Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №8**

**«ОСНОВЫ СОБЫТИЙНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ»**

**ПО**

**МДК 05.02 РАЗРАБОТКА КОДА ИНФОРМАЦИОННЫХ СИСТЕМ**

Выполнил: студент учебной группы

ИСПк-205-52-00

Ильин Тимофей Анатольевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров 2024

Цель работы: получить представление об организации хранения и использования информации посредством типизированных файлов, закрепить навыки создания пользовательского

**Задание**

1. ﻿﻿﻿Сформировать структуру записи, с которой будет осуществляться работа, и согласовать ее с преподавателем.
2. ﻿﻿﻿Разработать приложение, позволяющее извлекать набор записей из типизированного файла, визуализировать данный набор, а также позволяющее изменять данных и выполнять сохранение проделанных изменений.

**Описание алгоритма**

1. Открывается форма приложения TfMain с элементами управления.

2. Пользователь может добавлять, редактировать, удалять и сортировать данные о контактах с помощью кнопок bAdd, bEdit, bDel, и bSort.

3. При нажатии на кнопку bAdd, открывается форма редактирования fEdit, где пользователь может вводить данные о контакте.

4. После заполнения всех полей и нажатия кнопки "Сохранить" на форме fEdit, данные добавляются в таблицу SG на форме TfMain.

5. При нажатии на кнопку bDel, выбранная строка в таблице SG удаляется после подтверждения операции.

6. При нажатии на кнопку bEdit, выбранная строка в таблице SG загружается в форму редактирования fEdit, где пользователь может изменить данные о контакте.

7. После внесения изменений и нажатия "Сохранить" на форме fEdit, измененные данные обновляются в таблице SG.

8. При нажатии на кнопку bSort, данные в таблице SG сортируются по возрастанию в первом столбце.

9. В случае ошибочного ввода или незаполненных полей, программа выводит соответствующие сообщения об ошибке.

**Код основной программы**

unit Unit1;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls, Buttons, edit,

Grids, StdCtrls;

type

Contacts = record

Title: string[100];

Proizvoditel: string[100];

Model: string[100];

Ves: integer;

Price: double;

end;

type

{ TfMain }

TfMain = class(TForm)

Panel1: TPanel;

bAdd: TSpeedButton;

bEdit: TSpeedButton;

bDel: TSpeedButton;

bSort: TSpeedButton;

SG: TStringGrid;

procedure bAddClick(Sender: TObject);

procedure bDelClick(Sender: TObject);

procedure bEditClick(Sender: TObject);

procedure bSortClick(Sender: TObject);

procedure FormClose(Sender: TObject; var CloseAction: TCloseAction);

procedure FormCreate(Sender: TObject);

private

function ValidateInput: Boolean;

public

end;

var

fMain: TfMain;

adres: string;

implementation

{$R \*.lfm}

{ TfMain }

function TfMain.ValidateInput: Boolean;

var

Price: Double;

Ves: Integer;

begin

Result := False;

if (fEdit.eTitle.Text = '') or (fEdit.eProizvoditel.Text = '') or

(fEdit.eModel.Text = '') or (fEdit.eVes.Text = '') or (fEdit.ePrice.Text = '') then

begin

ShowMessage('Заполните все поля');

Exit;

end;

if not TryStrToInt(fEdit.eVes.Text, Ves) then

begin

ShowMessage('Введите целое число');

Exit;

end;

if not TryStrToFloat(fEdit.ePrice.Text, Price) then

begin

ShowMessage('Не вводите букавки');

Exit;

end;

Result := True;

end;

procedure TfMain.bAddClick(Sender: TObject);

begin

// Очищаем поля, если там что-то есть:

fEdit.eTitle.Text := '';

fEdit.eProizvoditel.Text := '';

fEdit.eVes.Text := '';

fEdit.ePrice.Text := '';

// Устанавливаем ModalResult редактора в mrNone:

fEdit.ModalResult := mrNone;

// Теперь выводим форму:

fEdit.ShowModal;

// Если пользователь ничего не ввел - выходим:

if not ValidateInput then Exit;

// Если пользователь не нажал "Сохранить" - выходим:

if fEdit.ModalResult <> mrOk then Exit;

// Иначе добавляем в сетку строку, и заполняем её:

SG.RowCount := SG.RowCount + 1;

SG.Cells[0, SG.RowCount-1] := fEdit.eTitle.Text;

SG.Cells[1, SG.RowCount-1] := fEdit.eProizvoditel.Text;

SG.Cells[2, SG.RowCount-1] := fEdit.eModel.Text;

SG.Cells[3, SG.RowCount-1] := fEdit.eVes.Text;

SG.Cells[4, SG.RowCount-1] := fEdit.ePrice.Text;

end;

procedure TfMain.bDelClick(Sender: TObject);

begin

// Если данных нет - выходим:

if SG.RowCount = 1 then Exit;

// Иначе выводим запрос на подтверждение:

if MessageDlg('Требуется подтверждение',

'Вы действительно хотите удалить модель миксера"' +

SG.Cells[0, SG.Row] + '"?',

mtConfirmation, [mbYes, mbNo, mbIgnore], 0) = mrYes then

SG.DeleteRow(SG.Row);

end;

procedure TfMain.bEditClick(Sender: TObject);

begin

// Если данных в сетке нет - просто выходим:

if SG.RowCount = 1 then Exit;

// Иначе записываем данные в форму редактора:

fEdit.eTitle.Text := SG.Cells[0, SG.Row];

fEdit.eProizvoditel.Text := SG.Cells[1, SG.Row];

fEdit.eModel.Text := SG.Cells[2, SG.Row];

fEdit.eVes.Text := SG.Cells[3, SG.Row];

fEdit.ePrice.Text := SG.Cells[4, SG.Row];

// Устанавливаем ModalResult редактора в mrNone:

fEdit.ModalResult := mrNone;

// Теперь выводим форму:

fEdit.ShowModal;

// Сохраняем в сетку возможные изменения,

// если пользователь нажал "Сохранить":

if fEdit.ModalResult = mrOk then

begin

if not ValidateInput then Exit;

SG.Cells[0, SG.Row] := fEdit.eTitle.Text;

SG.Cells[1, SG.Row] := fEdit.eProizvoditel.Text;

SG.Cells[2, SG.Row] := fEdit.eModel.Text;

SG.Cells[3, SG.Row] := fEdit.eVes.Text;

SG.Cells[4, SG.Row] := fEdit.ePrice.Text;

end;

end;

procedure TfMain.bSortClick(Sender: TObject);

begin

// Если данных в сетке нет - просто выходим:

if SG.RowCount = 1 then Exit;

// Иначе сортируем список:

SG.SortColRow(True, 0);

end;

procedure TfMain.FormClose(Sender: TObject; var CloseAction: TCloseAction);

var

MyCont: Contacts; // для очередной записи

f: file of Contacts; // файл данных

i: Integer; // счетчик цикла

begin

// Если строки данных пусты, просто выходим:

if SG.RowCount = 1 then Exit;

// Иначе открываем файл для записи:

try

AssignFile(f, adres + 'Micser.dat');

Rewrite(f);

// Теперь цикл - от первой до последней записи сетки:

for i := 1 to SG.RowCount-1 do

begin

// Получаем данные текущей записи:

MyCont.Title := SG.Cells[0, i];

MyCont.Proizvoditel := SG.Cells[1, i];

MyCont.Model := (SG.Cells[2, i]);

MyCont.Ves := StrToint(SG.Cells[3, i]);

MyCont.Price := StrTofloat(SG.Cells[4, i]);

// Записываем их:

Write(f, MyCont);

end;

finally

CloseFile(f);

end;

end;

procedure TfMain.FormCreate(Sender: TObject);

var

MyCont: Contacts; // для очередной записи

f: file of Contacts; // файл данных

i: Integer; // счетчик цикла

begin

// Сначала получим адрес программы:

adres := ExtractFilePath(ParamStr(0));

// Настроим сетку:

SG.Cells[0, 0] := 'Название миксера';

SG.Cells[1, 0] := 'Производитель';

SG.Cells[2, 0] := 'Модель';

SG.Cells[3, 0] := 'Вес(г)';

SG.Cells[4, 0] := 'Стоимость';

SG.ColWidths[0] := 200;

SG.ColWidths[1] := 200;

SG.ColWidths[2] := 100;

SG.ColWidths[3] := 150;

SG.ColWidths[4] := 150;

// Если файла данных нет, просто выходим:

if not FileExists(adres + 'Micser.dat') then Exit;

// Иначе файл есть, открываем его для чтения и

// считываем данные в сетку:

try

AssignFile(f, adres + 'Micser.dat');

Reset(f);

// Теперь цикл - от первой до последней записи сетки:

while not Eof(f) do

begin

// Считываем новую запись:

Read(f, MyCont);

// Добавляем в сетку новую строку, и заполняем её:

SG.RowCount := SG.RowCount + 1;

SG.Cells[0, SG.RowCount-1] := MyCont.Title;

SG.Cells[1, SG.RowCount-1] := MyCont.Proizvoditel;

SG.Cells[2, SG.RowCount-1] := (MyCont.Model);

SG.Cells[3, SG.RowCount-1] := IntToStr(MyCont.Ves);

SG.Cells[4, SG.RowCount-1] := FloatToStr(MyCont.Price);

end;

finally

CloseFile(f);

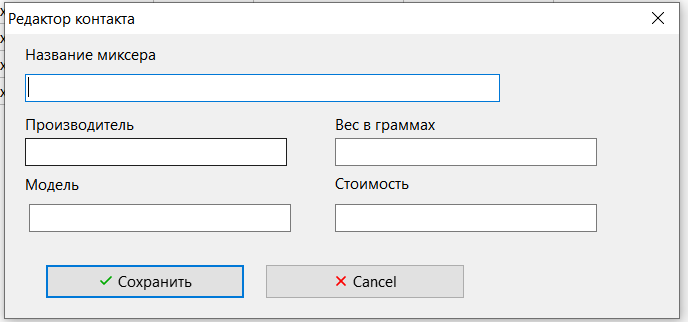
end;

end;

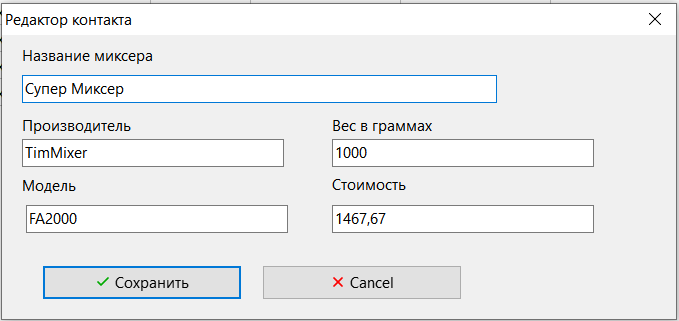
end.

**Результат выполнения программы**

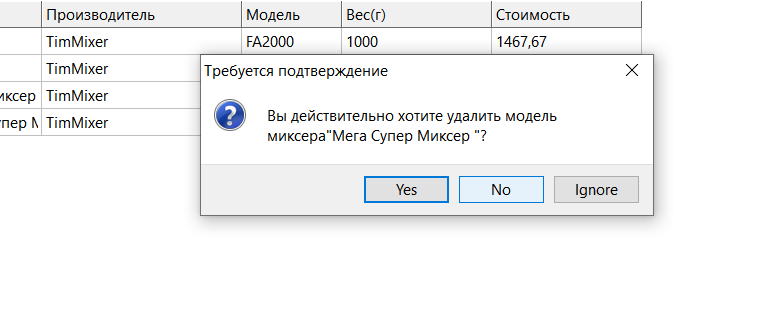
1. Добавление модели миксера



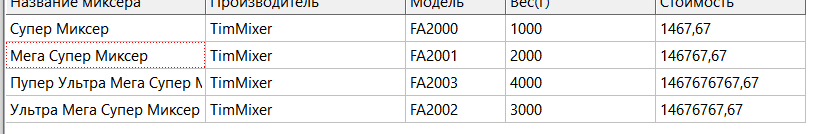
1. Редактирование модели миксера



1. Удаление модели миксера



1. Сортировка по алфавиту



Вывод: в ходе выполнения работы были успешно достигнуты поставленные цели. Было освоено использование типизированных файлов для организации хранения информации, что позволяет удобно структурировать данные и обеспечить их доступность. Также были закреплены навыки создания пользовательского интерфейса, что позволяет пользователям взаимодействовать с программой более эффективно и удобно.

Работа над проектом позволила не только расширить теоретические знания об организации хранения данных и создании интерфейсов, но и применить их на практике. Полученный опыт в разработке программы для работы с данными и пользовательским интерфейсом будет полезен для дальнейшего развития навыков программирования и проектирования ПО.

Таким образом, выполнение работы помогло не только понять основы организации информации и создания интерфейсов, но и приобрести опыт, который может быть использован в будущих проектах.