МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное благовещенское образовательное учреждение высшего образования «Благовещенский государственный педагогический университет»

Физико-математический факультет

Кафедра информатики и методики преподавания информатики

КУРСОВАЯ РАБОТА

на тему: Разработка логического калькулятора

по дисциплине: Программирование

|  |  |  |  |
| --- | --- | --- | --- |
| Исполнитель:  студент группы 3 «А» | ­­­­­­­­­­­­­­­­  \_\_\_\_\_\_\_\_\_\_\_\_\_  дата | \_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись | Т.А.Дробышевская |
| Руководитель: | ­­­­­­­­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_  дата | \_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись | А.Н. Землянский |

Благовещенск 2017

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc482082578)

[1 Языки программирования 4](#_Toc482082579)

[2 Язык программирования Java 7](#_Toc482082580)

[2.1 Апплеты и приложения 7](#_Toc482082581)

[2.2 Свойства языка Java 8](#_Toc482082582)

[3 Генератор парсеров ANTLR 10](#_Toc482082583)

[4 Алгебра высказываний 12](#_Toc482082584)

[5 Описание частей кода программы 16](#_Toc482082585)

[6 Руководство пользователя 17](#_Toc482082586)

[ЗАКЛЮЧЕНИЕ 18](#_Toc482082587)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 19](#_Toc482082588)

ВВЕДЕНИЕ

Целью данного курсового проекта является создание программы «Логический калькулятор», которая и будет являться объектом исследования. Логический калькулятор – устройство для построения таблицы истинности.

Нами были определены следующие функциональные требования к разрабатываемой программе:

1. соблюдение правильности вычислений;
2. программа должна иметь простой, но в то же время понятный и наглядный интерфейс, который не должен перегружать ресурсы компьютера;
3. программа должна иметь возможность сброса полученного результата;
4. программа не должна занимать большой объем памяти и не должна требовать установки на жесткий диск компьютера;
5. должна существовать возможность построения таблицы истинности для любого высказывания;
6. работоспособность приложения в среде Windows.

1 Языки программирования

С момента проявления математики существовало явление, которое начали чётко осознавать только в начале ХХ века. Это явление называется алгоритм.

Определение 1: Алгоритм – это предписание, однозначно задающее процесс преобразования исходной информации в виде последовательности элементарных дискретных шагов, приводящих за конечное число их применений к результату.

Требования к алгоритмам:

1. Детерминированность – определённость. В каждый момент времени следующий шаг работы однозначно определяется состоянием системы. Таким образом алгоритм выдает один и тот же результат для одних и тех же исходных данных.
2. Понятность. Алгоритм для исполнителя должен включать только те команды, которые ему (исполнителю) доступны, которые входят в его систему команд.
3. Конечность. При корректно заданных исходных данных алгоритм должен завершать работу и выдавать результат за конечное число шагов.
4. Массовость. Начальная система величин выбирается из заведомо большого множества.

С точки зрения математики приведённое выше определение некорректно, так как является не строгим с формальной точки зрения определением алгоритма, а интуитивным, ибо в них используются неточные интуитивные понятия: предписание, элементарные шаги и т.д.

Определение 2: В самом общем случае языком программирования называют фиксированную систему обозначений для описаний алгоритмов и структур данных.

Свойства языка программирования:

1. Небольшой ограниченный набор команд. Язык большинства современных ЭВМ крайне беден и состоит из команд типа «выделить память определенного размера», «запомнить информацию в определенном месте памяти», «считать информацию из определенного места памяти» и т.п. Простота и малочисленность команд позволяет легко реализовывать этот язык с помощью аппаратных средств.
2. Функциональная полнота. Используя команды из набора языка, можно описать любой алгоритм.

Приведем классификацию языков с точки зрения виртуальных машин.

1. Языки низкого уровня:
2. Язык микрокоманд. Микрокоманды обычно задают простые передачи данных между оперативной памятью и быстрыми регистрами, между самими регистрами и обрабатывающими элементами. На основе простейшего набора микрокоманд пишутся специальные микропрограммы, определяющие элементарные операции ЭВМ.
3. Машинный язык. Язык, команды которого построены из последовательностей микрокоманд. Синтаксически данный язык представляет собой последовательность нулей и единиц, реализуется аппаратурой ЭВМ.
4. Язык ассемблера. Язык символического кодирования, его операторы те же команды машинного языка, но имеют мнемонические названия, а в качестве операндов команд имеют не конкретные адреса в оперативной памяти, а их символические имена.
5. Макроязыки. Замена часто встречающихся последовательностей команд на более крупные единицы – макрокоманды. Макропроцессор – программа-переводчик с языка макрокоманд на более низкий язык, то есть замена макрокоманд на последовательности команд более низкого уровня.

Все языки низкого уровня ориентированы на конкретный тип ЭВМ, для которых они созданы.

1. Языки высокого уровня.

Отличие языков высокого уровня от предыдущей группы языков заключается в том, что они ориентированы не на систему команд той или иной ЭВМ, а на систему операторов, характерных для записи определенного класса алгоритмов.

Примерами языков программирования данного уровня являются: Fortran, Algol, Pascal, C++, Java.

1. Языки сверхвысокого уровня.

К языкам сверхвысокого уровня можно отнести лишь АЛГОЛ-68 и APL.

2 Язык программирования Java

Можно выделить три ключевых элемента, которые объединились в технологии языка Java и сделали ее в корне отличной от всего существующего на сегодняшний день.

1. Java предоставляет для широкого использования свои апплеты – небольшие, надежные, динамичные, не зависящие от платформы активные сетевые приложения, встраиваемые в страницы Web. Апплеты Java могут настраиваться и распространяться потребителям с такой же легкостью, как любые документы HTML.
2. Java является объектно-ориентированным языком, сочетая простой и знакомый синтаксис с надежной и удобной в работе средой разработки. Это позволяет широкому кругу программистов быстро создавать новые программы и новые апплеты.
3. Java предоставляет программисту богатый набор классов объектов для ясного абстрагирования многих системных функций, используемых при работе с окнами, сетью и для ввода-вывода. Ключевая черта этих классов заключается в том, что они обеспечивают создание независимых от используемой платформы абстракцией для широкого спектра системных интерфейсов.
   1. Апплеты и приложения

С помощью средств разработки на языке Java фирмы Sun Microsystems можно создавать два типа программ: апплеты и приложения.

Определение 3: Java-апплет – это небольшая программа, динамически загружаемая по сети – точно так же, как картинка, звуковой файл или элемент мультипликации, встроенные в web-страницу.

Апплеты исполняются внутри web-браузеров, которые по умолчанию имеют возможность интерпретировать откомпилированные коды апплетов. В распоряжении апплетов имеются все ресурсы и возможности web-браузера. В то же время апплет по умолчанию не имеет полного доступа к ресурсам локального компьютера, на котором исполняется.

Определение 4: Java-приложение – это полноценная программа, исполняемая на локальном компьютере и имеющая доступ ко всем системным ресурсам.

* 1. Свойства языка Java

Существует ряд требований к современному языку программирования, которые учитываются при выборе средства создания программ: простота и мощь, объектная ориентированность, надежность, безопасность, архитектурная независимость, возможность работы с параллельными потоками команд, возможность интерпретации, высокая производительность и легкость в изучении.

Рассмотрим некоторые из требований к современному языку программирования.

1. Объектная ориентированность. Поскольку при разработке языка Java отсутствовала тяжелая наследственность, как это было со многими известными языками при переходе к объектно-ориентированному подходу, то для реализации объектов был избран удобный прагматичный подход. Практически все в Java является объектами, в то же время, ради повышения производительности, числа и другие простые типы данных Java не являются объектами. Таким образом, достаточно познакомиться с основными понятиями объектно-ориентированного программирования, для того чтобы начать программировать на Java.
2. Архитектурная независимость. Среда Java – это нечто гораздо большее, чем просто язык программирования. В нее встроен набор ключевых объектов, содержащих основные абстракции реального мира, с которым придется иметь дело программам. Основой популярности Java являются встроенные классы-абстракции, сделавшие его языком, действительно независимым от платформы, в отличие от библиотек, подобных MFC/COM, OWL, VCL, NEXTStep, Motif и OpenDoc, привязанных к конкретным платформам.
3. Надежность. Среда Java ограничивает свободу в нескольких ключевых областях и таким образом способствует обнаружению ошибок на ранних стадиях разработки программы. В то же время в ней отсутствуют многие источники ошибок, свойственных другим языкам программирования.
4. Безопасность. Один из ключевых принципов разработки языка Java заключался в обеспечении защиты от несанкционированного доступа. Программы на Java не могут вызывать глобальные функции и получать доступ к произвольным системным ресурсам, что обеспечивает в Java уровень безопасности, недоступный для других языков.
5. Интерпретация и высокая производительность. Способность Java исполнять свой код на любой из поддерживаемых платформ достигается тем, что ее программы транслируются в некое промежуточное представление, называемое байт-кодом. Байт-код, в свою очередь, может интерпретироваться в любой системе, для которой существует реализация виртуальной Java-машины. Одним из недостатков интерпретаторов, обеспечивающих независимость от платформы, является потеря производительности. Для решения этой проблемы в Java существует несколько вариантов оптимизации выполнения программ. Например, использование JIT-компиляторов (Just In Time compilers), когда байт-код переводится непосредственно в машинные коды платформы, на которой исполняется. При этом достигается очень высокая производительность.

3 Генератор парсеров ANTLR

Для разработки нашего приложения необходимо использовать генератор парсеров ANTLR.

Определение 5: ANTLR (ANother Tool for Language Recognition) – набор средств для написания парсеров.

Подобные средства оказываются незаменимыми, если необходимо использовать в работе программы данные какого-нибудь сложного формата:

1. текстовые конфигурационные файлы;
2. файлы скриптов, данные стандартных протоколов, таких как HTTP, SQL и т. д.

Написание модулей, разбирающих эти данные, вручную является очень трудоемкой задачей и не всегда гарантирует хороший результат – результат, как правило, представляет собой многие килобайты кода, который достаточно сложно отлаживать. Использование готовых парсеров для стандартных типов данных (ASN.1, SQL и т. д.) тоже не всегда приемлемо, т.к. при этом навязывается структура хранения разобранных данных.

ANTLR, как и многие другие подобные средства, состоит из библиотеки классов, облегчающих основные операции при разборе (буферизация, поиск и т. д.), и утилиты, генерирующей код парсера на основе файла, описывающего грамматику разбираемого языка. Сам ANTLR написан на Java, но позволяет генерировать код на Java и C++. В качестве источника данных, используется класс потока, что позволяет разбирать данные, находящиеся в файле, в памяти процесса или приходящие по сети. Результатом работы ANTLR являются два класса – лексер и парсер. Лексер разбивает поток символов на поток токенов в соответствии с правилами, а парсер обрабатывает поток токенов в соответствии с другими правилами. Правила пишутся в грамматике на специальном языке, основанном на регулярных выражениях.

Краткая справка элементов языка грамматики приведена ниже в таблице 1.

Таблица 1 – Краткая справка элементов языка грамматики

|  |  |
| --- | --- |
| Символ | Описание |
| (...) | Подправило |
| (...)\* | Повторение подправила 0 или более раз |
| (...)+ | Повторение подправила 1 или более раз |
| (...)? | Подправило, которое может отсутствовать |
| {...} | Семантические действия (на языке, использующемся в качестве выходного, напр. Java) |
| [...] | Параметры правила |
| {...}? | Семантический предикат |
| (...)=> | Синтаксический предикат |
| | | Оператор альтернативы |
| .. | Оператор диапазона |
| ~ | Отрицание |
| . | Любой символ |
| = | Присвоение |
| : | Метка, начало правила |
| ; | Конец правила |
| class | Класс грамматики |
| extends | Определяет базовый класс для грамматики |
| returns | Описание возвращаемого значения для правила |
| options | Секция опций |
| tokens | Секция токенов |
| header | Заголовок |

4 Алгебра высказываний

Определение 6: Высказывание – это предложение, относительно содержания которого имеет смысл говорить истинно оно или ложно.

Все высказывания можно разделить на два класса:

1. простые (или атомарные) высказывания;
2. составные высказывания, которые являются комбинациями атомарных высказываний.

Определение 7: Атомарное высказывание – это высказывание, которое мы рассматриваем как не представимое в виде комбинации более простых высказываний.

При формальном анализе высказываний каждое атомарное высказывание будет обозначаться некоторым символом, который называется булевой переменной. Как правило, это – строчная буква латинского алфавита (возможно с индексом внизу). Булевы переменные, обозначающие атомарные высказывания, должны быть выбраны так, чтобы разным атомарным высказываниям соответствовали разные булевы переменные.

Из высказываний путем соединения их различными способами можно составлять новые, более сложные высказывания.

Определение 7: Образование новых высказываний из исходных с помощью логических связок «не», «и», «или», «если, то», «тогда и только тогда» называется логическими операциями.

Если высказывание является истинным, то мы будем говорить, что оно имеет значение 1, а если ложным, то будем говорить, что оно имеет значение 0.

Определение 8: Отрицанием высказывания называется новое высказывание, которое истинно тогда и только тогда, когда исходное высказывание ложно, и ложно тогда и только тогда, когда исходное высказывание истинно.

Определение 9: Конъюнкцией двух высказываний называется высказывание, которое истинно тогда и только тогда, когда оба исходных высказывания истинны.

Определение 10: Дизъюнкцией двух высказываний называется высказывание, которое ложно тогда и только когда, когда исходные высказывания ложны.

Определение 11: Импликацией двух высказываний называется такое высказывание, которое ложно тогда и только тогда, когда первое высказывание истинно, а второе ложно.

Определение 12: Эквиваленцией двух высказываний называется такое высказывание которое истинно тогда и только тогда, когда исходные высказывания принимают одновременно одинаковые значения (оба одновременно истинны или ложны).

Составим таблицу выражающую истинные значения вышеперечисленных сложных высказываний.

Таблица 2 – Таблица истинности для вышеперечисленных высказываний

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |

1. Определение 13: Логические переменные и символы 0, 1 будем считать формулами алгебры высказываний.
2. Если и формулы алгебры высказываний, то , , , , – формулы алгебры высказываний.
3. Других формул кроме перечисленных по правилам двух предыдущих пунктов нет.

Иначе говоря формула алгебры высказываний – это всякое сложное высказывание, составленное из некоторых исходных высказываний посредством применения логических операций.

Порядок выполнения операций:

1. отрицание, относящееся к переменной;
2. конъюнкция;
3. дизъюнкция;
4. импликация;
5. эквиваленция.

Замечание: Если без скобок записаны одна за другой несколько одинаковых операций, то они выполняются последовательно слева на право.

Для всякой формулы алгебры высказываний можно составить таблицу истинности.

Аксиомы:

1. , закон двойного отрицания;
2. ;
3. ;
4. ;
5. ;
6. ;
7. ;
8. ;
9. .

Стрелка Пирса и штрих Шеффера – бинарные логические операции, булевы функции над двумя переменными. Таблица истинности для этих операций приведены в таблице 3.

Таблица 3 – Таблица истинности для бинарных логических операций

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 1 | 2 | 3 | 4 |

Продолжение таблицы 3:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |

5 Описание частей кода программы

6 Руководство пользователя

ЗАКЛЮЧЕНИЕ

В результате выполнения данного курсового проекта был разработан игровой программный продукт «Логический калькулятор». При помощи этой программы можно построить таблицу истинности для любого логического высказывания. Программа не занимает много места, не требовательна к установленному программному обеспечению.

Также было проведено исследование полученного программного продукта. В результате были выявлены следующие достоинства и недостатки полученного программного продукта:

Достоинства:

* существует возможность сбрасывать результат;
* программный продукт малотребователен к системным ресурсам компьютера.

Недостатки:

* невозможность выполнения некоторых других математических операций;
* невозможность сохранения операндов или результатов в памяти программы.

В целом, поставленная в начале курсового проекта цель была достигнута. В программе выполняются все необходимые функциональные требования.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Семочкин, А.Н. Язык программирования Java: учебное пособие для студентов вузов / А.Н. Семочкин. - 2-е изд. Переработанное и дополненное – Благовещенск: Издательство БГПУ , 2009. - 89 с.
2. Написание парсеров с помощью ANTLR // [Электронный ресурс] / Статья. [Москва, 2003]. URL : <http://club.shelek.ru/viewart.php?id=39> (дата обращения 05.05.17).