|  |  |
| --- | --- |
|  | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ: Информатики и систем управления

КАФЕДРА: Компьютерные системы и сети

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

Студент Боярских Никита Игоревич

*фамилия, имя, отчество*

Группа ИУ6-22

Тип практики Практикум по программированию

Название предприятия МГТУ им. Н.Э. Баумана

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Боярских Н. И.\_\_\_**

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Черноусова Т. Г. \_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2017 г.*

Оглавление

Введение 2

1. Объектная декомпозиция 3

2. Главная форма 3

Диаграмма классов 3

Диаграмма состояний пользовательского интерфейса 4

Схемы алгоритмов программы 5

Схема алгоритма быстрой сортировки 5

Схема алгоритма загрузки таблицы из CSV файла 6

Схема алгоритма отрисовки графика по точкам 7

Заключение 10

Литература 11

Приложения 12

Приложение №1 (Код программы) 12

Приложение №2 (Скриншоты программы) 23

**Создание программной системы**

**Введение**

Файл содержит сведения об элементах: номер элемента по таблице Менделеева, название элемента, удельный вес, проводимость (проводник, изолятор, полупроводник). Программа должна в интерактивном режиме воспринимать каждый из перечисленных запросов и давать на него ответ.

1. Найти названия и удельные веса всех полупроводников.

2. Вывести сведения о проводниках в порядке возрастания их удельных весов.

3. Определить названия всех изоляторов, удельные веса которых не превышают заданного значения.

4. Построить график зависимости удельного веса элемента от его номера.

Целью работы является выполнение объектной декомпозиции, разработка интерфейса, диаграмм состояний интерфейса, диаграмм классов интерфейсной и предметной областей:разработка, тестирование и отладка программы в среде TurboDelphi.

**Объектная декомпозиция**



*Рисунок 1 – Объектная декомпозиция, созданная в среде Microsoft Office Visio © [4]*

**Главная форма**

Главное меню программы используется для выбора действия, необходимого пользователю. Диаграмма классов представлена на рисунке 2, диаграмма состояний пользовательского интерфейса на рисунке 3, изображение главной формы на рисунке 7 в приложении №1.

**Диаграмма классов**



*Рисунок 2 – диаграмма классов, созданная в среде Microsoft Office Visio © [4]*

**Диаграмма состояний пользовательского интерфейса**



*Рисунок 3 – диаграмма состояний пользовательского интерфейса, созданная в среде Microsoft Office Visio © [4]*

**Схемы алгоритмов программы ПР№1**



*Рисунок 4 – Схема алгоритма быстрой сортировки, созданная в среде*

*Microsoft Office Visio © [4]*



*Рисунок 5 – Схема алгоритма загрузки таблицы из CSV файла, созданная в среде*

*Microsoft Office Visio © [4]*



*Рисунок 6.1 – Схема алгоритма отрисовки графика по точкам, созданная в среде*

*Microsoft Office Visio © [4]*



*Рисунок 6.2 – Схема алгоритма отрисовки графика по точкам, созданная в среде*

*Microsoft Office Visio © [4]*



*Рисунок 6.3 – Схема алгоритма отрисовки графика по точкам, созданная в среде*

*Microsoft Office Visio © [4]*

**Текст программы ПР№1**

Код программы в приложении №1. [1 – 3, 5 – 9]

**Скриншоты, демонстрирующие работу программы ПР№1**

Скриншоты программы в приложении №2.

**Заключение**

Разработаны алгоритмы построения графика по точкам [3], парсинга CSV таблины [7], сортировки таблицы по типу проводимости, удельному весу и порядковому номеру элемента [9], вывода подготовленной таблицы на экран в удобном пользователю виде [7]; подготовлена CSV таблица, содержащая элементы периодической системы химических элементов [9], построены схемы алгоритмов, объектная декомпозиция, диаграмма классов и диаграмма состояний пользовательского интерфейса [2, 4], создана программа ПР№1 [1 – 3, 5 – 9], реализующая данные алгоритмы, успешно проведено функциональное и модульное тестирования программы (корректность работы программы подтверждают рисунки 1-11).

**Литература**

1. д.т.н., проф. Иванова Галина Сергеевна: Лекции по объектно-ориентированному программированию МГТУ им. Н.Э. Баумана, «факультет Информатика и системы управления», кафедра «Компьютерные системы и сети». МОСКВА. 2017 год МГТУ им. Баумана
2. д.т.н., проф. Иванова Галина Сергеевна: Лекции по основам программирования МГТУ им. Н.Э. Баумана, «факультет Информатика и системы управления», кафедра «Компьютерные системы и сети». МОСКВА. 2016 год МГТУ им. Баумана
3. д.т.н., проф. Иванова Галина Сергеевна: Методические материалы по основам программирования МГТУ им. Н.Э. Баумана, «факультет Информатика и системы управления», кафедра «Компьютерные системы и сети». МОСКВА. 2016 год МГТУ им. Баумана
4. «Создание схем алгоритмов средствами Microsoft Visio и OpenOffice Draw. Методические указания по выполнению лабораторных работ и практикума по курсу Алгоритмические языки и программирование», МГТУ им. Н.Э. Баумана, факультет «Информатика и Системы Управления», кафедра ИУ-6 «Компьютерные системы и сети», Самарев Роман Станиславович. МОСКВА. 2010 год МГТУ им. Баумана
5. «Программирование под Windows в среде TurboDELPHI. Методические указания по выполнению лабораторной работы по дисциплине Основы программирования», МГТУ им. Н.Э. Баумана, факультет «Информатика и Системы Управления», кафедра ИУ-6 «Компьютерные системы и сети», Г. С. Иванова, Т.Н. Ничушкина. МОСКВА. 2006 год МГТУ им. Баумана
6. «Интерфейсные компоненты Visual Components Library. Методические указания по выполнению лабораторных работ и домашних заданий в среде TurboDELPHI», МГТУ им. Н.Э. Баумана, факультет «Информатика и Системы Управления», кафедра ИУ-6 «Компьютерные системы и сети», Г. С. Иванова, Т.Н. Ничушкина. МОСКВА. 2006 год МГТУ им. Баумана
7. <http://www.cyberforum.ru/> (версия от 15.03.17)
8. <https://www.stackoverflow.com/> (версия от 15.03.17)
9. <https://www.wikipedia.org/> (версия от 15.03.17)

**Приложение №1**

Код программы ПР№1.

program PR1;

uses

Forms,

Main in 'Main.pas' {frmMain},

CSVParser in 'CSVParser.pas',

Graphic in 'Graphic.pas';

{$R \*.res}

begin

Application.Initialize;

Application.Title := 'Таблица Менделеева';

Application.CreateForm(TfrmMain, frmMain);

Application.Run;

end.

Модуль главной формы.

UNIT Main;

INTERFACE

uses

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

Dialogs, ExtCtrls, CustomizeDlg, StdCtrls, Grids, jpeg, CSVParser, Graphic;

type

TfrmMain = class(TForm)

imgGraphic: TImage;

btnIsolators: TButton;

btnConductors: TButton;

btnSemi: TButton;

strGrdElems: TStringGrid;

btnGraphic: TButton;

imgTable: TImage;

btnShowAll: TButton;

btnExit: TButton;

procedure FormCreate(Sender: TObject);

procedure strGrdElemsDrawCell(Sender: TObject; ACol, ARow: Integer;

Rect: TRect; State: TGridDrawState);

procedure btnExitClick(Sender: TObject);

procedure btnSemiClick(Sender: TObject);

procedure btnIsolatorsClick(Sender: TObject);

procedure btnConductorsClick(Sender: TObject);

procedure btnShowAllClick(Sender: TObject);

procedure btnGraphicClick(Sender: TObject);

private

// отсортированный по возрастанию удельного веса элементов массив

ConductorsSorted : Rows;

{ сортирует массив элементов element arr по x координате

от индекса first до индекса last }

procedure ConductorsSort(arr : PRows; first, last : Integer);

public

{ Public declarations }

end;

(Продолжение кода программы ПР№1 на следующей странице)

(Продолжение кода программы ПР№1)

var

frmMain: TfrmMain;

periodic\_chemical\_system : TextFile; // CSV файл с таблицей элементов

periodic\_table : CSVTable; // таблица элементов

IMPLEMENTATION

{$R \*.dfm}

// инициализация

procedure TfrmMain.FormCreate(Sender: TObject);

begin

Cursor := crAppStart; // показываем курсор загрузки

// загружаем таблицу элементов из файла CSV

periodic\_table := CSVTable.Create('Periodic\_table.csv');

Cursor := crDefault; // обычный курсор

end;

{$REGION ' Кнопки '}

// построить график

procedure TfrmMain.btnGraphicClick(Sender: TObject);

var

points : TPointArray; // массив точек

i : Integer;

begin

Cursor := crAppStart; // показываем курсор загрузки

imgTable.Visible := False; // скрываем картинку

strGrdElems.Visible := False; // скрываем таблицу

SetLength(points, length(periodic\_table.elements)); // задаём длину массиву точек

// заполняем массив точек

for i := 0 to periodic\_table.elements\_count - 1 do

begin

points[i].x := periodic\_table.elements[i].number; // номер элемента

points[i].y := round(periodic\_table.elements[i].weight\*100); // вес элемента \* 100 (для простоты дальнейшего построения)

end;

// рисуем график на всю ширину канвы графика по точкам из массива

Graphic.paint(points, periodic\_table.elements\_count, 0, 0, imgGraphic.Width, imgGraphic.Height, imgGraphic);

imgGraphic.Visible := True; // показываем график

Cursor := crDefault; // обычный курсор

end;

// проводники

procedure TfrmMain.btnConductorsClick(Sender: TObject);

begin

Cursor := crAppStart; // показываем курсор загрузки

imgTable.Visible := False; // скрываем картинку

imgGraphic.Visible := False; // скрываем график

strGrdElems.ColCount := 3; // количество колонок : 3

// заполняем заголовок таблицы

strGrdElems.Cells[0, 0] := 'Номер';

strGrdElems.Cells[1, 0] := 'Название';

strGrdElems.Cells[2, 0] := 'Удельный вес';

ConductorsSorted := Copy(periodic\_table.elements);

// сортируем элементы по возрастранию удельного веса

ConductorsSort(@ConductorsSorted, 0, periodic\_table.elements\_count - 1);

(Продолжение кода программы ПР№1 на следующей странице)

(Продолжение кода программы ПР№1)

// выводим содержимое таблицы

periodic\_table.showConductors(frmMain.strGrdElems, ConductorsSorted);

strGrdElems.Visible := True; // показываем таблицу

Cursor := crDefault; // обычный курсор

end;

// изоляторы

procedure TfrmMain.btnIsolatorsClick(Sender: TObject);

var

answer : string;

value : Real;

status : Integer;

begin

// выдаём окно с запросом

answer := InputBox('Изоляторы', 'Пожалуйста, введите максимальный удельный вес.'

+ ' Оставьте поле пустым, чтобы вывести все изоляторы.', '');

val(answer, value, status); // пытаемся привести к числу

// если пользователь ввёл некорректные данные, выводим все

if status <> 0 then value := -1;

Cursor := crAppStart; // показываем курсор загрузки

imgTable.Visible := False; // скрываем картинку

imgGraphic.Visible := False; // скрываем график

strGrdElems.ColCount := 3; // количество колонок : 3

// заполняем заголовок таблицы

strGrdElems.Cells[0, 0] := 'Номер';

strGrdElems.Cells[1, 0] := 'Название';

strGrdElems.Cells[2, 0] := 'Удельный вес';

// выводим содержимое таблицы

periodic\_table.showNonConductors(frmMain.strGrdElems, value);

strGrdElems.Visible := True; // показываем таблицу

Cursor := crDefault; // обычный курсор

end;

// полупроводники

procedure TfrmMain.btnSemiClick(Sender: TObject);

begin

Cursor := crAppStart; // показываем курсор загрузки

imgTable.Visible := False; // скрываем картинку

imgGraphic.Visible := False; // скрываем график

strGrdElems.ColCount := 3; // количество колонок : 3

// заполняем заголовок таблицы

strGrdElems.Cells[0, 0] := 'Номер';

strGrdElems.Cells[1, 0] := 'Название';

strGrdElems.Cells[2, 0] := 'Удельный вес';

// выводим содержимое таблицы

periodic\_table.showSemiConductors(frmMain.strGrdElems);

strGrdElems.Visible := True; // показываем таблицу

Cursor := crDefault; // обычный курсор

end;

(Продолжение кода программы ПР№1 на следующей странице)

(Продолжение кода программы ПР№1)

// показать все

procedure TfrmMain.btnShowAllClick(Sender: TObject);

begin

Cursor := crAppStart; // показываем курсор загрузки

imgTable.Visible := False; // скрываем картинку

imgGraphic.Visible := False; // скрываем график

strGrdElems.ColCount := 4; // количество колонок : 4

// заполняем заголовок таблицы

strGrdElems.Cells[0, 0] := 'Номер';

strGrdElems.Cells[1, 0] := 'Название';

strGrdElems.Cells[2, 0] := 'Удельный вес';

strGrdElems.Cells[3, 0] := 'Проводимость';

// выводим содержимое таблицы

periodic\_table.showAll(frmMain.strGrdElems);

strGrdElems.Visible := True; // показываем таблицу

Cursor := crDefault; // обычный курсор

end;

// выход

procedure TfrmMain.btnExitClick(Sender: TObject);

begin

Close;

end;

{$ENDREGION}

// быстрая сортировка

{ сортирует массив точек элементов element arr по x координате

от индекса first до индекса last }

procedure TfrmMain.ConductorsSort(arr : PRows; first, last : Integer);

var

i, j : Integer; // счётчики

middle : Real;

temp : element;

begin

i := first; j := last;

middle := arr^[(first + last) div 2].weight; // находим середину массива

// сортируем

repeat

begin

// проматываем i до тех пор, пока все элементы до i не будут меньше среднего

while (arr^[i].weight < middle) do inc(i);

// проматываем j до тех пор, пока все элементы после j не будут больше среднего

while (middle < arr^[j].weight) do dec(j);

// меняем местами i и j элементы

if (i <= j) then

begin

temp := arr^[i];

arr^[i] := arr^[j];

arr^[j] := temp;

inc(i);

dec(j);

end;

end;

until (i > j);

(Продолжение кода программы ПР№1 на следующей странице)

(Продолжение кода программы ПР№1)

// сортируем от начала до j

if (first < j) then

ConductorsSort(arr, first, j);

// и от i до конца

if (i < last) then

ConductorsSort(arr, i, last);

end;

// раскраска таблицы

procedure TfrmMain.strGrdElemsDrawCell(Sender: TObject; ACol, ARow: Integer;

Rect: TRect; State: TGridDrawState);

type

TSave = record

FontColor : TColor; // цвет шрифта

FontStyle : TFontStyles; // стиль шрифта

BrColor : TColor; // цвет кисти

end;

var

Sg : TStringGrid;

Save : TSave;

begin

Sg := Sender as TStringGrid;

with Sg.Canvas, Save do

begin

// запоминаем параметры шрифта и кисти.

FontColor := Font.Color;

FontStyle := Font.Style;

BrColor := Brush.Color;

// устанавливаем новые парметры

// фиксированные строки

if ARow <= Sg.FixedRows then

begin

Font.Color := clWhite; // белый

Font.Style := Font.Style + [fsBold]; // жирный шрифт

Brush.Color := RGB($CC, $99, $66); // коричневый

end

// нефиксированные строки с чётными индексами

else if ARow mod 2 = 0 then

begin

Font.Color := RGB(0, 0, 0); // чёрный

Brush.Color := RGB($FF, $FF, $CC); // светло-жёлтый

end

else

// нефиксированные строки с нечётными индексами

begin

Font.Color := clBlack; // чёрный

Brush.Color := RGB($CC, $FF, $FF); // светло-синий

end;

// закрашиваем прямоугольник

FillRect(Rect);

// прорисовываем в ячейке текст

TextOut(Rect.Left + 4, Rect.Top + 4, Sg.Cells[ACol, ARow]);

// восстанавливаем прежние параметры канвы

Font.Color := FontColor;

Font.Style := FontStyle;

Brush.Color := BrColor;

end;

end;

END.

Модуль парсера CSV таблиц

UNIT CSVParser;

INTERFACE

uses Grids;

// элемент периодической системы химических элементов

type element = record

number : Integer; // порядковый номер

name : String; // название

weight : Real; // удельный вес

conduction : (unknown, conductive, nonconductive, semiconductor); // проводимость

end;

// строки в таблице

type Rows = array of element;

type PRows = ^Rows;

// сама таблица

type CSVTable = class

private

const CHUNK = 8; // по сколько элементов за раз добалять в таблицу

var

text\_table : TextFile; // CSV файл, откуда читать таблицу

procedure addElement(table : TStringGrid; elements : Rows; i, j : Integer);

public

elements : Rows; // элементы - строки в таблице

elements\_count : Integer; // количество элементов в таблице

constructor Create(filename : string; separator : char = ';');

procedure showConductors(table : TStringGrid; ConductorsSorted : Rows);

procedure showSemiConductors(table : TStringGrid);

procedure showNonConductors(table : TStringGrid; maxWeight : Real = 0);

procedure showAll(table : TStringGrid);

end;

IMPLEMENTATION

uses Main, SysUtils;

// загрузка данных из CSV файла filename, разделённых separator

constructor CSVTable.Create(filename: string; separator: char = ';');

var

str : string;

status, i : integer;

begin

i := 0;

elements\_count := 0;

// открываем файл для чтения

AssignFile (text\_table, filename);

Reset(text\_table);

// читаем данные из файла

while not EOF(text\_table) do

begin

// увеличиваем размер таблицы, если нужно

if i >= elements\_count then

begin

elements\_count := elements\_count + CHUNK;

SetLength(elements, elements\_count);

end;

(Продолжение кода программы ПР№1 на следующей странице)

(Продолжение кода программы ПР№1)

ReadLn(text\_table, str);

val(copy(str, 1, pos(separator, str) - 1), elements[i].number, status);

// если данные некоректны, принимаем номер, равным 0

if status <> 0 then elements[i].number := 0;

delete (str, 1, pos(separator, str)); // отрезаем от строки прочитанную часть

elements[i].name := copy(str, 1, pos(separator, str) - 1);

delete (str, 1, pos(separator, str)); // отрезаем от строки прочитанную часть

val(copy(str, 1, pos(separator, str) - 1), elements[i].weight, status);

// если данные некоректны, принимаем удельный вес, равным 0

if status <> 0 then elements[i].weight := 0;

delete (str, 1, pos(separator, str)); // отрезаем от строки прочитанную часть

// устанавливаем проводимость

if str = 'проводник' then elements[i].conduction := conductive

else if str = 'изолятор' then elements[i].conduction := nonconductive

else if str = 'полупроводник' then elements[i].conduction := semiconductor

else elements[i].conduction := unknown;

inc(i); // увеличиваем счётчик

end;

elements\_count := i;

CloseFile(text\_table);

end;

// заполняем таблицу table всеми элементами

procedure CSVTable.showAll(table : TStringGrid);

var

i : Integer;

s : string;

begin

for i := 0 to elements\_count do

begin

addElement(table, elements, i, i);

// заполняем четвёртую ячейку типом проводимости i-го элемента

if elements[i].conduction = Conductive then s := 'Проводник'

else if elements[i].conduction = NonConductive then s := 'Изолятор'

else if elements[i].conduction = SemiConductor then s := 'Полупроводник'

else s := 'не известно'; // Если тип проводимости не известен, выведем "не известно"

table.Cells[3, i + 1] := s;

end;

end;

// заполняем таблицу table проводниками

procedure CSVTable.showConductors(table : TStringGrid; ConductorsSorted : Rows);

var

i, j : Integer;

begin

j := 0;

for i := 0 to elements\_count do

if elements[i].conduction = conductive then

begin

// добавляем новый элемент

addElement(table, ConductorsSorted, i, j);

inc(j); // увеличиваем счётчик

end;

end;

(Продолжение кода программы ПР№1 на следующей странице)

(Продолжение кода программы ПР№1)

// заполняем таблицу table полупроводниками

procedure CSVTable.showSemiConductors(table : TStringGrid);

var

i, j : Integer;

begin

j := 0;

for i := 0 to elements\_count do

if elements[i].conduction = semiConductor then

begin

// добавляем новый элемент

addElement(table, elements, i, j);

inc(j); // увеличиваем счётчик

end;

end;

// заполняем таблицу table изоляторами

procedure CSVTable.showNonConductors(table : TStringGrid; maxWeight : Real = 0);

var

i, j : Integer;

begin

j := 0;

for i := 0 to elements\_count do

if (elements[i].conduction = nonConductive) and ( (maxWeight < 0) or

(elements[i].weight <= maxWeight) ) then

begin

// добавляем новый элемент

addElement(table, elements, i, j);

inc(j); // увеличиваем счётчик

end;

end;

// добавляет i-ый элемент в j-ую строку таблицы table

procedure CSVTable.addElement(table: TStringGrid; elements : Rows; i, j: Integer);

var s : string;

begin

table.RowCount := j + 1;

// заполняем первую ячейку номером i-го элемента

if elements[i].number = 0 then

s := 'не известно'

else

Str(elements[i].number, s);

table.Cells[0, j + 1] := s;

// заполняем вторую ячейку названием i-го элемента

table.Cells[1, j + 1] := elements[i].name;

// заполняем третью ячейку удельным весом i-го элемента

if elements[i].weight = 0 then

s := 'не известно'

else

Str(elements[i].weight:4:2, s);

table.Cells[2, j + 1] := s;

end;

END.

Модуль отрисовки графика

UNIT Graphic;

INTERFACE

uses ExtCtrls, Graphics;

{$REGION ' Константы и типы '}

(Продолжение кода программы ПР№1 на следующей странице)

(Продолжение кода программы ПР№1)

const MARGIN = 40; // внутренние отступы вокруг графика

TEXT\_SIZE\_X = 6; // размер выводимых подписей по x

TEXT\_SIZE\_Y = 10; // размер выводимых подписей по y

GRID\_STEP\_X = 5; // Частота шага сетки по x (через каждые GRID\_STEP единиц)

GRID\_STEP\_Y = 200; // Частота шага сетки по y (через каждые GRID\_STEP единиц)

// тип "точка" - запись из 2-х координат

type TPoint = record x, y : Integer; end;

type TPointArray = array of TPoint;

type PTPointArray = ^TPointArray;

{$ENDREGION}

// быстрая сортировка

{ сортирует массив точек TPoint arr по x координате

от индекса first до индекса last }

procedure qSort(arr : PTPointArray; first, last : Integer);

// прототип процедуры, выводящей график по массиву точек

procedure paint(arr : TPointArray; // массив точек

len, // его длина

// координаты левой верхней точки прямоугольника для отрисовки

left\_top\_x, left\_top\_y,

right\_bottom\_x, right\_bottom\_y : integer;

Image : TImage // изображение, на которое производить отрисовку

);

IMPLEMENTATION

// быстрая сортировка

// сортирует массив точек TPoint arr по x координате от индекса first до индекса last

procedure qSort(arr : PTPointArray; first, last : Integer);

var

i, j : Integer; // счётчики

middle : Integer;

temp : TPoint;

begin

i := first; j := last;

middle := arr^[(first + last) div 2].x; // находим середину массива

// сортируем

repeat

begin

// проматываем i до тех пор, пока все элементы до i не будут меньше среднего

while (arr^[i].x < middle) do inc(i);

// проматываем j до тех пор, пока все элементы после j не будут больше среднего

while (middle < arr^[j].x) do dec(j);

// меняем местами i и j элементы

if (i <= j) then

begin

temp := arr^[i];

arr^[i] := arr^[j];

arr^[j] := temp;

inc(i);

dec(j);

end;

end;

until (i > j);

(Продолжение кода программы ПР№1 на следующей странице)

(Продолжение кода программы ПР№1)

// сортируем от начала до j

if (first < j) then

qSort(arr, first, j);

// и от i до конца

if (i < last) then

qSort(arr, i, last);

end;

// процедура, выводящая график по массиву точек

procedure paint(arr : TPointArray; // массив точек

len, // его длина

// координаты левой верхней точки прямоугольника для отрисовки

left\_top\_x, left\_top\_y,

right\_bottom\_x, right\_bottom\_y : integer;

Image : TImage // изображение, на которое производить отрисовку

);

var

y\_min, y\_max, // минимальное и максимальное значение y

mx, my : Double; // масштабы

screen\_width, screen\_height, // размеры "окна"

x, y, // координаты

i : integer; // счётчик

s : string[10]; // строки для вывода координат x и y на экран

begin

// рисуем рамку

Image.Canvas.Pen.Color := clGreen;

Image.Canvas.Rectangle(left\_top\_x, left\_top\_y,

right\_bottom\_x, right\_bottom\_y);

// определяем размеры "окна"

screen\_width := right\_bottom\_x - left\_top\_x + 1;

screen\_height := right\_bottom\_y - left\_top\_y + 1;

// сортируем массив по x координате

qSort(@arr, 0, len - 1);

// определяем минимальное и максимальное значения x и y

y\_min := 9999;

y\_max := -1;

for i:=1 to len - 1 do

begin

if arr[i].y > y\_max then

y\_max := arr[i].y;

if (arr[i].y > 0) and (arr[i].y < y\_min) then

y\_min := arr[i].y;

end;

// определяем масштабы по x и y

mx := (screen\_width - 2\*MARGIN) / abs(arr[1].x - arr[len - 1].x) ;

my := (screen\_height - 2\*MARGIN) / abs(y\_max - y\_min);

// рисуем график

Image.Canvas.Pen.Color := clRed; // красный

Image.Canvas.Pen.Width := 3; // шириной 3px

x := left\_top\_x + MARGIN;

y := right\_bottom\_y - round(my\*(arr[1].y - y\_min)) - MARGIN;

Image.Canvas.MoveTo(x, y); // начальное положение курсора

for i := 1 to len - 1 do

if arr[i].y > 0 then // Игнорируем неизвестные значения

begin

(Продолжение кода программы ПР№1 на следующей странице)

(Продолжение кода программы ПР№1)

x := round(mx\*(arr[i].x - arr[1].x)) + left\_top\_x + MARGIN;

y := right\_bottom\_y - round(my\*(arr[i].y - y\_min)) - MARGIN;

Image.Canvas.LineTo(x, y); // линия до следующей точки

end;

// Выводим вертикальные линии сетки

Image.Canvas.Pen.Width := 1; // шириной 1px

Image.Canvas.Pen.Color := clBlue; // синие

i := arr[0].x;

while i < arr[len - 1].x do

begin

Str(i, s); // преобразуем число в строку для вывода

// выводим подпись

x := round(mx\*(i - arr[0].x)) + left\_top\_x + MARGIN - TEXT\_SIZE\_X;

y := right\_bottom\_y - 5 - TEXT\_SIZE\_Y;

Image.Canvas.TextOut(x, y, s);

// рисуем линию

x := x + TEXT\_SIZE\_X;

y := left\_top\_y + MARGIN;

Image.Canvas.MoveTo(x, y);

Image.Canvas.LineTo(x, right\_bottom\_y - MARGIN);

// увеличиваем счётчик

i := i + GRID\_STEP\_X;

end;

// выводим горизонтальные линии сетки

i := round(y\_min);

while i < y\_max do

begin

Str((i/100):4:2, s); // преобразуем число в строку для вывода

// выводим подпись

x := left\_top\_x + 5;

y := right\_bottom\_y - round(my\*(i - y\_min)) - MARGIN - TEXT\_SIZE\_Y;

Image.Canvas.TextOut(x, y, s);

// рисуем линию

x := left\_top\_x + MARGIN;

y := y + TEXT\_SIZE\_Y;

Image.Canvas.MoveTo(x, y);

Image.Canvas.LineTo(right\_bottom\_x - MARGIN, y);

// увеличиваем счётчик

i := i + GRID\_STEP\_Y;

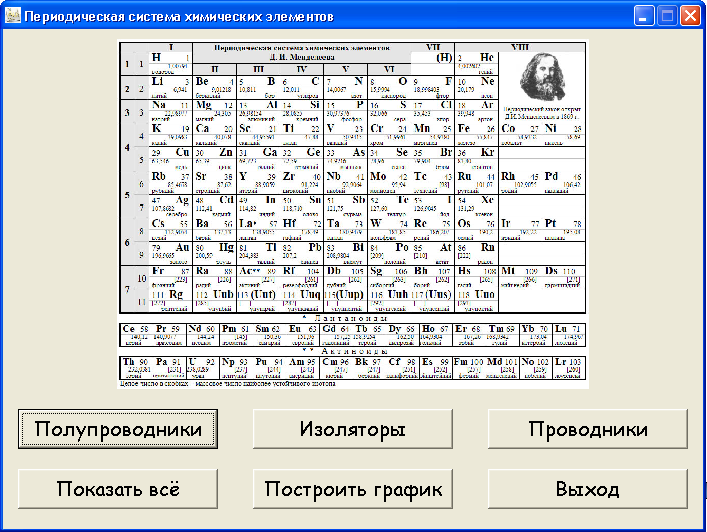
end;

end;

END.

**Приложение №2**

Скриншоты программы



*Рисунок 7 – Начальный вид приложения*



*Рисунок 8 – График зависимости удельного веса элемента от его номера*

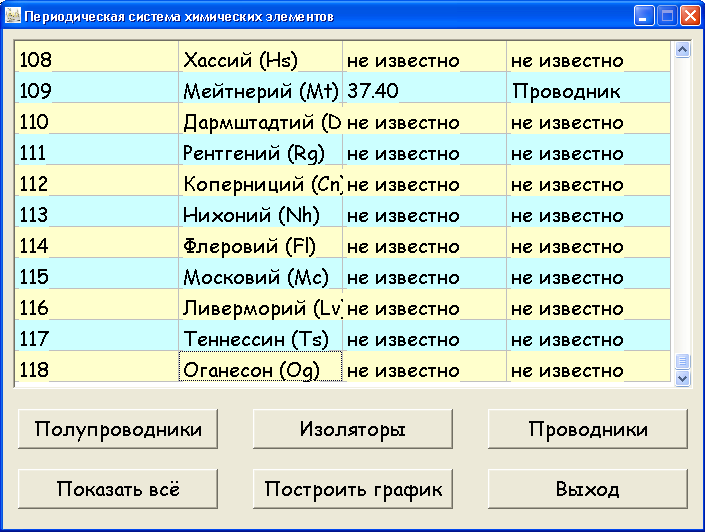
Все элементы



*Рисунок 9 – Вывод всех элементов (стр. 1)*

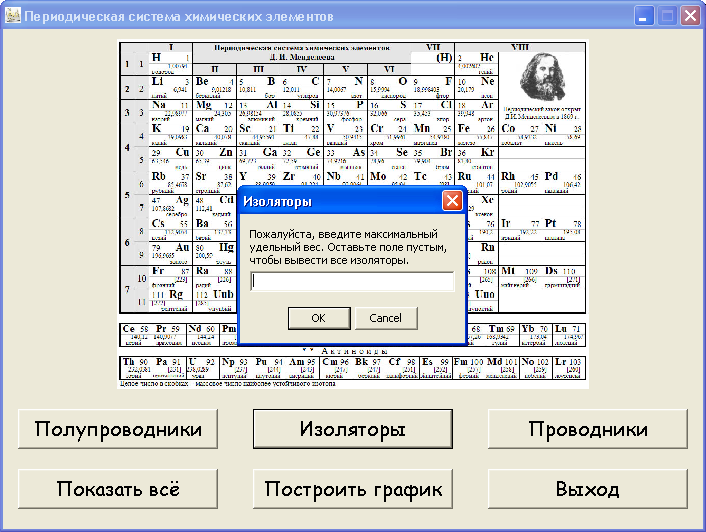


*Рисунок 10 – Вывод всех элементов (стр. 2)*

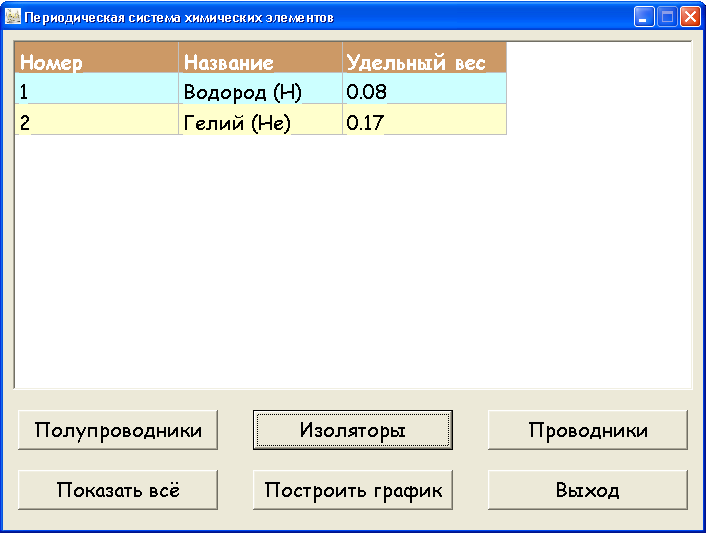


*Рисунок 11 – Вывод всех элементов (стр. 3)*

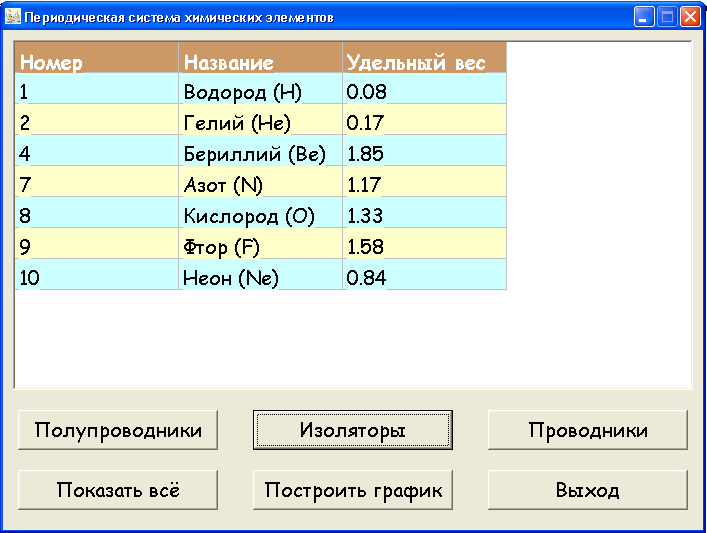
Изоляторы



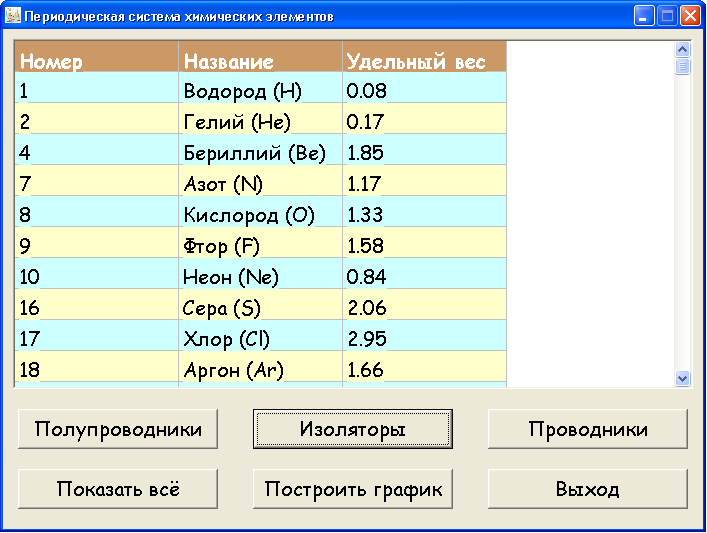
*Рисунок 12 – Запрос о вводе наибольшего удельного веса выводимых элементов*



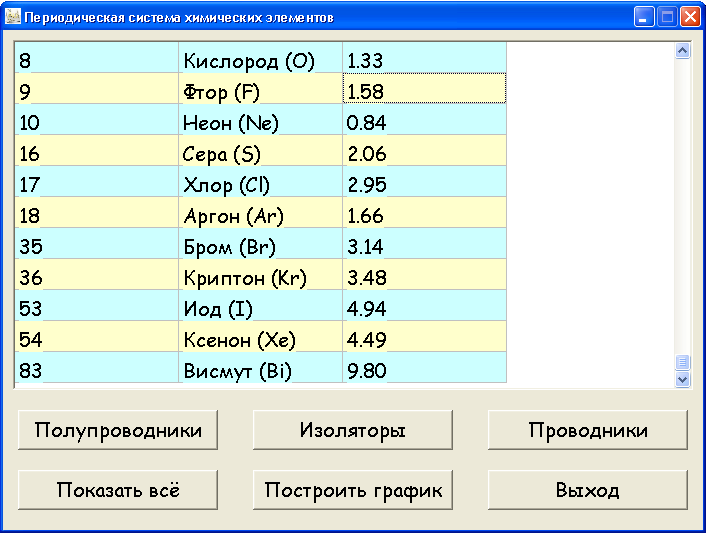
*Рисунок 13 – Вывод изоляторов с удельным весом меньше 1*



*Рисунок 14 – Вывод изоляторов с удельным весом меньше 2*



*Рисунок 15 – Вывод всех изоляторов (стр. 1)*



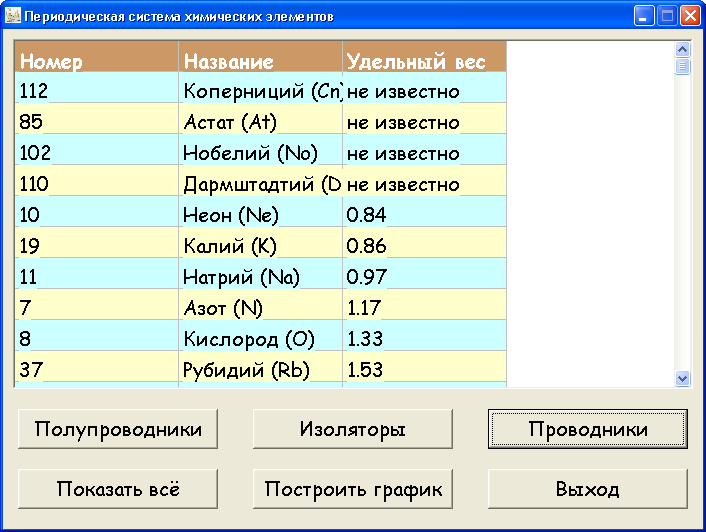
*Рисунок 16 – Вывод всех изоляторов (стр. 2)*

Полупроводники



*Рисунок 17 – Вывод всех полупроводников*

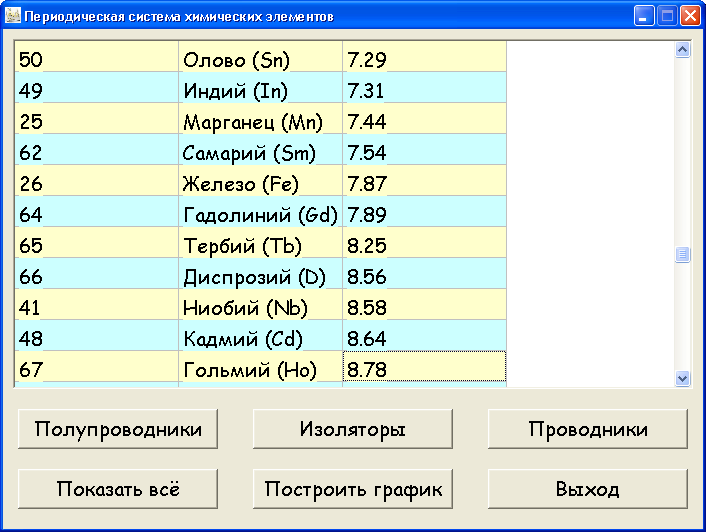
Проводники



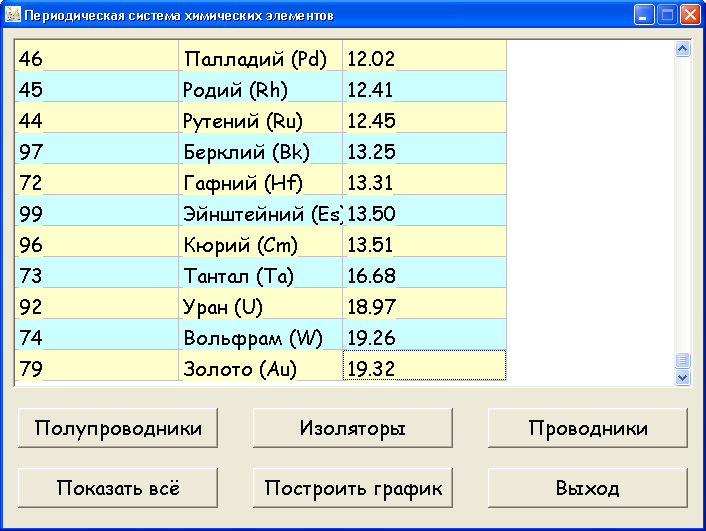
*Рисунок 18 – Проводники, упорядоченные по возрастанию удельного веса (стр. 1)*



*Рисунок 19 – Проводники, упорядоченные по возрастанию удельного веса (стр. 2)*



*Рисунок 20 – Проводники, упорядоченные по возрастанию удельного веса (стр. 3)*



*Рисунок 21 – Проводники, упорядоченные по возрастанию удельного веса (стр. 4)*