Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

“Рязанский государственный радиотехнический университет

имени В. Ф. Уткина”

**Пояснительная записка**

**к курсовой работе по дисциплине**

**«Алгоритмические языки и программирование»**

**на тему**

**«Информационная система»**

Выполнил: студент гр. 1413

Проверил: доцент

Антипов О.В.

Рязань, 2022

Содержание

[Задание 3](#_Toc103150823)

[Введение 4](#_Toc103150824)

[Анализ задания и математическая постановка задачи 5](#_Toc103150825)

[Разработка схемы алгоритма и ее описание 6](#_Toc103150826)

[Основной модуль 9](#_Toc103150827)

[Разработка программы и ее описание 37](#_Toc103150828)

[Инструкция по пользованию разработанной программой 38](#_Toc103150829)

[Минимальные системные требования 38](#_Toc103150830)

[Список файлов 38](#_Toc103150831)

[Результаты отладки 43](#_Toc103150832)

[Решение контрольных примеров и проверка правильности функционирования программы 44](#_Toc103150833)

[Календарь выполнения работ 45](#_Toc103150834)

[Заключение 46](#_Toc103150835)

[Приложение: код программы 47](#_Toc103150836)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

“Рязанский государственный радиотехнический университет

имени В. Ф. Уткина”

# Задание

на курсовую работу по дисциплине

«Алгоритмические языки и программирование»

студенту

Разработать программу создания-редактирования-удаления таблиц произвольной длины с произвольным числом столбцов. Для каждого столбца задаётся тип данных – текстовый или числовой. В программе должны быть предусмотрены функции добавления и удаления строк и столбцов в созданных таблицах, поиск информации по заданному ключу, а также сортировка строк по заданному столбцу по возрастанию или убыванию. Программа должна адекватно пролистывать страницу, отображая несколько столбцов и строк на одном экране, а также отображая информацию с какого по какой столбец и строка отображаются в данный момент. В программе должно быть создано не менее 4 таблиц, с количеством строк не менее 10, и различным числом столбцов от 4 до 10.

Срок представления работы к защите:

Дата выдачи задания:

Преподаватель: Антипов О.В.

Введение

Данная курсовая работа является заключительным этапом в изучении предмета «Алгоритмические языки и программирование». Программный продукт разрабатывается в программной среде Pascal ABC.

Целью проектирования данной работы является создание именованной совокупности данных, отражающей состояние объектов и их отношений в рассматриваемой предметной области, т.е. файловой (информационной) системы, а также системы управления спроектированной структурой.

Анализ задания и математическая постановка задачи

При запуске программы пользователь видит форму для ввода логина, а затем и пароля для доступа к базе данных с правами чтения и/или редактирования.

При неудачной авторизации программа завершает свою работу.

После успешной авторизации пользователь видит главное меню, в котором доступны следующие действия с таблицами базы данных:

* Просмотреть таблицу;
* Создать таблицу;
* Отредактировать таблицу;
* Удалить таблицу.

При выборе «Отредактировать» доступные следующие действия:

* Добавить столбец (текстовый или числовой);
* Добавить строку;
* Удалить столбец;
* Удалить строку;
* Задать значение ячейки таблицы;
* Поиск значения в таблице;
* Сортировка строк таблицы по заданному столбцу (по возрастанию или убыванию).

Разработка схемы алгоритма и ее описание

По результатам анализа задания можно составить упрощенную схему алгоритма (рис. 1-3). Детализируем алгоритм (рис. 4-40).



Рисунок 1 – Упрощенная схема алгоритма



Рисунок 2 – Главное меню



Рисунок 3 – Редактирование таблицы

## Основной модуль



Рисунок 4 – Основная функция программы



Рисунок 5 – Главное меню



Рисунок 6 - Показ таблицы



Рисунок 7 – Создание таблицы



Рисунок 8 – Редактирование таблицы



Рисунок 9 – Удаление таблицы



Рисунок 10 – Добавление столбца



Рисунок 11 – Удаление столбца



Рисунок 12 – Добавление строки



Рисунок 13 – Удаление строки



Рисунок 14 – Изменение ячейки



Рисунок 15 – Поиск значения в таблице



Рисунок 16 – Сортировка таблицы



Рисунок 17 – Подтверждение действия



Рисунок 18 – Выбор индекса в пределах массива



Рисунок 19 – Выбор операции в диапазон между мин и макс



Рисунок 20 – Ожидание ввода кода возврата



Рисунок 21 – Вывод таблицы



Рисунок 22 – Конвертирование в тип Table



Рисунок 23 – Конвертирование в тип FileTable



Рисунок 24 – Чтение БД пользователей



Рисунок 25 – Удаление таблицы из массива таблиц



Рисунок 26 – Удаление столбца



Рисунок 27 – Удаление строки



Рисунок 28 – Удаление строки



Рисунок 29 – Чтение таблиц из БД



Рисунок 30 – Запись таблиц в БД



Рисунок 31 – Отображение названий таблиц



Рисунок 32 – Сортировка по возрастанию



Рисунок 33 – Сортировка по убыванию

Разработка программы и ее описание

Первым этапом была разработана система для работы с текстовой базой данных, методы считывания и записи на жесткий диск данных.

Затем были реализованы универсальные вспомогательные функции для работы с типизированным файлом, добавления, удаления элемента в динамический массив.

Следующим шагом стал вывод содержимого таблиц, меню с использованием псевдографики.

Большая часть функций возвращает коды ошибок в или значение успешного/неудачного выполнения.

Инструкция по пользованию разработанной программой

Данный программный продукт был реализован на языке программирования Паскаль с использованием среды PascalABC.NET и разработан для машин на базе ОС Microsoft Windows.

## Минимальные системные требования

* ОС: Microsoft Windows 7
* Процессор: 1ГГц
* ОЗУ: 512 МБ
* Microsoft .NET framework 4.0

## Список файлов

Исходники:

* kurs.pas

Готовый продукт:

* kurs.exe
* user.data
* database.dat

Основное управление в программе происходит через выбор нужного действия с клавиатуры.

При запуске программы пользователь увидит окно авторизации.

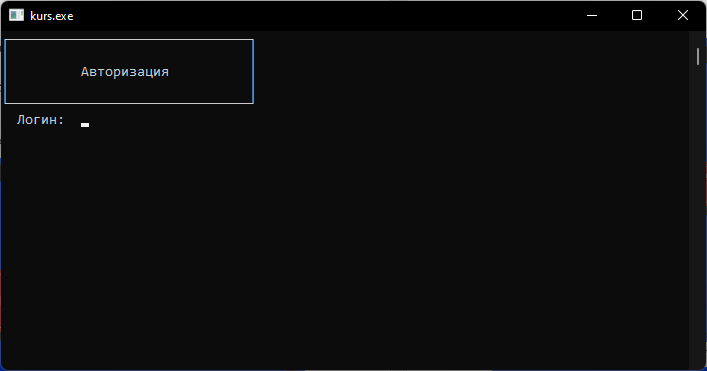


Рисунок 34 - Окно ввода логина

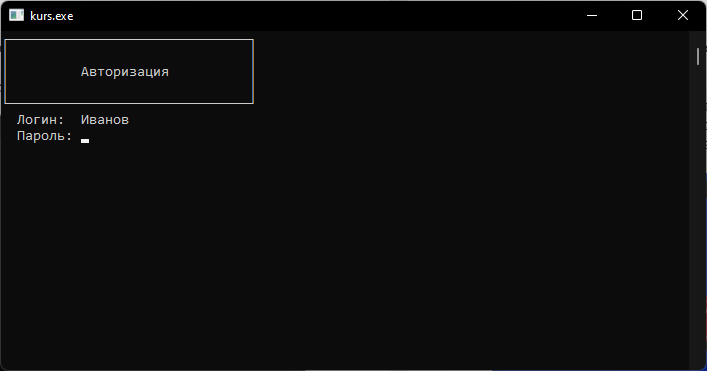


Рисунок 35 - Окно ввода пароля

В случае успешной авторизации откроется главное меню. Если пользователь имеет права для записи, об этом дополнительно появится уведомление.

В случае ввода неправильных учетных данных у пользователя есть в общем счете 5 попыток на вход в систему. После 5х неудачных попыток приложение завершит свою работу принудительно.

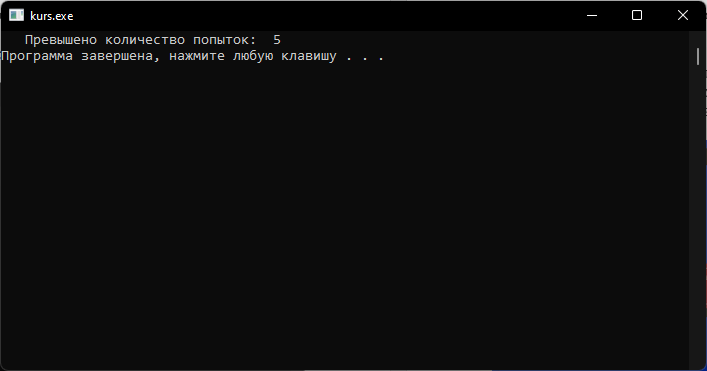


Рисунок 36 - Пример ошибки при авторизации

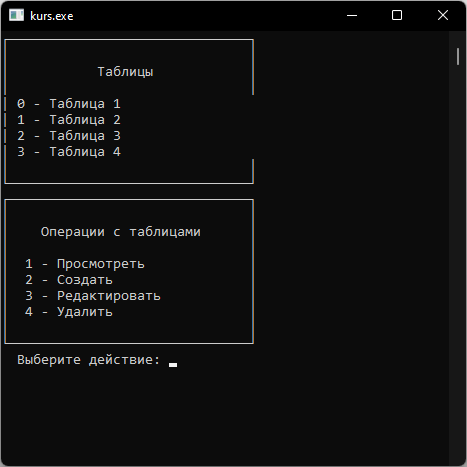


Рисунок 37 - Главное меню (решим администратора)

Из главного меню пользователь может перейти на меню просмотра, создания, редактирования и удаления таблицы.

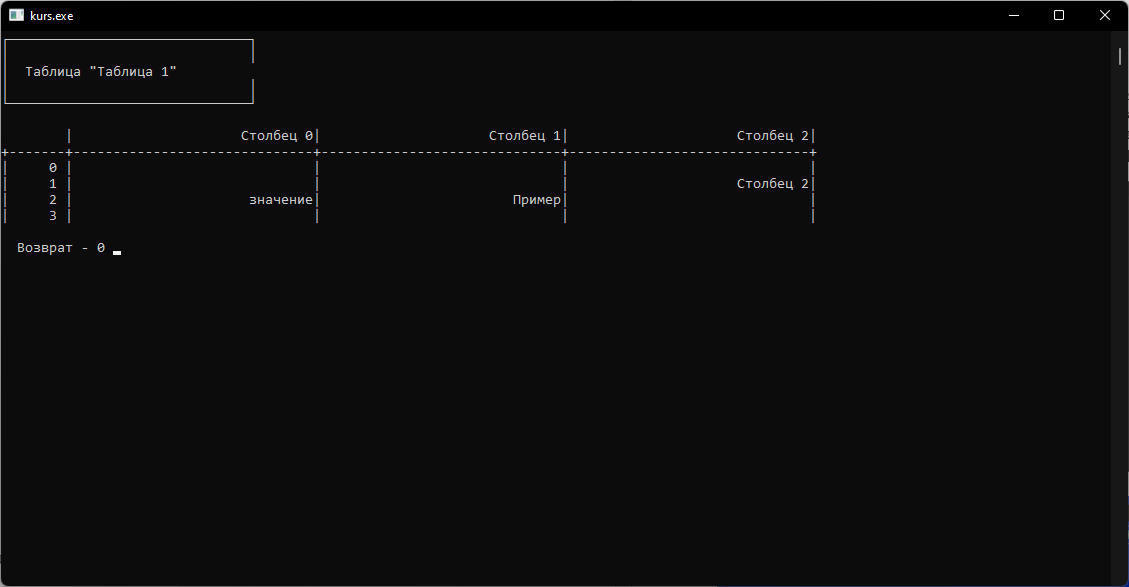


Рисунок 38 - Просмотр таблицы

При выборе действия «Редактировать» откроется меню с возможными действиями над таблицей, среди которых добавление/удаление столбцов, строк, изменение ячейки, поиск значений, сортировка.

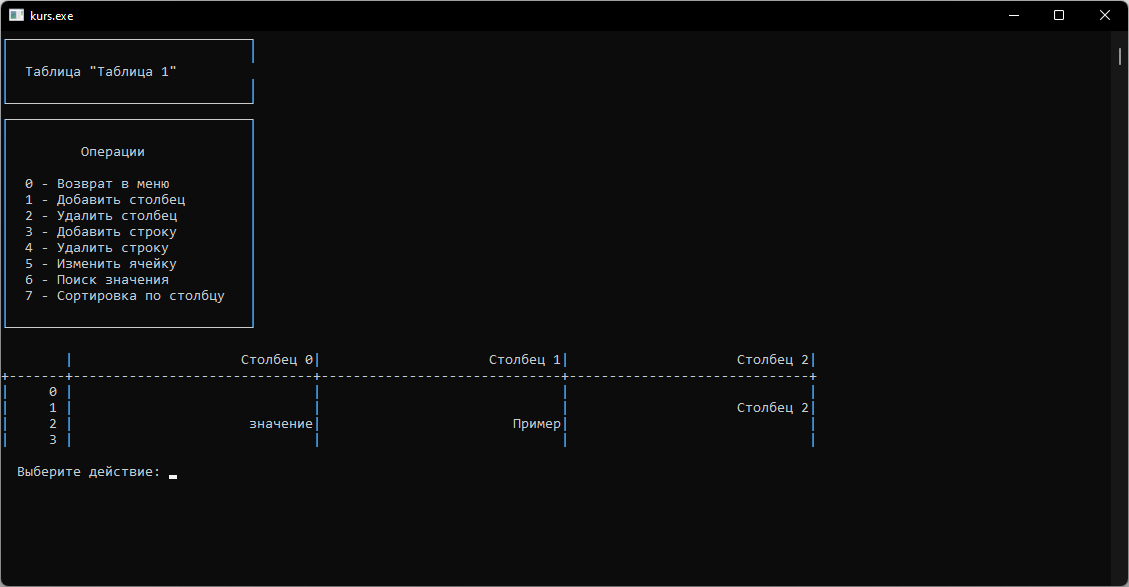


Рисунок 39 - Действия над базой данных

Результаты отладки

Была встречена ошибка в сегменте:

**var** count := Reset(f);

kurs.pas : Файл не найден

После анализа было выяснено, что до открытия файла отсутствовала проверка на его существование.

К сегменту была добавлена проверка:

if not fileexists('database.dat') then

begin

Rewrite(f);

Close(f);

exit;

end;

Была встречена ошибка в сегменте:

Result := false;

**for** i := 0 **to** countUsers **do**

**begin**

user := userList[i];

TUI.pas(304) : Неизвестное имя ‘countUsers’

В ходе анализа было выяснено, что константа countUsers не была определена.

Решение контрольных примеров и проверка правильности функционирования программы

Для проверки функциональности программы попробуем отсортировать таблицу.

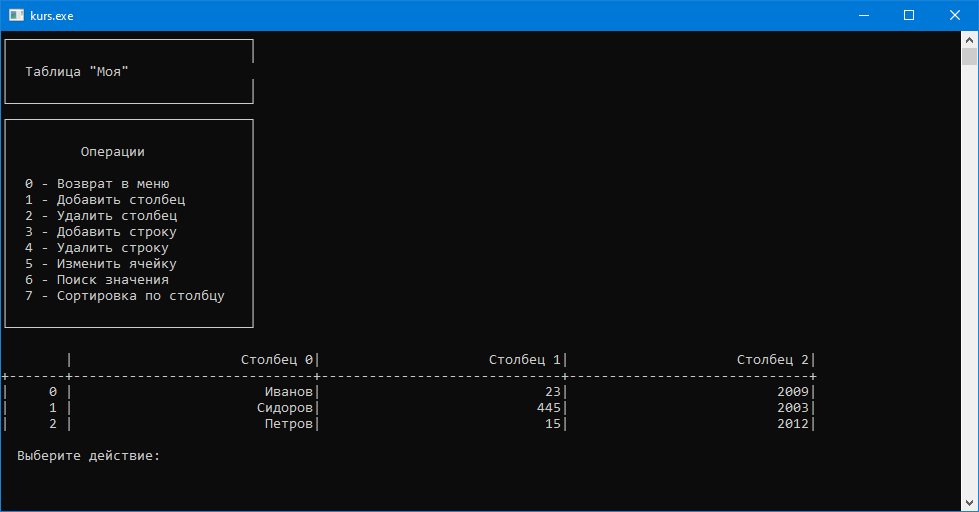


Рисунок 40 - Оригинальная БД

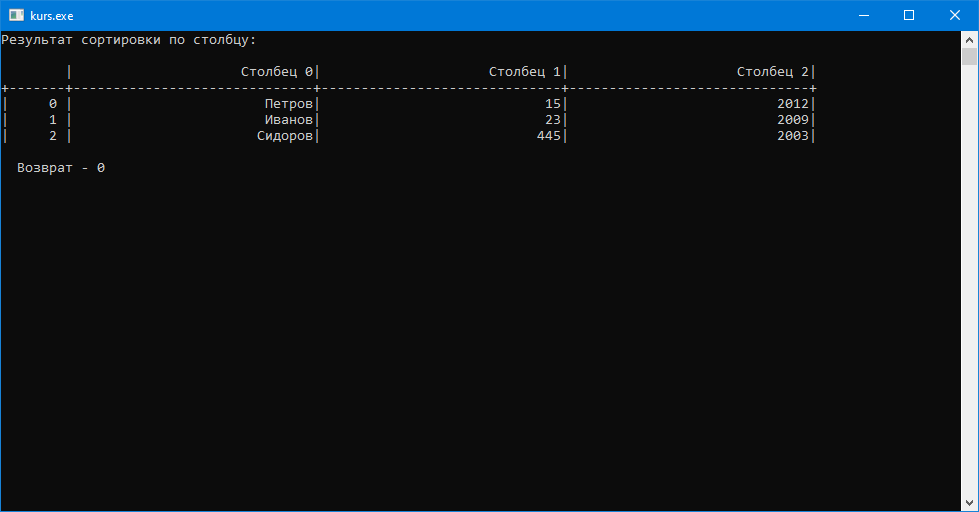


Рисунок 41 - Отсортированная по модели

Календарь выполнения работ

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Этапы решения задачи | Ориентировочная трудоёмкость, ч | Реально затраченное время, ч |
| 1 | Анализ предметной области и формализация задачи | 5 | 4 |
| 2 | Разработка структуры организации данных | 2 | 1 |
| 3 | Составление алгоритма | 4 | 9 |
| 4 | Разработка схемы интерфейса | 1 | 1 |
| 5 | Написание программы | 3 | 4 |
| 6 | Устранение синтаксических ошибок | 1 | 0,5 |
| 7 | Подготовка тестов |  | 0 |
| 8 | Тестирование и отладка | 5 | 10 |
| 9 | Подготовка документации | 10 | 5 |
| 10 | Внесение правок | 4 | 6 |
| Итого | | 35 | 40.5 |

Заключение

По результатам выполнения курсовой работы разработана программа создания-редактирования-удаления таблиц произвольной длины с произвольным числом столбцов. Пользователи имеют разграничение в правах. Все необходимые данные хранятся во внешних файлах. Также присутствует проверка на вводимые пользователем данные.

Таким образом задача курсовой работы выполнена.

Приложение: код программы

{ Программа: Информационная система }

{ Переменные: }

{ currentUser - текущий авторизованный пользователь }

{ isAuth - авторизован ли пользователь }

{ tables - массив таблиц }

{ userList - массив пользователей }

**program** informationSystem;

**uses** crt;

**const**

countUsers = 3;

sec = 2000;

countAttempts = 5;

maxLength = 30;

**type**

AuthData = **record**

login: ShortString;

pass: ShortString;

isAdmin: boolean;

**end**;

str = string[maxLength];

Users = **array**[0..countUsers] **of** AuthData;

{ Тип для записи в файл }

FileTable = **record**

name: str;

countRow: integer;

countColumn: integer;

data: **array** [0..100, 0..100] **of** str;

**end**;

StrArray = **record**

isDigit: boolean;

value: **array of** str;

**end**;

Table = **record**

name: str;

data: **array of** StrArray;

**end**;

**var**

currentUser : AuthData;

isAuth : boolean;

tables : **array of** Table;

userList : Users;

{ Функция для подтверждения выбора пользователя }

{ Параметры: message - выводимое пользователю сообщение }

{ value - введённое пользователем значение }

**function** actionConfirmed(message: string): boolean;

**var**

value: string;

**begin**

Write(message);

Readln(value);

Result := value = 'Да';

**end**;

{ Функция для подтверждения выбора индекса из массива }

{ Параметры: message - выводимое пользователю сообщение }

{ mas - массив, для которого проверяется индекс }

**function** indexConfirmed(message: string; mas: System.**Array**): integer;

**var**

temp: integer;

line: string;

code: integer;

**begin**

code := 0;

**repeat**

write(' ', message);

Readln(line);

VAL(line, temp, code);

**if** code <> 0 **then** writeln('Должно быть число');

**until** (code = 0) **and** (temp >= 0) **and** (temp < Length(mas));

Result := temp;

**end**;

{ Функция для подтверждения выбранной операции }

{ Параметры: min - минимальное число }

{ max - максимально число }

**function** selectOperation(min: integer; max: integer): integer;

**var**

temp: integer;

line: string;

code: integer;

**begin**

code := 0;

**repeat**

write(' Выберите действие: ');

Readln(line);

VAL(line, temp, code);

**if** code <> 0 **then** writeln('Должно быть число');

**until** (code = 0) **and** ((temp >= min) **and** (temp <= max));

Result := temp;

**end**;

{ Процедура для подтверждения ввода кода возврата }

**procedure** comeBack;

**var**

temp: integer;

code: integer;

answer: string;

**begin**

write(' Возврат - 0 ');

**repeat**

Readln(answer);

VAL(answer, temp, code);

**until** (code = 0) **and** (temp = 0);

writeln();

**end**;

{ Процедура для вывода содержимого массива в виде таблицы }

{ Параметры: data - данные для вывода в терминал }

**procedure** printTable(data: **array of** StrArray);

**var**

countColumn, countRow: integer;

**begin**

writeln();

write(' ':8, '|');

countColumn := Length(data);

countRow := Length(data[0].value);

**for var** i := 0 **to** countColumn - 1 **do**

write((' Столбец ' + inttostr(i)):30, '|');

writeln();

write('+', '-' \* 7, '+');

**for var** i := 0 **to** countColumn - 1 **do**

write('-' \* 30, '+');

writeln();

**for var** i := 0 **to** countRow - 1 **do**

**begin**

**for var** j := 0 **to** countColumn - 1 **do**

**begin**

**if** j = 0 **then**

write('| ', i:5, ' |');

write(data[j].value[i]:30, '|');

**end**;

writeln();

**end**;

writeln();

**end**;

{ Функция для преобразования FileTable в Table }

{ Параметры: fileTable - переменная для преобразования }

**function** convertToTable(fileTable: FileTable): Table;

**var**

table: Table;

**begin**

table.name := fileTable.name;

table.data := **new** StrArray[fileTable.countColumn];

**for var** i := 0 **to** fileTable.countColumn - 1 **do**

**begin**

table.data[i].value := **new** str[fileTable.countRow];

**for var** j := 0 **to** fileTable.countRow - 1 **do**

table.data[i].value[j] := fileTable.data[i][j];

**end**;

Result := table;

**end**;

{ Функция для преобразования Table в FileTable }

{ Параметры: table - переменная для преобразования }

**function** convertToFileTable(table: Table): FileTable;

**var**

fileTable: FileTable;

**begin**

fileTable.name := table.name;

fileTable.countColumn := Length(table.data);

**if** (table.data = nil) **or** (fileTable.countColumn = 0) **then**

**begin**

Result := fileTable;

**exit**;

**end**;

fileTable.countRow := Length(table.data[0].value);

**for var** i := 0 **to** fileTable.countRow - 1 **do**

**for var** j := 0 **to** fileTable.countColumn - 1 **do**

fileTable.data[j][i] := table.data[j].value[i];

Result := fileTable;

**end**;

{ Функция для чтения пользователей из файла }

**function** readAuthData(): Users;

**var**

user: AuthData;

users: Users;

f: **file of** AuthData;

**begin**

Assign(f, 'auth.data');

Reset(f);

**var** i := 0;

**while not** Eof(f) **do**

**begin**

Read(f, user);

users[i] := user;

i := i + 1;

**end**;

Close(f);

Result := users;

**end**;

{ Функция для удаления таблицы из массива таблиц по индексу }

{ Параметры: V - массив таблиц }

{ Index - индекс таблицы для удаления }

**function** deleteElement(V: **array of** Table; Index: Integer): **array of** Table;

**var**

NewSize: Integer;

Ind: Integer;

**begin**

NewSize := Length(V) - 1;

**if** Index > NewSize {Length(V) - 1} **then**

**raise** Exception.Create('Указанный элемент не существует');

SetLength(Result, NewSize);

Ind := 0;

**for var** I := 0 **to** High(V) **do**

**begin**

**if** I <> Index **then**

**begin**

Result[Ind] := V[I];

Inc(Ind);

**end**;

**end**;

**end**;

{ Функция для удаления столбца из массива столбцов по индексу }

{ Параметры: current - массив столбцов }

{ Index - индекс столца для удаления }

**function** deleteColumn(current: **array of** StrArray; Index: Integer): **array of** StrArray;

**var**

size: Integer;

Ind: Integer;

**begin**

size := Length(current) - 1;

**if** Index > size **then**

**begin**

Writeln('Указанный элемент не существует');

Result := current;

**exit**

**end**;

SetLength(Result, size);

Ind := 0;

**for var** I := 0 **to** High(current) **do**

**begin**

**if** I <> Index **then**

**begin**

Result[Ind] := current[I];

Inc(Ind);

**end**;

**end**;

**end**;

{ Функция для удаления строки из массива строк по индексу }

{ Параметры: current - массив строк }

{ Index - индекс строки для удаления }

**function** deleteRow(current: StrArray; Index: Integer): StrArray;

**var**

size: Integer;

I, Ind: Integer;

**begin**

size := Length(current.value) - 1;

**if** Index > size {Length(V) - 1} **then**

**begin**

Writeln('Указанный элемент не существует');

Result := current;

**exit**

**end**;

SetLength(Result.value, size);

Ind := 0;

**for** I := 0 **to** High(current.value) **do**

**begin**

**if** I <> Index **then**

**begin**

Result.value[Ind] := current.value[I];

Inc(Ind);

**end**;

**end**;

**end**;

{ Функция для проверки аутентификации пользователя }

{ Параметры: login - логин }

{ pass - пароль }

**function** isAuthSuccess(login, pass: string): boolean;

**var**

user: AuthData;

i: integer;

**begin**

Result := false;

**for** i := 0 **to** countUsers **do**

**begin**

user := userList[i];

**if** ((user.login = login) **and** (user.pass = pass)) **then**

**begin**

Result := true;

currentUser := user;

**break**;

**end**;

**end**;

**end**;

{ Процедура для вывода заставки }

**procedure** displaySplashScreen;

**begin**

writeln('┌──────────────────────────────┐');

writeln('│ │');

writeln('│ Информационная система │');

writeln('│ │');

writeln('└──────────────────────────────┘');

delay(sec);

ClrScr;

**end**;

{ Процедура для вывода приветственного сообщения }

**procedure** displayWelcomeMessage;

**begin**

writeln(' Добро пожаловать, ', currentUser.login);

writeln();

writeln(' Загрузка базы данных...');

delay(sec);

ClrScr;

**end**;

{ Процедура для вывода авторизации }

**procedure** displayAuthScreen;

**var**

login, pas: string;

**begin**

writeln('┌──────────────────────────────┐');

writeln('│ │');

writeln('│ Авторизация │');

writeln('│ │');

writeln('└──────────────────────────────┘');

write(' Логин: ');

Readln(login);

write(' Пароль: ');

Readln(pas);

isAuth := isAuthSuccess(login, pas);

**if** (isAuth) **then**

**begin**

Writeln(' Авторизация прошла успешно!');

**end**

**else**

**begin**

Writeln(' Неверный логин или пароль!');

delay(1000);

**end**;

ClrScr;

**end**;

{ Процедура для чтения таблиц из файла }

**procedure** readTables;

**var**

f: **file of** FileTable;

table: FileTable;

**begin**

Assign(f, 'database.dat');

**if not** fileexists('database.dat') **then**

**begin**

Rewrite(f);

Close(f);

**exit**;

**end**;

Reset(f);

**var** i := 0;

**var** count := FileSize(f);

SetLength(tables, count);

**while not** Eof(f) **do**

**begin**

Read(f, table);

tables[i] := convertToTable(table);

i := i + 1;

**end**;

Close(f);

**end**;

{ Процедура для записи таблиц в файл }

**procedure** writeTables;

**var**

f: **file of** FileTable;

table: FileTable;

i: integer;

**begin**

Assign(f, 'database.dat');

Rewrite(f);

**if** (Length(tables) = 0) **then**

**begin**

Close(f);

**exit**;

**end**;

**for** i := 0 **to** Length(tables) - 1 **do**

**begin**

table := convertToFileTable(tables[i]);

Write(f, table);

**end**;

Close(f);

**end**;

{ Процедура для вывода названий таблиц }

**procedure** displayTables;

**var**

i: integer;

**begin**

writeln('┌──────────────────────────────┐');

writeln('│ │');

writeln('│ Таблицы │');

writeln('│ │');

**if** (Length(tables) = 0) **then**

writeln('│ Не создано ни одной таблицы │')

**else**

**begin**

**var** table: Table;

**for** i := 0 **to** Length(tables) - 1 **do**

**begin**

table := tables[i];

writeln('| ', i, ' - ', table.name);

**end**;

**end**;

writeln('│ │');

writeln('└──────────────────────────────┘');

**end**;

{ Процедура для создания таблицы }

**procedure** createTable;

**var**

table: Table;

name: str;

last: integer;

**begin**

Write('Введите имя таблицы: ');

Readln(name);

table.name := name;

table.data := **new** StrArray[0];

last := Length(tables);

SetLength(tables, Length(tables) + 1);

tables[last] := table;

ClrScr;

**end**;

{ Процедура для удаления таблицы }

**procedure** deleteTable;

**var**

num: integer;

answer: string[3];

**begin**

num := indexConfirmed('Выберите № таблицы: ', tables);

**if not** actionConfirmed('Удалить таблицу? (Да/Нет)') **then**

**begin**

ClrScr;

**exit**;

**end**;

tables := deleteElement(tables, num);

ClrScr;

**end**;

{ Процедура для отображения имени текущей таблицы и её содержимого }

**procedure** viewTable();

**var**

current: Table;

oper: integer;

num: integer;

**begin**

num := indexConfirmed('Выберите № таблицы: ', tables);

ClrScr;

current := tables[num];

writeln('┌──────────────────────────────┐');

writeln('│ │');

writeln('│ Таблица "', current.name, '"');

writeln('│ │');

writeln('└──────────────────────────────┘');

**var** countRow := Length(current.data);

**if** (countRow <> 0) **then**

printTable(current.data);

comeBack();

ClrScr;

**end**;

{ Функция для добавления столбца в таблицу }

{ Параметры: current - текущая таблица }

**function** addColumn(current: Table): Table;

**var**

last: integer;

isDigit: boolean;

**begin**

last := Length(current.data);

SetLength(current.data, last + 1);

**var** len := Length(current.data[0].value);

current.data[last].value := **new** str[len];

isDigit := actionConfirmed('Тип столбца числовой? (Да/Нет)');

current.data[last].isDigit := isDigit;

Result := current;

**end**;

{ Функция для удаления столбца из таблицы }

{ Параметры: current - текущая таблица }

**function** delColumn(current: Table): Table;

**var**

num: integer;

answer: string[3];

**begin**

num := indexConfirmed('Выберите № столбца: ', current.data);

**if not** actionConfirmed('Удалить столбец? (Да/Нет)') **then**

**begin**

Result := current;

**exit**;

**end**;

current.data := deleteColumn(current.data, num);

Result := current;

**end**;

{ Функция для создания строки в таблице }

{ Параметры: current - текущая таблица }

**function** addRow(current: Table): Table;

**var**

last: integer;

**begin**

**if** Length(current.data) = 0 **then**

**begin**

Result := current;

**exit**;

**end**;

last := Length(current.data[0].value);

**for var** i := 0 **to** Length(current.data) - 1 **do**

SetLength(current.data[i].value, last + 1);

Result := current;

**end**;

{ Функция для удаления строки из таблицы }

{ Параметры: current - текущая таблица }

**function** delRow(current: Table): Table;

**var**

num: integer;

**begin**

**if** Length(current.data) = 0 **then**

**begin**

Result := current;

**exit**;

**end**;

num := indexConfirmed('Выберите № строки: ', current.data[0].value);

**if not** actionConfirmed('Удалить строку? (Да/Нет)') **then**

**begin**

Result := current;

**exit**;

**end**;

**for var** i := 0 **to** Length(current.data) - 1 **do**

current.data[i] := deleteRow(current.data[i], num);

Result := current;

**end**;

{ Функция для добавления значения в ячейку таблицы }

{ Параметры: current - текущая таблица }

**function** insertValue(current: Table): Table;

**var**

row, column: integer;

value, line: string;

code: integer;

temp: real;

**begin**

column := indexConfirmed('Выберите № столбца: ', current.data);

row := indexConfirmed('Выберите № строки: ', current.data[0].value);

**if** current.data[column].isDigit **then**

**begin**

**repeat**

write(' Введите значение: ');

Readln(line);

VAL(line, temp, code);

**if** code <> 0 **then** writeln('Должно быть число');

**until** code = 0;

value := FloatToStr(temp);

**end**

**else**

**begin**

write(' Введите значение: ');

Readln(value);

**end**;

current.data[column].value[row] := value;

Result := current;

**end**;

{ Процедура для поиска значения в таблице }

{ Параметры: current - текущая таблица }

**procedure** findValue(current: Table);

**var**

value: string;

currentData: string;

count: integer;

**begin**

write(' Введите значение: ');

Readln(value);

**for var** i := 0 **to** Length(current.data) - 1 **do**

**for var** j := 0 **to** Length(current.data[i].value) - 1 **do**

**begin**

currentData := current.data[i].value[j];

**if** currentData = value **then**

count := count + 1;

**end**;

**if** count = 0 **then**

Writeln('"', value, '" не найдено')

**else**

Writeln('"', value, '" найдено ', count, ' раз');

comeBack();

**end**;

{ Функция для сортировки по возрастанию }

{ Параметры: array - массив для сортировки }

{ k - номер столбца, по которому сортируем }

**function** sortDescending(arr: **array of** StrArray; k: integer): **array of** StrArray;

**var**

temp: str;

**begin**

**var** a := copy(arr);

**for var** i := 0 **to** Length(arr) - 1 **do**

a[i].value := copy(arr[i].value);

**for var** i := 0 **to** Length(arr[0].value) - 1 **do**

**for var** j := i + 1 **to** Length(arr[0].value) - 1 **do**

**begin**

write(i, j, ' ');

**if** (a[k].value[i] > a[k].value[j]) **then**

**begin**

**for var** d := 0 **to** Length(arr) - 1 **do**

**begin**

temp := a[d].value[i];

a[d].value[i] := a[d].value[j];

a[d].value[j] := temp;

**end**;

**end**;

**end**;

Result := a;

**end**;

{ Функция для сортировки по убыванию }

{ Параметры: array - массив для сортировки }

{ k - номер столбца, по которому сортируем }

**function** sortAscending(arr: **array of** StrArray; k: integer): **array of** StrArray;

**var**

temp: str;

**begin**

**var** a := copy(arr);

**for var** i := 0 **to** Length(arr) - 1 **do**

a[i].value := copy(arr[i].value);

**for var** i := 0 **to** Length(arr[0].value) - 1 **do**

**for var** j := i + 1 **to** Length(arr[0].value) - 1 **do**

**begin**

write(i, j, ' ');

**if** (a[k].value[i] < a[k].value[j]) **then**

**begin**

**for var** d := 0 **to** Length(arr) - 1 **do**

**begin**

temp := a[d].value[i];

a[d].value[i] := a[d].value[j];

a[d].value[j] := temp;

**end**;

**end**;

**end**;

Result := a;

**end**;

{ Процедура для сортировки текущей таблицы }

{ Параметры: current - текущая таблица }

**procedure** sortTable(current: Table);

**var**

column: integer;

sortData: **array of** StrArray;

isDesc: boolean;

**begin**

column := indexConfirmed('Выберите № столбца: ', current.data);

isDesc := actionConfirmed('Сортировать по возрастанию? (Да/Нет)');

**if** isDesc **then**

sortData := sortDescending(current.data, column)

**else**

sortData := sortAscending(current.data, column);

ClrScr;

writeln('Результат сортировки по столбцу:');

printTable(sortData);

comeBack();

**end**;

{ Процедура для изменения ячейки таблицы }

**procedure** editTable();

**var**

current: Table;

num: integer;

**begin**

num := indexConfirmed('Выберите № таблицы: ', tables);

ClrScr;

current := tables[num];

**while** true **do**

**begin**

writeln('┌──────────────────────────────┐');

writeln('│ │');

writeln('│ Таблица "', current.name, '"');

writeln('│ │');

writeln('└──────────────────────────────┘');

writeln('┌──────────────────────────────┐');

writeln('│ │');

writeln('│ Операции │');

writeln('│ │');

writeln('│ 0 - Возврат в меню │');

writeln('│ 1 - Добавить столбец │');

writeln('│ 2 - Удалить столбец │');

writeln('│ 3 - Добавить строку │');

writeln('│ 4 - Удалить строку │');

writeln('│ 5 - Изменить ячейку │');

writeln('│ 6 - Поиск значения │');

writeln('│ 7 - Сортировка по столбцу │');

writeln('│ │');

writeln('└──────────────────────────────┘');

**var** countColumn := Length(current.data);

**if** (countColumn <> 0) **then**

printTable(current.data);

**var** oper: integer;

oper := selectOperation(0, 7);

**case** oper **of**

0: **begin** ClrScr; **exit**; **end**;

1: current := addColumn(current);

2: current := delColumn(current);

3: current := addRow(current);

4: current := delRow(current);

5: current := insertValue(current);

6: findValue(current);

7: sortTable(current);

**end**;

tables[num] := current;

ClrScr;

**end**;

**end**;

{ Процедура для вывода меню админа }

**procedure** displayAdminMenu;

**var**

oper: integer;

**begin**

writeln('┌──────────────────────────────┐');

writeln('│ │');

writeln('│ Операции с таблицами │');

writeln('│ │');

writeln('│ 1 - Просмотреть │');

writeln('│ 2 - Создать │');

writeln('│ 3 - Редактировать │');

writeln('│ 4 - Удалить │');

writeln('│ │');

writeln('└──────────────────────────────┘');

oper := selectOperation(1, 4);

**case** oper **of**

1: viewTable();

2:

**begin**

createTable();

writeTables();

**end**;

3:

**begin**

editTable();

writeTables();

**end**;

4:

**begin**

deleteTable();

writeTables();

**end**;

**end**;

**end**;

{ Процедура для вывода меню пользователя }

**procedure** displayUserMenu;

**var**

oper: integer;

**begin**

writeln('┌──────────────────────────────┐');

writeln('│ │');

writeln('│ Операции с таблицами │');

writeln('│ │');

writeln('│ 1 - Просмотреть │');

writeln('│ 2 - Редактировать │');

writeln('│ │');

writeln('└──────────────────────────────┘');

oper := selectOperation(1, 2);

**case** oper **of**

1: viewTable();

2:

**begin**

editTable();

writeTables();

**end**;

**end**;

**end**;

**begin**

displaySplashScreen();

userList := readAuthData();

**var** attempt := 0;

**while** (attempt < countAttempts) **and not** isAuth **do**

**begin**

displayAuthScreen();

attempt := attempt + 1;

**end**;

**if** (**not** isAuth) **then**

**begin**

writeln(' Прeвышено количество попыток: ', countAttempts);

**exit**;

**end**;

displayWelcomeMessage();

readTables();

**while** True **do**

**begin**

displayTables();

**if** (currentUser.isAdmin) **then**

displayAdminMenu()

**else**

displayUserMenu();

**end**;

**end**.