МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
федеральное государственное бюджетное образовательное учреждение высшего образования
КРАСНОЯРСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ
им.В.П.АСТАФЬЕВА
(КГПУ им.В.П.Астафьева)

Институт/факультет Институт математики, физики и информатики
(полное наименование института/факультета/филиала)

Выпускающая кафедра Базовая кафедра информатики и информационных технологий в образовании
(полное наименование кафедры)

Шаповалов Егор Владимирович

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Тема **Среда разработки для обучения незрячих обучающихся программированию**

Направление подготовки 44.03.05 Педагогическое образование
(код и наименование направления)

Профиль Физика и информатика
(наименование профиля для бакалавриата)

ДОПУСКАЮ К ЗАЩИТЕ
Заведующий кафедрой
д.п.н., профессор Пак Н.И.
(ученая степень, ученое звание, фамилия, инициалы)

(дата, подпись)

Руководитель д.ф.-м.н., профессор кафедры ИИТО Романов В.А.
(ученая степень, ученое звание, фамилия, инициалы)

Дата защиты

Обучающийся Шаповалов Е. В.
(фамилия, инициалы)

(дата, подпись)

Оценка
(прописью)

Красноярск 20__

Содержание

Введение

В связи с развитием информационно-комуникационных технологий люди с нарушением зрения получают все больше возможностей для реализации себя в различных профессиях.

В наше время незрячие программисты не редкость.

T. V. Raman — один из первых незрячих программистов, добившихся успеха. Он с рождения плохо видел, а в 14 лет полностью лишился зрения из-за глаукомы. Он получил степень бакалавра, а затем и магистра в родной Индии и защитить докторскую диссертацию на тему «Audio System For Technical Readings» в США. С 2005 года он работает в Google. Там он решает проблемы адаптации Google Chrome и Android для слепых. На основе своей докторской он разработал голосовой интерфейс Emacspeak, которым пользуются многие слепые программисты для чтения текста на экране. Лукас Радаэлли - слепой программист, работает в Google. Рон Морфорд — владел компанией Automated Functions.

Обучение информатике на определённом этапе интенсивно использует программирование на ряде языков для развития профессиональных навыков, ряда компетенций и алгоритмического мышления.

Ключевым отличием в обучении является то, что незрячие люди при работе с компьютером воспринимают информацию на слух. Да, помимо восприятия информации на слух существует возможность восприятия информации тактильно, при помощи брайлевских дисплеев - специальных устройств, выводящих текстовую информацию шрифтом Брайля, но из-за малой распространенности данных устройств рассматривать этот способ мы не будем.

В главе 1 будут подробно разобраны вопросы восприятия информации при работе с экранными дикторами, а пока уже можно озвучить

актуальность настоящей работы, которая заключается в принципиально ином доминирующем канале восприятия у незрячих учеников.

Особенно важны:

- линейный характер восприятия информации на слух;
- отсутствие, в противоположность двумерному зрительному восприятию, аналога поля зрения и возможности перемещаться в визуальном поле в произвольном направлении;
- невозможность перехода скачком от участка экрана к участку как у зрячих пользователей, использующих такие эволюционно развитые механизмы системы визуального поиска и непроизвольной фокусировки внимания.

Дополнительно **актуальность** работы обусловлена рядом особенностей синтаксиса и сложившейся культуры работы с промышленными языками программирования, например:

- использование отступов как инструмента добавления мета-информации;
- невозможность использования скобок в ряде языков (Питон, регулярные выражения, ФОРТРАН).

Проблема исследования — (исторически сложившаяся) опора на визуальное восприятие при работе с ПК. На визуальное восприятие опирается и практически вся методика обучения, что делает её в нашем случае неэффективной из-за большого количества технических и когнитивных трудностей.

Объектом исследования выбран процесс выполнения лабораторных работ по программированию.

Цель работы: ускорение обучения алгоритмическому мышлению незрячих людей за счёт устранения целого ряда системных недостатков существующих языков и сред программирования.

Задачи работы:

1. Организация работы в среде программирования, позволяющей эффективно использовать программы экранного доступа.
2. Доработка инструментария среды, особенно навигации и сообщений об ошибках.
3. Доработка существующего языка на уровне внесения изменений в синтаксис (транспайлинг в существующий промышленный язык) с сохранением набора библиотек.
4. Покрытие кода юнит-тестами (особенно блока грамматики) для поддержки работы с бегущими релизами ключевых библиотек.
5. Анализ необходимых изменений в методике преподавания, с учётом новых возможностей среды и языка программирования.

В главе 1 настоящей работы подробно рассмотрены вопросы перехода на слуховой канал восприятия при работе с ПК и при выполнении лабораторных работ по программированию, разработана среда для изучения и выполнения программ, подведены промежуточные итоги.

В главе 2 описан процесс разработки грамматики нового языка и его внедрения.

Заключение содержит обзор результатов работы, а приложения — необходимую, но не существенную с точки зрения понимания информацию.

Глава 1. Среда разработки и выполнения программ

1.1. Выбор программы экранного доступа

Во введении было сказано, что незрячие люди при работе с компьютером воспринимают информацию на слух. Организация взаимодействия незрячего пользователя с компьютером происходит посредством программ экранного доступа. Программы экранного доступа - это программы или программные комплексы, обеспечивающие доступ (как правило, тактильный или речевой) незрячих и слабовидящих пользователей к информации, выводимой на экране компьютера [1].

Программа экранного доступа (ПЭД) является центральным звеном системы компьютерных тифлосредств и осуществляет передачу информации между операционной системой и прикладными программами, с одной стороны, и средствами рельефно-точечного и/или речевого вывода, с другой, обеспечивая как управление этими средствами, так и содержательное формирование информационного потока [2].

ПЭД позволяет читать текст, выводимый на экран. Наиболее популярными программами экранного доступа являются Jaws for Windows (Job Access With Speech) , коммерческая разработка фирмы Freedom Scientific (США) (официальный сайт — http://www.freedomscientific.com) и NVDA (NonVisual Desktop Access) — свободно распространяемая программма (официальный сайт проэкта https://www.nvaccess.org).

Несмотря на то, что NVDA является некомерческой программой, на данное время она уже не уступает платным аналогам, так как поддерживается большим числом разработчиков.

В своей работе мы будим опираться на использование NVDA как более доступный в финансовом отношении вариант для простого пользователя.

Кроме всего прочего, NVDA является продуктом с открытым кодом, что позволяет дорабатывать программу под свои нужды.

1.2. Выбор среды разработки (IDE)

Незрячие программисты при работе сталкиваются с рядом специфических проблем с интерфейсом.

В современных IDE (интегрированных средах разработки, от англ. Integrated development environment) присутствует большое количество визуальных элементов управления (кнопки, иконки, цветовое выделение и подчеркивание и др.) что обеспечивает высокую наглядность. К сожалению, эти средства визуализации оказываются неэффективны для людей с нарушением зрения, особенно для тотально незрячих [2].

Более того, визуальные элементы могут затруднить работу незрячему программисту. Примером может служить цветовое выделение ошибок в средах разработки языка Pascal — фрагмент кода с ошибкой выделяется цветом, но при этом не происходит вывода информации об ошибке в доступной незрячему программисту форме. Управление графическими элементами в некоторых средах разработки возможно только при помощи мыши, что также затрудняет работу незрячим.

Также встречаются интегрированные среды разработки, в которых текст программы во встроенном редакторе не читается программами экранного доступа ("КУМИР"). Экранным дикторам недоступны все среды разработки компании JetBrains [2]. В некоторых средах текст читается, но весь целиком, т.е. недоступна навигация по тексту. И так как воспринять большой фрагмент текста на не естественном (формальном) языке (т.е. не родном для человека, а значит тяжело воспринимаемом на слух) очень затруднительно, то эти среды программирования оказываются также малопригодными (Python IDLE, Microsoft Visual Studio).

В ряде языков программирования используются графический способ построения программы, например:

-язык "Дракон" [3];

-язык "Scratch" [4].

Также существуют обучающие среды программирования, в которых, для повышения наглядности обучения, используются формальные графические исполнители, например [4]:

-"кумир" - исполнители "Робот" и "Чертежник""

-"Robomind" - исполнитель робот".

Естественно, использование их при обучении незрячих и слабовидящих обучающихся невозможно.

Требования, которым должна отвечать среда разработки:

1. возможна работа в среде при помощи программы экранного доступа;
2. навигация по пунктам меню возможна при помощи клавиатуры без использования мыши;
3. возможность использования горячих клавиш для запуска определенных действий (запуск выполнения программы, навигация по тексту программы и пр.) (по возможности);
4. простота создания новых программ и последующей работы с ними;
5. вывод результата выполнения программы в удобной для просмотра форме.

Найти готовую бесплатную и лицензионно чистую среду разработки, отвечающую всем перечисленным критериям, не удалось и было принято создать такую среду, взяв мощных редактор текста и расширив его возможности при помощи дополнений (plugins).

При выборе основы для среды разработки учитывались следующие требования:

1. доступность для работы с программами экранного доступа;

2. удобство работы без использования мыши, т.е. удобная навигация по элементам управления с клавиатуры, возможность использования горячих клавиш;
3. возможность подключения расширений (плагинов) для организации обработки написанного кода программы и вывода результата (при отсутствии в исходной среде данных возможностей.

Плагин (англ. plug-in, от plug in «подключать») — независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения и/или использования её возможностей. Плагины обычно выполняются в виде библиотек общего пользования [5].

При выборе основы для среды разработки был рассмотрен ряд текстовых редакторов:

1. Notepad++
2. Atom
3. Sublime Text
4. Редактор ACE, встраиваемый в любой веб-интерфейс.

1.3. Установка Notepad++

В качестве основы для среды разработки был выбран Notepad++ (сайт - https://notepad-plus-plus.org). Notepad++ - бесплатный текстовый редактор с открытым исходным кодом для Windows, с подсветкой синтаксиса большого количества языков программирования и разметки. Базовая функциональность программы может быть расширена как за счёт плагинов, так и сторонних модулей, таких как компиляторы и препроцессоры [6].

Основными достоинствами Notepad++являются:

- доступен для программ экранного доступа;
- удобная навигация по пунктам меню;

- возможность использования горячих клавиш;
- возможность расширений при помощи плагинов;
- поддержка большого числа языков программирования;
- является бесплатной программой;
- высокое быстродействие.

Для работы была выбрана версия 7.3.1. На сайте предлагается для скачивания выбрать вариант программы в зависимости от вашей операционной системы - 32 или 64 бита, установочный файл или архив с портабельной программой, которая не требует установки на компьютер и готова к работе сразу после распаковки архива. Нами был выбран *установочный файл*. После скачивания на компьютер необходимо установить программу, для чего нужно кликнуть по установочному файлу клавишей "Enter" и следовать инструкциям мастера установки.

1.4. Создание расширений для Notepad++

Notepad++ поддерживает автоматизацию работы при помощи скриптов на языках Python, JScript, Lua, и других [6], т.е. имеется возможность управления функционалом Notepad++ посредством написания скриптов - файлов, содержащих сценарий действий, написанный на каком-либо языке программирования.

В качестве языка программирования был выбран язык Python.

Достоинства языка Python:

- расширяемость языка, т.е. имеется возможность совершенствования языка, исходный код доступен для любых манипуляций;
- наличие большого числа подключаемых к программе модулей, обеспечивающих различные дополнительные возможности.

Для редактора Notepad++ существует плагин Python Script (сайт - http://npppythonscript.sourceforge.net). Python Script позволяет управлять программой Notepad++ при помощи команд Python, а также создавать и запускать программы на языке Python при помощи Notepad++.

Рекомендации по установке, настройке и применению плагинов в редакторе Notepad++ можно найти здесь [7, 8, 9]

Плагин Python Script можно скачать по адресу - http://npppythonscript.sourceforge.net/download.shtml. На сайте предложен выбор между MSI (MicroSoft Installer), которая сама устанавливает Python Script, и архивом с файлами плагина, которые необходимо в ручную поместить в папке программы Notepad++ в директорию plugins.

Мы выбрали первый вариант. По окончанию скачивания, необходимо кликнуть по загруженному на компьютер файлу PythonScript_1.0.8.0.msi. Если плагин установлен правильно, то в Notepad++ в меню Plugins должен появиться пункт "Python Script" (Приложение А. Рис.1).

1.5. Доработка Notepad++ до IDE

Создаем скрипт, который будет организовывать считывание написанного в редакторе программного кода, выполнение его и выдачу результата в конце редактируемого файла (в нижней части окна редактора, под текстом прпрограммы после специального символа, обозначающего границу между кодом программы и результатом ее выполнения).

Для этого идем в меню "Plugins", выбираем пункт "Python Script", и в нем выбираем пункт "New Script" (Рис.2).

В результате у нас получилась среда разработки, отвечающая изначально заданным требованиям:

1. возможна работа в среде при помощи программы экранного доступа;
2. навигация по пунктам меню возможна при помощи клавиатуры без использования мыши;

3. есть возможность использования горячих клавиш для запуска определенных действий (запуск выполнения программы, навигация по тексту программы и пр.);
4. простота создания новых программ и последующей работы с ними;
5. вывод результата выполнения программы в удобной для просмотра форме.

   (Рис.3)

Для запуска выполнения написанной программы можно задать горячие клавиши (мы установили "ctrl" + "F5"). Для работы с редактором можно использовать распространенные сочетания горячих клавиш, например:

Сочетание	Действие
"ctrl" + "s"	сохранение изменений в файле;
"ctrl" + "a"	выделение всего текста в поле редактора;
"ctrl" + "c"	скопировать выделенный фрагмент текста;
"ctrl" + "x"	вырезать выделенный фрагмент текста;
"ctrl" + "v"	вставить текст, находящийся в буфере обмена на место курсора;
"ctrl" + "z"	отмена предыдущего действия (в Notepad++ возможно отменять действия до самого первого);
"ctrl" + "g"	вызов меню перехода к строке или символу по заданному номеру;
"ctrl" + "h"	вызов меню поиска заданного фрагмента текста с возможностью его замены;
"ctrl" + "tab"	переход к следующей вкладке;

"ctrl" + "shift" + "tab"	переход к предыдущей вкладке№
"ctrl" + "w"	закрыть текущую вкладку;
"alt" + "F4"	закрыть текущее окно редактора.

1.6. Процесс редактирования и выполнения программы

Для создания файла новой программы необходимо создать только текстовый документ. Открыть его возможно, наведя на него курсор , нажав клавишу вызова контекстного меню и выбрав пункт "edit with notepad++".

Результат выполнения программы (текст) выводится в конце поля редактора после текста программы и отделяется от него строкой, состоящей из символов дефис (минус) "-".

(Рис.4)

Результат выполнения программы можно скопировать в нужное место на компьютере: в текстовый файл, в текст сообщения электронной почты или, оставив его в конце файла программы, сохранить файл для последующей работы.

В получившейся среде возможно писать программы на языке Python, запуская редактируемую программу сочетанием клавиш Ctrl + F5.

Глава 2. Разработка языка программирования для выполнения лабораторных работ

2.1. Ключевые проблемы при выборе языка программирования

Если язык программирования расширений целиком определяется вопросами выразительной мощи и возможностями взаимодействия с внешним миром (стандартная библиотека и наличие развитой экосистемы пользователей), то язык, на котором будут выполняться лабораторные работы, выбирается совершенно из иных критериев.

Ниже будут перечислены ключевые причины, по которым вместо существующего языка был написан новый.

В первую очередь, при программировании крайне важно получать от программы, выполняемой в рамках лабораторной работы, отклик в виде гибко программируемого звука. Ярким примером такого дизайна служит язык BGT, созданный для незрячих пользователей. К сожалению, в BGT отсутствует возможность создания графических приложений и скромная стандартная библиотека, что значительно ограничивает возможности программиста, так как большинство современных программ имеет графический интерфейс.

Вторую большую проблему для учащихся составляет тот факт, что подавляющее большинство языков программирования имеют в основе английский язык - т.е. не родной язык.

Школьный курс английского языка практически не содержит терминов, употребляемых в программировании. Следовательно, обучающимся приходится фактически учить два языка.

Для учащихся с глубокими нарушениями зрения английский язык как основа языка программирования порождает еще одну большую проблему в

связи с тем, что озвучиваемые учителем команды языка пишутся не так как слышатся, например:

write - "райт";

while - "вайл";

case - "кейс" и др.

Следовательно, помимо логических ошибок, появляются еще и ошибки, связанные с неправильным восприятием учащимися.

"Полезным в обучении может быть возможность использования национальной лексики (родного языка обучаемого) для ключевых слов и идентификаторов…" [10].

"Служебные слова языка и сообщения системы должны быть понятны обучаемому. Иностранные слова создают дополнительную сложность, их заучивание отвлекает от основной задачи – создания правильных алгоритмов и программ.

Именно ведение обучения на основе родного языка с применением языково-программных средств и технологических приемов, которые разработаны на базе национального интерфейса помогает выработке и быстрому развитию алгоритмического мышления обучаемого" [11].

Приведённые примеры озвучивают сделанный выбор в пользу использования национального языка для программирования. Примерами учебных языков программирования, использующих национальную (русскую) лексику являются "КУМИР", "Рапира", "Робик", "Scratch", "Robomind" и др., но использовать их мы не можем, так как:

- среда "КУМИР" недоступна для работы с программами экранного доступа;
- языки "Робик" и "рапира" создавались для советской ЭВМ "Агат", в конце 70-ых, начале 80-ых годов XX века, и использовать их на современных компьютерах мы не имеем возможности.

Также существуют языки программирования, использующие русские ключевые слова, но не подходящие для обучения незрячих и слабовидящих; к примеру, языки, в которых программа строится графически:

- язык "Дракон";
- язык "Scratch".

Третья причина заключается в том, что языки и примеры на них несут отпечаток культуры работы в соответствующей нише. Это выражается в правилах именования и форматирования кода, использовании отступов как инструмента добавления мета-информации, в специфическом разбиении текста программы на модули. Для незрячих программистов использование отступов, например, делает исходный код на языке Си++ практически нечитаемым, потому что уменьшает отношение «сигнал/шум» в многие десятки раз.

Использование отступов для незрячего программиста заменяет использование скобок, но во многих языках невозможно использование скобок для группирования операторов (Питон, регулярные выражения, ФОРТРАН).

Таким образом было необходимо разработать учебный язык программирования, использующий русскую лексику, который будет возможно использовать для обучения незрячих и слабовидящих обучающихся в разработанной среде программирования. Учитывая огромное количество новых языков и их незавидную судьбу, следовало сразу решить проблему сохранения всех преимуществ использования широко известного и популярного языка программирования, в том числе и для мотивации обучающихся. Эти два противоречивых требования удалось совместить.

2.2. Разработка языка программирования для выполнения работ

Для работы с языком программирования необходимы:

1. грамотно построенная и выученная грамматика языка;
2. компилятор [12].

Начнём с п. 1. Язык программирования описывается формальной грамматикой. Грамматика — это совокупность определенных правил, по которым строятся предложения языка. Эти грамматические правила приписывают предложениям языка некоторую синтаксическую структуру, которая используется в дальнейшем при определении смысла высказываний.

Определение формальной грамматики (ФГ) включает в себя [13]:

- множество (алфавит) терминальных символов T;
- множество (алфавит) нетерминальных символов N;
- начальный нетерминальный символ Z из множества N;
- множество порождающих правил вида A :: B, где B - цепочка из любых символов грамматики (NET), A-цепочка, содержащая хотя бы один нетерминальный символ.

Компилятор — это программа, производящая компиляцию [14]. Компиляция - процесс, состоящий из следующих этапов:

1. Лексический анализ последовательности символов исходного кода программы, преобразуется в последовательность лексем;
2. Синтаксический (грамматический) анализ (парсинг) — последовательность лексем преобразуется в дерево разбора;
3. Семантический анализ — дерево разбора обрабатывается с целью установления его семантики (смысла);
4. Оптимизация — выполняется удаление излишних конструкций и упрощение кода с сохранением его смысла. Оптимизация может быть на разных уровнях и этапах — например, над промежуточным кодом или над конечным машинным кодом.

5. Генерация кода - из промежуточного представления порождается код на целевом языке.

Противоречие между новизной языка и, одновременно, необходимостью обеспечения надёжной базы пользователей и поддержки мирового масштаба разрешается как раз в последнем пункте 5, достаточно выбрать в качестве целевого языка хорошо известный язык мирового уровня. Этот приём не нов и называется *транспайлинг* исходного кода программы, записанной на создаваемом языке, в программный код на промежуточном (уже существующем) языке; средствами последнего затем уже и будет осуществляться окончательная трансляция и выполнение программы.

Транспайлинг, — преобразование исходного кода программы с одного языка программирования на другой язык [15], — не новая техника. Таким образом делаются языки, работающие поверх виртуальной машины Java (Kotlin, Scala, Groovy), например. Или EcmaScript 8/ ES.Next.

2.3. Транспайлинг в Питон

В результате поиска программных средств, позволяющих осуществлять транспайлинг с одного языка программирования на другой, были найдены следующие варианты:

Plyplus - мощный парсер написанный на языке Python [16].

Пакет Plyplus принимает данную ему грамматику языка программирования, и производит разбор программы на языке , преобразуя её в синтаксическое дерево.

По адресу: https://github.com/erezsh/plyplus/tree/master/examples помещены примеры - грамматика языка "Calc" (калькулятор), по которым возможно понять принципы создания грамматик (приложение В. лист 1).

Для дальнейшей работы с синтаксическим деревом создается программа, преобразующая получившееся дерево в код программы на языке

Python и производящая запуск выполнения программы (приложение В. лист 2-4).

К сожалению, Plyplus не поддерживает возможность работы с русским алфавитом, и, так как создаваемый язык должен иметь русские ключевые слова, то нам пришлось отказаться от использования Plyplus.

Так же было рассмотрено несколько библиотек с PEG-грамматикой (grako, tatsu), но от их использования пришлось отказаться по техническим причинам после обсуждения нашей проблемы с автором библиотек.

В итоге для построения и обхода AST-дерева был выбран пакет ANTLR4. ANTLR4 (от англ. - ANother Tool for Language Recognition) - генератор нисходящих анализаторов для формальных языков.

ANTLR преобразует контекстно-свободную грамматику в виде РБНФ в программу на C++, Java, C#, Python, Ruby. Используется для разработки компиляторов, интерпретаторов и трансляторов [17].

К достоинствам данной программы можно отнести:

- является свободным программным обеспечением;
- может преобразовывать программу на заданном языке в программу на широко распространенных языках программирования;
- имеется большая база грамматик распространенных языков программирования, таких как Pascal, C++ и т. д.;
- существуют наборы тестов для проверки созданных компиляторов для грамматик, входящих в базу программы. Таким образом, возможна при необходимости смена основного языка с Python на любой из списка;
- наличие большого количества литературы и документации по данной программе.

Наличие готовых файлов грамматик распространённых языков программирования позволит в дальнейшем создать грамматику

разрабатываемого языка, дополнив алфавит существующего языка символами Кириллицы, и, заменив английские ключевые слова на русские.

Также, достоинством данной программы по сравнением с Plyplus является то, что анализ синтаксического дерева и построение по нему программы на основном языке производит сам ANTLR4.

2.4. Установка ANTLR4

Инструкцию по установке можно найти по адресу: https://github.com/antlr/antlr4/blob/master/doc/getting-started.md.

Для работы ANTLR4 необходимо наличие на компьютере JAVA. Необходимо установить на компьютер JRE и JDK. JRE - Java Runtime Environment; пакет, включающий в себя JVM и минимальный набор библиотек для работы программ. Минимальная реализация виртуальной машины, необходимая для исполнения Java-приложений, без компилятора и других средств разработки.

Скачать установочные файлы можно по адресу: https://www.java.com/en/download/windows-64bit.jsp.

jvm - Java Virtual Machine - виртуальная машина Java исполняет байт-код Java, предворительно скомпилированный компилятором JAVAC из программного кода [18].

Скачать установочные файлы можно по адресу: https://www.java.com/en/download/windows-64bit.jsp.

После скачивания, необходимо кликнуть клавишей "Enter" по файлу jre-8u131-windows-x64.exe и следовать инструкциям мастера по установке.

После установки необходимо перезагрузить компьютер.

JDK (Java Development Kit) - комплект разработчика приложений на языке Java, включающий в себя компилятор Java (javac), стандартные библиотеки классов Java, примеры, документацию, различные утилиты и исполнительную систему Java (JRE).

Скачать можно по адресу: http://www.oracle.com/technetwork/pt/java/javase/downloads/jdk8-downloads-2133151.html

После того как JRE и JDK установлены на компьютер, нужно создать текстовый файл **javac_Egor.bat** и в этом файле поместить текст:

```
SET CLASSPATH=.;путь_к_файлу\antlr-4.5.3-complete.jar;%CLASSPATH
%

"c:\Program Files\Java\jdk1.8.0_131\bin\javac.exe" %*
```

Затем необходимо создать текстовый файл с названием grun и расширением .bat и поместить туда текст:

```
SET CLASSPATH=.;путь_к_файлу\antlr-4.5.3-complete.jar;%CLASSPATH
%

java org.antlr.v4.gui.TestRig %*
```

После чего необходимо создать текстовый файл с именем antlr4 и расширением .bat и поместить туда текст:

```
SET CLASSPATH=.;путь_к_файлу\ANTLR4\antlr-4.5.3-complete.jar;
%CLASSPATH%

java org.antlr.v4.Tool %*
```

Важно - необходимо всех трех файлах в место надписи "путь_к_файлу" вписать путь до файла **antlr-4.5.3-complete.jar** на вашем компьютере, иначе программа работать не будет.

2.5. Использование AnTLR4 для транспайлинга

Благодаря базе грамматик, полностью соответствующих стандартам и покрытых тестами, теперь возможно как перевести исходный код на большинстве языков программирования (ЯП) в AST-дерево, так и перевести AST-дерево в программу на любом выбранном языке.

В будущем это позволит как преобразовывать из популярного ЯП в созданный, так и изучать новые языки уже разработанными средствами.

Детали и прокомментированные примеры прилагаются в исходных кодах на диске.

Заключение

В работе разработана среда программирования и язык, полностью ориентированные на незрячих пользователей:

- редактор кода дружелюбен к программам экранного чтения;
- среда расширяется с помощью мощного языка программирования Питон;
- разработан инструментарий перевода программ из одного языка программирования в другой, что позволяет вести обучение на аккуратно спроектированном языке, но не думать о портируемости, написании стандартной библиотеки и так далее;
- создана база примеров и юнит-тестов для гарантии качества и документирования написанного кода.

Это позволит разработать методику, свободную от ряда значительных технических недостатков, приводящих к потерям до 70% времени урока.

Список литературы

1] Программы экранного доступа для Windows [электронный ресурс] - http://win.tiflocomp.ru

[2] Рощина М. А., Швецов В. И. Доступность Интернет-ресурсов для незрячих пользователей как фактор обеспечения им доступа к открытому образованию // Открытое образование. – 2010. – №. 1.

[3] Язык программирования "Дракон" [электронный ресурс] - https://ru.wikipedia.org/wiki/ДРАКОН

[4] Национальные» языки программирования [электронный ресурс] - https://habrahabr.ru/post/176243/

[5] Плагин - это [электронный ресурс] - https://ru.wikipedia.org/wiki/Плагин

[6] Notepad++ - это [электронный ресурс] - https://ru.wikipedia.org/wiki/Notepad%2B%2B

[7] Скрипт для Notepad++ на Python [электронный ресурс] - https://habrahabr.ru/post/135822/

[8] Notepad++ и Python [электронный ресурс] - http://python.su/forum/topic/8372/

[9] Ускоряем написание кода при помощи плагина [электронный ресурс] - http://seoskop.ru/entries/13-reviews/30-notepad-emmet-uskoryaem-napisanie-koda.html

[10] Учебный язык программирования [электронный ресурс] - https://ru.wikipedia.org/wiki/Учебный_язык_программирования#cite_note-kobilov-8

[11] Кобилов С. С. ОБРАЗОВАТЕЛЬНАЯ ИНФОРМАТИКА: ПОДХОД К ОБУЧЕНИЮ, ВЫБОР УЧЕБНЫХ ЯЗЫКОВ И СОЗДАНИЕ ПРОГРАММНЫХ СИСТЕМ [электронный ресурс] - http://ict.edu.ru/vconf/files/3197.rtf

[12] Компилятор - это [электронный ресурс] - https://ru.wikipedia.org/wiki/Компилятор

[13] Формальные грамматики и языки [электронный ресурс] - http://ermak.cs.nstu.ru/trans/Trans312.htm

[14] Компиляция [электронный ресурс] - https://ru.wikipedia.org/wiki/Компиляция

[15] Разница между: транспайлер, транслятор, компилятор [электронный ресурс] - https://toster.ru/q/231556

[16] Plyplus [электронный ресурс] - https://github.com/erezsh/plyplus

[17] ANTLR - это [электронный ресурс] - https://ru.wikipedia.org/wiki/ANTLR

[18] Вертуальная машина JAVA [электронный ресурс] - https://ru.wikipedia.org/wiki/Java_Virtual_Machine

[19] Швецов В. И., Рощина М. А. Компьютерные тифлотехнологии в социальной интеграции лиц с глубокими нарушениями зрения: Учебное пособие // Нижний Новгород: Нижегородский государственный университет им. НИ Лобачевского. – 2007.

[20] Как работают слепые программисты [электронный ресурс] - https://dev.by/lenta/main/kak-rabotayut-slepye-programmisty

[21] "Тот день, когда мы взяли на работу слепого программиста" [электронный ресурс] - https://geektimes.ru/post/277348/

[22] ANTLR4 примеры [электронный ресурс] - https://github.com/antlr/antlr4/blob/master/doc/getting-started.md#a-first-example

[23] ANTLR4 иPython2 [электронный ресурс] - https://dzone.com/articles/using-antlr-4-with-python-2

[24] ANTLR4 Грамматики [электронный ресурс] - https://github.com/antlr/grammars-v4

[25] Рувинская В. М., Пригожев А. С. Средства построения звукового интерфейса для незрячих пользователей ПК // Проблеми програмування. – 2003. – №. 4. – С. 100-106.

[26] ОСОБЕННОСТИ ОБУЧЕНИЯ ПРОГРАММИРОВАНИЮ НЕЗРЯЧИХ СТУДЕНТОВ МГППУ Нуркаева Ирина Михайловна [электронный ресурс] -http://ito.edu.ru/2004/Moscow/V/V-0-4335.html

Приложение А. Установка плагина и создание среды программирования на базе Notepad++

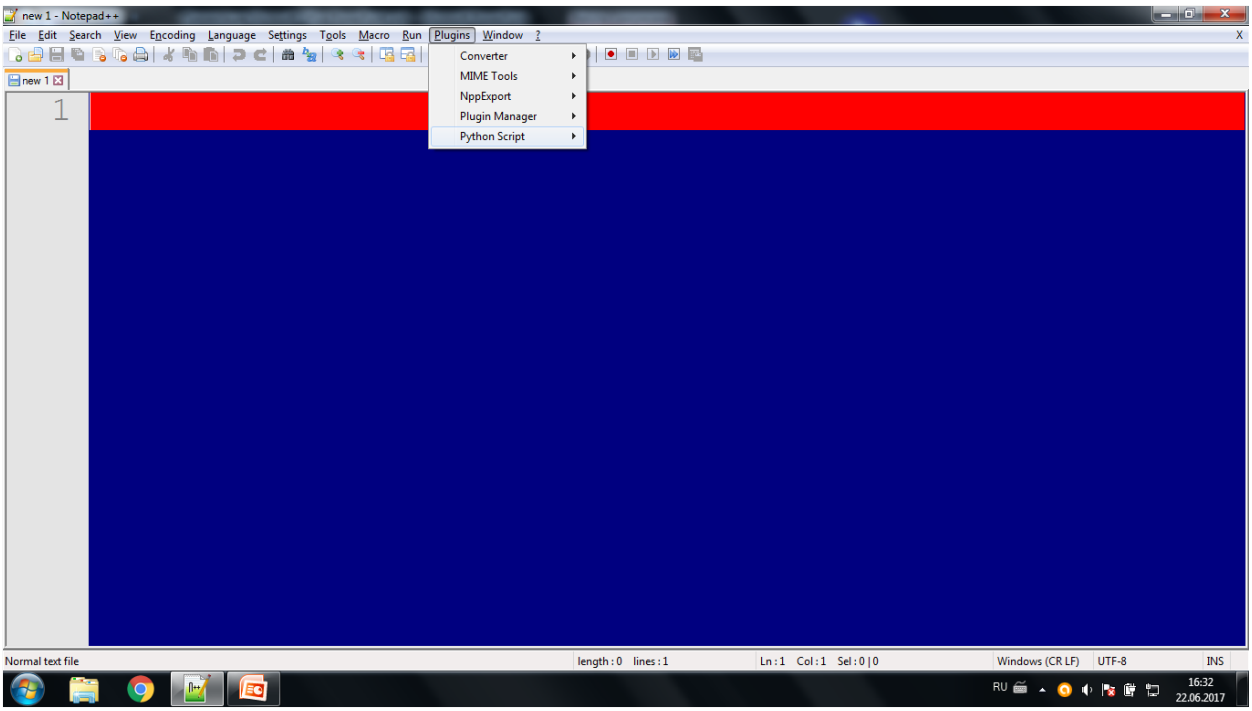

Рис. 1. Вид меню PlugIns в случае успешной установки PythonScript.

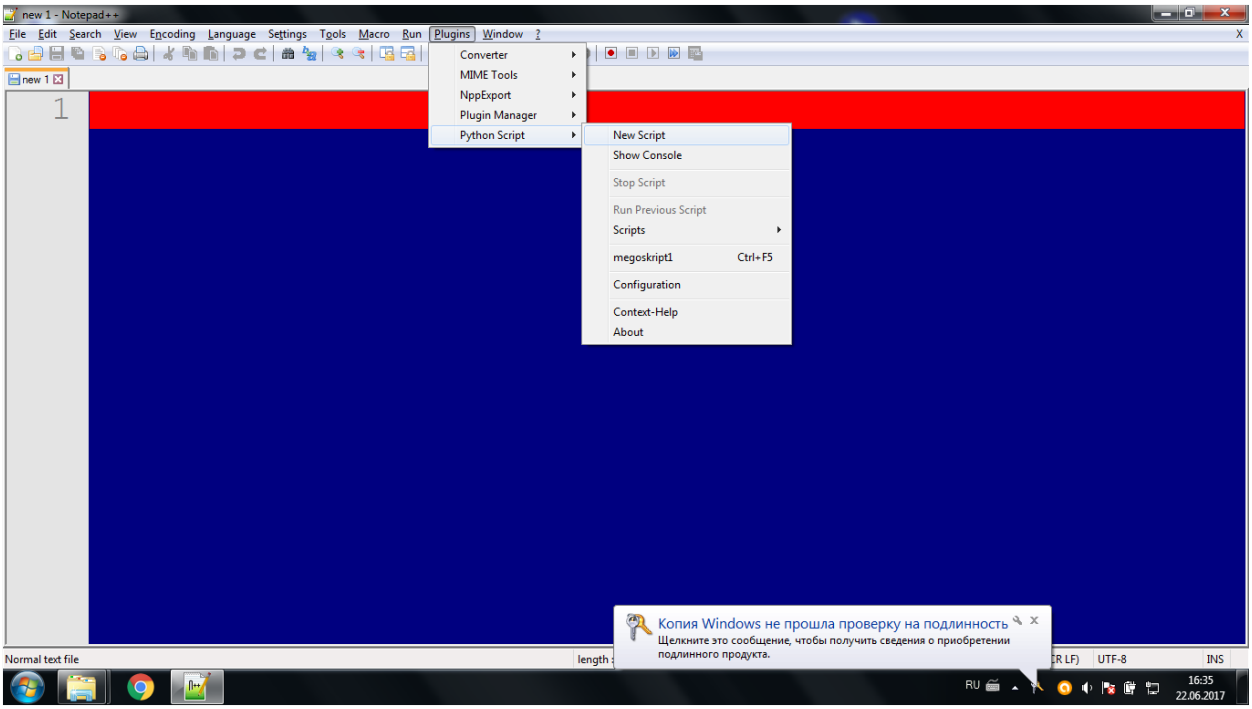

Рис. 2. Создание скрипта для обработки буфера редактора.

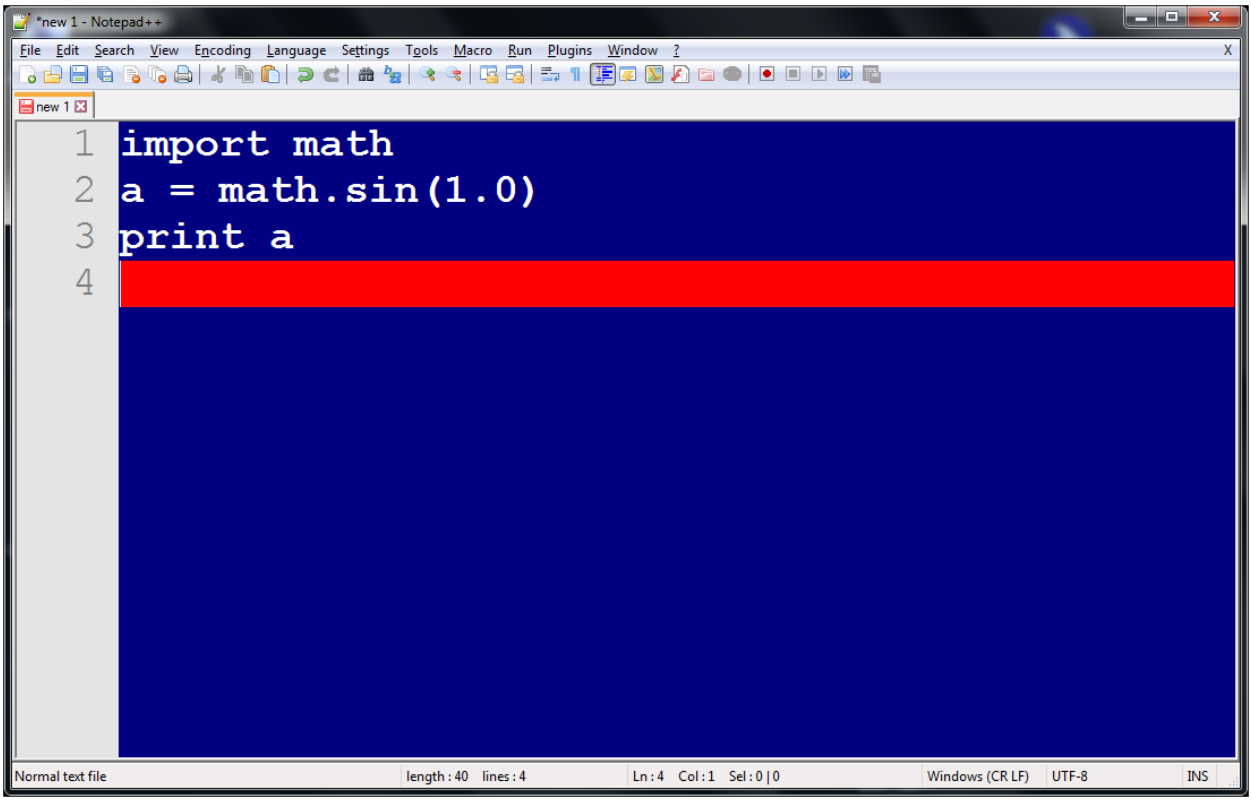

Рис. 3. Среда программирования с кодом для запуска по горячей клавише.

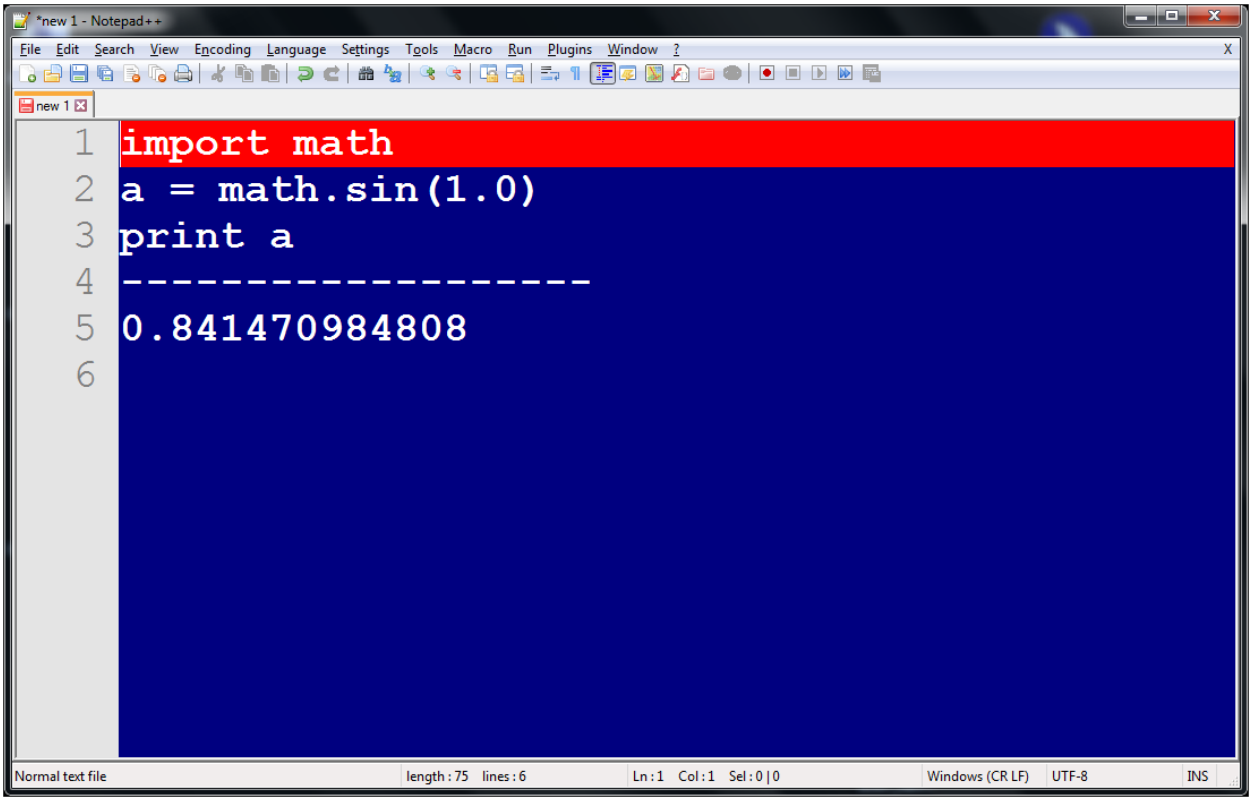

Рис. 4. Результат выполнения кода, изображённого на рис. 3.

Приложение B. Пример создания грамматики и обхода AST-дерева для выполнения транспайлинга

Листинг 1. Запись грамматики языка "Калькулятор"

```
start: add;

// Правила
?add: (add add_symbol)? mul;
?mul: (mul mul_symbol)? atom;
@atom: neg | number | '\(' add '\)';
neg: '-' atom;

// Tokens
number: '[\d.]+';
mul_symbol: '\*' | '/';
add_symbol: '\+' | '-';

WS: '[ \t]+' (%ignore);
```

Листинг 2. Программа на языке "Калькулятор":

```
(1 + 2) * -3
```

Синтаксическое дерево и результат его выполнения:

```
start(mul(add(number(u'1'), add_symbol(u'+'), number(u'2')),
mul_symbol(u'*'), neg(number(u'3'))))


1.0 + 2.0
3.0 * -3.0
-9.0
```

Листинг 3. Программа, организующая разбор программного кода по грамматике и переводящая ее в программу на языке Python

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import plyplus
```

```
#загрузка грамматики из файла в переменную
calc_grammar = open("grammar/calc.ply").read()

#обработка правил грамматики библиотекой плайплас и создание
грамматики
grammar = plyplus.Grammar(calc_grammar)

#обработка программы на нашем языке и сохранение дерева
программы в переменную аст
ast = grammar.parse(open("code_samples/calc_000.ply").read())
print ast


class Calc(plyplus.STransformer):
    def number(self, arg):
        return float(arg.tail[0])

    def neg(self, arg):
        return -(arg.tail[0])


    def add_symbol(self, arg):
        return str(arg.tail[0])

    def add(self, arg):
        a, oper, b = arg.tail
        print a, oper, b
        if oper == '+':
            return a + b
        if oper == '-':
            return a - b
        return 'ERROR_ADD: unknown operation ' + oper

#умножение
    def mul_symbol(self, arg):
        return str(arg.tail[0])

    def mul(self, arg):
        a, oper, b = arg.tail
        print a, oper, b
        if oper == '*':
            return a * b
        if oper == '/'

            return a / b
        return 'ERROR_ADD: unknown operation ' * oper
```

```
    def start(self, arg):
        return arg.tail[0]

calc = Calc()
print calc.transform(ast)
```