Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Издательского дела и полиграфии

Кафедра Информационных систем и технологий

Специальность Информационные системы и технологии

Специализация 1-40 01 02-03

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОЙ РАБОТЫ**

по дисциплине «Специализированные информационные системы в медиа, издательском деле и полиграфии»

Тема:

Преобразование алгоритмов с языка Геракл на Язык DELPHI\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Исполнитель

студент(ка)\_\_\_ курса группы\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата Ф.И.О.

Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

должность, ученая степень, ученое звание подпись, дата Ф.И.О.

Курсовая работа защищена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись Ф.И.О.

Минск 2013

# **Реферат**

Пояснительная записка содержит 16 страниц, 4 рисунков, 1 схемы и 2 источников.

СРЕДСТВО АВТОМАТИЗАЦИИ РАБОТЫ, ЯЗЫК ПРОГРАММИРОВАНИЯ DELPHI, ИНСТРУМЕНТАЛЬНАЯ СРЕДА RAD STUDIO XE5, АВТОМАТИЗИРОВАННАЯ СИСТЕМА, ИНФОРМАЦИОННАЯ СИСТЕМА

Курсовой проект состоит из пояснительной записки и программы на языке Delphi.

Задание: преобразование алгоритмов с языка Геракл на язык DELPHI .

* 1. В приложении Гиперсистема выполнить преобразование алгоритмов Algor1, Algor2, Algor3.
  2. Добавить в программу проверку синтаксиса алгоритмов.

Цель выполнения курсового проекта: разработать программное средство выполняющее синтаксический анализ языка Геракл.

Во введении приводятся понятие электронного учебника, данные по использованию, защите от копирования и разработке пособия;

В первом разделе проводится анализ литературы и рассматриваются наиболее доступные технологии и инструменты для разработки.

Во втором разделе рассматривается структура и метод создания программного средства, а так же обзор основных функций и методов, используемых при разработке.

В третьем описывается тестирование и анализ полученных результатов.

В заключении – вывод о проделанной работе.

**Оглавление**

[Реферат 3](#_Toc375493414)

[Введение 5](#_Toc375493415)

[1 Аналитический обзор литературы 6](#_Toc375493416)

[1.1 Язык Герекл 6](#_Toc375493417)

[1.2 Среда разработки RAD Studio XE5 6](#_Toc375493418)

[1.3 Язык программирования Delphi 8](#_Toc375493419)

[1.4 Патентный поиск 9](#_Toc375493420)

[2 Разработка программной системы 10](#_Toc375493421)

[2.1 Описание структуры программного средства 10](#_Toc375493422)

[2.2 Разработка интерфейса 11](#_Toc375493423)

[2.3 Программирование элементов управления 12](#_Toc375493424)

[2.3.1 Листинг кода проверки на заполнение полей 12](#_Toc375493425)

[2.3.2 Листинг кода загрузки файла 12](#_Toc375493426)

[2.3.3 Листинг кода сохранения файла 12](#_Toc375493427)

[2.3.4 Листинг кода выполняющего синтаксический анализ 13](#_Toc375493428)

[3 Тестирование и анализ результатов 14](#_Toc375493429)

[Заключение 16](#_Toc375493430)

[Литература 17](#_Toc375493431)

# **Введение**

Целью данного курсового проекта является разработка программного средства, выполняющего синтаксический анализ блок-схемы алгоритма на языке Геракл. Программное средство должно иметь возможность принимать в качестве входного параметра путь к файлу на локальном и или удалённом компьютере в сети, а в качестве выходного параметра – проанализированный и скорректированный текстовый файл на языке Геракл, с последующим сохранением в указанной папке и именем файла.

Синтаксический анализ – это процесс, который определяет, принадлежит ли некоторая последовательность лексем языку, порождаемому грамматикой. В принципе, по любой грамматике можно построить синтаксический анализатор, но грамматики, используемые на практике, имеют специальную форму. Например, известно, что для любой контекстно-свободной грамматики может быть построен анализатор, сложность которого не превышает O(n3) для входной строки длины n, но в большинстве случаев по заданному языку программирования мы можем построить такую грамматику, которая позволит сконструировать и более быстрый анализатор. Анализаторы реально используемых языков обычно имеют линейную сложность; это достигается, например, за счет просмотра исходной программы слева направо с заглядыванием вперед на один терминальный символ (лексический класс).

Вход синтаксического анализатора – последовательность лексических и таблицы, например, таблица внешних представлений, которые являются выходом лексического анализатора.

Выход синтаксического анализатора – дерево разбора и таблицы, например, таблица идентификаторов и таблица типов, которые являются входом для следующего просмотра компилятора (например, это может быть просмотр, осуществляющий контроль типов).

Отметим, что совсем необязательно, чтобы фазы лексического и синтаксического анализа выделялись в отдельные просмотры. Обычно эти фазы взаимодействуют друг с другом на одном просмотре. Основной фазой такого просмотра считается фаза синтаксического анализа, при этом синтаксический анализатор обращается к лексическому анализатору каждый раз, когда у него появляется потребность в очередном терминальном символе.

# **Аналитический обзор литературы**

# **Язык Герекл**

Запись блок-схемы алгоритма на языке ГЕРАКЛ выполняется с помощью фраз, представленных в виде текстовых строк. Ключевые слова ГЕРАКЛ записываются на русском языке.

Синтаксис языка

* Нач:<ИМЯ> - блок начала алгоритма;
  + ИМЯ – идентификатор
* Кон:<ИМЯ> - блок конца алгоритма;
* Ком:<ТЕКСТ> - блок комментария;
  + ТЕКСТ – однострочный текст
* <ИМЯ>=< ЛОГИЧЕСКОЕ ВЫРАЖЕНИЕ> - блок действия;
  + ЛОГИЧЕСКОЕ ВЫРАЖЕНИЕ – выражение с операндами перечислимого типа, операциями отношения и логическими операциями
* <ИМЯ>(<СПИСОК ПАРАМЕТРОВ>) - блок вызова функции;
  + СПИСОК ПАРАМЕТРОВ – список переменных и констант перечислимого типа
* <ИМЯ>() - блок вызова процедуры;
* Если (<ЛОГИЧЕСКОЕ ВЫРАЖЕНИЕ>) То <ЧИСЛО> Иначе <ЧИСЛО> - блок решения;
  + ЧИСЛО – целое без знака
* <ЧИСЛО> - точка слияния;
* <ЧИСЛО>: - метка блока.

Семантика языка:

* Линейная последовательность блоков на схеме алгоритма, заканчивающаяся *блоком решение,* *точкой слияния* или *блоком конца алгоритма* считается одним объединенным блоком;
* Перед каждым объединенным блоком ставится *метка блока*. Первый объединенный блок (после *блока начала алгоритма*) должен иметь метку «1:»;
* Параметр *<*ЧИСЛО*> точки слияния* должен совпадать с параметром *<*ЧИСЛО*>* *метки блока,* следующего за *точкой слияния;*
* Если после объединенного блока следует *точка слияния*, то она записывается в следующей за блоком строке.

# **Среда разработки RAD Studio XE5**

RAD Studio XE5 - это среда разработки native-приложений для рабочих групп, которая позволяет быстро создавать приложения для Android и iOS, загружать их в магазины приложений и внедрять в организациях. Создавайте приложения с помощью единой базы исходного кода без ущерба для качества, возможностей подключения и производительности. Благодаря native-поддержке Android и iOS вы сможете выйти на крупнейшие в мире рынки приложений для мобильных устройств.

* **Delphi XE5. Embarcadero® Delphi® XE5** - это простая в освоении среда разработки native-приложений для рабочих групп, которая помогает создавать приложения для Android и iOS и быстро загружать их в магазины приложений и предоставлять предприятиям.
* **C++Builder XE5. Embarcadero® C++Builder® XE5** - это стандартная среда разработки для рабочих групп, которая помогает создавать native-приложения для Windows и OS X и предоставлять их пользователям и предприятиям.
* **HTML5 Builder XE5**. Средство для максимально простой и быстрой разработки мобильных и веб-приложений на основе единой базы исходных кодов HTML5, CSS3, PHP и JavaScript.
* **DB PowerStudio XE3.5** Developer Edition (только в редакции Ultimate). Средство для быстрой разработки баз данных SQL, управления изменениями в базах данных, профилирования и настройки SQL.
* **ER/Studio 9.5 Developer Edition** (только в редакции Architect). ER/Studio помогает проектировщикам баз данных находить, документировать и повторно использовать данные и предоставляет средства обратного проектирования, анализа и оптимизации существующих баз данных.
* **InterBase XE3 Developer Edition** и **InterBase XE3 ToGo** (в редакции Professional и более старших редакциях). InterBase Developer Edition - это лучшая кросс-платформенная база данных для создания и тестирования приложений баз данных, включая встраиваемые, предназначенных для малых и средних предприятий.
* **База данных IBLite для Android и iOS** с лицензией на бесплатное развертывание (в редакции Professional и в редакциях Enterprise, Ultimate и Architect). Новая редакция InterBase, которую можно бесплатно распространять и использовать в ваших мобильных приложениях. Разработчики, желающие встроить в приложение базу данных с более широкими возможностями, могут использовать InterBase ToGo для iOS (продается отдельно).
* **Дополнительные средства. RAD Studio содержит широкий набор дополнительных средств, помогающих реализовывать недоступные ранее возможности.**
  + **FastReport VCL RAD Edition** и **FastReport FMX Embarcadero Edition** от компании Fast Reports - быстрое проектирование и формирование отчетов.
  + **Documentation Insight Express** от компании DevJET Software - наглядные средства для чтения и написания документации в формате XML в интегрированной среде разработки Delphi.
  + **TeeChart Standard** от компании Steema - быстрое и простое создание графиков и диаграмм.
  + **VCL для веб-решений (IntraWeb)** компании AtoZed - полнофункциональная инфраструктура разработки веб-приложений RAD.
  + **CodeSite Express** - ведение журналов для повышения качества приложений.
  + **AQtime Standard** от компании SmartBear - профилирование производительности.
  + **Beyond Compare Text Compare** - сравнение файлов исходного кода.

# **Язык программирования Delphi**

Delphi (Де́лфи) — императивный, структурированный, объектно-ориентированный язык программирования, диалект Object Pascal. Начиная со среды разработки Delphi 7.0, в официальных документах Borland стала использовать название Delphi для обозначения языка Object Pascal. Начиная с 2007 года уже язык Delphi (производный от Object Pascal) начал жить своей самостоятельной жизнью и претерпевал различные изменения, связанные с современными тенденциями (например, с развитием платформы .NET) развития языков программирования: появились class helpers, перегрузки операторов и другое.

Расширения файлов:

* .pas — исходный код модуля (pascal)
* .dpr — исходный код проекта (pascal)
* .dproj — исходный код проекта (xml)
* .dproj.local — исходный код проекта (xml)
* .dfm — исходный код формы
* .dpk — скомпилированный пакет
* .dcu — скомпилированный модуль
* .exe — скомпилированное приложение
* .res — ресурсы
* .dsk — привязки к файлам
* .identcache — кэшированные привязки к файлам

Среди многих распространённых программных продуктов, написанных на Delphi, можно найти:

* Продукция Embarcadero: Embarcadero Delphi, Embarcadero C++ Builder, Borland JBuilder 1 и 2 версии;
* Администрирование и разработка баз данных: MySQL Tools, IBExpert;
* Инженерное программное обеспечение: Altium Designer;
* Файловые менеджеры: Total Commander, Frigate, ViewFD;
* Просмотрщики графики: FastStone Image Viewer, FuturixImager, drComRead;
* Видео- и аудиопроигрыватели: Light Alloy, The KMPlayer, AIMP, X-Amp, Nata Player;
* Программы мгновенного обмена сообщениями: Skype, QIP 2012, R&Q, The Bat!, PopTray, FeedDemon;
* Создание музыки: FL Studio, Guitar Pro (до версии 6.0);
* Разработка программного обеспечения: Dev-C++, DUnit, Game Maker, Inno Setup, PyScripter;
* Веб-разработка: Macromedia HomeSite;
* Текстовые редакторы: BirEdit, Notepad GNU, Bred;
* Бухучёт и налогообложение: «ПАРУС», AVARDA (до версии 6.x включительно);
* Программы для создания анимаций: Pivot Stickfigure Animator;
* Программы для сжатия данных: ALZip, PowerArchiver;
* Компьютерные игры: Age of Wonders, «Космические рейнджеры», Venom. Codename: Outbreak, Space Empires V, «Правда о девятой роте»;
* Графические редакторы: Real Paint;
* Системные утилиты: Auslogics BoostSpeed;
* Системы управления очередью: МАКСИМА;
* Биллинговые системы: Петер-Сервис, Аргус.

# **Патентный поиск**

При осуществлении патентного поиска не были найдены патенты, патентующих «Преобразование алгоритмов с языка Геракл на язык DELPHI». Это означает, что на момент написания курсового проекта не существует аналогов разрабатываемого программного средства.

# **Разработка программной системы**

# **Описание структуры программного средства**

Реализация программного средства выполняющего синтаксический анализ кода на языке Геракл осуществлялось в среде разработки RAD Studio XE5 на языке Delphi.

Результат работы программного средства и анализа текстового файла с описанием блок-схемы алгоритма на языке Геракл представлено на следующей блок-схеме (рисунок 2.1).

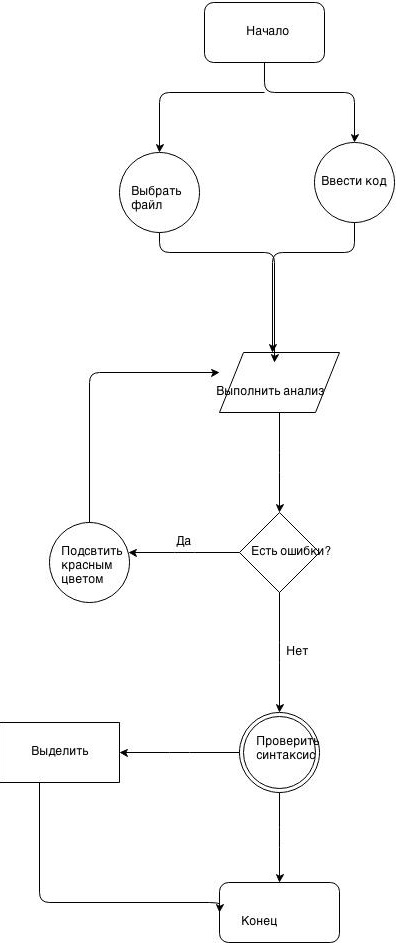


Рисунок 2.1 — Блок-схема работы программного средства

Структура работы программного средства сочетает в себе простую, и в тоже время, понятную логику выполнения анализа ошибок для пользователя.

# **Разработка интерфейса**

Приступая к разработке пользовательского интерфейса было проанализировано большое количество уже существующих синтаксических анализаторов разных языков описания алгоритмов. Проанализировав их структуру было выбрано такое количество элементов управления, которое необходимо для реализации поставленной задачи. Все элементы должны выполнять строго определённые действия возложенные на них и иметь удобное расположение на форме.

В итоге после долгого анализа был подобран следующий интерфейс, изображённый на рисунке 2.2.

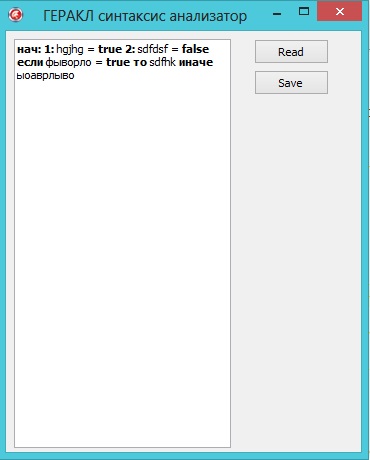
****

Рисунок 2.2 – Пользовательский интерфейс программного средства

Для начала работы пользователь должен выбрать файл для синтаксического анализа нажав кнопку «Read» или просто ввести код. После выбора файла, программное средство считывает весь алгоритм в элемент управления Memo и выводит содержимое файла с возможностью корректировки.

На следующем этапе необходимо нажать кнопку «Проверить». Программа-анализатор выполнит проверку текста на соответствие его содержимого синтаксису языка Геракл. Если содержимое файла соответствует языку Геракл, то выводится соответствующее сообщение. Если же содержимое файла не соответствует синтаксису языка Геракл, то выводится сообщение об ошибках. Сообщение содержит строки и их номера в элементе управления Memo. Далее пользователь основываясь на информации представленной в сообщении может отредактировать текст и снова проанализировать его.

# **Программирование элементов управления**

# **Листинг кода проверки на заполнение полей**

Во всем проекте для каждого элемента написаны проверки на заполнение полей, Которое выдает сообщение о типе незаполненного поля, устанавливает фокус на это поле и прерывает выполнение программы:

if Trim(edTextFilePath.Text) = '' then

begin

ShowMessage(Укажите путь к файлу!');

edTextFilePath.SetFocus;

Exit;

end;

# **Листинг кода загрузки файла**

Загрузка текстового файла осуществляется с помощью элемента управления OpenTextFileDialog. Код, выполняющий данные действия представлен ниже:

procedure TForm1.btnOpenFileClick(Sender: TObject);

begin

if OpenTextFileDialog1.Execute

then

begin

edTextFilePath.Text := OpenTextFileDialog1.FileName;

mTextFromFile.Lines.LoadFromFile(OpenTextFileDialog1.FileName);

end;

end;

Если пользователь указал путь к файлу, то выполняется считывание файла в элемент управления Memo.

# **Листинг кода сохранения файла**

Загрузка текстового файла осуществляется с помощью элемента управления SaveTextFileDialog. Код, выполняющий данные действия представлен ниже:

procedure TForm1.Button1Click(Sender: TObject);

begin

if SaveTextFileDialog1.Execute

then

begin

mTextFromFile.Lines.SaveToFile(SaveTextFileDialog1.FileName + '.txt');

end;

end;

Если пользователь ввёл имя файла для сохранения, процедура выполняет сохранение файла под пользовательским именем и указывает формат сохранения \*.txt.

# **Листинг кода выполняющего синтаксический анализ**

Выполнение синтаксического анализа представлено в следующей процедуре:

procedure TForm1.btnGoClick(Sender: TObject);

var

i: Integer;

begin

mErrors.Lines.Clear();

mErrors.Text:='';

for i:=0 to mTextFromFile.Lines.Count-1 do

begin

if (AnsiPos('Нач:', Copy(mTextFromFile.Lines[i],1, 4))<>0) and not(i = 0) then

begin

mErrors.Lines.Add('Ошибка в строке №' + IntToStr(i) + ' - неверная позиция "Нач:"');

continue;

end;

if (AnsiPos('Кон:', Copy(mTextFromFile.Lines[i],1, 4))<>0) and not(i = mTextFromFile.Lines.Count-1) then

begin

mErrors.Lines.Add('Ошибка в строке №'+IntToStr(i) + ' - неверная позиция "Кон:"');

continue;

end;

if Trim(mTextFromFile.Lines[i]) = '' then

begin

mErrors.Lines.Add('Ошибка в строке №'+IntToStr(i) + ' - присутствует пустая строка!');

continue;

end;

if Trim(mTextFromFile.Lines[i]) = '' then

begin

mErrors.Lines.Add('Ошибка в строке №'+IntToStr(i) + ' - присутствует пустая строка!');

continue;

end;

end;

end;

# **Тестирование и анализ результатов**

На данном этапе проведём тестирование реализованного программного средства. Выберем файл для тестирования (рисунок 3.1).

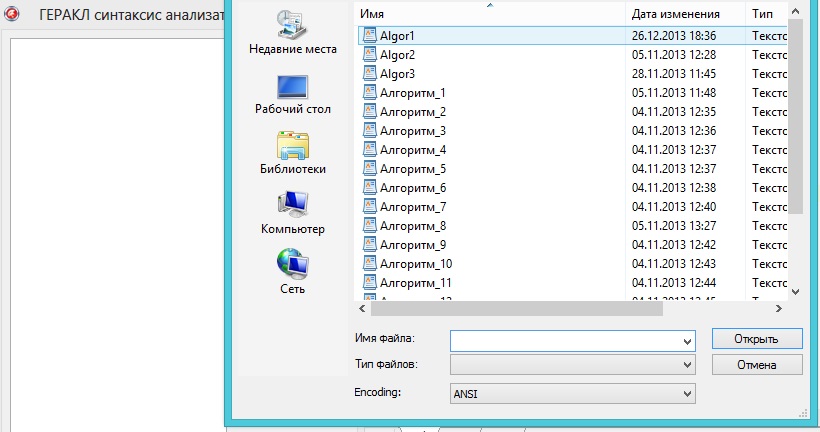


Рисунок 3.1 ­ Выбор файла

После нажатия кнопки «Открыть» содержимое файла подгружается в пользовательский элемент управления Memo (рисунок 3.2).

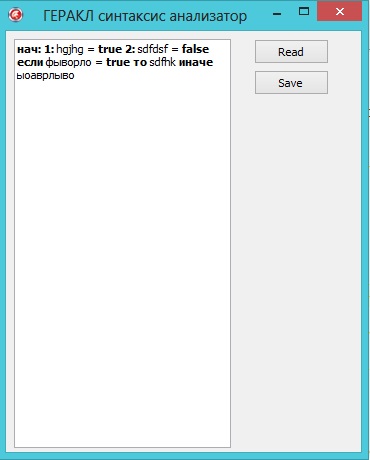


Рисунок 3.2 – Загрузка файла в пользовательский элемент управления

Далее удалим из пользовательского элемента управления первую строку «Нач:» и вставим её в любое другое место. После этого выполним проверку на соответствие синтаксиса языку Геракл (рисунок 3.3).

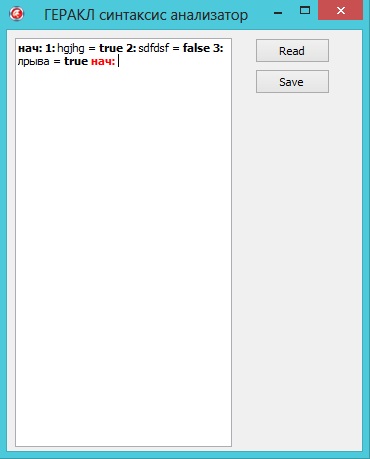


Рисунок 3.3 ­ Результат работы анализатора

Исходя из полученного результата можно сделать вывод, что разработанное программное средство выполняет поставленные перед собой задачи.

# **Заключение**

Современная степень развития коммуникационных ресурсов открыла перед разумным человечеством новые горизонты на поле образовательной деятельности, но при этом поставила и новые задачи. Решение одной из них - суть проделанной работы.

Бурное развитие информационных технологий, медленное, но неуклонное превращение компьютера из сакрального предмета, доступного лишь узкому кругу посвященных, в явление повседневной обыденности, появление Internet и т.д. — все это рано или поздно должно было затронуть и такую традиционно консервативную область, как отечественное образование. Автоматизированная система - это система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая автоматизированную технологию выполнения установленных функций.

В результате выполнения курсового проекта было разработано программное средство, выполняющее синтаксический анализ языка Геракл. Данное программное средство отлично справляется с поставленными задачами.

# **Литература**

1. Гультяев, А. К. Проектирование и дизайн пользовательского интерфейса [Текст] / А. К. Гультяев, В. А. Машин. − М.: КОРОНА принт, 2010. − 190 с.
2. Википедия [Электронный ресурс]. — Режим доступа: [http://ru.wikipedia.org](http://ru.wikipedia.org/). — Дата доступа: 20.12.2013.