result := sLANG\_RU;

EN:

result := sLANG\_EN;

CH:

result := sLANG\_CH;

BY:

result := sLANG\_BY;

JP:

result := sLANG\_JP;

SP:

result := sLANG\_SP;

end;

end;

// Конструктор TBook

function toBook(const code: Integer;

const authorSurname, bookTitle: String;

const publishYear: Integer;

const publishLang: TLang):TBook;

begin

result.code := code;

result.authorSurname := authorSurname;

result.bookTitle := bookTitle;

result.publishYear := publishYear;

result.publishLang := publishLang;

end;

// Добавление элемента в список

procedure insert(list: TBookList; book: TBook);

var

temp: TBookList;

begin

new(temp);

temp^.book := book;

temp^.next := list^.next;

list^.next := temp;

end;

// Добавление книги

procedure addBook(book: TBook);

var

list: TBookList;

begin

list := books;

while (list^.next <> nil) and (list^.next^.book.bookTitle

< book.bookTitle) do

list := list^.next;

insert(list, book);

if book.code > booksLen then

booksLen := book.code;

end;

// Изменение книги по коду

procedure changeByCode(code: Integer; newBook: TBook);

var

list: TBookList;

begin

list := books;

while (list^.next <> nil) and

(list^.next^.book.code <> code) do

list := list^.next;

if list^.next <> nil then

list^.next^.book := newBook;

end;

// Поиск книги по имени автора

function findByAuthor(name: String):TBooks;

var

list: TBookList;

begin

list := books;

SetLength(result, 0);

while list^.next <> nil do

begin

list := list^.next;

if list^.book.authorSurname = name then

begin

SetLength(result, length(result) + 1);

result[length(result) - 1] := list^.book;

end;

end;

end;

// Поиск книги по названию

function findByTitle(title: String):TBooks;

var

list: TBookList;

begin

list := books;

SetLength(result, 0);

while list^.next <> nil do

begin

list := list^.next;

if list^.book.bookTitle = title then

begin

SetLength(result, length(result) + 1);

result[length(result) - 1] := list^.book;

end;

end;

end;

// Поиск книги по коду

function findByCode(code: Integer):TBook;

var

list: TBookList;

begin

list := books;

result.code := code - 1;

while list^.next <> nil do

begin

list := list^.next;

if list^.book.code = code then

result := list^.book;

end;

end;

// Удаление элемента списка

procedure delete(list: TBookList);

var

temp: TBookList;

begin

if list^.next <> nil then

begin

temp := list^.next;

list^.next := temp^.next;

dispose(temp);

end;

end;

// Удаление книги по коду

procedure deleteByCode(code: Integer);

var

list: TBookList;

begin

list := books;

while list^.next <> nil do

begin

if list^.next^.book.code = code then

delete(list)

else

list := list^.next;

end;

end;

// Возвращает все книги библиотеки

function returnAllBooks:TBooks;

var

list: TBookList;

begin

list := books;

SetLength(result, 0);

while list^.next <> nil do

begin

list := list^.next;

SetLength(result, length(result) + 1);

result[length(result) - 1] := list^.book;

end;

end;

// Проверяет, существует ли книга с кодом

function bookExist(code: Integer):Boolean;

var

list: TBookList;

begin

list := books;

result := false;

while list^.next <> nil do

begin

list := list^.next;

if list^.book.code = code then

result := true;

end;

end;

// Очистка списков

procedure clear(list: TBookList);

begin

while list^.next <> nil do

delete(list);

end;

// Создание списка

procedure create(var list: TBookList);

begin

new(list);

list^.next := nil;

end;

// Удаление списка

procedure destroy(var list: TBookList);

begin

clear(list);

dispose(list);

list := nil;

end;

initialization

create(books);

finalization

destroy(books);

end.

unit UVisitorList;

interface

var

visitorsLen: Integer = 0;

type

// Адрес посетителся библиотеки

TAddress = packed record

city: String[10];

street: String[20];

houseNumber: Integer;

case blockOfFlats: Boolean of

true: (flatNumber: Integer);

false: ();

end;

// ФИО

TName = packed record

firstName: String[20];

middleName: String[20];

lastName: String[20];

end;

// Запись о пользователе

TVisitor = packed record

ID: Integer;

name: TName;

address: TAddress;

phoneNumber: String[20];

end;

TVisitors = array of TVisitor;

// Конструктор TName

function toName(const firstName,

middleName,

lastName: String):TName;

// Конструктор TAddress

// flatNumber = -1, выбирается вариант blockOfFlats = false

function toAddress(const city,

street: String;

const houseNumber,

flatNumber: Integer):TAddress;

// Конструктор TVisitor

function toVisitor(const ID: Integer;

const name: TName;

const address: TAddress;

const phoneNumber: String):TVisitor;

// Добавление посетителся библиотеки

procedure addVisitor(visitor: TVisitor);

// Поиск посетителей по имени

function findByName(visitorName: TName):TVisitors;

// Поиск посетителя по номеру

function findByID(ID: Integer):TVisitor;

// Удаление посетителя по номеру

procedure deleteByID(ID: Integer);

// Изменение посетителя по номеру

procedure changeByID(ID: Integer; newVisitor: TVisitor);

// Возвращает записи о всех посетителях библиотеки

function returnAllVisitors:TVisitors;

// Перевод TName в строку

function nameToStr(name: TName):String;

// Первод TAddress в строку

function addressToStr(address: TAddress):String;

// Проверяет, существует ли посетитель с кодом

function visitorExist(code: Integer):Boolean;

implementation

uses

SysUtils, UMain;

type

// Список посетителей библиотеки

TVisitorList = ^TVisitorElem;

TVisitorElem = packed record

visitor: TVisitor;

next: TVisitorList;

end;

var

visitors: TVisitorList;

// Перевод TName в строку

function nameToStr(name: TName):String;

begin

result := name.lastName + ' ' + name.firstName + ' ' +

name.middleName;

end;

// Первод TAddress в строку

function addressToStr(address: TAddress):String;

begin

result := sCITY\_SHORT + ' ' + address.city + ', ' +

address.street +

' ' + sHOUSE\_SHORT + ' ' + IntToStr(address.houseNumber);

if address.blockOfFlats then

result := result + ' ' + sFLAT\_SHORT + ' ' +

IntToStr(address.flatNumber);

end;

// Конструктор TName

function toName(const firstName,

middleName,

lastName: String):TName;

begin

result.firstName := firstName;

result.middleName := middleName;

result.lastName := lastName;

end;

// Конструктор TAddress

function toAddress(const city,

street: String;

const houseNumber,

flatNumber: Integer):TAddress;

begin

result.city := city;

result.street := street;

result.houseNumber := houseNumber;

if flatNumber = -1 then

result.blockOfFlats := false

else

begin

result.blockOfFlats := true;

result.flatNumber := flatNumber;

end;

end;

// Конструктор TVisitor

function toVisitor(const ID: Integer;

const name: TName;

const address: TAddress;

const phoneNumber: String):TVisitor;

begin

result.ID := ID;

result.name := name;

result.address := address;

result.phoneNumber := phoneNumber;

end;

// Добавление элемента в список

procedure insert(list: TVisitorList; visitor: TVisitor);

var

temp: TVisitorList;

begin

new(temp);

temp^.visitor := visitor;

temp^.next := list^.next;

list^.next := temp;

end;

// Сравнение двух записей TName

function lessThan(const nameA, nameB: TName):Boolean;

begin

result := true;

if nameA.lastName > nameB.lastName then

result := false

else

begin

if nameA.lastName = nameB.lastName then

begin

if nameA.firstName > nameB.firstName then

result := false

else

begin

if nameA.firstName = nameB.firstName then

begin

if nameA.middleName > nameB.middleName then

result := false;

end;

end;

end;

end;

end;

// Добавление посетителся библиотеки

procedure addVisitor(visitor: TVisitor);

var

list: TVisitorList;

begin

list := visitors;

while (list^.next <> nil) and

lessThan(list^.next^.visitor.name, visitor.name) do

list := list^.next;

insert(list, visitor);

if visitor.ID > visitorsLen then

visitorsLen := visitor.ID;

end;

// Изменение посетителя по номеру

procedure changeByID(ID: Integer; newVisitor: TVisitor);

var

list: TVisitorList;

begin

list := visitors;

while (list^.next <> nil) and (list^.next^.visitor.ID <>

ID) do

list := list^.next;

if list^.next <> nil then

list^.next^.visitor := newVisitor;

end;

// Равенство двух записей TName

function equal(recA, recB: TName):Boolean;

begin

if (recB.firstName = '') and (recB.middleName = '') then

begin

result := (recA.lastName = recB.lastName);

end

else

begin

result := (recA.firstName = recB.firstName) and

(recA.middleName = recB.middleName) and

(recA.lastName = recB.lastName);

end;

end;

// Поиск посетителей по имени

function findByName(visitorName: TName):TVisitors;

var

list: TVisitorList;

begin

list := visitors;

SetLength(result, 0);

while list^.next <> nil do

begin

list := list^.next;

if equal(list^.visitor.name, visitorName) then

begin

SetLength(result, length(result) + 1);

result[length(result) - 1] := list^.visitor;

end;

end;

end;

// Поиск посетителя по номеру

function findByID(ID: Integer):TVisitor;

var

list: TVisitorList;

begin

list := visitors;

result.ID := ID - 1;

while list^.next <> nil do

begin

list := list^.next;

if list^.visitor.ID = ID then

result := list^.visitor;

end;

end;

// Удаление элемента из списка

procedure delete(list: TVisitorList);

var

temp: TVisitorList;

begin

if list^.next <> nil then

begin

temp := list^.next;

list^.next := temp^.next;

dispose(temp);

end;

end;

// Удаление посетителя по номеру

procedure deleteByID(ID: Integer);

var

list: TVisitorList;

begin

list := visitors;

while list^.next <> nil do

begin

if list^.next^.visitor.ID = ID then

delete(list)

else

list := list^.next;

end;

end;

// Возвращает записи о всех посетителях библиотеки

function returnAllVisitors:TVisitors;

var

list: TVisitorList;

begin

list := visitors;

SetLength(result, 0);

while list^.next <> nil do

begin

list := list^.next;

SetLength(result, length(result) + 1);

result[length(result) - 1] := list^.visitor;

end;

end;

// Проверяет, существует ли посетитель с кодом

function visitorExist(code: Integer):Boolean;

var

list: TVisitorList;

begin

list := visitors;

result := false;

while list^.next <> nil do

begin

list := list^.next;

if list^.visitor.ID = code then

result := true;

end;

end;

// Очистка списка

procedure clear(list: TVisitorList);

begin

while list^.next <> nil do

delete(list);

end;

// Создание списка

procedure create(var list: TVisitorList);

begin

new(list);

list^.next := nil;

end;

// Удаление списка

procedure destroy(var list: TVisitorList);

begin

clear(list);

dispose(list);

list := nil;

end;

initialization

create(visitors);

visitorsLen := 0;

finalization

destroy(visitors);

end.

unit UBorrowList;

interface

var

borrowsLen: Integer = 0;

type

// Запись о взятии книги из библиотеки

TBorrow = packed record

code: Integer;

visitorID: Integer;

bookCode: Integer;

borrowDate: TDateTime;

expReturnDate: TDateTime;

realReturnDate: TDateTime;

end;

TBorrows = array of TBorrow;

// Конструктор TBorrow

function toBorrow(const code,

visitorID,

bookCode: Integer;

const borrowDate,

expReturnDate,

realReturnDate:

TDateTime):TBorrow;

// Добавление записи о взятии книги

procedure addBorrow(borrow: TBorrow);

// Возвращает записи о всех взятых книгах библиотеки

function returnAllBorrows:TBorrows;

// Возвращает записи о открытых записях

function returnOpenBorrows:TBorrows;

// Возвращает записи о закрытых записях

function returnClosedBorrows:TBorrows;

// Изменение записи по коду

procedure changeByBorrowCode(code: Integer;

newBorrow: TBorrow);

// Поиск записи по коду

function findByBorrowCode(code: Integer):TBorrow;

// Поиск записи по коду посетителя

procedure deleteByVisitorCode(code: Integer);

// Поиск записи по коду книги

procedure deleteByBookCode(code: Integer);

// Удаление записи по коду

procedure deleteByBorrowCode(code: Integer);

// Проверяет, существует ли запись с кодом

function borrowExist(code: Integer):Boolean;

implementation

type

TBorrowList = ^TBorrowElem;

TBorrowElem = packed record

borrow: TBorrow;

next: TBorrowList;

end;

var

borrows: TBorrowList;

// Конструктор TBorrow

function toBorrow(const code,

visitorID,

bookCode: Integer;

const borrowDate,

expReturnDate,

realReturnDate:

TDateTime):TBorrow;

begin

result.code := code;

result.visitorID := visitorID;

result.bookCode := bookCode;

result.borrowDate := borrowDate;

result.expReturnDate := expReturnDate;

result.realReturnDate := realReturnDate;

end;

// Добавление записи о взятии книги

procedure addBorrow(borrow: TBorrow);

var

temp: TBorrowList;

begin

new(temp);

temp^.borrow := borrow;

temp^.next := borrows^.next;

borrows^.next := temp;

inc(borrowsLen);

end;

// Возвращает записи о всех взятых книгах библиотеки

function returnAllBorrows:TBorrows;

var

list: TBorrowList;

begin

list := borrows;

SetLength(result, 0);

while list^.next <> nil do

begin

list := list^.next;

SetLength(result, length(result) + 1);

result[length(result) - 1] := list^.borrow;

end;

end;

// Возвращает записи о открытых записях

function returnOpenBorrows:TBorrows;

var

list: TBorrowList;

begin

list := borrows;

SetLength(result, 0);

while list^.next <> nil do

begin

list := list^.next;

if list^.borrow.realReturnDate = -1 then

begin

SetLength(result, length(result) + 1);

result[length(result) - 1] := list^.borrow;

end;

end;

end;

// Возвращает записи о закрытых записях

function returnClosedBorrows:TBorrows;

var

list: TBorrowList;

begin

list := borrows;

SetLength(result, 0);

while list^.next <> nil do

begin

list := list^.next;

if list^.borrow.realReturnDate <> -1 then

begin

SetLength(result, length(result) + 1);

result[length(result) - 1] := list^.borrow;

end;

end;

end;

// Изменение записи по коду

procedure changeByBorrowCode(code: Integer;

newBorrow: TBorrow);

var

list: TBorrowList;

begin

list := borrows;

while (list^.next <> nil) and

(list^.next^.borrow.code <> code) do

list := list^.next;

if list^.next <> nil then

list^.next^.borrow := newBorrow;

end;

// Поиск записи по коду

function findByBorrowCode(code: Integer):TBorrow;

var

list: TBorrowList;

begin

list := borrows;

result.code := code - 1;

while list^.next <> nil do

begin

list := list^.next;

if list^.borrow.code = code then

result := list^.borrow;

end;

end;

// Удаление элемента из списка

procedure delete(list: TBorrowList);

var

temp: TBorrowList;

begin

if list^.next <> nil then

begin

temp := list^.next;

list^.next := temp^.next;

dispose(temp);

end;

end;

// Удаление записи по коду

procedure deleteByBorrowCode(code: Integer);

var

list: TBorrowList;

begin

list := borrows;

while list^.next <> nil do

begin

if list^.next^.borrow.code = code then

delete(list)

else

list := list^.next;

end;

end;

// Удаление записи по коду посетителя

procedure deleteByVisitorCode(code: Integer);

var

list: TBorrowList;

begin

list := borrows;

while list^.next <> nil do

begin

if list^.next^.borrow.visitorID = code then

delete(list)

else

list := list^.next;

end;

end;

// Удаление записи по коду книги

procedure deleteByBookCode(code: Integer);

var

list: TBorrowList;

begin

list := borrows;

while list^.next <> nil do

begin

if list^.next^.borrow.bookCode = code then

delete(list)

else

list := list^.next;

end;

end;

// Проверяет, существует ли запись с кодом

function borrowExist(code: Integer):Boolean;

var

list: TBorrowList;

begin

list := borrows;

result := false;

while list^.next <> nil do

begin

list := list^.next;

if list^.borrow.code = code then

result := true;

end;

end;

// Очистка списка

procedure clear(list: TBorrowList);

begin

while list^.next <> nil do

delete(list);

end;

// Создание списка

procedure create(var list: TBorrowList);

begin

new(list);

list^.next := nil;

end;

// Удаление списка

procedure destroy(var list: TBorrowList);

begin

clear(list);

dispose(list);

list := nil;

end;

initialization

create(borrows);

finalization

destroy(borrows);

end.

unit UAddBook;

interface

uses

SysUtils, Windows, Messages, Classes, Graphics, Controls,

StdCtrls, ExtCtrls, Forms,

ComCtrls, UBookList, USelectCode;

type

TfrmAddBook = class(TForm)

btnComplete: TButton;

editTitle: TEdit;

labelTitle: TLabel;

editAuthor: TEdit;

labelAuthor: TLabel;

udPublishYear: TUpDown;

comboLang: TComboBox;

labelPublishYear: TLabel;

labelPublishLanguage: TLabel;

editPublishYear: TEdit;

procedure editPublishYearKeyPress

(Sender: TObject; var Key: Char);

procedure udPublishYearClick

(Sender: TObject; Button: TUDBtnType);

procedure btnCompleteClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormClose

(Sender: TObject; var Action: TCloseAction);

public

code: Integer;

procedure showBook(code: Integer);

end;

var

frmAddBook: TfrmAddBook;

implementation

{$R \*.DFM}

uses

UMain, UBorrowList;

// Запрещает ввод не-цифр в поле ввода года

procedure TfrmAddBook.editPublishYearKeyPress

(Sender: TObject;

var Key: Char);

begin

case key of

'0'..'9', #8: ;

else

key := #0;

end;

end;

// Увеличивает значение в поле года при нажатии клавиш

procedure TfrmAddBook.udPublishYearClick

(Sender: TObject;

Button: TUDBtnType);

var

current: Integer;

begin

if editPublishYear.Text = '' then

begin

current := 1;

udPublishYear.Position := 0;

end

else

current := StrToInt(editPublishYear.Text);

if button = btNext then

inc(current)

else

dec(current);

editPublishYear.Text := IntToStr(current);

end;

// Проверяет введённые значения

// В зависимости от типа вызова диалого

// Добавляет, изменяет или удаляет запись в списке

procedure TfrmAddBook.btnCompleteClick(Sender: TObject);

var

borrow: TBorrows;

i: Integer;

begin

SetLength(borrow, 0);

if editPublishYear.Text <> '' then

begin

case frmMain.dialogType of

cCHANGE:

changeByCode(code, toBook(code, editAuthor.Text,

editTitle.Text, StrToInt(editPublishYear.Text),

TLang(comboLang.ItemIndex)));

cADD:

addBook(toBook(booksLen + 1, editAuthor.Text,

editTitle.Text, StrToInt(editPublishYear.Text),

TLang(comboLang.ItemIndex)));

cDELETE:

begin

deleteByCode(code);

borrow := returnAllBorrows;

for i := 0 to length(borrow) - 1 do

begin

if borrow[i].bookCode = code then

deleteByBookCode(borrow[i].bookCode);

end;

end;

end;

frmMain.showList;

Self.Close;

end

else

MessageBox(Handle, PChar(sENTER\_PUBLISH\_YEAR),

PChar(sERROR), MB\_OK or MB\_ICONWARNING);

end;

// При создании формы вызывает диалог ввода кода записи

procedure TfrmAddBook.FormCreate(Sender: TObject);

begin

Visible := false;

labelTitle.Caption := sTITLE;

labelAuthor.Caption := sAUTHOR;

labelPublishYear.Caption := sPUBLISH\_YEAR;

labelPublishLanguage.Caption := sPUBLISH\_LANG;

code := 0;

if (frmMain.dialogType = cCHANGE) or

(frmMain.dialogType = cDELETE) then

begin

if frmMain.dialogType = cCHANGE then

begin

btnComplete.Caption := sCHANGE\_BOOK;

Caption := sCHANGE\_BOOK;

end

else

begin

btnComplete.Caption := sDELETE\_BOOK;

Caption := sDELETE\_BOOK;

end;

if (Assigned(frmSelectCode)) then

frmSelectCode.Close;

frmSelectCode:=TfrmSelectCode.Create(Self);

frmSelectCode.dialogType := cBOOK;

frmSelectCode.Visible := true;

end

else

begin

Caption := sADD\_BOOK;

btnComplete.Caption := sADD\_BOOK;

visible := true;

end;

end;

// Заполняет поля значениями выбранной записи

procedure TfrmAddBook.showBook(code: Integer);

var

book: TBook;

begin

book := findByCode(code);

editTitle.Text := book.bookTitle;

editAuthor.Text := book.authorSurname;

editPublishYear.Text := IntToStr(book.publishYear);

comboLang.ItemIndex := ord(book.publishLang);

Self.code := book.code;

end;

// Закрывает вызванный диалог

procedure TfrmAddBook.FormClose

(Sender: TObject; var Action: TCloseAction);

begin

if (Assigned(frmSelectCode)) then

frmSelectCode.Close;

end;

end.

unit UAddVisitor;

interface

uses

SysUtils, Windows, Messages, Classes, Graphics, Controls,

StdCtrls, ExtCtrls, Forms, UVisitorList;

type

TfrmAddVisitor = class(TForm)

labelLastName: TLabel;

editLastName: TEdit;

editFirstName: TEdit;

editMiddleName: TEdit;

labelFirstName: TLabel;

labelMiddleName: TLabel;

labelStreet: TLabel;

editStreet: TEdit;

editHouse: TEdit;

editPhone: TEdit;

labelHouse: TLabel;

labelPhone: TLabel;

editFlat: TEdit;

labelFlat: TLabel;

btnAddVisitor: TButton;

labelCity: TLabel;

editCity: TEdit;

procedure editHouseKeyPress

(Sender: TObject; var Key: Char);

procedure editFlatKeyPress

(Sender: TObject; var Key: Char);

procedure editPhoneKeyPress

(Sender: TObject; var Key: Char);

procedure btnAddVisitorClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormClose

(Sender: TObject; var Action: TCloseAction);

public

code: Integer;

procedure showVisitor(code: Integer);

end;

var

frmAddVisitor: TfrmAddVisitor;

implementation

uses UMain, USelectCode, UBorrowList;

{$R \*.DFM}

// Запрещает ввод не-цифр в поле ввода номера дома

procedure TfrmAddVisitor.editHouseKeyPress

(Sender: TObject; var Key: Char);

begin

case key of

'0'..'9', #8: ;

else

key := #0;

end;

end;

// Запрещает ввод не-цифр в поле ввода номера квартиры

procedure TfrmAddVisitor.editFlatKeyPress

(Sender: TObject; var Key: Char);

begin

case key of

'0'..'9', #8: ;

else

key := #0;

end;

end;

// Запрещает ввод не-цифр в поле ввода номера телефона

procedure TfrmAddVisitor.editPhoneKeyPress

(Sender: TObject; var Key: Char);

begin

case key of

'0'..'9', #8, '-': ;

else

key := #0;

end;

end;

// Проверяет введённые значения

// В зависимости от типа вызова диалого

// Добавляет, изменяет или удаляет запись в списке

procedure TfrmAddVisitor.btnAddVisitorClick

(Sender: TObject);

var

borrow: TBorrows;

i: Integer;

begin

SetLength(borrow, 0);

if editHouse.Text <> '' then

begin

if editFlat.Text = '' then

editFlat.Text := '-1';

case frmMain.dialogType of

cCHANGE:

changeByID(code, toVisitor(code,

toName(editFirstName.Text, editMiddleName.Text,

editLastName.Text), toAddress(editCity.Text,

editStreet.Text, StrToInt(editHouse.Text),

(editFlat.Text)), editPhone.Text));

cADD:

addVisitor(toVisitor(visitorsLen + 1,

toName(editFirstName.Text, editMiddleName.Text,

editLastName.Text), toAddress(editCity.Text,

editStreet.Text, StrToInt(editHouse.Text),

StrToInt(editFlat.Text)), editPhone.Text));

cDELETE:

begin

deleteByID(code);

borrow := returnAllBorrows;

for i := 0 to length(borrow) - 1 do

begin

if borrow[i].visitorID = code then

deleteByVisitorCode(borrow[i].visitorID);

end;

end;

end;

frmMain.showList;

Self.Close;

end

else

MessageBox(Handle, PChar(sENTER\_HOUSE\_NUMBER),

PChar(sERROR), MB\_OK or MB\_ICONWARNING);

end;

// Заполняет поля значениями выбранной записи

procedure TfrmAddVisitor.showVisitor(code: Integer);

var

visitor: TVisitor;

begin

visitor := findByID(code);

editLastName.Text := visitor.name.lastName;

editFirstName.Text := visitor.name.firstName;

editMiddleName.Text := visitor.name.middleName;

editCity.Text := visitor.address.city;

editStreet.Text := visitor.address.street;

editHouse.Text := IntToStr(visitor.address.houseNumber);

if visitor.address.blockOfFlats then

editFlat.Text := IntToStr(visitor.address.flatNumber);

editPhone.Text := visitor.phoneNumber;

Self.code := visitor.ID;

end;

// При создании формы вызывает диалог ввода кода записи

procedure TfrmAddVisitor.FormCreate(Sender: TObject);

begin

Visible := false;

labelLastName.Caption := sLAST\_NAME;

labelFirstName.Caption := sFIRST\_NAME;

labelMiddleName.Caption := sMIDDLE\_NAME;

labelStreet.Caption := sSTREET;

labelCity.Caption := sCITY;

labelHouse.Caption := sHOUSE;

labelFlat.Caption := sFLAT;

labelPhone.Caption := sPHONE;

code := 0;

if (frmMain.dialogType = cCHANGE) or

(frmMain.dialogType = cDELETE) then

begin

if frmMain.dialogType = cCHANGE then

begin

Caption := sCHANGE\_VISITOR;

btnAddVisitor.Caption := sCHANGE\_VISITOR;

end

else

begin

btnAddVisitor.Caption := sDELETE\_VISITOR;

Caption := sDELETE\_VISITOR;

end;

if (Assigned(frmSelectCode)) then

frmSelectCode.Close;

frmSelectCode:=TfrmSelectCode.Create(Self);

frmSelectCode.dialogType := cVISITOR;

frmSelectCode.Visible := true;

end

else

begin

Caption := sADD\_VISITOR;

btnAddVisitor.Caption := sADD\_VISITOR;

visible := true;

end;

end;

// Закрывает вызванный диалог

procedure TfrmAddVisitor.FormClose(Sender: TObject;

var Action: TCloseAction);

begin

if (Assigned(frmSelectCode)) then

frmSelectCode.Close;

end;

end.

unit UAddBorrow;

interface

uses

SysUtils, Windows, Messages, Classes, Graphics, Controls,

StdCtrls, ExtCtrls, Forms, ComCtrls,

UBorrowList, UVisitorList, UBookList;

type

TfrmAddBorrow = class(TForm)

dtpBorrowDate: TDateTimePicker;

labelBookCode: TLabel;

editBookCode: TEdit;

editVisitorID: TEdit;

labelVisitorID: TLabel;

labelBorrowDate: TLabel;

labelPeriod: TLabel;

editPeriod: TEdit;

udPeriod: TUpDown;

btnAddBorrow: TButton;

procedure editBookCodeKeyPress

(Sender: TObject; var Key: Char);

procedure editVisitorIDKeyPress

(Sender: TObject; var Key: Char);

procedure editPeriodKeyPress

(Sender: TObject; var Key: Char);

procedure udPeriodClick

(Sender: TObject; Button: TUDBtnType);

procedure btnAddBorrowClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormClose

(Sender: TObject; var Action: TCloseAction);

public

code: Integer;

procedure showBorrow(code: Integer);

end;

var

frmAddBorrow: TfrmAddBorrow;

implementation

uses UMain, USelectCode;

{$R \*.DFM}

// Запрещает ввод не-цифр в поле ввода кода книги

procedure TfrmAddBorrow.editBookCodeKeyPress

(Sender: TObject;

var Key: Char);

begin

case key of

'0'..'9', #8: ;

else

key := #0;

end;

end;

// Запрещает ввод не-цифр в поле ввода кода посетителя

procedure TfrmAddBorrow.editVisitorIDKeyPress

(Sender: TObject;

var Key: Char);

begin

case key of

'0'..'9', #8: ;

else

key := #0;

end;

end;

// Запрещает ввод не-цифр в поле ввода кода срока возврата

procedure TfrmAddBorrow.editPeriodKeyPress

(Sender: TObject; var Key: Char);

begin

case key of

'0'..'9', #8: ;

else

key := #0;

end;

end;

// Увеличивает значение в поле периода при нажатии клавиш

procedure TfrmAddBorrow.udPeriodClick

(Sender: TObject; Button: TUDBtnType);

var

current: Integer;

begin

if editPeriod.Text = '' then

begin

current := 1;

udPeriod.Position := 0;

end

else

current := StrToInt(editPeriod.Text);

if button = btNext then

inc(current)

else

dec(current);

editPeriod.Text := IntToStr(current);

end;

// Проверяет введённые значения

// В зависимости от типа вызова диалого

// Добавляет, изменяет или закрывает запись в списке

procedure TfrmAddBorrow.btnAddBorrowClick(Sender: TObject);

begin

if (editBookCode.Text <> '') and

(editVisitorID.Text <> '') and

(editPeriod.Text <> '') then

begin

if bookExist(StrToInt(editBookCode.Text)) then

begin

if visitorExist(StrToInt(editVisitorID.Text)) then

begin

case frmMain.dialogType of

cCHANGE:

changeByBorrowCode(code, toBorrow(code,

StrToInt(editVisitorID.Text),

StrToInt(editBookCode.Text),

dtpBorrowDate.DateTime,

dtpBorrowDate.DateTime +

StrToInt(editPeriod.Text), -1));

cADD:

addBorrow(toBorrow(borrowsLen + 1,

StrToInt(editVisitorID.Text),

StrToInt(editBookCode.Text),

dtpBorrowDate.DateTime,

dtpBorrowDate.DateTime +

StrToInt(editPeriod.Text),

-1));

cDELETE:

changeByBorrowCode(code, toBorrow(code,

StrToInt(editVisitorID.Text),

StrToInt(editBookCode.Text),

dtpBorrowDate.DateTime,

dtpBorrowDate.DateTime +

StrToInt(editPeriod.Text), date));

end;

frmMain.showList;

Self.Close;

end

else

MessageBox(Handle, PChar(sINCORRECT\_VISITOR\_ID),

PChar(sERROR), MB\_OK or MB\_ICONWARNING);

end

else

MessageBox(Handle, PChar(sINCORRECT\_BOOK\_CODE),

PChar(sERROR), MB\_OK or MB\_ICONWARNING);

end

else

MessageBox(Handle, PChar(sENTER\_CORRECT\_DATA),

PChar(sERROR), MB\_OK or MB\_ICONWARNING);

end;

// При создании формы вызывает диалог ввода кода записи

procedure TfrmAddBorrow.FormCreate(Sender: TObject);

begin

Visible := false;

labelBookCode.Caption := sBOOK\_CODE;

labelVisitorID.Caption := sVISITOR\_ID;

labelBorrowDate.Caption := sBORROW\_DATE;

labelPeriod.Caption := sRETURN\_PERIOD;

code := 0;

if (frmMain.dialogType = cCHANGE) or

(frmMain.dialogType = cDELETE) then

begin

if frmMain.dialogType = cCHANGE then

begin

btnAddBorrow.Caption:= sCHANGE\_BORROW;

Caption := sCHANGE\_BORROW;

end

else

begin

btnAddBorrow.Caption := sCLOSE\_BORROW;

Caption := sCLOSE\_BORROW;

end;

if (Assigned(frmSelectCode)) then

frmSelectCode.Close;

frmSelectCode:=TfrmSelectCode.Create(Self);

frmSelectCode.dialogType := cBORROW;

frmSelectCode.Visible := true;

end

else

begin

Caption := sADD\_BORROW;

btnAddBorrow.Caption := sADD\_BORROW;

visible := true;

end;

end;

// Заполняет поля значениями выбранной записи

procedure TfrmAddBorrow.showBorrow(code: Integer);

var

borrow: TBorrow;

begin

borrow := findByBorrowCode(code);

editBookCode.Text := IntToStr(borrow.bookCode);

editVisitorID.Text := IntToStr(borrow.visitorID);

editPeriod.Text := IntToStr(round(borrow.expReturnDate –

borrow.borrowDate));

dtpBorrowDate.DateTime := borrow.borrowDate;

Self.code := borrow.code;

end;

// Закрывает вызванный диалог

procedure TfrmAddBorrow.FormClose(Sender: TObject;

var Action: TCloseAction);

begin

if (Assigned(frmSelectCode)) then

frmSelectCode.Close;

end;

end.

unit UFilesIntegration;

interface

uses

UBookList, UVisitorList, UBorrowList;

implementation

uses SysUtils;

// Пути к файлам записей

const

BOOK\_PATH = 'books.dat';

VISITOR\_PATH = 'visitors.dat';

BORROW\_PATH = 'borrows.dat';

type

TBookFile = file of TBook;

TVisitorFile = file of TVisitor;

TBorrowFile = file of TBorrow;

// Создаёт несуществующие файлы

procedure checkFiles;

var

bookFile: TBookFile;

visitorFile: TVisitorFile;

borrowfile: TBorrowFile;

begin

if not FileExists(BOOK\_PATH) then

begin

AssignFile(bookFile, BOOK\_PATH);

Rewrite(bookFile);

CloseFile(bookFile);

end;

if not FileExists(VISITOR\_PATH) then

begin

AssignFile(visitorFile, VISITOR\_PATH);

Rewrite(visitorFile);

CloseFile(visitorFile);

end;

if not FileExists(BORROW\_PATH) then

begin

AssignFile(borrowFile, BORROW\_PATH);

Rewrite(borrowFile);

CloseFile(borrowFile);

end;

end;

// Прочитывает файлы и записывает в списки

procedure readFiles;

var

bookFile: TBookFile;

visitorFile: TVisitorFile;

borrowfile: TBorrowFile;

tempBook: TBook;

tempVisitor: TVisitor;

tempBorrow: TBorrow;

begin

checkFiles;

AssignFile(bookFile, BOOK\_PATH);

AssignFile(visitorFile, VISITOR\_PATH);

AssignFile(borrowfile, BORROW\_PATH);

Reset(bookFile);

Reset(visitorFile);

Reset(borrowfile);

while not eof(bookFile) do

begin

read(bookFile, tempBook);

addBook(tempBook);

end;

while not eof(visitorFile) do

begin

read(visitorFile, tempVisitor);

addVisitor(tempVisitor);

end;

while not eof(borrowFile) do

begin

read(borrowFile, tempBorrow);

addBorrow(tempBorrow);

end;

CloseFile(bookFile);

CloseFile(visitorFile);

CloseFile(borrowfile);

end;

// Сохраняет файлы из списков

procedure saveFiles;

var

arBooks: TBooks;

arVisitors: TVisitors;

arBorrows: TBorrows;

bookFile: TBookFile;

visitorFile: TVisitorFile;

borrowfile: TBorrowFile;

i: Integer;

begin

arBooks := returnAllBooks;

arVisitors := returnAllVisitors;

arBorrows := returnAllBorrows;

AssignFile(bookFile, BOOK\_PATH);

AssignFile(visitorFile, VISITOR\_PATH);

AssignFile(borrowfile, BORROW\_PATH);

Rewrite(bookFile);

Rewrite(visitorFile);

Rewrite(borrowfile);

for i := 0 to length(arBooks) - 1 do

begin

write(bookFile, arBooks[i]);

end;

for i := 0 to length(arVisitors) - 1 do

begin

write(visitorFile, arVisitors[i]);

end;

for i := 0 to length(arBorrows) - 1 do

begin

write(borrowFile, arBorrows[i]);

end;

CloseFile(bookFile);

CloseFile(visitorFile);

CloseFile(borrowfile);

end;

initialization

readFiles;

finalization

saveFiles;

end.

unit USelectCode;

interface

uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls, Buttons, ExtCtrls;

type

TDialogType = (cBOOK, cVISITOR, cBORROW);

TfrmSelectCode = class(TForm)

editCode: TEdit;

labelCode: TLabel;

btnComplete: TButton;

procedure editCodeKeyPress

(Sender: TObject; var Key: Char);

procedure btnCompleteClick(Sender: TObject);

procedure FormShow(Sender: TObject);

private

{ Private declarations }

public

dialogType: TDialogType;

result: Integer;

end;

var

frmSelectCode: TfrmSelectCode;

implementation

uses

UBookList, UVisitorList, UBorrowList, UMain,

UAddBook, UAddVisitor, UAddBorrow;

{$R \*.dfm}

// Запрещает ввод не-цифр в поле ввода кода

procedure TfrmSelectCode.editCodeKeyPress

(Sender: TObject; var Key: Char);

begin

case key of

'0'..'9', #8: ;

else

key := #0;

end;

end;

// Проверяет код и возвращает в вызывающий диалог код

procedure TfrmSelectCode.btnCompleteClick(Sender: TObject);

begin

if editCode.Text <> '' then

begin

case dialogType of

cBOOK:

begin

if not bookExist(StrToInt(editCode.Text)) then

begin

MessageBox(Handle, PChar(sINCORRECT\_BOOK\_CODE),

PChar(sERROR), MB\_OK or MB\_ICONWARNING);

end

else

begin

frmAddBook.code := StrToInt(editCode.Text);

frmAddBook.showBook(frmAddBook.code);

Visible := false;

frmAddBook.Visible := true;

end;

end;

cVISITOR:

begin

if not visitorExist(StrToInt(editCode.Text)) then

begin

MessageBox(Handle, PChar(sINCORRECT\_VISITOR\_ID),

PChar(sERROR), MB\_OK or MB\_ICONWARNING);

end

else

begin

frmAddVisitor.code := StrToInt(editCode.Text);

frmAddVisitor.showVisitor(frmAddVisitor.code);

Visible := false;

frmAddVisitor.Visible := true;

end;

end;

cBORROW:

begin

if not borrowExist(StrToInt(editCode.Text)) then

begin

MessageBox(Handle, PChar(sINCORRECT\_BORROW\_CODE),

PChar(sERROR), MB\_OK or MB\_ICONWARNING);

end

else

begin

frmAddBorrow.code := StrToInt(editCode.Text);

frmAddBorrow.showBorrow(frmAddBorrow.code);

Visible := false;

frmAddBorrow.Visible := true;

end;

end;

end;

end

else

MessageBox(Handle, PChar(sFIELD\_MUST\_BE\_FILLED),

PChar(sERROR), MB\_OK or MB\_ICONWARNING);

end;

// Заполняет начальные значения метки и кнопки

procedure TfrmSelectCode.FormShow(Sender: TObject);

begin

labelCode.Caption := sENTER\_CODE;

Caption := sENTER\_CODE;

btnComplete.Caption := sSELECT;

end;

end.