**Введение**

На курсовой проект была поставлена задача: «Разработка электронного средства обучения «Алгоритмы поиска».

Цель проекта заключается в разработке электронного средства обучения «Алгоритмы поиска» которое будет объяснять и демонострировать работу алгоритмов поиска.

Оно будет предназначено для пользователей, изучающих «Основы алгоритмизации и программирования», и интересующихся программированием.

Далее приведем краткое описание разделов пояснительной записки.

“Анализ задачи”. Можно ознакомиться с постановкой задачи, которая включает в себя: исследование предметной области поставленной задачи. Все входные и выходные данные будут описаны в первом разделе. В подразделе “Инструменты разработки” будет рассмотрена среда, в которой создается данный курсовой проект. Здесь также будут установлены минимальные и оптимальные требования к аппаратным характеристикам, обеспечивающим правильное функционирование поставленной задачей.

В разделе “Проектирование задачи” будут рассмотрены основные аспекты разработки программного продукта. Здесь можно будет узнать об организации данных в контексте среды разработки. В данном разделе будет четко описан пользовательский интерфейс, составлены алгоритмы процесса обработки информации, описана разработка системы справочной информации.

“Реализация задачи” – раздел пояснительной записки, который описывает все элементы и объекты, которые будут использованы при реализации данного приложения. В этом разделе будут четко описаны функции пользователя и их структура.

“Тестирование”. В нем будет описано полное и функциональное тестирование данной программы, т.е. будет оттестирован каждый пункт меню, каждая операция, которая выполняется приложением.

В разделе “Применение” будет описано назначение, область применения, среда функционирования курсовой программы. Также в нем будет описано использование справочной системы.

“Заключение” будет содержать краткую формулировку задачи, результаты проделанной работы, описание использованных методов и средств, описание степени автоматизации процессов на различных этапах разработки.

В “Литературе” будет приведен список используемых при разработке источников.

В приложениях к пояснительной записке будет приведен листинг программы с необходимыми комментариями.

Схема работы системы будет представлена в графической части.

**1 Анализ задачи**

**1.1 Постановка задачи**

**Наименование задачи:** «Разработка электронного средства обучения «Алгоритмы поиска».

**Цель разработки:** «Описание и демонстрация алгоритмов поиска»**.**

**Назначение:** продукт разрабатывается для людей знакомых с основами алгоритмизации и программирования.

**Периодичность использования:** зависит от нужд потребителя, может использоваться ежедневно.

**Источники и способы получения данных:** Входная, выходная и условно-постоянная информация

К входной информации можно отнести текстовое описание 5 алгоритмов поиска, вводимые пользователем значения,необходимы для их работы и демонстрации, например, количество элементов, число для поиска.

К выходной – демонстрация работы алгоритма, результат его работы

Постоянной информацией в проекте будут являться текстовые файлы, графические изображения.

Разрабатываемое ЭСО должно позволять пользователю следующее :

* изучить теоритические сведения об 5 алгоритмах поиска;
* просмотерть демонстрацию работы алгоритмов поиска;
* просмотреть код реализации на языках: PascalABC.Net и Python.

**1.2 Инструменты разработки**

Для разработки данного проекта выбрана среда Delphi (Delphi 10.4), так как это среда объектно-ориентированного программирования, относящаяся к классу RAD (Rapid Application Development), реализованная на Object Pascal и используемая для разработки визуализированного представления программного обеспечения.

Выгоды от проектирования в среде Windows с помощью Delphi:

* обеспечение согласованности проекта и его реализации;
* устранение необходимости в повторном вводе данных;
* увеличение производительности разработки и переносимости программ.

Firefox – для нахождения требуемой информации;

Help + Manual 8 – создание справочной системы;

Figma – требуется для создания изображений;

Inno Setup – требуется для создания инсталлятора приложения;

ONLYOFFICE Desktop Editors – требуется для создания пояснительной записки и отчетной презентации.

Для оптимального функционирования разрабатываемого приложения необходимо наличие у компьютера следующих параметров:

* процессор: 1.8 GHz;
* оперативная память: 1 GB;
* место на диске: 20 MB.

**1.3 Требования к приложению**

На этапе исследования предметной области был установленный целый ряд требований, которые предъявляются к разрабатываемой задаче. При моделировании форм следует учесть такие моменты:

- интуитивно понятный интерфейс;

- небольшое количество информации на каждой форме;

- небольшие окна формы;

- визуально понятный и приятный глазу интерфейс;

- небольшие кнопки.

Также следует обратить внимание на такие моменты:

* оптимизация программного кода приложения;

Минимальные системные требования:

* процессор: 1.5 GHz;
* Оперативная память: 512 MB;
* место на диске: 20 MB.

Рекомендуемые:

* процессор: 1.8 GHz;
* оперативная память: 1 GB;
* место на диске: 20 MB.

При разработке интерфейса приложения должны быть использованы цвета, преимущественно бело/серых оттенков. Основные разделы приложения должны быть доступны с первой страницы. Грамотный пользовательский интерфейс.

Следовательно, каждое окно должно иметь ясную визуальную иерархию своих элементов. Фрагменты текста должны располагаться на экране так, чтобы пользователю было просто и понятно принимать информацию.

Ошибки приложения вследствие некорректного взаимодействия пользователя при работе с приложением через графический интерфейс не должны влиять на работоспособность.

Надёжность приложения должна быть на высоком уровне, не допуская ситуаций программных ошибок и некорректной работы приложения.

# **2 Проектирование задачи**

**2.1 Организация данных**

Необходимым условием при разработке данного ЭСО является  описание организации данных, т.е. логическая и физическая структура данных  в контексте среды разработки. В разрабатываемой программе будут  использоваться три вида данных.

Первым видом являются данные, которые будут введены разработчиком  на этапе реализации задачи. Сюда можно отнести изображения (иконки),  описание алгоритмов, исходные коды поиска на языках: PascalABC.Net, Python.

Вторым видом данных, используемых в программе, является вводимая  пользователем информация. Входной информацией в разрабатываемой  программе будут являться данные: количество  данных (количество элементов) и элемент для поиска . Количество возможных введенных пользователем элементов будет ограничено, а именно  13 штук. Тип данных: *integer* (т. е пользователь может вводить только целые числа). Необязательно заполнять сами элементы вручную, поскольку эти  данные, по желанию пользователя, могут быть сгенерированы самой  программой. Подходящей структурой данных для хранения элементов является динамический одномерный массив*.*

Третьим видом данных является результат программы – выполнение поиска по пользовательским данным. Его также относят к отдельному  виду, так как ни пользователь, ни разработчик его не вводят, а программа сама  получает его в результате выполнения определенных действий.

Таким образом, организация данных является важной задачей при  разработке данной и любой программы.

**2.2 Процессы**

Согласно всем перечисленным требованиям и указаниям, которые были  рассмотрены в разделе «Анализ задачи», было определено, чем конкретно  должна заниматься разрабатываемое электронное средство обучения. Главной ее задачей  будет являться: демонстрация работы 5 алгоритмов поиска данных. Весь  данный процесс будет заключаться в следующем.

Пользовательские данные, хранятся в линейке, состоящие из ячеек. В  зависимости от этапа выполнения алгоритма поиска, эти ячейки будут  подсвечиваться различными цветами. А именно: желтый цветобозначает  рассматриваемые в текущий момент времени элементы данных, зеленый – найденный элемент (рисунок 1):



Рисунок 1 - Пример процесса демонстрации алгоритма линейного поиска

Для реализации поставленной задачи будут использованы процедуры.  Например, с помощью процедур, будет осуществляться алгоритмы поиска.

**2.3 Описание внешнего пользовательского интерфейса**

Важным при выполнении курсового проекта является организация диалога между пользователем и самой программой. Во многом это зависит от того, как программист разработает данную программу, какие компоненты будут использованы и какие методы будут автоматизированы. Во-первых, особое внимание следует уделить интерфейсу. Разработчик должен так организовать внешний вид своей программы, чтобы пользователь понял, как работать с данной программой.

Для организации эффективной работы пользователя нужно создать целостное приложение данной предметной области, в которой все компоненты приложения будут сгруппированы по функциональному назначению. При этом необходимо обеспечить удобный графический интерфейс пользователя. Приложение должно позволить пользователю решать задачи, затрачивая значительно меньше усилий, чем при работе с разрозненными объектами.

Ниже представлена система меню и организация навигации между окнами программы (рисунок 2):



Рисунок 2 - система меню и организация навигации между окнами программы

**Прототип** – это наглядная модель пользовательского интерфейса. В сущности это «черновик», созданный на основе представления разработчика о потребностях пользователя. **Итоговое отображение программы может отличаться от прототипа.** Ниже можно наблюдать прототипы:

1. Заставки (рисунок 3)

2. Главного окна (рисунок 4)

3. Окна, содержащее основную информацию о алгоритме (рисунок 5)

4. Окна, демонстрирующее работу алгоритмов сортировки (рисунок 6)

5. Окна, содержащее исходный код алгоритмов сортировки на языках  Pascalи Python (рисунок 7)

6. Информационного окна (рисунок 8)



Рисунок 3 – Прототип заставки.

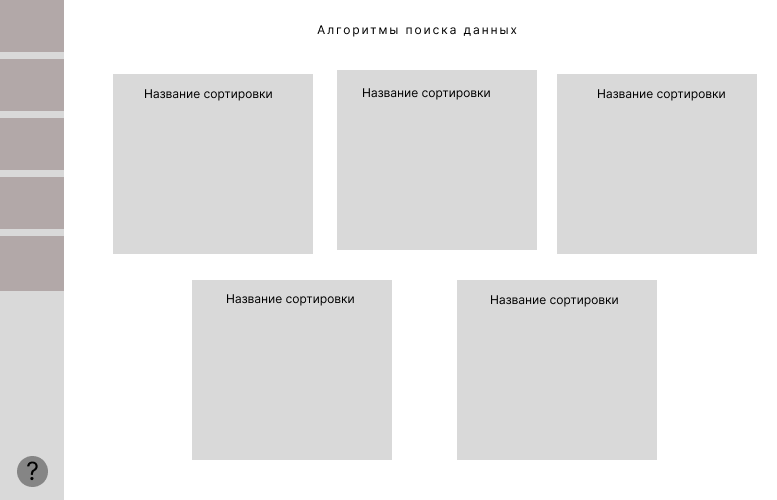


Рисунок 4 – Прототип главного окна.

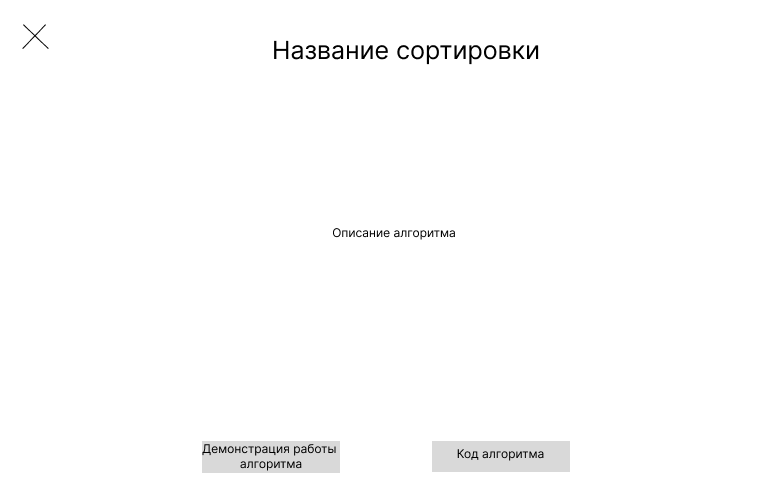


Рисунок 5 – Прототип содержащее основную информацию о  алгоритме.



Рисунок 6 – Прототип окна, демонстрирующее работу алгоритмов



Рисунок 7 – Прототип окна, исходный код алгоритмов сортировки на языках  Pascalи Python



Рисунок 8 – Прототип информационного окна

Таким образом, для успешной работы всего проекта в целом следует гарантировать эффективный в плане производительности и пользования программный продукт.

**3 Реализация**

**3.1 Реализация проекта**

**3.1.1 Структура программы**

Данный курсовой проект содержит 8 модулей. Далее будет описано назначение каждого из них.

Модуль Main\_FormUnit – это главная форма приложения.

Модуль SplashFormUnit – это форма заставки приложения, появляющаяся на несколько секунд и содержащая название приложения. После данной формы появляется главная форма.

Модуль DiscribeFormUnit – это форма с описание алгоритма.

Модуль DemonstrationFormUnit – это форма демонстрации работы алгоритма.

Модуль CodeFormUnit – это форма с исходным код алгоритма.

Модуль AboutFormUnit – это форма с описание проекта.

Модуль DesignUnit – это форма с процедурами взаимодействия с дизайном.

Модуль SearchUnit – это форма с алгоритмами поиска.

**3.1.2 Структура и описание процедур и функций пользователя**

Описание разработанных процедур и функций приводится в Таблице 1.

Таблица 1 – Процедуры и функции

|  |  |  |  |
| --- | --- | --- | --- |
| Имя процедуры (функции) | В каком модуле находится | За каким элементом управления закреплена | Назначение |
| 1 | 2 | 3 | 4 |
| procedure TMainForm.LinearSearchImageClick(Sender: TObject); | Main\_FormUnit | LinearSearchImage | Открытие формы с описанием |
| procedure TMainForm.BinarySearchImageClick(Sender: TObject); | Main\_FormUnit | BinarySearchImage | Открытие формы с описанием |
| procedure TMainForm.JumpSearchImageClick(Sender: TObject); | Main\_FormUnit | JumpSearchImage | Открытие формы с описанием |
| procedure TMainForm.InterpolationSearchImageClick(Sender: TObject); | Main\_FormUnit | InterpolationSearchImage | Открытие формы с описанием |
| procedure TMainForm.ExponentionalSearchImageClick(Sender: TObject); | Main\_FormUnit | ExponentionalSearchImage | Открытие формы с описанием |
| procedure TMainForm.LSImageClick(Sender: TObject); | Main\_FormUnit | LSImage | Открытие формы с описанием |
| procedure TMainForm.BSImageClick(Sender: TObject); | Main\_FormUnit | BSImage | Открытие формы с описанием |
| procedure TMainForm.JSImageClick(Sender: TObject); | Main\_FormUnit | JSImage | Открытие формы с описанием |
| procedure TMainForm.ISImageClick(Sender: TObject); | Main\_FormUnit | ISImage | Открытие формы с описанием |
| procedure TMainForm.ESImageClick(Sender: TObject); | Main\_FormUnit | ESImage | Открытие формы с описанием |
| procedure TMainForm.LSImageMouseMove(Sender: TObject; Shift: TShiftState; X,  Y: Integer); | Main\_FormUnit | LSImage | Вызов фукции selectMainMenu |
| procedure TMainForm.BSImageMouseMove(Sender: TObject; Shift: TShiftState; X,  Y: Integer); | Main\_FormUnit | BSImage | Вызов фукции selectMainMenu |
| procedure TMainForm.JSImageMouseMove(Sender: TObject; Shift: TShiftState; X,  Y: Integer); | Main\_FormUnit | JSImage | Вызов фукции selectMainMenu |
| procedure TMainForm.ESImageMouseMove(Sender: TObject; Shift: TShiftState; X,  Y: Integer); | Main\_FormUnit | ISImage | Вызов фукции selectMainMenu |
| procedure TMainForm.ISImageMouseMove(Sender: TObject; Shift: TShiftState; X,  Y: Integer); | Main\_FormUnit | ESImage | Вызов фукции selectMainMenu |
| procedure TMainForm.LSImageMouseLeave(Sender: TObject); | Main\_FormUnit | LSImage | Вызов функции UnselectMainMenu |

Продолжение Таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| procedure TMainForm.BSImageMouseLeave(Sender: TObject); | Main\_FormUnit | BSImage | Вызов функции UnselectMainMenu |
| procedure TMainForm.JSImageMouseLeave(Sender: TObject); | Main\_FormUnit | JSImage | Вызов функции UnselectMainMenu |
| procedure TMainForm.ESImageMouseLeave(Sender: TObject); | Main\_FormUnit | ISImage | Вызов функции UnselectMainMenu |
| procedure TMainForm.ISImageMouseLeave(Sender: TObject); | Main\_FormUnit | ESImage | Вызов функции UnselectMainMenu |
| procedure TMainForm.LinearSearchImageMouseMove(Sender: TObject;  Shift: TShiftState; X, Y: Integer); | Main\_FormUnit | LinearSearchImage | Вызов функции SelectSearch |
| procedure TMainForm.BinarySearchImageMouseMove(Sender: TObject;  Shift: TShiftState; X, Y: Integer); | Main\_FormUnit | BinarySearchImage | Вызов функции SelectSearch |
| procedure TMainForm.JumpSearchImageMouseMove(Sender: TObject;  Shift: TShiftState; X, Y: Integer); | Main\_FormUnit | JumpSearchImage | Вызов функции SelectSearch |
| procedure TMainForm.InterpolationSearchImageMouseMove(Sender: TObject;  Shift: TShiftState; X, Y: Integer); | Main\_FormUnit | InterpolationSearchImage | Вызов функции SelectSearch |
| procedure TMainForm.ExponentionalSearchImageMouseMove(Sender: TObject;  Shift: TShiftState; X, Y: Integer); | Main\_FormUnit | ExponentionalSearchImage | Вызов функции SelectSearch |
| procedure TMainForm.AboutSearchAlghoritmsImageMouseMove(Sender: TObject;  Shift: TShiftState; X, Y: Integer); | Main\_FormUnit | AboutSearchAlghoritmsImage | Вызов функции SelectSearch |
| procedure TMainForm.LinearSearchImageMouseLeave(Sender: TObject); | Main\_FormUnit | LinearSearchImage | Вызов функции UnselectSearch |
| procedure TMainForm.BinarySearchImageMouseLeave(Sender: TObject); | Main\_FormUnit | BinarySearchImage | Вызов функции UnselectSearch |
| procedure TMainForm.JumpSearchImageMouseLeave(Sender: TObject); | Main\_FormUnit | JumpSearchImage | Вызов функции UnselectSearch |
| procedure TMainForm.InterpolationSearchImageMouseLeave(Sender: TObject); | Main\_FormUnit | InterpolationSearchImage | Вызов функции UnselectSearch |
| procedure TMainForm.ExponentionalSearchImageMouseLeave(Sender: TObject); | Main\_FormUnit | ExponentionalSearchImage | Вызов функции UnselectSearch |
| procedure TMainForm.AboutSearchAlghoritmsImageMouseLeave(Sender: TObject); | Main\_FormUnit | AboutSearchAlghoritmsImage | Вызов функции UnselectSearch |

Продолжение Таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| procedure TMainForm.AboutSearchAlghoritmsImageClick(Sender: TObject); | Main\_FormUnit | AboutSearchAlghoritmsImage | Открытие формы «О программе» |
| procedure TCodeForm.Image1Click(Sender: TObject); | CodeFormUnit | Image1 | Закрытие формы |
| procedure TCodeForm.PascalCodeClick(Sender: TObject); | CodeFormUnit | PascalCode | Загрузка кода на языке Pascal |
| procedure TCodeForm.PythonCodeClick(Sender: TObject); | CodeFormUnit | PythonCode | Загрузка кода на языке Python |
| procedure selectMainMenu(Panel:TPanel; SelectInd:TShape); | DesignUnit | Panel, Shape | Выделение панели меню |
| procedure UnselectMainMenu(Panel:TPanel; SelectInd:TShape); | DesignUnit | Panel, Shape | Возвращение в обычное состояние |
| procedure selectButton(Panel:TPanel); | DesignUnit | Panel | Изменение курсора и цвета панели |
| procedure UnselectButton(Panel:TPanel); | DesignUnit | Panel | Возвращение в обычное состояние |
| procedure selectSearch(Panel:TPanel; Image:TImage); | DesignUnit | Panel, Image | Выделение сортировки, изменение курсора |
| procedure UnselectSearch(Panel:TPanel; Image:TImage); | DesignUnit | Panel, Image | Возвращение в обычное состояние |
| procedure setCellColor(StringGrid:TStringGrid; idx,color:integer;dTime:real); | SearchUnit | StringGrid | Изменение цвета ячейки |
| procedure LinearSearch(StringGrid:TStringGrid; element:integer); | SearchUnit | StringGrid | Алгоритм линейного поиска |
| procedure BinarySearch(StringGrid:TStringGrid; element:integer); | SearchUnit | StringGrid | Алгоритм бинарного поиска |
| procedure JumpSearch(StringGrid:TStringGrid; element:integer); | SearchUnit | StringGrid | Алгоритм прыжкового поиска |
| procedure InterpolationSearch(StringGrid:TStringGrid; element:integer); | SearchUnit | StringGrid | Алгоритм интерполяционного поиска |
| procedure ExponentionalSearch(StringGrid:TStringGrid; element:integer); | SearchUnit | StringGrid | Алгоритм экспоненциального поиска поиска |
| procedure ExBinarySearch(StringGrid:TStringGrid; left, right, element:integer); | SearchUnit | StringGrid | Алгоритм бинарного поиска |
| procedure Delay(Value: Cardinal); | SearchUnit | StringGrid | Время прорисовки ячейки |
| procedure SetSearchName(); | DemonstrationFormUnit | Label | Название алгоритма |
| procedure Sort(StringGrid:TStringGrid); | DemonstrationFormUnit | StringGrid | Сортировка значений |

Продолжение Таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| function isGridEmpty(StringGrid:TStringGrid):boolean; | DemonstrationFormUnit | StringGrid | Проверка пустой ли String Grid |
| procedure FormShow(Sender: TObject); | DemonstrationFormUnit | StringGrid | Увлечиение ячеек StringGrid, центрирование названия алгоритма |
| procedure TDemonstrationForm.FormClose(Sender: TObject;  var Action: TCloseAction); | DemonstrationFormUnit | StringGrid | Очиста StringGrid |
| procedure SubmitCountDataClick(Sender: TObject); | DemonstrationFormUnit | SubmitCountData | StringGrid принимает значение которое ввел пользователь |
| procedure TDemonstrationForm.ElementsCountChange(Sender: TObject); | DemonstrationFormUnit | ElementsCount | Изменение цвета Edit |
| procedure TDemonstrationForm.DataGridDrawCell(Sender: TObject; ACol,  ARow: Integer; Rect: TRect; State: TGridDrawState); | DemonstrationFormUnit | DataGrid | Чтение значений ячейки  Задание цветов ячееек |
| procedure TDemonstrationForm.GenerateDataClick(Sender: TObject); | DemonstrationFormUnit | GenerateData | Заполнение DataGrid отсортированными рандомными значениями |
| procedure TDemonstrationForm.FindNumberClick(Sender: TObject); | DemonstrationFormUnit | FindNumbe | Запуск демонстрации алгоритма поиска |
| procedure TAboutForm.CloseButtonClick(Sender: TObject); | AboutFormUnit | CloseButton | Закрытие формы «О программе» |
| procedure TSplashForm.Timer1Timer(Sender: TObject); | SplashFormUnit | Timer1 | Время отображения заставки |
| procedure TDiscribeForm.FormShow(Sender: TObject); | DiscribeFormUnit | Webbrowser  Label | Загрузка описания алгоритма |
| procedure TDiscribeForm.SetSearchDiscription(); | DiscribeFormUnit | WebBrowser | Загрузка описания алгоритма |
| procedure TDiscribeForm.CloseButtonClick(Sender: TObject); | DiscribeFormUnit | CloseButton | Закрытие формы |
| procedure TDiscribeForm.ShowCodeButtonClick(Sender: TObject); | DiscribeFormUnit | ShowCodeButton | Открытие формы с кодом алгоритма |
| procedure TDiscribeForm.ShowDemoLabelClick(Sender: TObject); | DiscribeFormUnit | ShowDemoLabel | Открытие формы с демонстрацией работы алгоритма |
| procedure TCodeForm.FormShow(Sender: TObject); | CodeFormUnit | WebBrowser | Загурзка кода алгоритма на языке Pascal |

**3.1.3 Описание использованных компонентов**

Описание использованных при разработке при разработке приложения компонентов приводится в таблице 2.

Таблица 2 – Использованные компоненты

|  |  |  |
| --- | --- | --- |
| Компонент | На какой форме расположен | Назначение |
| TPanel | Main\_Form, DiscribeForm, | Распологаяю на них картинки и надписи |
| TImage | Main\_Form,DiscribeForm, SplashForm, AboutForm,CodeForm | Картинки на формах |
| TShape | Main\_Form,DemostrationForm | Подсвечивание выбранной панели |
| TWebBrowser | DiscribeForm, CodeForm | Браузер для чтения |
| TLabel | Main\_Form,DiscribeForm, SplashForm, AboutForm,CodeForm, DemostrationForm | Вывод информации |
| TEdit | DemostrationForm | Ввод информации |
| TStringGrid | DemostrationForm | Демонстрация алгоритма |
| TCheckBox | DemostrationForm | Демонтсрировать ли поиск |

**3.2 Спецификация программы**

Точное название проекта и его состав приводится в таблице 3.

Таблица 3 – Спецификация программы

| Имя файла | Назначение |
| --- | --- |
| Searchalghoritms.exe | Исполняемый файл проекта, используется для запуска программы на выполнение. |
| Searchalghoritms.dproj | Файл проекта, связывает все файлы из которых состоит приложение. |
| Main\_FormUnit.pas | Файл программного модуля главной формы |
| DiscribeFormUnit.pas | Файл программного модуля описания алгоритма |
| CodeFormUnit.pas | Файл программного модуля код алгоритма |
| DemonstrationFormUnit.pas | Файл программного модуля для демонстрации работы алгоритма |
| DesignUnit.pas | Файл программного модуля дизайна |
| SearchUnit.pas | Файл программного модуля с алгоритмами поиска |
| SplashFormUnit.pas | Файл программного модуля заставки |
| AboutFormUnit.pas | Файл программного модуля о приложении |
| Main\_FormUnit.dfm | Главная форма |
| DiscribeFormUnit.dfm | Форма с описанием алгорита |
| CodeFormUnit.dfm | Форма с исходным кодом алгоритма |
| DemonstrationFormUnit.dfm | Форма с демонстрацией работы алгоритма |
| SplashFormUnit.dfm | Форма с заставкой |
| AboutFormUnit.dfm | Форма с информацией |
| Pas.html | Коды алгоритмов на Pascal |
| Py.html | Коды алгоритмов на Python |
| Setup.exe | Установочный пакет программы |
| index.html | Описание алгоритмов |

**4 Тестирование**

При разработке данной программы многие возникающие ошибки и недоработки были исправлены на этапе реализации проекта. После завершения этапа написания программы было проведено тщательное функциональное тестирование. Функциональное тестирование должно гарантировать работу всех элементов программы в автономном режиме.

Отчет о результатах тестирования представлен в таблице 4.

Таблица 4 – Отчет о результатах тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| Тест | Ожидаемый результат | Фактический результат | Результат тестирования |
| 1 | 2 | 3 | 4 |
| Проверка элементов меню | Переход на форму с описание алгоритма | Переход на форму | Выполнено |
| Проверка элементов главного экрана | Переход на форму с описание алгоритма | Переход на форму | Выполнено |
| Проверка пункта меню «О приложении» | Переход на форму «О приложении» | Открылась форма «О приложении» | Выполнено |
| Проверка кнопки «Код алгоритма» | Переход на форму  «Код алгоритма» | Открылась форма «Код алгоритма» | Выполнено |
| Проверка кнопки «Демонстрация работы алгоритма» | Переход на форму  «Демонстрация работы алгоритма» | Открылась форма «Демонстрация работы алгоритма» | Выполнено |
| Проверка кнопки «X» | Переход на главную форму | Переход на главную форму | Выполнено |
| Проверка кнопки «Сгенерировать отсортированный данные» | Заполнение StringGrid отсортированными данными | Заполнение StringGrid отсортированными данными | Выполнено |
| Проверка кнопки «Найти число» | Запуск демонстрации | Запуск демонстрации | Выполнено |
| Проверка кнопки «Применить» | Изменение размеров StringGrid и подтверждение числа для поиска | Изменение размеров StringGrid и подтверждение числа для поиска | Выполнено |
| Проверка кнопки «Pascal» | Загрузка кода на PascalABC.NET | Загрузка кода на PascalABC.NET | Выполнено |
| Проверка кнопки «Python» | Загрузка кода на Python | Загрузка кода на Python | Выполнено |

Результат данного тестирования: приложение полностью справилось с функциональным тестированием, выполнив его успешно, что означает полную отказоустойчивость и работоспособность приложения во время его взаимодействия с пользователем.

**5 Руководство пользователя**

**5.1 Общие сведения о программном продукт**

Разработанная программа является электронным средством обучения «Алгоритмы поиска».

Данное средство обучения предназначено для объяснение и десонстрации алгоритмов поиска

Основными функциями приложения являются:

- Изучение теоретической информации о алгоритмах поиска;

- Просмотр исходного кода алгоритма;

- Просмотр демонстрации алгоритмов поиска;

- Изучение справочной информации про функционал программs

Быстродействие любой программы во многом зависит от характеристик выбранного персонального компьютера: рабочей частоты процессора, объема оперативной памяти и т.д. Несмотря на все реализованные задачи в ней, она легко запускается и функционирует на любых компьютерах.

Тестирование проводилось на различных классах ЭВМ и работать с данной программой было комфортно. Но даже на системе следующего типа работать было комфортно:

Разработка происходила на ноутбуке со следующими характеристиками:

- процессор: Intel Core i7;

- ОЗУ: 8Gb;

- память: SSD 256Gb;

- ОС – Windows 10.

**5.2 Инсталляция**

Для того что бы установить программу необходимо запустить файл setup.exe и выбираем каталог(рисунок 9).

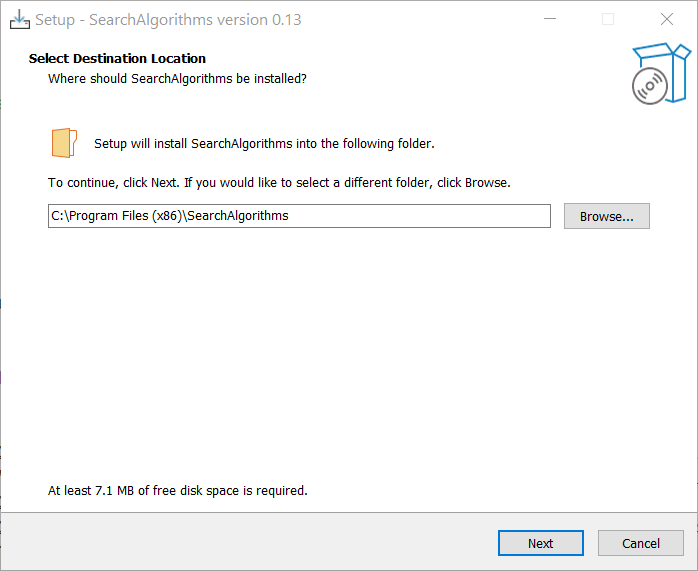


Рисунок 9 – устоновка ЭСО SearchAlgorithms

Далее выводится окно предлагающее создать ярлык на рабочем столе, (рисунок 10).

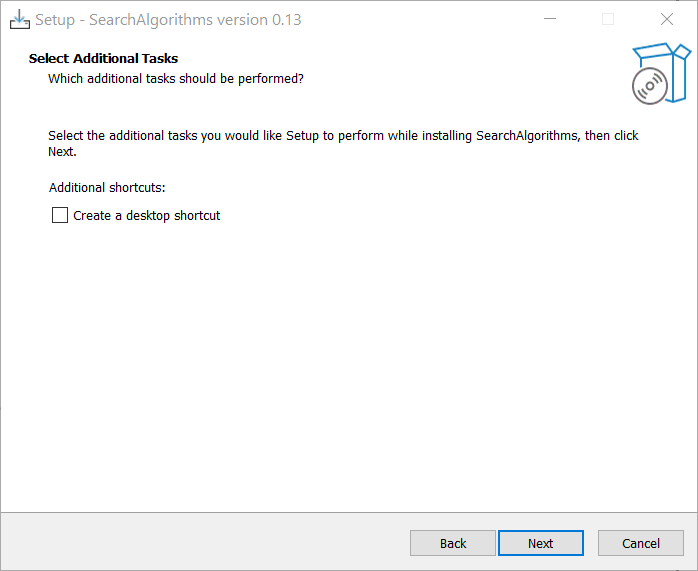


Рисунок 10 – создание ярлыка на рабочем столе.

Далее появится завершающее окно установки (рисунок 11)

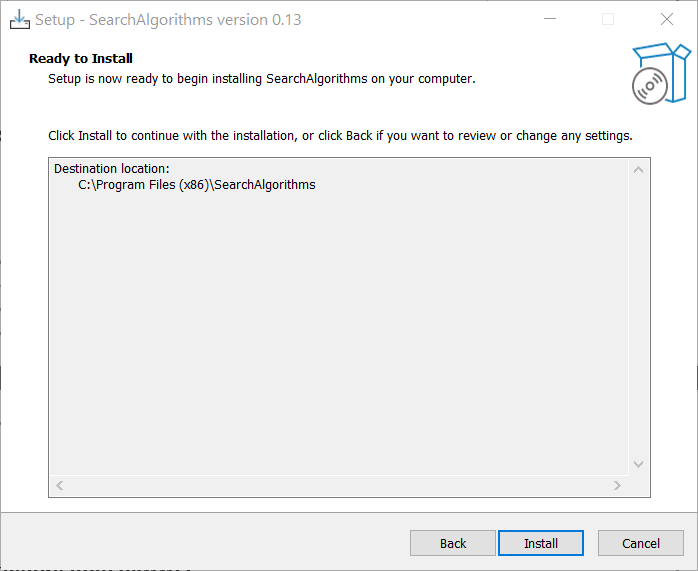


Рисунок 11 **-** завершающее окно установки**.**

**5.3 Выполнение программы**

**5.3.1 Запуск программы**

Данную программу можно запустить различными способами.

Первым из них является запуск с помощью ярлыка на рабочем столе. Необходимо дважды щелкнуть левой клавишей мыши на ярлык под названием SearchAlgorithms (Рисунок 12).

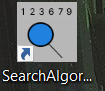


Рисунок 12 – Ярлык программы

Вторым способом является запуск из каталога, в который установилась программа.

**5.3.2 Инструкции по работе с программой**

После запуска программы на экране появляется главный экран (рисунок 13).



Рисунок 13 – Главный экран

После нажатия на один из пунктов меню откроется форма с описанием (Рисунок 14).

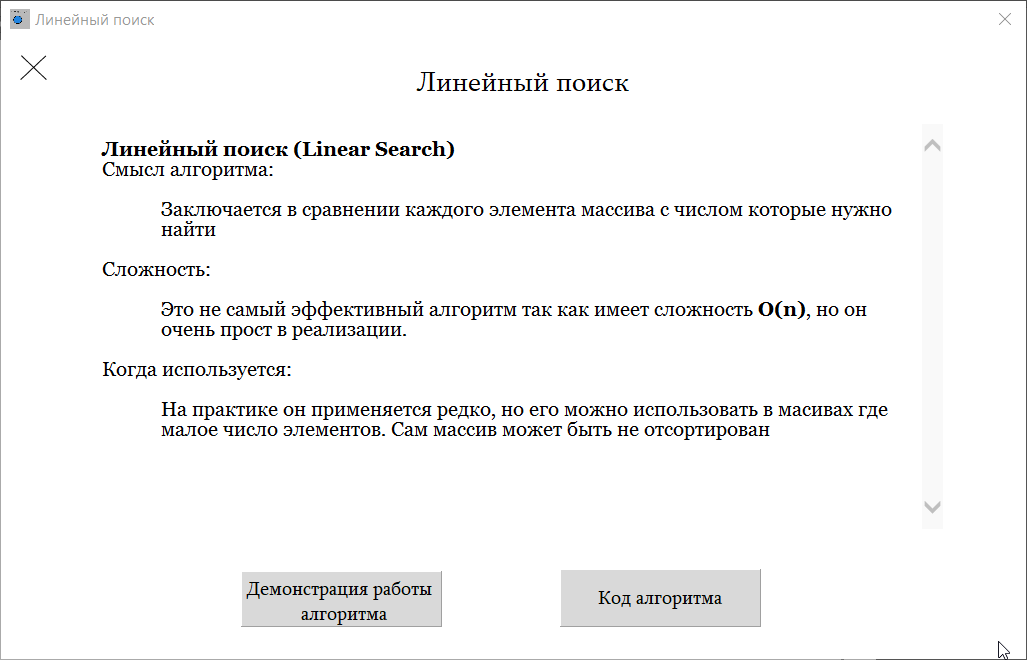


Рисунок 14 – Форма с описанием

Далее можно перейти на форму с демонстрацией работы алгоритма (рисунок 15) или на форму с исходным кодом алгоритма (рисунок 16)

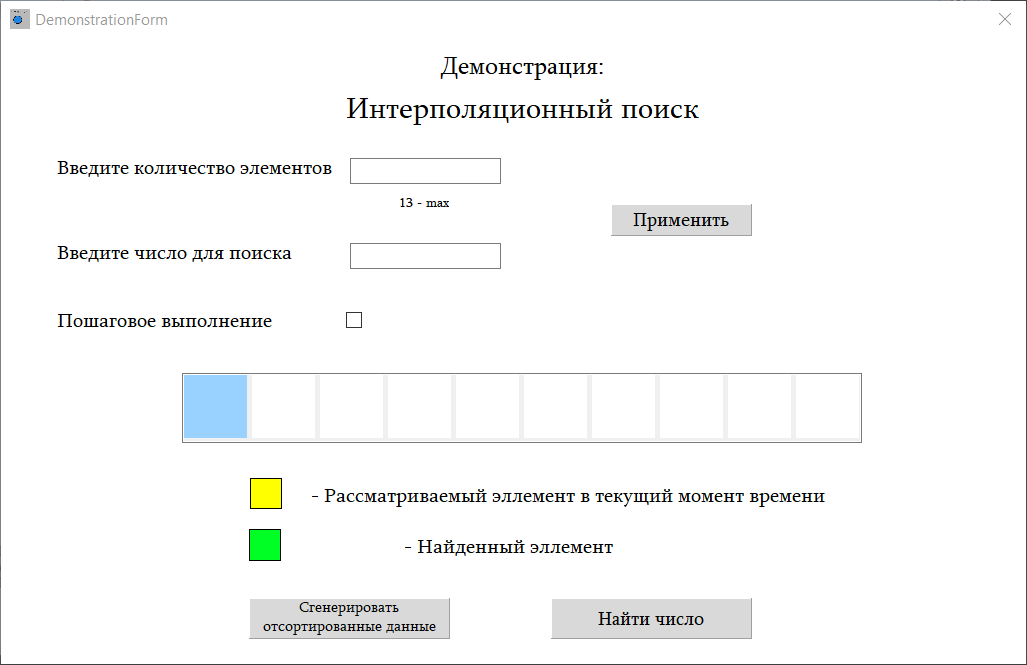


Рисунок 15 – Окно с демонстрацией алгоритма

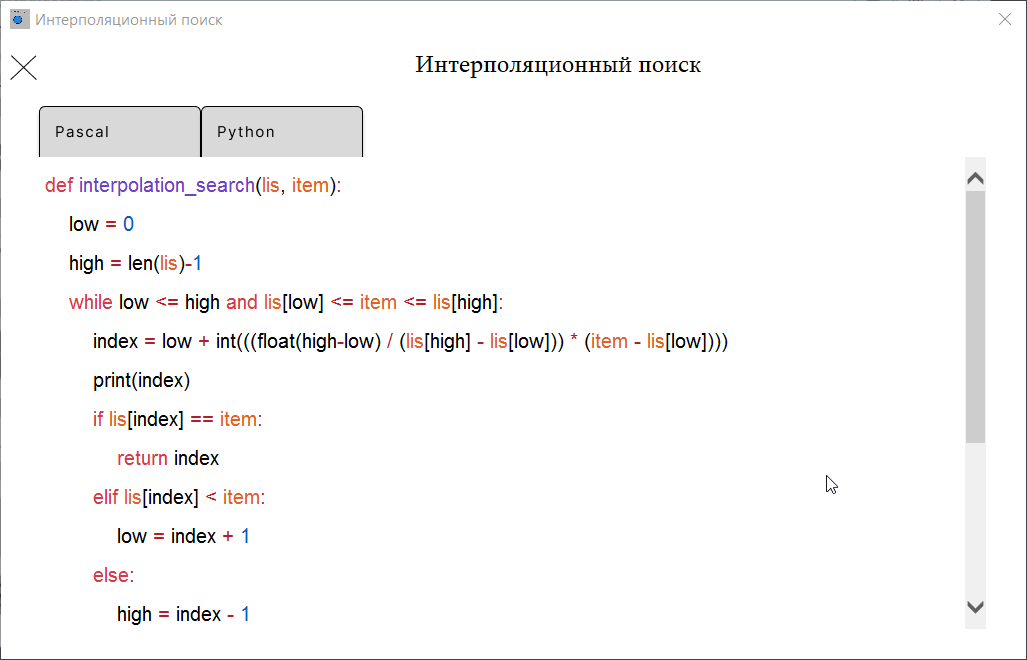


Рисунок 16 – Окно с исходным кодом

Нажав на кнопку в меню можно прочитать информацию о проекте (рисунок 17)



Рисунок 17 – окно «О приложении»

**5.3.3 Завершение работы с программой**

Единственный способ выйти из приложения нажать на крестик в правом верхнем углу на любом из окон (рисунок 18):



Рисунок 18 - Выход из приложения

**5.4 Использование системы справочной информации**

Справочную систему можно запустить с помощью меню – О программе (рисунок 18).

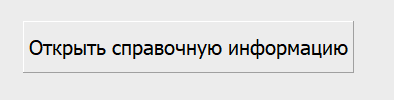


Рисунок 19 - справка

После нажатия появиться окно с справочной информацией о программе (Рисунок 19).

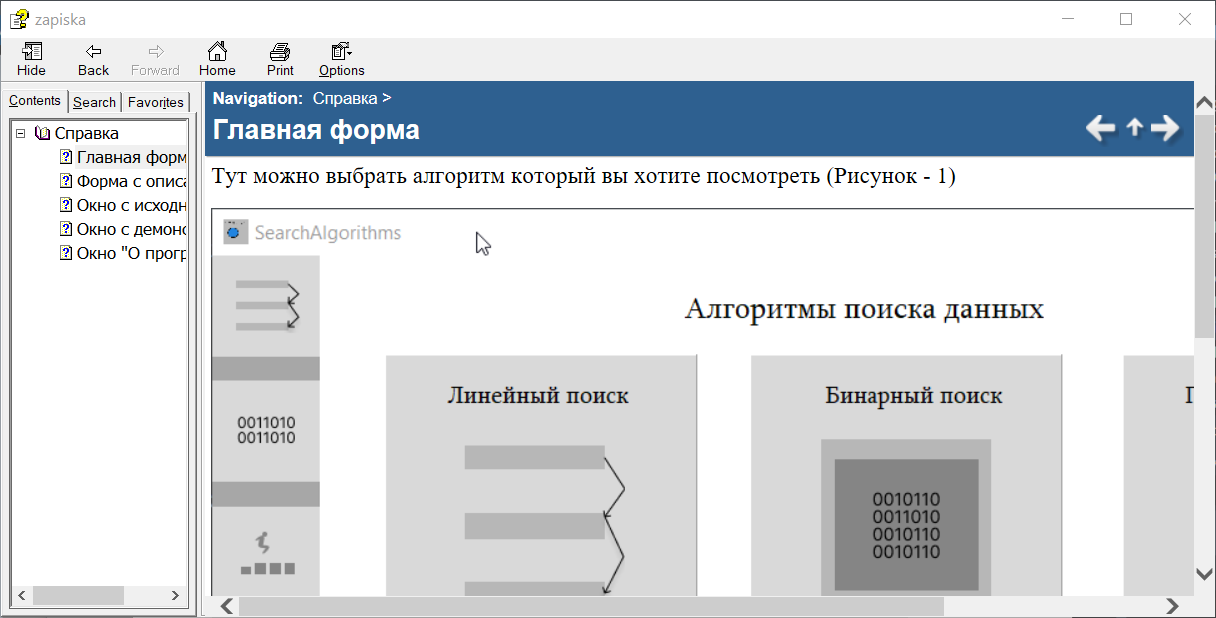


Рисунок 19 – справочная информация

**Заключение**

Цель данного проекта была реализована - создано электронного средства обучения «Алгоритмы поиска».

Данное электронное средство обучения будет интересно изучающим «Основы алгоритмизации и программирования», и интересующихся программированием.

Был реализован простой и понятный пользовательский интерфейс, который позволяет использовать приложение пользователю, не обладающему дополнительными знаниями ЭВМ.

В ходе реализации поставленной задачи был укреплён пройденный курс программирования в среде Delphi, а также получено много дополнительной информации о её возможностях.

После тщательного тестирования приложения были выявлены некоторые недоработки, которые были частично исправлены на стадии проектирования, и полностью исключены на стадии тестирования программы. В целом при реализации программы, были выполнены все условия, перечисленные в предыдущих разделах пояснительной записки.

Таким образом, можно сказать, что поставленная задача была реализована успешно.

**Список используемых источников**

1. Электронный ресурс – Режим доступа: <https://iq.opengenus.org> – Дата доступа 16.06.2022.

Приложение A

Листинг программы

unit Main\_FormUnit;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.StdCtrls,

Vcl.Imaging.pngimage;

type

TMainForm = class(TForm)

Algorithms: TLabel;

LinearSearch: TPanel;

LinearSearchLabel: TLabel;

BinarySearch: TPanel;

BinarySearchLabel: TLabel;

JumpSearch: TPanel;

JumpSearchLabel: TLabel;

InterpolationSearch: TPanel;

ExponentionalSearch: TPanel;

LinearSearchImage: TImage;

BinarySearchImage: TImage;

JumpSearchImage: TImage;

ExponentionalSearchImage: TImage;

ExponentionalSearchLabel: TLabel;

InterpolationSearchLabel: TLabel;

InterpolationSearchImage: TImage;

MainMenu: TPanel;

LinearSearchMenu: TPanel;

BinarySearchMenu: TPanel;

LSImage: TImage;

BSImage: TImage;

JumpSearchMenu: TPanel;

JSImage: TImage;

InterpolationSearchMenu: TPanel;

ISImage: TImage;

ExponentionalSearchMenu: TPanel;

ESImage: TImage;

SelectLS: TShape;

SelectBS: TShape;

SelectJS: TShape;

SelectIS: TShape;

SelectES: TShape;

AboutPanel: TPanel;

AboutSearchAlghoritmsImage: TImage;

procedure ExponentionalSearchImageClick(Sender: TObject);

procedure InterpolationSearchImageClick(Sender: TObject);

procedure JumpSearchImageClick(Sender: TObject);

procedure BinarySearchImageClick(Sender: TObject);

procedure LinearSearchImageClick(Sender: TObject);

procedure LSImageMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure LSImageMouseLeave(Sender: TObject);

procedure BSImageMouseLeave(Sender: TObject);

procedure BSImageMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure JSImageMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure ISImageMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure ISImageMouseLeave(Sender: TObject);

procedure ESImageMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure ESImageMouseLeave(Sender: TObject);

procedure JSImageMouseLeave(Sender: TObject);

procedure LSImageClick(Sender: TObject);

procedure BSImageClick(Sender: TObject);

procedure JSImageClick(Sender: TObject);

procedure ISImageClick(Sender: TObject);

procedure ESImageClick(Sender: TObject);

procedure AboutSearchAlghoritmsImageClick(Sender: TObject);

procedure LinearSearchImageMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure LinearSearchImageMouseLeave(Sender: TObject);

procedure BinarySearchImageMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure BinarySearchImageMouseLeave(Sender: TObject);

procedure JumpSearchImageMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure JumpSearchImageMouseLeave(Sender: TObject);

procedure InterpolationSearchImageMouseMove(Sender: TObject;

Shift: TShiftState; X, Y: Integer);

procedure InterpolationSearchImageMouseLeave(Sender: TObject);

procedure ExponentionalSearchImageMouseMove(Sender: TObject;

Shift: TShiftState; X, Y: Integer);

procedure ExponentionalSearchImageMouseLeave(Sender: TObject);

procedure AboutSearchAlghoritmsImageMouseMove(Sender: TObject;

Shift: TShiftState; X, Y: Integer);

procedure AboutSearchAlghoritmsImageMouseLeave(Sender: TObject);

private

public

end;

var

MainForm: TMainForm;

implementation

{$R \*.dfm}

uses DiscribeFormUnit, AboutFormUnit, DesignUnit;

procedure TMainForm.LSImageMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

begin

selectMainMenu(LinearSearchMenu, SelectLS)

end;

procedure TMainForm.BSImageMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

begin

selectMainMenu(BinarySearchMenu, SelectBS)

end;

procedure TMainForm.JSImageMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

begin

selectMainMenu(JumpSearchMenu, SelectJS)

end;

procedure TMainForm.ESImageMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

begin

selectMainMenu(ExponentionalSearchMenu, SelectES)

end;

procedure TMainForm.ISImageMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

begin

selectMainMenu(InterpolationSearchMenu, SelectIS)

end;

procedure TMainForm.LSImageMouseLeave(Sender: TObject);

begin

UnselectMainMenu(LinearSearchMenu, SelectLS)

end;

procedure TMainForm.BSImageMouseLeave(Sender: TObject);

begin

UnselectMainMenu(BinarySearchMenu, SelectBS)

end;

procedure TMainForm.JSImageMouseLeave(Sender: TObject);

begin

UnselectMainMenu(JumpSearchMenu, SelectJS)

end;

procedure TMainForm.ESImageMouseLeave(Sender: TObject);

begin

UnselectMainMenu(ExponentionalSearchMenu, SelectES)

end;

procedure TMainForm.ISImageMouseLeave(Sender: TObject);

begin

UnselectMainMenu(InterpolationSearchMenu, SelectIS)

end;

procedure TMainForm.LinearSearchImageMouseMove(Sender: TObject;

Shift: TShiftState; X, Y: Integer);

begin

selectSearch(LinearSearch, LinearSearchImage)

end;

procedure TMainForm.BinarySearchImageMouseMove(Sender: TObject;

Shift: TShiftState; X, Y: Integer);

begin

selectSearch(BinarySearch, BinarySearchImage)

end;

procedure TMainForm.JumpSearchImageMouseMove(Sender: TObject;

Shift: TShiftState; X, Y: Integer);

begin

selectSearch(JumpSearch, JumpSearchImage)

end;

procedure TMainForm.InterpolationSearchImageMouseMove(Sender: TObject;

Shift: TShiftState; X, Y: Integer);

begin

selectSearch(InterpolationSearch, InterpolationSearchImage)

end;

procedure TMainForm.ExponentionalSearchImageMouseMove(Sender: TObject;

Shift: TShiftState; X, Y: Integer);

begin

selectSearch(ExponentionalSearch, ExponentionalSearchImage)

end;

procedure TMainForm.AboutSearchAlghoritmsImageMouseMove(Sender: TObject;

Shift: TShiftState; X, Y: Integer);

begin

selectSearch(AboutPanel, AboutSearchAlghoritmsImage)

end;

procedure TMainForm.LinearSearchImageMouseLeave(Sender: TObject);

begin

UnselectSearch(LinearSearch, LinearSearchImage)

end;

procedure TMainForm.BinarySearchImageMouseLeave(Sender: TObject);

begin

UnselectSearch(BinarySearch, BinarySearchImage)

end;

procedure TMainForm.JumpSearchImageMouseLeave(Sender: TObject);

begin

UnselectSearch(JumpSearch, JumpSearchImage)

end;

procedure TMainForm.InterpolationSearchImageMouseLeave(Sender: TObject);

begin

UnselectSearch(InterpolationSearch, InterpolationSearchImage)

end;

procedure TMainForm.ExponentionalSearchImageMouseLeave(Sender: TObject);

begin

UnselectSearch(ExponentionalSearch, ExponentionalSearchImage)

end;

procedure TMainForm.AboutSearchAlghoritmsImageMouseLeave(Sender: TObject);

begin

UnselectSearch(AboutPanel, AboutSearchAlghoritmsImage)

end;

procedure TMainForm.AboutSearchAlghoritmsImageClick(Sender: TObject);

begin

AboutForm.ShowModal;

end;

procedure TMainForm.LinearSearchImageClick(Sender: TObject);

begin

SearchID := 0;

DiscribeForm.Show;

MainForm.Hide;

end;

procedure TMainForm.LSImageClick(Sender: TObject);

begin

SearchID := 0;

DiscribeForm.Show;

MainForm.Hide;

end;

procedure TMainForm.BinarySearchImageClick(Sender: TObject);

begin

SearchID := 1;

DiscribeForm.Show;

MainForm.Hide;

end;

procedure TMainForm.BSImageClick(Sender: TObject);

begin

SearchID := 1;

DiscribeForm.Show;

MainForm.Hide;

end;

procedure TMainForm.JumpSearchImageClick(Sender: TObject);

begin

SearchID := 2;

DiscribeForm.Show;

MainForm.Hide;

end;

procedure TMainForm.JSImageClick(Sender: TObject);

begin

SearchID := 2;

DiscribeForm.Show;

MainForm.Hide;

end;

procedure TMainForm.InterpolationSearchImageClick(Sender: TObject);

begin

SearchID := 3;

DiscribeForm.Show;

MainForm.Hide;

end;

procedure TMainForm.ISImageClick(Sender: TObject);

begin

SearchID := 3;

DiscribeForm.Show;

MainForm.Hide;

end;

procedure TMainForm.ExponentionalSearchImageClick(Sender: TObject);

begin

SearchID := 4;

DiscribeForm.Show;

MainForm.Hide;

end;

procedure TMainForm.ESImageClick(Sender: TObject);

begin

SearchID := 4;

DiscribeForm.Show;

MainForm.Hide;

end;

end.

unit DiscribeFormUnit;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.Imaging.pngimage,

Vcl.OleCtrls, SHDocVw, Vcl.StdCtrls;

type

TDiscribeForm = class(TForm)

Search\_name: TLabel;

SearchDescription: TWebBrowser;

CloseButton: TImage;

ShowDemoButton: TPanel;

ShowCodeButton: TPanel;

ShowDemoLabel: TLabel;

procedure CloseButtonClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure SetSearchDiscription();

procedure ShowDemoLabelClick(Sender: TObject);

procedure ShowCodeButtonClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

DiscribeForm: TDiscribeForm;

SearchID: integer;

SearchPathsName: array of string = [

'LinearSearch',

'BinarySearch',

'JumpSearch',

'InterpolationSearch',

'ExponentionalSearch'

];

SearchName: array of string = [

'Линейный поиск',

'Бинарный поиск',

'Прыжковый поиск',

'Интерполяционный поиск',

'Экспоненциальный поиск'

];

implementation

{$R \*.dfm}

uses Main\_FormUnit, CodeFormUnit, DemonstrationFormUnit, DesignUnit;

procedure TDiscribeForm.CloseButtonClick(Sender: TObject);

begin

DiscribeForm.Close;

MainForm.Show();

end;

procedure TDiscribeForm.FormCreate(Sender: TObject);

begin

SearchDescription.Silent := true;

end;

procedure TDiscribeForm.SetSearchDiscription();

begin

SearchDescription.Navigate(ExtractFilePath(paramstr(0))+ 'SearchDiscriptions/' + SearchPathsName[SearchID] + '/index.html');

end;

procedure TDiscribeForm.ShowCodeButtonClick(Sender: TObject);

begin

DiscribeForm.Hide;

CodeForm.Show;

end;

procedure TDiscribeForm.ShowDemoLabelClick(Sender: TObject);

begin

DemonstrationForm.Show;

end;

procedure TDiscribeForm.FormShow(Sender: TObject);

begin

DiscribeForm.Caption := SearchName[SearchID];

Search\_Name.Caption := SearchName[SearchID];

Search\_Name.Left:=(DiscribeForm.Width-Search\_Name.Width) div 2;

SetSearchDiscription();

end;

end.

unit DemonstrationFormUnit;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.Grids, Vcl.StdCtrls;

type

TDemonstrationForm = class(TForm)

AlgorithmName: TLabel;

Label1: TLabel;

ElementsCount: TEdit;

MaxCountElementsLabel: TLabel;

DataSearchLabel: TLabel;

Element: TEdit;

DataGrid: TStringGrid;

GenerateData: TPanel;

SubmitCountData: TPanel;

FindNumber: TPanel;

Shape1: TShape;

Label2: TLabel;

Shape2: TShape;

Label3: TLabel;

GenerateDataLabel: TLabel;

Label4: TLabel;

procedure SetSearchName();

procedure Sort(StringGrid:TStringGrid);

function isGridEmpty(StringGrid:TStringGrid):boolean;

procedure FormShow(Sender: TObject);

procedure SubmitCountDataClick(Sender: TObject);

procedure ElementsCountChange(Sender: TObject);

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

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

Rect: TRect; State: TGridDrawState);

procedure FindNumberClick(Sender: TObject);

procedure GenerateDataLabelClick(Sender: TObject);

procedure SubmitCountDataMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure SubmitCountDataMouseLeave(Sender: TObject);

private

public

end;

var

DemonstrationForm: TDemonstrationForm;

CellsState: array of integer;

implementation

{$R \*.dfm}

uses DiscribeFormUnit, SearchUnit, DesignUnit;

procedure TDemonstrationForm.FindNumberClick(Sender: TObject);

begin

if isGridEmpty(DataGrid) = true then

begin

showmessage('Требуется заполнить массив');

exit;

end;

for var i := 0 to DataGrid.ColCount-1 do

begin

DataGrid.Rows[0].Objects[i] := TObject(1);

end;

case SearchID of

0: LinearSearch(DataGrid, StrToInt(Element.text));

1: BinarySearch(DataGrid, StrToInt(Element.text));

2: JumpSearch(DataGrid, StrToInt(Element.text));

3: InterpolationSearch(DataGrid, StrToInt(Element.text));

4: ExponentionalSearch(DataGrid, StrToInt(Element.Text))

end;

end;

procedure TDemonstrationForm.DataGridDrawCell(Sender: TObject; ACol,

ARow: Integer; Rect: TRect; State: TGridDrawState);

begin

var flag:integer;

flag := Integer(DataGrid.Rows[ARow].Objects[ACol]);

if (flag < 1) and (flag > 3) then

exit;

with DataGrid.Canvas do

begin

case flag of

1:

{default}

Brush.Color := clWhite;

2:

{select}

Brush.Color := clYellow;

3:

{find}

Brush.Color := $0024FF00;

end;

FillRect(rect);

TextOut(rect.Left + 2, rect.Top+2, DataGrid.Cells[ACol, ARow]);

end;

end;

function TDemonstrationForm.isGridEmpty(StringGrid:TStringGrid):boolean;

begin

for var i := 0 to StringGrid.ColCount-1 do

if StringGrid.Cells[i, 0] = '' then

begin

isGridEmpty := true;

exit

end;

isGridEmpty := false;

end;

procedure TDemonstrationForm.Sort(StringGrid:TStringGrid);

begin

var temp:string;

for var i := 0 to StringGrid.ColCount-1 do

for var j := 0 to StringGrid.ColCount - i - 1 do

begin

if StrToInt(StringGrid.Cells[j, 0]) > StrToInt(StringGrid.Cells[j+1, 0]) then

begin

temp := StringGrid.Cells[j, 0];

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

StringGrid.Cells[j+1, 0] := temp

end;

end;

end;

procedure TDemonstrationForm.GenerateDataLabelClick(Sender: TObject);

begin

for var i := 0 to StrToInt(ElementsCount.Text) do

begin

DataGrid.Cells[i, 0]:= IntToStr(Random(25));

DataGrid.Rows[0].Objects[i] := TObject(1);

CellsState[i]:=0;

end;

Sort(DataGrid);

end;

procedure TDemonstrationForm.ElementsCountChange(Sender: TObject);

begin

var dataCount: integer;

try

dataCount := StrToInt(ElementsCount.text);

if (dataCount > 13) or (dataCount < 1) then

ElementsCount.Color := clRed

else

ElementsCount.Color := clWhite;

except

ElementsCount.Color := clWhite;

end;

end;

procedure TDemonstrationForm.FormShow(Sender: TObject);

begin

SetSearchName();

DataGrid.Width:=DataGrid.ColCount\*68;

DataGrid.Left:=(DemonstrationForm.Width-DataGrid.Width) div 2;

SetLength(CellsState, DataGrid.ColCount);

end;

procedure TDemonstrationForm.FormClose(Sender: TObject;

var Action: TCloseAction);

begin

for var i:=0 to DataGrid.ColCount-1 do

begin

DataGrid.Cells[i, 0] := '';

DataGrid.Rows[0].Objects[i] := TObject(1);

end;

ElementsCount.Clear;

Element.Clear;

DataGrid.ColCount := 10;

end;

procedure TDemonstrationForm.SubmitCountDataClick(Sender: TObject);

begin

DataGrid.ColCount := StrToInt(ElementsCount.Text);

DataGrid.Width := DataGrid.ColCount \* 68;

DataGrid.Left :=(DemonstrationForm.Width - DataGrid.Width) div 2;

SetLength(CellsState, DataGrid.ColCount-1);

end;

procedure TDemonstrationForm.SetSearchName();

begin

AlgorithmName.Caption := SearchName[SearchID];

AlgorithmName.Left:=(DemonstrationForm.Width-AlgorithmName.Width) div 2;

Label4.Left:=(DemonstrationForm.Width-Label4.Width) div 2;

end;

procedure TDemonstrationForm.SubmitCountDataMouseLeave(Sender: TObject);

begin

selectButton(SubmitCountData);

end;

procedure TDemonstrationForm.SubmitCountDataMouseMove(Sender: TObject;

Shift: TShiftState; X, Y: Integer);

begin

selectButton(SubmitCountData);

end;

end.

unit SearchUnit;

interface

uses Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls, Vcl.Grids, Math, DemonstrationFormUnit;

procedure Delay(Value: Cardinal);

procedure setCellColor(StringGrid:TStringGrid; idx,color:integer;dTime:real);

procedure LinearSearch(StringGrid:TStringGrid; element:integer);

procedure BinarySearch(StringGrid:TStringGrid; element:integer);

procedure JumpSearch(StringGrid:TStringGrid; element:integer);

procedure InterpolationSearch(StringGrid:TStringGrid; element:integer);

procedure ExponentionalSearch(StringGrid:TStringGrid; element:integer);

procedure ExBinarySearch(StringGrid:TStringGrid; left, right, element:integer);

var kd:real = 1;

implementation

procedure Delay(Value: Cardinal);

var

F, N: Cardinal;

begin

N := 0;

while N <= (Value div 10) do

begin

SleepEx(1, True);

Application.ProcessMessages;

Inc(N);

end;

F := GetTickCount;

repeat

Application.ProcessMessages;

N := GetTickCount;

until (N - F >= (Value mod 10)) or (N < F);

end;

procedure setCellColor(StringGrid:TStringGrid; idx,color:integer;dTime:real);

begin

{

colors:

1 - default (White),

2 - selected (Yellow),

3 - find (Green)

}

StringGrid.Rows[0].Objects[idx] := TObject(color);

Delay(trunc(dTime));

end;

procedure LinearSearch(StringGrid:TStringGrid; element:integer);

begin

for var i := 0 to StringGrid.ColCount-1 do

begin

if StrToInt(StringGrid.Cells[i, 0]) = element then

begin

setCellColor(StringGrid, i, 3, 1000);

exit;

end

else

begin

setCellColor(StringGrid, i, 2, 1000);

setCellColor(StringGrid, i, 1, 0)

end;

end;

showmessage('Число не найдено')

end;

procedure BinarySearch(StringGrid:TStringGrid; element:integer);

begin

var left :=0;

var right := StringGrid.ColCount-1;

var mid:integer;

while left<=right do

begin

mid := left+(right-left) div 2;

if element = StrToInt(StringGrid.Cells[mid, 0]) then

begin

setCellColor(StringGrid, mid, 3, 1000);

exit;

end

else if element < StrToInt(StringGrid.Cells[mid, 0]) then

begin

right := mid ;

setCellColor(StringGrid, mid, 2, 1000);

setCellColor(StringGrid, mid, 1, 0);

end

else if element > StrToInt(StringGrid.Cells[mid, 0]) then

begin

left := mid + 1;

setCellColor(StringGrid, mid, 2, 1000);

setCellColor(StringGrid, mid, 1, 0);

end

end;

end;

procedure JumpSearch(StringGrid:TStringGrid; element:integer);

begin

var high := StringGrid.ColCount-1;

var jump:integer := floor(sqrt(high));

var left:integer := 0;

var right:integer := 0;

var i:integer := 0;

while (left < high) and (StrToInt(StringGrid.Cells[left, 0]) <= element) do

begin

right := min(high-1, left+jump);

setCellColor(StringGrid, right, 2, 1000);

setCellColor(StringGrid, right, 1, 0);

if (StrToInt(StringGrid.Cells[left, 0]) <= element) and (StrToInt(StringGrid.Cells[right, 0]) >= element) then

break;

left := left + jump;

setCellColor(StringGrid, left, 2, 1000);

setCellColor(StringGrid, left, 1, 0);

end;

if (left >= high) or (StrToInt(StringGrid.Cells[left, 0]) > element) then

exit;

right := min(high-1, right);

setCellColor(StringGrid, right, 2, 1000);

setCellColor(StringGrid, right, 1, 0);

i := left;

while (i <= right) and (StrToInt(StringGrid.Cells[i, 0]) <= element) do

begin

if StrToInt(StringGrid.Cells[i, 0]) = element then

begin

setCellColor(StringGrid, i, 3, 1000);

exit;

end;

setCellColor(StringGrid, i, 2, 1000);

setCellColor(StringGrid, i, 1, 0);

i:=i+1;

end;

showmessage('Число не найдено')

end;

procedure ExBinarySearch(StringGrid:TStringGrid; left, right, element:integer);

begin

if left > right then

begin

showmessage('Число не найдено');

exit;

end;

var mid := left+(right-left) div 2;

setCellColor(StringGrid, mid, 2, 1000);

setCellColor(StringGrid, mid, 1, 0);

if element = StrToInt(StringGrid.Cells[mid, 0]) then

begin

setCellColor(StringGrid, mid, 3, 1000);

exit

end

else if StrToInt(StringGrid.Cells[mid, 0]) > element then

ExBinarySearch(StringGrid, left, mid - 1, element)

else

ExBinarySearch(StringGrid, mid + 1, right, element);

end;

procedure ExponentionalSearch(StringGrid:TStringGrid; element:integer);

begin

var bound := 1;

var high := StringGrid.ColCount-1;

while (bound < high) and (StrToInt(StringGrid.Cells[bound, 0]) < element) do

begin

bound := bound \* 2;

setCellColor(StringGrid, bound, 2, 1000);

setCellColor(StringGrid, bound, 1, 0);

end;

ExBinarySearch(StringGrid, bound div 2, min(bound, high), element)

end;

procedure InterpolationSearch(StringGrid:TStringGrid; element:integer);

begin

var low := 0;

var high := StringGrid.ColCount-1;

var index: integer;

var a := StrToInt(StringGrid.Cells[low, 0]);

var b := StrToInt(StringGrid.Cells[high, 0]);

while (low <= high) and (StrToInt(StringGrid.Cells[low, 0]) <= element) and (StrToInt(StringGrid.Cells[high, 0]) >= element) do

begin

index := low + floor((((high-low)/ (b - a)) \* (element - a)));

if StrToInt(StringGrid.Cells[index, 0]) = element then

begin

setCellColor(StringGrid, index, 3, 1000);

exit;

end;

if StrToInt(StringGrid.Cells[index, 0]) < element then

begin

low := index + 1;

setCellColor(StringGrid, low, 2, 1000);

setCellColor(StringGrid, low, 1, 0);

end

else

begin

high := index - 1;

setCellColor(StringGrid, high, 2, 1000);

setCellColor(StringGrid, low, 1, 0);

end;

end;

showmessage('Число не найдено')

end;

end.

unit DesignUnit;

interface

uses Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.WinXCtrls, Vcl.Grids,

Vcl.ExtCtrls, Vcl.Buttons, Vcl.WinXPickers, Vcl.Imaging.pngimage;

procedure selectMainMenu(Panel:TPanel; SelectInd:TShape);

procedure UnselectMainMenu(Panel:TPanel; SelectInd:TShape);

procedure selectSearch(Panel:TPanel; Image:TImage);

procedure UnselectSearch(Panel:TPanel; Image:TImage);

procedure UnselectButton(Panel:TPanel);

procedure selectButton(Panel:TPanel);

implementation

uses Main\_FormUnit;

procedure selectMainMenu(Panel:TPanel; SelectInd:TShape);

begin

Panel.Color:=$a7a7a7;

SelectInd.Visible:=true;

SelectInd.Brush.Color:=$717171;

end;

procedure UnselectMainMenu(Panel:TPanel; SelectInd:TShape);

begin

Panel.Color:=$d9d9d9;

SelectInd.Visible:=false;

end;

procedure selectSearch(Panel:TPanel; Image:TImage);

begin

Screen.Cursor:=crHandPoint;

Panel.Color:=$d5d5d5;

end;

procedure UnselectSearch(Panel:TPanel; Image:TImage);

begin

Panel.Color:=$d9d9d9;

Screen.Cursor:=crDefault;

end;

procedure selectButton(Panel:TPanel);

begin

Screen.Cursor:=crHandPoint;

Panel.Color:=$d5d5d5;

end;

procedure UnselectButton(Panel:TPanel);

begin

Panel.Color:=$d9d9d9;

Screen.Cursor:=crDefault;

end;

end.

unit CodeFormUnit;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.OleCtrls, SHDocVw, Vcl.StdCtrls,

Vcl.Imaging.pngimage, Vcl.ExtCtrls;

type

TCodeForm = class(TForm)

SearchCode: TWebBrowser;

CodeLabel: TLabel;

PythonCode: TImage;

PascalCode: TImage;

Image1: TImage;

procedure FormShow(Sender: TObject);

procedure PascalCodeClick(Sender: TObject);

procedure PythonCodeClick(Sender: TObject);

procedure Image1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

CodeForm: TCodeForm;

implementation

{$R \*.dfm}

uses DiscribeFormUnit, Main\_FormUnit;

procedure TCodeForm.FormShow(Sender: TObject);

begin

CodeLabel.Left:=(CodeForm.Width-CodeLabel.Width) div 2;

CodeLabel.Caption := SearchName[SearchID];

CodeForm.Caption := SearchName[SearchID];

SearchCode.Silent := true;

SearchCode.Navigate(ExtractFilePath(paramstr(0))+ 'SearchCodes/' + SearchPathsName[SearchID] + '/pas.html');

end;

procedure TCodeForm.Image1Click(Sender: TObject);

begin

CodeForm.close;

DiscribeForm.Show;

end;

procedure TCodeForm.PascalCodeClick(Sender: TObject);

begin

SearchCode.Navigate(ExtractFilePath(paramstr(0))+ 'SearchCodes/' + SearchPathsName[SearchID] + '/pas.html');

end;

procedure TCodeForm.PythonCodeClick(Sender: TObject);

begin

SearchCode.Navigate(ExtractFilePath(paramstr(0))+ 'SearchCodes/' + SearchPathsName[SearchID] + '/py.html');

end;

end.

unit AboutFormUnit;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Imaging.pngimage,

Vcl.ExtCtrls, ShellAPI;

type

TAboutForm = class(TForm)

LogoImage: TImage;

Label1: TLabel;

Label2: TLabel;

CloseButton: TImage;

Label3: TLabel;

Label4: TLabel;

Panel1: TPanel;

procedure CloseButtonClick(Sender: TObject);

procedure Panel1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

AboutForm: TAboutForm;

implementation

{$R \*.dfm}

procedure TAboutForm.CloseButtonClick(Sender: TObject);

begin

AboutForm.close;

end;

procedure TAboutForm.Panel1Click(Sender: TObject);

begin

ShellExecute(0, Pchar('Open'), PChar('spravka.chm'), nil, nil, SW\_SHOW);

end;

end.

unit SplashFormUnit;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.Imaging.pngimage;

type

TSplashForm = class(TForm)

Timer1: TTimer;

Image1: TImage;

procedure Timer1Timer(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

SplashForm: TSplashForm;

implementation

{$R \*.dfm}

procedure TSplashForm.Timer1Timer(Sender: TObject);

begin

Timer1.Enabled := False;

end;

end.