Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Информационные технологии»

О. А. Кравченко

МЕТОДЫ ТРАНСЛЯЦИИ

ПРАКТИКУМ
по выполнению лабораторных работ для студентов специальности 1-40 04 01 «Информатика и технологии программирования» дневной и заочной форм обучения

Гомель 2021

Библиотека ГГТУ им.П.О.Сухого

УДК 004.4’42(075.8)
ББК 32.972.11я73
К78

Рекомендовано научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 10 от 04.05.2020 г.*)

Рецензент: проф. каф. «Промышленная электроника» ГГТУ им. П. О. Сухого
д-р техн. наук, проф. *В. П. Кудин*

Кравченко, О. А.

К78 Методы трансляции : практикум по выполнению лаборатор. работ для студентов специальности 1-40 04 01 «Информатика и технологии программирования» днев. и заоч. форм обучения / О. А. Кравченко. – Гомель : ГГТУ им. П. О. Сухого, 2021. – 79 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

Содержит задания для выполнения студентами лабораторных работ по учебной дисциплине «Методы трансляции», способствующие усвоению пройденного теоретического материала. Темы лабораторных работ ориентированы на получение студентами навыков алгоритмизации и программной реализации наиболее важных аспектов процесса создания компиляторов и интерпретаторов.

Для студентов специальности 1-40 04 01 «Информатика и технологии программирования» дневной и заочной форм обучения.

УДК 004.4’42(075.8)
ББК 32.972.11я73

© Учреждение образования «Гомельский государственный технический университет имени П. О. Сухого», 2021

Библиотека ГГТУ им. П.О.Сухого

СОДЕРЖАНИЕ

Библиотека ГГТУ им. П.О. Сухого

Библиотека ГГТУ им. П.О. Сухого

ВВЕДЕНИЕ

Трансляция – это процесс преобразования программы на исходном языке программирования в эквивалентную по результатам выполнения программу на объектном языке программирования.

В качестве объектного языка программирования чаще всего используется язык, который понятен установленной на компьютере операционной системе. Например, система команд компьютера.

Транслятор – это программа, которая выполняет трансляцию.

Существуют два основных вида трансляции – компиляция и интерпретация.

Компиляция – это такой процесс трансляция, в результате которого получается программа на объектном языке программирования в виде exe-файла. Таким образом, объектная программа будет выполняться после отдельного запуска.

Компилятор – транслятор, работающий в режиме компиляции. Примеры компиляторов: компилирующие системы для языков C#, C++, PYTON и т. п.

Интерпретация – это процесс трансляции, при котором последовательно переводятся отдельные конструкции исходной программы и после перевода каждая конструкция сразу выполняется. Так работают интерпретаторы формул в текстовом процессоре Word и в табличном процессоре Excel.

Рассмотрим основные этапы процесса компиляции. Их два – синтаксический и семантический анализ.

На этапе синтаксического анализа проверяется синтаксис исходной программы, т. е. соответствие написания конструкций правилам синтаксиса исходного языка. Примеры синтаксических ошибок: использование недопустимых символов в идентификаторах и числах, ошибки в написании служебных слов, дисбаланс скобок и кавычек и пр.

Если на этапе синтаксического анализа компилятор не выявил ошибок, то выполняется семантический анализ, в ходе которого для каждой смысловой конструкции исходной программы (например, для оператора) генерируется совокупность эквивалентных по смыслу конструкций объектного языка. Таким, образом, этот процесс полностью аналогичен работе переводчика, который переводит исходный английский текст, на русский язык отдельными предложениями.

Библиотека ГГТУ им. П.О. Сухого

При создании практических компиляторов, как правило, выделяют не два, а, по крайней мере, четыре этапа компиляции – лексический анализ, синтаксический анализ, семантический анализ и сборка (компоновка) программы.

Выделение этапа лексического анализа – это чисто практический прием, позволяющий упростить синтаксический и семантический анализ и, тем самым, уменьшить общее время компиляции программы.

Основные задачи, решаемые на этапе лексического анализа:

- исключение из текста исходной программы комментариев и незначащих пробелов;
- выделение строковых и числовых констант;
- выделение служебных слов;
- проверка наличия в тексте символов, не принадлежащих исходному языку;
- составление таблиц идентификаторов.

Основные результаты работы лексического анализатора состоят в следующем:

- в исходном тексте программы идентификаторы заменяются ссылками на таблицы идентификаторов;
- аналогично, как правило, поступают с числовыми и строковыми константами;
- осуществляется проверка несовпадения идентификаторов пользователя со служебными словами.

Все перечисленное облегчает последующую работу синтаксического анализатора, который оперирует не отдельными лексемами (символами), а специальными ссылками на таблицы.

Лексический анализатор может взять на себя часть функций семантического анализа, например, следующих:

- обнаружение неописанных и дважды описанных идентификаторов;
- разделение идентификаторов на различные виды – простые переменные, имена структур, имена функций и процедур (внутренних и внешних).

После лексического анализа синтаксический анализатор будет тратить основное время не на работу с исходным текстом, а на работу с таблицами. Таким образом, при построении компиляторов большое внимание уделяется технологии организации таблиц и алгоритмам быстрого поиска в этих таблицах.

Библиотека ГГТУ им. П.О. Сухого

Этап синтаксического анализа наиболее хорошо формализуется при условии описания исходного языка программирования с помощью формальных грамматик.

Возникновение теории формальных грамматик относится ко второй половине 50-х годов прошлого века, ее основоположником по праву считается американский лингвист Н. Хомский. В своих работах он ввел специальные исчисления, названные им грамматиками, для порождения множеств правильных предложений естественных языков. В дальнейшем при развитии системного программирования математики-программисты с успехом использовали исчисления Хомского для описаний языков программирования и конструирования алгоритмов их синтаксического анализа.

Как говорилось ранее, задача семантического анализа состоит в замене конструкций исходной программы эквивалентными по результатам выполнения конструкциями объектного языка. Так как до сего времени не предложена конструктивная классификация семантических единиц языка, то создание семантического анализатора представляется, как правило, искусством разработчиков.

На последнем этапе рассматриваемого процесса компиляции выполняется подключение к сформированной семантическим анализатором объектной программе необходимых внешних библиотек, модулей и объектов, на которые имелись ссылки в исходной программе.

На основании вышеизложенных теоретических положений были выбраны следующие темы лабораторных работ:

1. Строки языка. В данной работе студенты закрепляют навыки по применению операций обработки строк – выделение подстроки, поиск подстроки, конкатенация и замыкание строк. Данные операции лежат в основе работы лексического анализатора.
2. Организация таблиц идентификаторов. Цель работы: изучить основные методы организации таблиц идентификаторов, получить представление о преимуществах и недостатках, присущих различным методам организации таблиц идентификаторов и определяющих временные характеристики работы лексического анализатора.
3. Синтаксический анализ по КС-грамматике. При выполнении данной работы студенты получают навыки проведения безвозвратного синтаксического анализа на основе КС-

Библиотека ГГТУ им. П.О. Сухого

грамматики для определения принадлежности заданной строки языку, порожденному грамматикой.

4. Разработка КС-грамматик. Цель работы: получить навыки создания КС-грамматик для заданного языка программирования. Для созданной грамматики необходимо разработать алгоритм и программу формирования множеств левых и правых символов для нетерминальных символов грамматики.
5. Построение матрицы предшествования. При выполнении данной работы студенты разрабатывают алгоритм и программу построения матрицы предшествования для заданной КС-грамматики.
6. Преобразование КС-грамматики в простую грамматику предшествования. Цель работы: проведение исследования заданной КС-грамматики на предмет того, является ли она простой грамматикой предшествования. При отрицательном ответе студент должен провести преобразование КС-грамматики в простую грамматику предшествования.
7. Построение детерминированного конечного автомата. В данной работе студенты получают навыки проведения синтаксического анализа конструкции языка программирования с помощью детерминированного конечного автомата.

Библиотека ГГТУ им. П.О.Сухого

1 ЛАБОРАТОРНАЯ РАБОТА № 1. СТРОКИ ЯЗЫКА

1.1 Цель и время выполнения работы

Цель работы: усвоить операции обработки строк языка.
Время выполнения работы – 2 часа.

1.2 Краткие теоретические сведения

Термин *алфавит* означает любое конечное множество символов. Множество {0, 1} представляет собой *бинарный алфавит*. Другой пример алфавита – код ASCII, использующийся во многих программных системах.

Строка над некоторым алфавитом – это конечная последовательность символов, взятых из этого алфавита. В теории языков термины *предложение* и *слово* часто используются как синонимы термина *строка*. Длина строки s, обычно обозначаемая как $|s|$, равна количеству символов в строке. Пустая строка, обозначаемая как ε, представляет собой строку нулевой длины.

Язык представляет собой любое счетное множество строк над некоторым фиксированным алфавитом.

Если x и y – строки, то *конкатенация* этих строк, записываемая как xy, является строкой, образованной путем добавления y к x справа.

Если рассматривать конкатенацию как умножение, то можно определить операцию «возведение в степень» следующим образом: полагаем s^0 как ε, а для всех $i>0$ определяем $s^i=s^{i-1}s$. Таким образом, $s^1=s$, $s^2=ss$ и т. д.

В таблице 1.1 приведены основные операции над языками.

Таблица 1.1 – Определение операций над языками

Операция	Определение и обозначение
Объединение L и M	$L \cup M = \{s \mid s \in L \text{ или } s \in M\}$
Конкатенация L и M	$LM = \{st \mid s \in L \text{ или } t \in M\}$
Замыкание Клини языка L	$L^* = \bigcup_{i=0}^{\infty} L^i$
Позитивное замыкание языка L	$L^+ = \bigcup_{i=1}^{\infty} L^i$

Библиотека ГГТУ им. П.О. Сухого

Обычно используются следующие термины, связанные со строками:

- *префиксом* строки *s* является любая строка, полученная удалением нуля или нескольких последних символов строки *s*;
- *суффиксом* строки *s* является любая строка, полученная удалением нуля или нескольких первых символов строки *s*;
- *подстрока* строки *s* получается удалением произвольного префикса и произвольного суффикса из строки *s*;
- *правильными* (*истинными*) префиксами, суффиксами и подстроками строки *s* являются соответственно префиксы, суффиксы и подстроки *s*, которые не являются пустыми и не совпадают с самой строкой *s*;
- *подпоследовательностью* строки *s* является любая строка, полученная удалением нуля или нескольких не обязательно смежных позиций строки *s*.

1.3 Список рекомендуемой литературы

1. Ахо А. Теория синтаксического анализа, перевода и компиляции. Том 1. Синтаксический анализ / А. Ахо, Дж. Ульман. – М. : Мир, 1978. – 612 с.
2. Ахо, А. Теория синтаксического анализа, перевода и компиляции. Том 2. Компиляция / А. Ахо, Дж. Ульман. – М. : Мир, 1978. – 488 с.
3. Мартыненко. Б. К. Языки и трансляции : учеб. пособие / Б. К. Мартыненко. – Изд. 2-е, испр. и доп. – СПб. : Изд-во С.-Петерб. ун-та, 2013. – 265 с.
4. Молчанов, А. Ю. Системное программное обеспечение. Лабораторный практикум. – СПб. : Питер, 2005. – 284 с.
5. Опалева, Э. А. Языки программирования и методы трансляции / Э. А. Опалева, В. П. Самойленко. – СПб. : БХВ-Петербург, 2005. – 480 с.
6. Орлов, С. А. Теория и практика языков программирования. (Стандарт 3-го поколения) / С. А. Орлов. – СПб. : Питер, 2014. – 88 с.

Библиотека ГГТУ им. П.О. Сухого

1.4 Задания и указания для выполнения лабораторной работы

Указания:

- программа должна быть написана на выбранном студентом языке программирования;
- в программе должен быть реализован графический интерфейс;
- исходные данные должны читаться из текстового файла и с консоли. Результаты должны выводиться на экран и в текстовый файл;
- исходные данные для выполнения работы (язык и строки) студент формирует самостоятельно в соответствии с условиями, указанными в заданиях.

1. Разработать алгоритм, написать и отладить программу, определяющую принадлежность строки *S* языку *L*.

Программа (с графическим интерфейсом) должна обеспечивать чтение из текстового файла и с консоли символов заданного языка (алфавит языка должен содержать не менее семи символов) и при вводе с консоли некоторой строки выдавать ответ, принадлежит ли эта строка языку или нет. Например, для языка $L=\{c, i, n, s, r, t, u, o\}$ при вводе строки *construction* ответ – «Принадлежит», а для строки *structure* – «Не принадлежит».

2. Разработать алгоритм, написать и отладить программу получения всех правильных префиксов, суффиксов и подстрок строки *S* языка *L*.

Программа должна обеспечивать чтение из текстового файла и с консоли символов заданного языка (не менее семи символов) и одной строки (*S*) этого языка. Графический интерфейс программы должен предоставлять пользователю возможность выбрать один из трех вариантов: префиксы, суффиксы и подстроки. Для каждого из указанных вариантов программа должна выводить на консоль и в текстовый файл все правильные ответы в виде строк (все префиксы, все суффиксы или все подстроки строки *S*).

3. Разработать алгоритм, написать и отладить программу определения наличия и вида подпоследовательности *S*1 в строке *S* языка *L*.

Программа (с графическим интерфейсом) должна обеспечивать чтение из текстового файла и с консоли символов заданного языка

Библиотека ГГТУ им. П.О. Сухого

(не менее семи символов), и двух строк этого языка (S и S1). Возможные результаты выполнения программы:

- вторая подстрока (*S*1) не является подпоследовательностью первой строки (*S*);
- вторая подстрока (*S*1) является подпоследовательностью первой строки (*S*);
- вторая подстрока (*S*1) является префиксом первой строки (*S*);
- вторая подстрока (*S*1) является суффиксом первой строки (*S*);
- вторая подстрока (*S*1) является подстрокой первой строки (*S*).

1.5 Структура отчета по лабораторной работе

Отчет по лабораторной работе должен содержать следующие элементы:

- цель работы;
- задание (для каждого из трех заданий);
- тесты (для каждого из трех заданий);
- графические схемы алгоритмов (для каждого из трех заданий);
- результаты выполнения программы (для каждого из трех заданий);
- листинг программы (для каждого из трех заданий).

Библиотека ГГТУ им. П.О. Сухого

2 ЛАБОРАТОРНАЯ РАБОТА № 2. ОРГАНИЗАЦИЯ ТАБЛИЦ ИДЕНТИФИКАТОРОВ

2.1 Цель и время выполнения работы

Цель работы: изучить основные методы организации таблиц идентификаторов, получить представление о преимуществах и недостатках, присущих различным методам организации таблиц идентификаторов.

Время выполнения работы – 6 часов.

2.2 Краткие теоретические сведения

Теоретические сведения к работе изложены на страницах 13-27 в лабораторном практикуме:

Молчанов А. Ю. Системное программное обеспечение. Лабораторный практикум. – СПб. Питер, 2005. – 284 с.: ил.

2.3 Содержание работы и указания по ее выполнению

Разработать алгоритм, написать и отладить программу с графическим интерфейсом поиска заданного значения в таблице идентификаторов двумя методами и сравнения быстродействия этих методов в соответствии с индивидуальным заданием.

Программа должна быть написана на выбранном студентом языке программирования. В программе должен быть реализован графический интерфейс.

Исходные данные должны читаться из текстового файла и с консоли. Результаты должны выводиться на экран и в текстовый файл.

Функции программы:

- первоначальное заполнение таблицы идентификаторов (не менее 50 значений, длина идентификаторов – не более 32 символов) посредством ввода информации из текстового файла в заданную структуру данных;
- ввод нового идентификатора;
- вывод информации о месте нахождения введенного идентификатора в существующей таблице или включение в таблицу нового идентификатора, если его нет в таблице;

Библиотека ГГТУ им. П.О. Сухого

– вывод информации о времени (скорости) поиска идентификатора в таблице для каждого из рассматриваемых методов.

Значения элементов таблицы идентификаторов и идентификаторы для поиска в таблице подобрать самостоятельно.

2.4 Варианты индивидуальных заданий

Вариант 1. Первый метод: таблица идентификаторов – неупорядоченный массив.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – строка текста произвольной длины. Преобразование строки – конкатенация битовых образов символов. Метод хеширования – модульный. Метод разрешения коллизий – линейный.

Вариант 2. Первый метод: таблица идентификаторов – упорядоченный массив, метод просмотра – двоичный поиск.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – вещественное число из отрезка [-1 000.00, +10 000.00]. Метод хеширования – мультипликативный. Метод разрешения коллизий – квадратичный.

Вариант 3. Первый метод: таблица идентификаторов – линейный односвязный нециклический список.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – целое число из отрезка [0, +1000 000 000]. Метод хеширования – выбор цифр. Метод разрешения коллизий – двойное хеширование.

Вариант 4. Первый метод: таблица идентификаторов – линейный односвязный циклический список.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – строка текста произвольной длины. Метод хеширования – модульный. Метод разрешения коллизий – отдельное связывание.

Вариант 5. Первый метод: таблица идентификаторов – линейный двусвязный нециклический список.

Библиотека ГГТУ им. П.О. Сухого

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – вещественное число из отрезка [100 000.00, +150 000.00]. Метод хеширования – модульный. Метод разрешения коллизий – двойное хеширование.

Вариант 6. Первый метод: таблица идентификаторов – линейный двусвязный циклический список.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – строка текста произвольной длины. Преобразование строки – конкатенация битовых образов символов. Метод хеширования – мультипликативный. Метод разрешения коллизий – квадратичный.

Вариант 7. Первый метод: таблица идентификаторов – бинарное дерево, обход Left-Root-Right.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – вещественное число из отрезка [-5 000.000, +5 000.000]. Метод хеширования – свёртка. Метод разрешения коллизий – квадратичный.

Вариант 8. Первый метод: таблица идентификаторов – бинарное дерево, обход Root-Left-Right.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – 12-значное натуральное число. Метод хеширования – свёртка, комбинированная с выбором цифр. Метод разрешения коллизий – квадратичный.

Вариант 9. Первый метод: таблица идентификаторов – бинарное дерево, обход Left-Right-Root.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – строка текста произвольной длины. Преобразование строки – конкатенация битовых образов символов. Метод хеширования – модульный. Метод разрешения коллизий – квадратичный.

Вариант 10. Первый метод: таблица идентификаторов – неупорядоченный массив.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – вещественное число из отрезка

Библиотека ГГТУ им. П.О. Сухого

[-10 000.00, +10 000.00]. Метод хеширования – мультипликативный. Метод разрешения коллизий – линейный.

Вариант 11. Первый метод: таблица идентификаторов – упорядоченный массив, метод просмотра – двоичный поиск.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – вещественное число из отрезка [-100 000.00, +100 000.00]. Метод хеширования – мультипликативный. Метод разрешения коллизий – двойное хеширование.

Вариант 12. Первый метод: таблица идентификаторов – линейный односвязный нециклический список.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – вещественное число из отрезка [-1 000.00, +1 000.00]. Метод хеширования – мультипликативный. Метод разрешения коллизий – отдельное связывание.

Вариант 13. Первый метод: таблица идентификаторов – линейный односвязный циклический список.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – целое число из отрезка [0, +1 000 000 000]. Метод хеширования – свертка с выбором цифр. Метод разрешения коллизий – двойное хеширование.

Вариант 14. Первый метод: таблица идентификаторов – линейный двусвязный нециклический список.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – целое число из отрезка [0, +1 000 000]. Метод хеширования – свертка с выбором цифр. Метод разрешения коллизий – отдельное связывание.

Вариант 15. Первый метод: таблица идентификаторов – линейный двусвязный циклический список.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – целое число из отрезка [0, +1 000 000 000]. Метод хеширования – свертка с выбором цифр. Метод разрешения коллизий – квадратичный.

Библиотека ГГТУ им. П.О. Сухого

Вариант 16. Первый метод: таблица идентификаторов – бинарное дерево, обход Left-Root-Right.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – вещественное число из отрезка [100 000.00, +150 000.00]. Метод хеширования – выбор цифр. Метод разрешения коллизий – линейный.

Вариант 17. Первый метод: таблица идентификаторов – бинарное дерево, обход Root-Left-Right.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – строка текста произвольной длины. Преобразование строки – конкатенация битовых образов символов. Метод хеширования – модульный. Метод разрешения коллизий – отдельное связывание.

Вариант 18. Первый метод: таблица идентификаторов – бинарное дерево, обход Left-Right-Root.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – строка текста произвольной длины. Преобразование строки – конкатенация битовых образов символов. Метод хеширования – свертка с выбором цифр. Метод разрешения коллизий – отдельное связывание.

Вариант 19. Первый метод: таблица идентификаторов – неупорядоченный массив.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – строка текста произвольной длины. Преобразование строки – конкатенация битовых образов символов. Метод хеширования – свертка. Метод разрешения коллизий – двойное хеширование.

Вариант 20. Первый метод: таблица идентификаторов – упорядоченный массив, метод просмотра – двоичный поиск.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – вещественное число из отрезка [100 000.00, +150 000.00]. Метод хеширования – свертка. Метод разрешения коллизий – двойное хеширование.

Библиотека ГГТУ им. П.О.Сухого

Вариант 21. Первый метод: таблица идентификаторов – линейный односвязный нециклический список.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – строка текста произвольной длины. Преобразование строки – конкатенация битовых образов символов. Метод хеширования – мультипликативный. Метод разрешения коллизий – линейный.

Вариант 22. Первый метод: таблица идентификаторов – линейный односвязный циклический список.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – целое число из отрезка [-5 000, +5 000]. Метод хеширования – свёртка с выбором цифр. Метод разрешения коллизий – квадратичный.

Вариант 23. Первый метод: таблица идентификаторов – линейный двусвязный нециклический список.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – 12-значное натуральное число. Метод хеширования – свёртка, комбинированная с выбором цифр. Метод разрешения коллизий – линейный.

Вариант 24. Первый метод: таблица идентификаторов – линейный двусвязный циклический список.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – строка текста произвольной длины. Преобразование строки – конкатенация битовых образов символов. Метод хеширования – модульный. Метод разрешения коллизий – отдельное связывание.

Вариант 25. Первый метод: таблица идентификаторов – бинарное дерево, обход Left-Root-Right.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – строка текста из 12 символов. Преобразование строки – конкатенация битовых образов символов. Метод хеширования – свертка с выбором цифр. Метод разрешения коллизий – отдельное связывание.

Библиотека ГГТУ им. П.О. Сухого

Вариант 26. Первый метод: таблица идентификаторов – бинарное дерево, обход Root-Left-Right.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – целое 12-разрядное число. Метод хеширования – свертка. Метод разрешения коллизий – квадратичный.

Вариант 27. Первый метод: таблица идентификаторов – бинарное дерево, обход Left-Right-Root.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – вещественное число из отрезка [-100 000.00, +150 000.00]. Метод хеширования – свертка с выбором цифр. Метод разрешения коллизий – двойное хеширование.

Вариант 28. Первый метод: таблица идентификаторов – неупорядоченный массив.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – строка текста произвольной длины. Преобразование строки – конкатенация битовых образов символов. Метод хеширования – мультипликативный. Метод разрешения коллизий – квадратичный.

Вариант 29. Первый метод: таблица идентификаторов – упорядоченный массив, метод просмотра – двоичный поиск.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – целое число из отрезка [-15 000, +15 000]. Метод хеширования – свёртка с выбором цифр. Метод разрешения коллизий – двойное хеширование.

Вариант 30. Первый метод: таблица идентификаторов – линейный односвязный нециклический список.

Второй метод: таблица идентификаторов – массив, полученный методом хеширования. Тип ключа – 12-значное натуральное число. Метод хеширования – модульный. Метод разрешения коллизий – линейный.

Библиотека ГГТУ им. П.О. Сухого

2.5 Структура отчета по лабораторной работе

Отчет по лабораторной работе должен содержать следующие элементы:

- цель работы;
- задание (для каждого из двух методов);
- тесты (для каждого из двух методов);
- графические схемы алгоритмов (для каждого из двух методов);
- результаты выполнения программы (для каждого из двух методов);
- листинг программы (для каждого из двух методов);
- результаты сравнения используемых методов.

Библиотека ГГТУ им.П.О.Сухого

3 ЛАБОРАТОРНАЯ РАБОТА № 3. СИНТАКСИЧЕСКИЙ АНАЛИЗ ПО КС-ГРАММАТИКЕ

3.1 Цель и время выполнения работы

Цель работы: получить навыки проведения безвозвратного синтаксического анализа на основе КС-грамматики для определения принадлежности заданной строки языку, порожденному грамматикой.

Время выполнения работы – 2 часа.

3.2 Краткие теоретические сведения

Грамматика $G=(N, \sum, P, S)$, где N – нетерминальный словарь, $\sum$ – терминальный словарь, S – начальный нетерминальный символ, P – множество правил вывода вида $B \rightarrow \varphi$, при этом $B \in N$ и $\varphi \in (N \cup \sum)^*$, называется контекстно-свободной грамматикой или просто КС-грамматикой.

Таким образом, из общего класса формальных грамматик КС-грамматика отличается ограничением на вид правил: в левой части каждого правила может находиться только один нетерминальный символ.

Правая часть каждого правила, как и в общем классе формальных грамматик, представляет собой произвольную цепочку нетерминальных и терминальных символов.

КС-грамматика порождает КС-язык, т. е. множество цепочек терминальных символов, которые выводятся из начального нетерминального символа последовательным применением правил грамматики:

$L(G)=\{\alpha \in \sum^* | S \overset{*}{\Rightarrow} \alpha\}$

Без каких-либо дополнительных ограничений на рассматриваемый класс КС-грамматик мы будем в дальнейшем полагать, что начальный нетерминальный символ S не находится в правой части ни одного из правил множества P.

Дерево разбора используется для проверки адекватности построенной грамматике неформальному описанию языка программирования. Когда грамматика создана, синтаксический анализатор на ее основе проверяет синтаксис исходного текста программы. При этом строится *дерево разбора*.

Библиотека ГГТУ им. П.О. Сухого

Построение дерева разбора заканчивается, когда будет получен начальный нетерминальный символ *S*.

Если при этом вся анализируемая цепочка «свернулась» в *S*, то эта цепочка принадлежит языку, порожденному грамматикой.

Если же начальный нетерминальный символ *S* получен не из всей цепочки (остались непроанализированные символы), то эта ситуация свидетельствует либо о существовании альтернативного варианта разбора, либо о наличии синтаксической ошибки (анализируемая цепочка не принадлежит языку).

При анализе исходной строки возможна ситуация, когда не существует правило для продолжения, но анализ строки еще не закончился. Данная ситуация не свидетельствует об ошибке в исходной строке, а является «тупиком» анализа, т. к. еще не рассмотрены все возможные варианты разбора.

Ясно, что существует очень много вариантов разбора, приводящих к тупиковой ситуации. Поэтому возникает задача построения «умного» анализатора, который будет выбирать алгоритм разбора, не приводящий к тупиковым ситуациям. Такой алгоритм называется *разбор без возвратов*.

3.3 Список рекомендуемой литературы

1. Ахо, А. Теория синтаксического анализа, перевода и компиляции. Том 1. Синтаксический анализ / А. Ахо, Дж. Ульман. – М. : Мир, 1978. – 612 с.
2. Ахо, А. Теория синтаксического анализа, перевода и компиляции. Том 2. Компиляция / А. Ахо, Дж. Ульман. – М. : Мир, 1978. – 488 с.
3. Мартыненко, Б. К. Языки и трансляции : учеб. пособие / Б. К. Мартыненко. – Изд. 2-е, испр. и доп. – СПб. : Изд-во С.-Петерб. ун-та, 2013. – 265 с.
5. Опалева, Э. А. Языки программирования и методы трансляции / Э. А. Опалева, В. П. Самойленко – СПб. : БХВ-Петербург, 2005. – 480 с.
6. Орлов, С. А. Теория и практика языков программирования. (Стандарт 3-го поколения) / С. А. Орлов. – СПб. : Питер, 2014. – 688 с.

3.4 Содержание работы и указания по ее выполнению

Дана КС-грамматика $G=(N, \sum, P, S)$,
где $N=\{A, B, D, F, H, I, L, M, R, S, T, U, V, Z\}$ – нетерминальный словарь,

Библиотека ГГТУ им. П.О.Сухого

∑={, (запятая), +, -, *, /, ^, . (точка), (,), λ (здесь под λ понимается любая строчная латинская буква: *a*, *b*, …, *z*), δ (δ – любая цифра: 0, 1, …, 9)} – терминальный словарь,

S – начальный нетерминальный символ,

P – множество правил вывода:

1. $S \rightarrow B$
2. $B \rightarrow A$
3. $B \rightarrow B, A$
4. $A \rightarrow T$
5. $A \rightarrow ZT$
6. $A \rightarrow AZT$
7. $T \rightarrow U$
8. $T \rightarrow TMU$
9. $U \rightarrow H$
10. $U \rightarrow U\text{^}H$
11. $H \rightarrow F$
12. $H \rightarrow V$
13. $H \rightarrow R$
14. $H \rightarrow (A)$
15. $F \rightarrow V(B)$
16. $V \rightarrow L$
17. $V \rightarrow VL$
18. $V \rightarrow VD$
19. $R \rightarrow I$
20. $R \rightarrow I.$
21. $R \rightarrow .I$
22. $R \rightarrow I.I$
23. $I \rightarrow D$
24. $I \rightarrow ID$
25. $Z \rightarrow +$
26. $Z \rightarrow -$
27. $M \rightarrow *$
28. $M \rightarrow /$
29. $L \rightarrow \lambda$
30. $D \rightarrow \delta$

Требуется провести безвозвратный синтактический анализ для определения принадлежности языку, порожденному грамматикой *G*, каждой из трех цепочек индивидуального задания. Соответствующие деревья разбора построить в текстовом процессоре или в любом графическом редакторе.

3.5 Варианты индивидуальных заданий

Вариант 1

-*sin*(*f*2**k*^3+*x*2*y*)/*exp*(4.98)+*ab*5*v*^(-3),6
a-*m*7**f*(*d*,*g*,5*a*)/*jk*6*v*-(.123)^+4.10
h,*k*,l9,2,3+5^*ln*(2-7**g*)

Вариант 2

*k*9+*fm*12/*tj*3(*d*,*g*,6*g*5*a*)*(*jk*6*v*)-(10.),9(4)
8,+*cos*(*bf*6^*k**3/2-*y*2*x*)+*ln*(4.98)-*bf*3*w*^(-.89),0
-4^*tg*(2.2-8**pi*),*k*,*h*-76,22.22

Библиотека ГГТУ им. П.О.Сухого

Вариант 3

a,33.34+8^*k*1(0.2-*pi*/52),*bc*,*x**11,78.02
-(50.),0,*a*15**hd*39/*ci*(2*d*,3**g*,1,*a*4*c*)/(7+*d*^2))
*sa*7,25,-*sec*(*yf*^2**dsa*/*f*^2-*n*6*yu*)+*exp*(13.91)/*bc*^(-.77),1

Вариант 4

09,66,*m*7*n*,+*abs*(-*k*3*gz*/*sd*^3.1-*kwa*/*v**7)*(*h*(5))/*z*^(-0.2)
99.0103**y*,*a*,33.34+12^*re*(*og*/19/1.7-*y*),*ui*
4/(7+4*h*^8)/*dsa*(*a*,*b*,*c*-4)-(123.),.0,

Вариант 5

1,*dsa*(*v*-5,*b*,23)-(.4565),7./8-34*5*gf*/3)
ka,56.3004**ds*,88.12,*c*-*re*(*are*/3.1/*k*+*ab*)^(-6/2)
exp(09**s*(*a*,11,*j*8*j*,+*abs*(-*k*3*gz*/*sd*^3.1-*kwa*/*v**4)))*10/*z*^(-0.2),7

Вариант 6

5^*gg*/1.1*(*dsa*(*v*-5,*b*,23)-(.4565),7.)/8-34*s*
nh^(-9/.2)+76.302-*ccs*,22.19,*d*-*ke*(*ek*/1.5/*k*,*mm*)
ln(106**a*(*s*,22,*y*7,+*exp*(-*m*9*nn*/*rr*^.8-*av*/*w**3.)))*b^*c*/6.6

Вариант 7

(13.23+*cos*(*x*+*z*)^2)/*abs*(*asd*(3,*x*)-2**y*^2)
(*g*(-1.13,43,7),+*tan*(23.2))/(*x*-2*m*)**y*^2,7,-9
(*sin*(-1.13)+*co*(*x*+*z*,456)^2)/(*x*-2)**y*^2,9.7

Вариант 8

x^2+41.345/*ln*(75+29.6),54*x**exp*(*y*3-*tr*(*x*,*y*3,*x*54)
cos(-3.12+*x*),*x*/83.6,-1.65*(*x*5^(21.5-*s*))
6,28.32,*exp*(*sin*(*x*))^(1/3)*(3.-*x*)/2.15

Вариант 9

exp(*b*8+*dd*(*z*1,*j*7,*k*23))+*f*6^3.1+90.524/*sin*(.002-23.),*b**36
5.64*(*y*3^(.304-*gg*))-*ln*(-9.21+*h*8*h*),*z*/43.,-11
(24/6.28)*(8.-3*cd*)/9.421,.666,*abs*(*cos*(*f*))^+5.

Вариант 10

37.2/*cos*(*a*-69.33),*dd**5/*ln*(*v*88+*mn*(1,*k*l4,*j*45))+*b*12^.077
exp(*ln*(*f*))^-.32,(55.2/7.48)*(*k*-2*cc*)/.47,*d*,9
26,*vv*/88.,22.94*(*as*7^(*cvc*-.25))-*abs*(*rs*-1.28)

Библиотека ГГТУ им.П.О.Сухого

Вариант 11

(.496/35.99)*(3*bb*-1.576)/44.22,*xx*,9+*ln*(*ln*(*vv*))^+6.66
*vv*8^3.7+11.49/*sin*(23.-*kfd*),56**pp*/*exp*(*gng*+*a*5(3,*h*,*h*11))
1.554*(*k*^(*j*3-9.92))-*cos*(7.115-*t*4),49.,*g*5/.232

Вариант 12

2.22**a*8/*ln*(4.4+*d*(*k*,15,*ss*9)),*nbd*^.95+237.1/*cos*(*a*52-5.22)
.67+*ss*1/44.,*d*77*(*f*33^(0.55-*h*94))-*sin*(*f*12-36.36)
0.36+*exp*(*cos*(*a*9))^+.4,(3.14/*k*89)*(2*b*2-5.)/1.234,*d*6*h*

Вариант 13

12.^*gsa*+4.29/*sin*(*zyx*-7.87-*m*9*m*),.56**c*3*c*/*exp*(*k*9(23,*s*,*t*5*t*)+8.02)
(*d*11/.567)*(4.-5*k*5)/888,*n*6*n*,21.4+*abs*(*sin*(3.14-*m*))^-1.1
*abc**(*pp*^(1.21-*ii*6))-*cos*(0.57-*ha*6),2.+*uo*0/.79

Вариант 14

ln(*tg*(*d*86-6.28))^+54.2+(9.17/*j*64)*(*f*48-.441)/1.23,74.1,2*m*
*z*9*.456/*abs*(*sab*(*g*44,3.14,*t*)+74.),*m*55^0.39+.102/*cos*(1.57-*vw*-0.66)
-*sin*(*x*31-22.22),*wu*2+11./.58+*f*6*g**(*c*8^(*b*4-3.15))

Вариант 15

2.1^*fab*+11./*tg*(*hh*-55.4-*bv*),*nm*4*.33/*ln*(*jf*(2.5,*d*,6.,*y*6)+021)
cos(3.48-*mn*5),*bac*+2.34/.66+*pf*8*(*mx*^(5/18-*v*6))
(12.1-*b*5*a*)/44.,25.8,4*n*3**exp*(*sin*(7.46-*pf*4))^-3.1+.87+(*a*5/0/13)

Вариант 16

-*sin*(*f*2**k*^3+*x*2*y*)/*exp*(4.98)+*ab*5*v*^(-3),6
a-*m*7**f*(*d*,*g*,5*a*)/*jk*6*v*-(.123)^+4.10
h,*k*,l9,2,3+5^*ln*(2-7**g*)

Вариант 17

*k*9+*fm*12/*tj*3(*d*,*g*,6*g*5*a*)*(*jk*6*v*)-(10.),9(4)
8,+*cos*(*bf*6^*k**3/2-*y*2*x*)+*ln*(4.98)-*bf*3*w*^(-.89),0
-4^*tg*(2.2-8**pi*),*k*,*h*-76,22.22

Вариант 18

a,33.34+8^*k*1(0.2-*pi*/52),*bc*,*x**11,78.02
-(50.),0,*a*15**hd*39/*ci*(2*d*,3**g*,1,*a*4*c*)/(7+*d*^2))
*sa*7,25,-*sec*(*yf*^2**dsa*/*f*^2-*n*6*yu*)+*exp*(13.91)/*bc*^(-.77),1

Библиотека ГГТУ им. П.О.Сухого

Вариант 19

09,66,*m*7*n*,+*abs*(-*k*3*gz*/*sd*^3.1-*kwa*/*v**7)*(*h*(5))/*z*^(-0.2)
99.0103**y*,*a*,33.34+12^*re*(*og*/19/1.7-*y*),*ui*
4/(7+4*h*^8)/*dsa*(*a*,*b*,*c*-4)-(123.),.0,

Вариант 20

1,*dsa*(*v*-5,*b*,23)-(.4565),7./8-34*5*gf*/3)
ka,56.3004**ds*,88.12,*c*-*re*(*are*/3.1/*k*+*ab*)^(-6/2)
exp(09**s*(*a*,11,*j*8*j*,+*abs*(-*k*3*gz*/*sd*^3.1-*kwa*/*v**4)))*10/*z*^(-0.2),7

Вариант 21

5^*gg*/1.1*(*dsa*(*v*-5,*b*,23)-(.4565),7.)/8-34*s*
nh^(-9/.2)+76.302-*ccs*,22.19,*d*-*ke*(*ek*/1.5/*k*,*mm*)
ln(106**a*(*s*,22,*y*7,+*exp*(-*m*9*nn*/*rr*^.8-*av*/*w**3.)))**b*^*c*/6.6

Вариант 22

(13.23+*cos*(*x*+*z*)^2)/*abs*(*asd*(3,*x*)-2**y*^2)
(*g*(-1.13,43,7),+*tan*(23.2))/(*x*-2*m*)**y*^2,7,-9
(*sin*(-1.13)+*co*(*x*+*z*,456)^2)/(*x*-2)**y*^2,9.7

Вариант 23

x^2+41.345/*ln*(75+29.6),54*x***exp*(*y*3-*tr*(*x*,*y*3,*x*54)
cos(-3.12+*x*),*x*/83.6,-1.65*(*x*5^(21.5-*s*))
6,28.32,*exp*(*sin*(*x*))^(1/3)*(3.-*x*)/2.15

Вариант 24

exp(*b*8+*dd*(*z*1,*j*7,*k*23))+*f*6^3.1+90.524/*sin*(.002-23.),*b**36
5.64*(*y*3^(.304-*gg*))-*ln*(-9.21+*h*8*h*),*z*/43.,-11
(24/6.28)*(8.-3*cd*)/9.421,.666,*abs*(*cos*(*f*))^+5.

Вариант 25

37.2/*cos*(*a*-69.33),*dd**5/*ln*(*v*88+*mn*(1,*k*l4,*j*45))+*b*12^.077
exp(*ln*(*f*))^-.32,(55.2/7.48)*(*k*-2*cc*)/.47,*d*,9
26,*vv*/88.,22.94*(*as*7^(*cvc*-.25))-*abs*(*rs*-1.28)

Вариант 26

(.496/35.99)*(3*bb*-1.576)/44.22,*xx*,9+*ln*(*ln*(*vv*))^+6.66
*vv*8^3.7+11.49/*sin*(23.-*kfd*),56**pp*/*exp*(*gng*+*a*5(3,*h*,*h*11))
1.554*(*k*^(*j*3-9.92))-*cos*(7.115-*t*4),49.,*g*5/.232

Библиотека БГТУ им.П.О.Сухого

Вариант 27

2.22*a8/ln(4.4+d(k,15,ss9)),nbd^.95+237.1/cos(a52-5.22)
.67+ss1/44.,d77*(f33^(0.55-h94))-sin(f12-36.36)
0.36+exp(cos(a9))^+.4,(3.14/k89)*(2b2-5.)/1.234,d6h

Вариант 28

12.^gsa+4.29/sin(zyx-7.87-m9m),.56*c3c/exp(k9(23,s,t5t)+8.02)
(d11/.567)*(4.-5k5)/888,n6n,21.4+abs(sin(3.14-m))^-1.1
abc*(pp^(1.21-ii6))-cos(0.57-ha6),2.+uo0/.79

Вариант 29

ln(tg(d86-6.28))^+54.2+(9.17/j64)*(f48-.441)/1.23,74.1,2m
z9*.456/abs(sab(g44,3.14,t)+74.),m55^0.39+.102/cos(1.57-vw-0.66)
-sin(x31-22.22),wu2+11./.58+f6g*(c8^(b4-3.15))

Вариант 30

2.1^fab+11./tg(hh-55.4-bv),nm4*.33/ln(jf(2.5,d,6.,y6)+021)
cos(3.48-mn5),bac+2.34/.66+pf8*(mx^(5/18-v6))
(12.1-b5a)/44.,25.8,4n3*exp(sin(7.46-pf4))^-3.1+.87+(a5/0/13)

3.6 Структура отчета по лабораторной работе

Отчет по лабораторной работе должен содержать следующие элементы:

- цель работы;
- задание;
- дерево разбора первой цепочки и вывод о принадлежности цепочки языку (при наличии синтаксических ошибок указать все допущенные ошибки);
- дерево разбора второй цепочки и вывод о принадлежности цепочки языку (при наличии синтаксических ошибок указать все допущенные ошибки);
- дерево разбора третьей цепочки и вывод о принадлежности цепочки языку (при наличии синтаксических ошибок указать все допущенные ошибки);
- общий вывод по всем построенным деревьям разбора.

Библиотека ГГТУ им. П.О.Сухого

4 ЛАБОРАТОРНАЯ РАБОТА № 4. РАЗРАБОТКА КС-ГРАММАТИК

4.1 Цель и время выполнения работы

Цель работы: получить навыки создания КС-грамматик для заданного языка, разработка алгоритма и программы формирования множеств левых и правых символов для нетерминальных символов грамматики.

Время выполнения работы – 4 часа.

4.2 Краткие теоретические сведения

Идея создания безвозвратного синтаксического анализа, основанного на простых грамматиках предшествования, была предложена в 60-ых годах прошлого века американскими учеными Н. Виртом и В. Вебером.

Для каждого нетерминального символа введем понятия множеств левых и правых символов:

$\mathfrak{L}(U)=\{W|\exists(U\rightarrow Wx)$ □ $\exists(U\rightarrow Zx)$ & $W\in\mathfrak{L}(Z)\}$ – множество левых символов нетерминального символа U состоит из символов, которые начинают цепочки, выводимые из U.

$\mathfrak{R}(U)=\{W|\exists(U\rightarrow xW)$ □ $\exists(U\rightarrow xZ)$ & $W\in\mathfrak{R}(Z)\}$ – множество правых символов нетерминального символа U состоит из символов, которые заканчивают цепочки, выводимые из U.

4.3 Список рекомендуемой литературы

1. Ахо А. Теория синтаксического анализа, перевода и компиляции. Том 1. Синтаксический анализ / А. Ахо, Дж. Ульман. – М. : Мир, 1978. – 612 с.

2. Ахо А. Теория синтаксического анализа, перевода и компиляции. Том 2. Компиляция / А. Ахо, Дж. Ульман. – М. : Мир, 1978. – 488 с.

4.4 Содержание работы и указания по ее выполнению

Лабораторная работа предполагает выполнение двух этапов.

Библиотека ГГТУ им. П.О. Сухого

На первом этапе требуется разработать КС-грамматику, порождающую заданный язык.

На втором этапе необходимо разработать программу, которая для каждого нетерминального символа определяет множество его левых или правых символов:

4.5 Варианты индивидуальных заданий

Вариант 1

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – сложение и умножение.

Пример предложения языка: ((52+37)*14+8*92+673*(12+600))*4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его левых символов.

Вариант 2

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – идентификаторы неограниченной длины, состоящие из латинских строчных букв, а операции – сложение и умножение.

Пример предложения языка: ((*fd*+*jh*)***vv*+*s***sa*+*kiu**(*mm*+*cxz*))**r*

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

Вариант 3

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – сложение и деление.

Пример предложения языка: ((52+37)/14+8/92+673/(12+600))/4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

Вариант 4

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого –

Библиотека ГГТУ им. П.О.Сухого

идентификаторы неограниченной длины, состоящие из латинских строчных букв, а операции – сложение и деление.

Пример предложения языка: ((*fd*+*jh*)/*vv*+*s*/*sa*+*kiu*/(*mm*+*cxz*))/*r*

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его левых символов.

Вариант 5

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – сложение и возведение в степень.

Пример предложения языка: ((52+37)^14+8^92+673^(12+600))^4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его левых символов.

Вариант 6

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – идентификаторы неограниченной длины, состоящие из латинских строчных букв, а операции – сложение и возведение в степень.

Пример предложения языка: ((*fd*+*jh*)^*vv*+*s*^*sa*+*kiu*^(*mm*+*cxz*))^*r*

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

Вариант 7

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – вычитание и умножение.

Пример предложения языка: ((52-37)*14-8*92-673*(12-600))*4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

Вариант 8

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого –

Библиотека ГГТУ им. П.О.Сухого

идентификаторы неограниченной длины, состоящие из латинских строчных букв, а операции – вычитание и умножение.

Пример предложения языка: ((*fd-jh*)**vv-s*sa-kiu**(*mm-cxz*))**r*

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его левых символов.

Вариант 9

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – вычитание и деление.

Пример предложения языка: ((52-37)/14-8/92-673/(12-600))/4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его левых символов.

Вариант 10

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – идентификаторы неограниченной длины, состоящие из латинских строчных букв, а операции – вычитание и деление.

Пример предложения языка: ((*fd-jh*)/*vv-s*/*sa-kiu*/(*mm-cxz*))/*r*

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

Вариант 11

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – вычитание и возведение в степень.

Пример предложения языка: ((52-37)^14-8^92-673^(12-600))^4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

Вариант 12

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого –

Библиотека ГГТУ им. П.О. Сухого

идентификаторы неограниченной длины, состоящие из латинских строчных букв, а операции – вычитание и возведение в степень.

Пример предложения языка: ((*fd-jh*)^*vv-s*^*sa-kiu*^(*mm-cxz*))^*r*

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его левых символов.

Вариант 13

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – возведение в степень и умножение.

Пример предложения языка: ((52*37)^14*8^92*673^(12*600))^4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его левых символов.

Вариант 14

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – идентификаторы неограниченной длины, состоящие из латинских строчных букв, а операции – умножение и возведение в степень.

Пример предложения языка: ((*fd*jh*)^*vv*s*^*sa*kiu*^(*mm*cxz*))^*r*

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

Вариант 15

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – деление и умножение.

Пример предложения языка: ((52/37)*14/8*92/673*(12/600))*4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

Библиотека ГГТУ им. П.О.Сухого

Вариант 16

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – сложение и умножение.

Пример предложения языка: ((52+37)*14+8*92+673*(12+600))*4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его левых символов.

Вариант 17

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – идентификаторы неограниченной длины, состоящие из латинских строчных букв, а операции – сложение и умножение.

Пример предложения языка: ((*fd*+*jh*)**vv*+*s***sa*+*kiu***mm*+*cxz*))**r*

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

Вариант 18

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – сложение и деление.

Пример предложения языка: ((52+37)/14+8/92+673/(12+600))/4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

Вариант 19

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – идентификаторы неограниченной длины, состоящие из латинских строчных букв, а операции – сложение и деление.

Пример предложения языка: ((*fd*+*jh*)/*vv*+*s*/*sa*+*kiu*/(*mm*+*cxz*))/*r*

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его левых символов.

Библиотека ГГТУ им. П.О.Сухого

Вариант 20

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – сложение и возведение в степень.

Пример предложения языка: ((52+37)^14+8^92+673^(12+600))^4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его левых символов.

Вариант 21

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – идентификаторы неограниченной длины, состоящие из латинских строчных букв, а операции – сложение и возведение в степень.

Пример предложения языка: ((*fd+jh*)^*vv+s^sa+kiu*^(*mm+cxz*))^*r*

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

Вариант 22

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – вычитание и умножение.

Пример предложения языка: ((52-37)*14-8*92-673*(12-600))*4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

Вариант 23

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – идентификаторы неограниченной длины, состоящие из латинских строчных букв, а операции – вычитание и умножение.

Пример предложения языка: ((*fd-jh*)**vv-s**sa-kiu**(*mm-cxz*))**r*

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его левых символов.

Библиотека ГГТУ им. П.О.Сухого

Вариант 24

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – вычитание и деление.

Пример предложения языка: ((52-37)/14-8/92-673/(12-600))/4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его левых символов.

Вариант 25

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – идентификаторы неограниченной длины, состоящие из латинских строчных букв, а операции – вычитание и деление.

Пример предложения языка: ((*fd-jh*)/*vv-s*/*sa-kiu*/(*mm-cxz*))/*r*

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

Вариант 26

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – вычитание и возведение в степень.

Пример предложения языка: ((52-37)^14-8^92-673^(12-600))^4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

Вариант 27

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – идентификаторы неограниченной длины, состоящие из латинских строчных букв, а операции – вычитание и возведение в степень.

Пример предложения языка: ((*fd-jh*)^*vv-s*^*sa-kiu*^(*mm-cxz*))^*r*

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его левых символов.

Библиотека ГГТУ им. П.О.Сухого

Вариант 28

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – возведение в степень и умножение.

Пример предложения языка: ((52*37)^14*8^92*673^(12*600))^4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его левых символов.

Вариант 29

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – идентификаторы неограниченной длины, состоящие из латинских строчных букв, а операции – умножение и возведение в степень.

Пример предложения языка: ((*fd*jh*)^*vv*s^sa*kiu*^(*mm*cxz*))^*r*

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

Вариант 30

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – деление и умножение.

Пример предложения языка: ((52/37)*14/8*92/673*(12/600))*4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

4.6 Структура отчета по лабораторной работе

Отчет по лабораторной работе должен содержать следующие элементы:

- цель работы;
- задание;
- КС-грамматика, порождающая заданный язык;
- пример дерева вывода по КС-грамматики цепочки заданного языка;

Библиотека ГГТУ им. П.О.Сухого

- описание алгоритма и структур данных программы, определяющей множество левых (правых) символов для нетерминальных символов грамматики;
- скриншоты результатов выполнения программы;
- рационально прокомментированный текст программы;
- общий вывод по лабораторной работе.

Библиотека ГГТУ им.П.О.Сухого

5 ЛАБОРАТОРНАЯ РАБОТА № 5. ПОСТРОЕНИЕ МАТРИЦЫ ПРЕДШЕСТВОВАНИЯ

5.1 Цель и время выполнения работы

Цель работы: разработка алгоритма и программы построения матрицы предшествования для заданной КС-грамматики.

Время выполнения работы – 4 часа.

5.2 Краткие теоретические сведения

Простой грамматикой предшествования называется КС-грамматика, удовлетворяющая двум условиям:

1) в матрице предшествования этой грамматике каждый элемент содержит не более одного соотношения;
2) в грамматике нет правил с одинаковыми правыми частями.

Заметим, что второе условие является наиболее сильным ограничением, т. к. при наличии матричного элемента с несколькими соотношениями (наличие конфликтов) всегда можно преобразовать правила грамматики (путем введения новых нетерминальных символов и сокращения длины правых частей правил) так, что конфликты исчезнут.

Отметим, что грамматика, порождающая множество идентификаторов, является простой грамматикой предшествования.

Если грамматика является ПГП-грамматикой, то синтаксический разбор ведется по следующему алгоритму:

1) в анализируемой строке находится самая левая подцепочка символов $A_iA_{i+1}\ldots A_{i+n}$, удовлетворяющая условию:

$$A_{i-1} <\cdot A_i \doteq A_i \doteq \ldots \doteq A_n \cdot> A_{n+1}$$

2) в множестве правил P выбирается правило $U \rightarrow A_iA_{i+1}\ldots A_{i+n}$ и в анализируемой строке подцепочка символов $A_iA_{i+1}\ldots A_{i+n}$ заменяется символом U, после чего переходим к первому пункту;
3) разбор заканчивается, когда анализируемая строка превращается (свертывается) в единственный начальный нетерминальный символ S – анализируемая строка принадлежит языку;

Библиотека ГГТУ им. П.О.Сухого

4) предполагается, что перед первым символом строки стоит отношение «меньше», а после последнего – «больше».

Данный алгоритм обеспечивает безвозвратный разбор.

5.3 Список рекомендуемой литературы

1. Ахо А. Теория синтаксического анализа, перевода и компиляции. Том 1. Синтаксический анализ / А. Ахо, Дж. Ульман. – М. : Мир, 1978. – 612 с.

2. Ахо А. Теория синтаксического анализа, перевода и компиляции. Том 2. Компиляция / А. Ахо, Дж. Ульман. – М. : Мир, 1978. – 488 с.

5.4 Содержание работы и указания по ее выполнению

Лабораторная работа предполагает выполнение нескольких этапов.

На первом этапе следует разработать структуры данных для ввода и хранения заданной КС-грамматики. Далее необходимо разработать структуры данных и алгоритм формирования множеств левых и правых символов для нетерминальных символов грамматики. На заключительном этапе надо разработать алгоритм построения матрицы предшествования и программу, реализующую перечисленные действия. После выполнения программы требуется определить, является ли заданная КС-грамматика простой грамматикой предшествования.

5.5 Варианты индивидуальных заданий

Вариант 1

1. $S \rightarrow A$
2. $A \rightarrow Y$
3. $A \rightarrow A+Y$
4. $Y \rightarrow T$
5. $T \rightarrow H$
6. $T \rightarrow T*H$
7. $H \rightarrow I$
8. $H \rightarrow (C$
9. $C \rightarrow A)$
10. $I \rightarrow D$
11. $I \rightarrow ID$

12-21. $D \rightarrow 0|1|\ldots|9$

Библиотека ГГТУ им. П.О. Сухого

Вариант 2

1.	$S \to B$	7.	$Z \to N$
2.	$B \to W$	8.	$Z \to (C$
3.	$B \to B+W$	9.	$C \to B)$
4.	$W \to X$	10.	$N \to L$
5.	$X \to H$	11.	$N \to NL$
6.	$X \to X*Z$	12-37.	$L \to a\|b\|\ldots\|z$

Вариант 3

1.	$S \to A$	7.	$H \to I$
2.	$A \to Y$	8.	$H \to (C$
3.	$A \to A+Y$	9.	$C \to A)$
4.	$Y \to T$	10.	$I \to D$
5.	$T \to H$	11.	$I \to ID$
6.	$T \to T/H$	12-21.	$D \to 0\|1\|\ldots\|9$

Вариант 4

1.	$S \to B$	7.	$Z \to N$
2.	$B \to W$	8.	$Z \to (C$
3.	$B \to B+W$	9.	$C \to B)$
4.	$W \to X$	10.	$N \to L$
5.	$X \to H$	11.	$N \to NL$
6.	$X \to X/Z$	12-37.	$L \to a\|b\|\ldots\|z$

Вариант 5

1.	$S \to A$	7.	$H \to I$
2.	$A \to Y$	8.	$H \to (C$
3.	$A \to A+Y$	9.	$C \to A)$
4.	$Y \to T$	10.	$I \to D$
5.	$T \to H$	11.	$I \to ID$
6.	$T \to T\hat{}H$	12-21.	$D \to 0\|1\|\ldots\|9$

Вариант 6

1.	$S \to B$	7.	$Z \to N$
2.	$B \to W$	8.	$Z \to (C$
3.	$B \to B+W$	9.	$C \to B)$
4.	$W \to X$	10.	$N \to L$
5.	$X \to H$	11.	$N \to NL$
6.	$X \to X\hat{}Z$	12-37.	$L \to a\|b\|\ldots\|z$

Библиотека ГГТУ им. П.О. Сухого

Вариант 7

1. $S \to A$ 7. $H \to I$
2. $A \to Y$ 8. $H \to (C$
3. $A \to A\text{-}Y$ 9. $C \to A)$
4. $Y \to T$ 10. $I \to D$
5. $T \to H$ 11. $I \to ID$
6. $T \to T*H$ 12-21. $D \to 0|1|\ldots|9$

Вариант 8

1. $S \to B$ 7. $Z \to N$
2. $B \to W$ 8. $Z \to (C$
3. $B \to B\text{-}W$ 9. $C \to B)$
4. $W \to X$ 10. $N \to L$
5. $X \to H$ 11. $N \to NL$
6. $X \to X*Z$ 12-37. $L \to a|b|\ldots|z$

Вариант 9

1. $S \to A$ 7. $H \to I$
2. $A \to Y$ 8. $H \to (C$
3. $A \to A\text{-}Y$ 9. $C \to A)$
4. $Y \to T$ 10. $I \to D$
5. $T \to H$ 11. $I \to ID$
6. $T \to T/H$ 12-21. $D \to 0|1|\ldots|9$

Вариант 10

1. $S \to B$ 7. $Z \to N$
2. $B \to W$ 8. $Z \to (C$
3. $B \to B\text{-}W$ 9. $C \to B)$
4. $W \to X$ 10. $N \to L$
5. $X \to H$ 11. $N \to NL$
6. $X \to X/Z$ 12-37. $L \to a|b|\ldots|z$

Вариант 11

1. $S \to A$ 7. $H \to I$
2. $A \to Y$ 8. $H \to (C$
3. $A \to A\text{-}Y$ 9. $C \to A)$
4. $Y \to T$ 10. $I \to D$
5. $T \to H$ 11. $I \to ID$
6. $T \to T\hat{}H$ 12-21. $D \to 0|1|\ldots|9$

Библиотека ГГТУ им. П.О. Сухого

Вариант 12

1.	$S \to B$	7.	$Z \to N$
2.	$B \to W$	8.	$Z \to (C$
3.	$B \to B\text{-}W$	9.	$C \to B)$
4.	$W \to X$	10.	$N \to L$
5.	$X \to H$	11.	$N \to NL$
6.	$X \to X\text{^}Z$	12-37.	$L \to a\|b\|\ldots\|z$

Вариант 13

1.	$S \to A$	7.	$H \to I$
2.	$A \to Y$	8.	$H \to (C$
3.	$A \to A*Y$	9.	$C \to A)$
4.	$Y \to T$	10.	$I \to D$
5.	$T \to H$	11.	$I \to ID$
6.	$T \to T\text{^}H$	12-21.	$D \to 0\|1\|\ldots\|9$

Вариант 14

1.	$S \to B$	7.	$Z \to N$
2.	$B \to W$	8.	$Z \to (C$
3.	$B \to B*W$	9.	$C \to B)$
4.	$W \to X$	10.	$N \to L$
5.	$X \to H$	11.	$N \to NL$
6.	$X \to X\text{^}Z$	12-37.	$L \to a\|b\|\ldots\|z$

Вариант 15

1.	$S \to A$	7.	$H \to I$
2.	$A \to Y$	8.	$H \to (C$
3.	$A \to A*Y$	9.	$C \to A)$
4.	$Y \to T$	10.	$I \to D$
5.	$T \to H$	11.	$I \to ID$
6.	$T \to T/H$	12-21.	$D \to 0\|1\|\ldots\|9$

Вариант 16

7.	$S \to A$	7.	$H \to I$
8.	$A \to Y$	8.	$H \to (C$
9.	$A \to A+Y$	9.	$C \to A)$
10.	$Y \to T$	10.	$I \to D$
11.	$T \to H$	11.	$I \to ID$
12.	$T \to T*H$	12-21.	
	$D \to 0\|1\|\ldots\|9$		

Библиотека ГГТУ им. П.О. Сухого

Вариант 17

1. $S \rightarrow B$
2. $B \rightarrow W$
3. $B \rightarrow B+W$
4. $W \rightarrow X$
5. $X \rightarrow H$
6. $X \rightarrow X*Z$
7. $Z \rightarrow N$
8. $Z \rightarrow (C$
9. $C \rightarrow B)$
10. $N \rightarrow L$
11. $N \rightarrow NL$

12-37. $L \rightarrow a|b|...|z$

Вариант 18

1. $S \rightarrow A$
2. $A \rightarrow Y$
3. $A \rightarrow A+Y$
4. $Y \rightarrow T$
5. $T \rightarrow H$
6. $T \rightarrow T/H$
7. $H \rightarrow I$
8. $H \rightarrow (C$
9. $C \rightarrow A)$
10. $I \rightarrow D$
11. $I \rightarrow ID$

12-21. $D \rightarrow 0|1|\ldots|9$

Вариант 19

1. $S \rightarrow B$
2. $B \rightarrow W$
3. $B \rightarrow B+W$
4. $W \rightarrow X$
5. $X \rightarrow H$
6. $X \rightarrow X/Z$
7. $Z \rightarrow N$
8. $Z \rightarrow (C$
9. $C \rightarrow B)$
10. $N \rightarrow L$
11. $N \rightarrow NL$

12-37. $L \rightarrow a|b|\ldots|z$

Вариант 20

1. $S \rightarrow A$
2. $A \rightarrow Y$
3. $A \rightarrow A+Y$
4. $Y \rightarrow T$
5. $T \rightarrow H$
6. $T \rightarrow T\hat{}H$
7. $H \rightarrow I$
8. $H \rightarrow (C$
9. $C \rightarrow A)$
10. $I \rightarrow D$
11. $I \rightarrow ID$

12-21. $D \rightarrow 0|1|\ldots|9$

Вариант 21

1. $S \rightarrow B$
2. $B \rightarrow W$
3. $B \rightarrow B+W$
4. $W \rightarrow X$
5. $X \rightarrow H$
6. $X \rightarrow X\hat{}Z$
7. $Z \rightarrow N$
8. $Z \rightarrow (C$
9. $C \rightarrow B)$
10. $N \rightarrow L$
11. $N \rightarrow NL$

12-37. $L \rightarrow a|b|\ldots|z$

Библиотека ГГТУ им. П.О. Сухого

Вариант 22

1. $S \to A$
2. $A \to Y$
3. $A \to A\text{-}Y$
4. $Y \to T$
5. $T \to H$
6. $T \to T*H$
7. $H \to I$
8. $H \to (C$
9. $C \to A)$
10. $I \to D$
11. $I \to ID$
12-21. $D \to 0|1|\ldots|9$

Вариант 23

1. $S \to B$
2. $B \to W$
3. $B \to B\text{-}W$
4. $W \to X$
5. $X \to H$
6. $X \to X*Z$
7. $Z \to N$
8. $Z \to (C$
9. $C \to B)$
10. $N \to L$
11. $N \to NL$
12-37. $L \to a|b|\ldots|z$

Вариант 24

1. $S \to A$
2. $A \to Y$
3. $A \to A\text{-}Y$
4. $Y \to T$
5. $T \to H$
6. $T \to T/H$
7. $H \to I$
8. $H \to (C$
9. $C \to A)$
10. $I \to D$
11. $I \to ID$
12-21. $D \to 0|1|\ldots|9$

Вариант 25

1. $S \to B$
2. $B \to W$
3. $B \to B\text{-}W$
4. $W \to X$
5. $X \to H$
6. $X \to X/Z$
7. $Z \to N$
8. $Z \to (C$
9. $C \to B)$
10. $N \to L$
11. $N \to NL$
12-37. $L \to a|b|\ldots|z$

Вариант 26

1. $S \to A$
2. $A \to Y$
3. $A \to A\text{-}Y$
4. $Y \to T$
5. $T \to H$
6. $T \to T\text{^}H$
7. $H \to I$
8. $H \to (C$
9. $C \to A)$
10. $I \to D$
11. $I \to ID$
12-21. $D \to 0|1|\ldots|9$

Библиотека ГГТУ им. П.О. Сухого

Вариант 27

1.	$S \to B$	7.	$Z \to N$
2.	$B \to W$	8.	$Z \to (C$
3.	$B \to B\text{-}W$	9.	$C \to B)$
4.	$W \to X$	10.	$N \to L$
5.	$X \to H$	11.	$N \to NL$
6.	$X \to X\text{^}Z$	12-37.	$L \to a\|b\|\ldots\|z$

Вариант 28

1.	$S \to A$	7.	$H \to I$
2.	$A \to Y$	8.	$H \to (C$
3.	$A \to A*Y$	9.	$C \to A)$
4.	$Y \to T$	10.	$I \to D$
5.	$T \to H$	11.	$I \to ID$
6.	$T \to T\text{^}H$	12-21.	$D \to 0\|1\|\ldots\|9$

Вариант 29

1.	$S \to B$	7.	$Z \to N$
2.	$B \to W$	8.	$Z \to (C$
3.	$B \to B*W$	9.	$C \to B)$
4.	$W \to X$	10.	$N \to L$
5.	$X \to H$	11.	$N \to NL$
6.	$X \to X\text{^}Z$	12-37.	$L \to a\|b\|\ldots\|z$

Вариант 30

1.	$S \to A$	7.	$H \to I$
2.	$A \to Y$	8.	$H \to (C$
3.	$A \to A*Y$	9.	$C \to A)$
4.	$Y \to T$	10.	$I \to D$
5.	$T \to H$	11.	$I \to ID$
6.	$T \to T/H$	12-21.	$D \to 0\|1\|\ldots\|9$

5.6 Структура отчета по лабораторной работе

Отчет по лабораторной работе должен содержать следующие элементы:

- цель работы;
- задание;

Библиотека ГГТУ им. П.О. Сухого

- описание структур данных для ввода и хранения заданной КС-грамматики;
- описание структуры данных и алгоритма формирования множеств левых и правых символов для нетерминальных символов грамматики;
- описание алгоритма построения матрицы предшествования;
- скриншоты результатов выполнения программы;
- рационально прокомментированный текст программы;
- вывод о принадлежности исследуемой грамматики классу ПГП.

Библиотека ГГТУ им.П.О.Сухого

6 ЛАБОРАТОРНАЯ РАБОТА № 6. ПРЕОБРАЗОВАНИЕ КС-ГРАММАТИКИ В ПРОСТУЮ ГРАММАТИКУ ПРЕДШЕСТВОВАНИЯ

6.1 Цель и время выполнения работы

Цель работы: проведение исследования заданной КС-грамматики на предмет того, является ли она простой грамматикой предшествования (ПГП); преобразование КС-грамматики в ПГП.

Время выполнения работы – 4 часа.

6.2 Краткие теоретические сведения

Необходимые теоретические сведения изложены в лабораторной работе № 5.

6.3 Содержание работы и указания по ее выполнению

С помощью программы, разработанной в лабораторной работе № 5, определить, является ли заданная КС-грамматика ПГП. В случае, если исходная грамматика не удовлетворяет условиям ПГП, то ввести необходимые новые нетерминальные символы и преобразовать правила таким образом, чтобы получилась эквивалентная по порождающим свойствам грамматика удовлетворяющая условиям ПГП.

6.4 Варианты индивидуальных заданий

Вариант 1

1. $S \to B$
2. $B \to X$
3. $B \to B+X$
4. $X \to Z$
5. $X \to X*Z$
6. $Z \to N$
7. $Z \to (B)$
8. $N \to L$
9. $N \to NL$
10-35. $L \to a|b|\ldots|z$

Вариант 2

1. $S \to A$
2. $A \to T$
3. $A \to A+T$
4. $T \to H$
5. $T \to T*H$
6. $H \to I$
7. $H \to (A)$
8. $I \to D$
9. $I \to ID$
10-19. $D \to 0|1|\ldots|9$

Библиотека ГГТУ им. П.О. Сухого

Вариант 3

1. $S \to B$ 6. $Z \to N$
2. $B \to X$ 7. $Z \to (B)$
3. $B \to B+X$ 8. $N \to L$
4. $X \to Z$ 9. $N \to NL$
5. $X \to X/Z$ 10-35. $L \to a|b|\ldots|z$

Вариант 4

1. $S \to A$ 6. $H \to I$
2. $A \to T$ 7. $H \to (A)$
3. $A \to A+T$ 8. $I \to D$
4. $T \to H$ 9. $I \to ID$
5. $T \to T/H$ 10-19. $D \to 0|1|\ldots|9$

Вариант 5

1. $S \to B$ 6. $Z \to N$
2. $B \to X$ 7. $Z \to (B)$
3. $B \to B+X$ 8. $N \to L$
4. $X \to Z$ 9. $N \to NL$
5. $X \to X\hat{}Z$ 10-35. $L \to a|b|\ldots|z$

Вариант 6

1. $S \to A$ 6. $H \to I$
2. $A \to T$ 7. $H \to (A)$
3. $A \to A+T$ 8. $I \to D$
4. $T \to H$ 9. $I \to ID$
5. $T \to T\hat{}H$ 10-19. $D \to 0|1|\ldots|9$

Вариант 7

1. $S \to B$ 6. $Z \to N$
2. $B \to X$ 7. $Z \to (B)$
3. $B \to B-X$ 8. $N \to L$
4. $X \to Z$ 9. $N \to NL$
5. $X \to X*Z$ 10-35. $L \to a|b|\ldots|z$

Вариант 8

1. $S \to A$ 6. $H \to I$
2. $A \to T$ 7. $H \to (A)$
3. $A \to A-T$ 8. $I \to D$
4. $T \to H$ 9. $I \to ID$
5. $T \to T*H$ 10-19. $D \to 0|1|\ldots|9$

Библиотека ГГТУ им. П.О.Сухого

Вариант 9

1. $S \to B$
2. $B \to X$
3. $B \to B\text{-}X$
4. $X \to Z$
5. $X \to X/Z$
6. $Z \to N$
7. $Z \to (B)$
8. $N \to L$
9. $N \to NL$
10-35. $L \to a|b|\ldots|z$

Вариант 10

1. $S \to A$
2. $A \to T$
3. $A \to A\text{-}T$
4. $T \to H$
5. $T \to T/H$
6. $H \to I$
7. $H \to (A)$
8. $I \to D$
9. $I \to ID$
10-19. $D \to 0|1|\ldots|9$

Вариант 11

1. $S \to B$
2. $B \to X$
3. $B \to B\text{-}X$
4. $X \to Z$
5. $X \to X\text{^}Z$
6. $Z \to N$
7. $Z \to (B)$
8. $N \to L$
9. $N \to NL$
10-35. $L \to a|b|\ldots|z$

Вариант 12

1. $S \to A$
2. $A \to T$
3. $A \to A\text{-}T$
4. $T \to H$
5. $T \to T\text{^}H$
6. $H \to I$
7. $H \to (A)$
8. $I \to D$
9. $I \to ID$
10-19. $D \to 0|1|\ldots|9$

Вариант 13

1. $S \to B$
2. $B \to X$
3. $B \to B*X$
4. $X \to Z$
5. $X \to X\text{^}Z$
6. $Z \to N$
7. $Z \to (B)$
8. $N \to L$
9. $N \to NL$
10-35. $L \to a|b|\ldots|z$

Вариант 14

1. $S \to A$
2. $A \to T$
3. $A \to A*T$
4. $T \to H$
5. $T \to T\text{^}H$
6. $H \to I$
7. $H \to (A)$
8. $I \to D$
9. $I \to ID$
10-19. $D \to 0|1|\ldots|9$

Библиотека ГГТУ им. П.О.Сухого

Вариант 15

1. $S \to A$ 6. $H \to I$
2. $A \to T$ 7. $H \to (A)$
3. $A \to A+T$ 8. $I \to D$
4. $T \to H$ 9. $I \to ID$
5. $T \to T*H$ 10-19. $D \to 0|1|\ldots|9$

Вариант 16

1. $S \to A$ 6. $H \to I$
2. $A \to T$ 7. $H \to (A)$
3. $A \to A*T$ 8. $I \to D$
4. $T \to H$ 9. $I \to ID$
5. $T \to T/H$ 10-19. $D \to 0|1|\ldots|9$

Вариант 17

1. $S \to A$ 6. $H \to I$
2. $A \to T$ 7. $H \to (A)$
3. $A \to A+T$ 8. $I \to D$
4. $T \to H$ 9. $I \to ID$
5. $T \to T/H$ 10-19. $D \to 0|1|\ldots|9$

Вариант 18

1. $S \to B$ 6. $Z \to N$
2. $B \to X$ 7. $Z \to (B)$
3. $B \to B+X$ 8. $N \to L$
4. $X \to Z$ 9. $N \to NL$
5. $X \to X*Z$ 10-35. $L \to a|b|\ldots|z$

Вариант 19

1. $S \to A$ 6. $H \to I$
2. $A \to T$ 7. $H \to (A)$
3. $A \to A+T$ 8. $I \to D$
4. $T \to H$ 9. $I \to ID$
5. $T \to T\wedge H$ 10-19. $D \to 0|1|\ldots|9$

Вариант 20

1. $S \to B$ 6. $Z \to N$
2. $B \to X$ 7. $Z \to (B)$
3. $B \to B+X$ 8. $N \to L$
4. $X \to Z$ 9. $N \to NL$
5. $X \to X/Z$ 10-35. $L \to a|b|\ldots|z$

Библиотека ГГТУ им. П.О.Сухого

Вариант 21

1. $S \to A$
2. $A \to T$
3. $A \to A\text{-}T$
4. $T \to H$
5. $T \to T*H$
6. $H \to I$
7. $H \to (A)$
8. $I \to D$
9. $I \to ID$
10-19. $D \to 0|1|\ldots|9$

Вариант 22

1. $S \to B$
2. $B \to X$
3. $B \to B+X$
4. $X \to Z$
5. $X \to X\text{^}Z$
6. $Z \to N$
7. $Z \to (B)$
8. $N \to L$
9. $N \to NL$
10-35. $L \to a|b|\ldots|z$

Вариант 23

1. $S \to A$
2. $A \to T$
3. $A \to A\text{-}T$
4. $T \to H$
5. $T \to T/H$
6. $H \to I$
7. $H \to (A)$
8. $I \to D$
9. $I \to ID$
10-19. $D \to 0|1|\ldots|9$

Вариант 24

1. $S \to B$
2. $B \to X$
3. $B \to B\text{-}X$
4. $X \to Z$
5. $X \to X*Z$
6. $Z \to N$
7. $Z \to (B)$
8. $N \to L$
9. $N \to NL$
10-35. $L \to a|b|\ldots|z$

Вариант 25

1. $S \to A$
2. $A \to T$
3. $A \to A\text{-}T$
4. $T \to H$
5. $T \to T\text{^}H$
6. $H \to I$
7. $H \to (A)$
8. $I \to D$
9. $I \to ID$
10-19. $D \to 0|1|\ldots|9$

Вариант 26

1. $S \to B$
2. $B \to X$
3. $B \to B\text{-}X$
4. $X \to Z$
5. $X \to X/Z$
6. $Z \to N$
7. $Z \to (B)$
8. $N \to L$
9. $N \to NL$
10-35. $L \to a|b|\ldots|z$

Библиотека ГГТУ им. П.О.Сухого

Вариант 27

1. $S \rightarrow A$ 6. $H \rightarrow I$
2. $A \rightarrow T$ 7. $H \rightarrow (A)$
3. $A \rightarrow A*T$ 8. $I \rightarrow D$
4. $T \rightarrow H$ 9. $I \rightarrow ID$
5. $T \rightarrow T\hat{}H$ 10-19. $D \rightarrow 0|1|\ldots|9$

Вариант 28

1. $S \rightarrow B$ 6. $Z \rightarrow N$
2. $B \rightarrow X$ 7. $Z \rightarrow (B)$
3. $B \rightarrow B-X$ 8. $N \rightarrow L$
4. $X \rightarrow Z$ 9. $N \rightarrow NL$
5. $X \rightarrow X\hat{}Z$ 10-35. $L \rightarrow a|b|\ldots|z$

Вариант 29

1. $S \rightarrow A$ 6. $H \rightarrow I$
2. $A \rightarrow T$ 7. $H \rightarrow (A)$
3. $A \rightarrow A*T$ 8. $I \rightarrow D$
4. $T \rightarrow H$ 9. $I \rightarrow ID$
5. $T \rightarrow T/H$ 10-19. $D \rightarrow 0|1|\ldots|9$

Вариант 30

1. $S \rightarrow B$ 6. $Z \rightarrow N$
2. $B \rightarrow X$ 7. $Z \rightarrow (B)$
3. $B \rightarrow B*X$ 8. $N \rightarrow L$
4. $X \rightarrow Z$ 9. $N \rightarrow NL$
5. $X \rightarrow X\hat{}Z$ 10-35. $L \rightarrow a|b|\ldots|z$

6.5 Структура отчета по лабораторной работе

Отчет по лабораторной работе должен содержать следующие элементы:

- цель работы;
- задание;
- скриншоты результатов выполнения программы для исходной грамматики;
- вывод о принадлежности исследуемой грамматики классу ПГП;
- описание новой грамматики (введение новых нетерминальных символов и новых правил);

Библиотека ГГТУ им. П.О.Сухого

- доказательство эквивалентности исходной и преобразованных грамматик;
- скриншоты результатов выполнения программы для новой грамматики;
- вывод о принадлежности новой грамматики классу ПГП.

Библиотека ГГТУ им.П.О.Сухого

7 ЛАБОРАТОРНАЯ РАБОТА № 7. ПОСТРОЕНИЕ ДЕТЕРМИНИРОВАННОГО КОНЕЧНОГО АВТОМАТА

7.1 Цель и время выполнения работы

Цель работы: проведение синтаксического анализа конструкции языка программирования с помощью детерминированного конечного автомата (ДКА).

Время выполнения работы – 4 часа.

7.2 Краткие теоретические сведения

Необходимые теоретические сведения изложены в лекциях по учебной дисциплине «Методы трансляции», помещенных на учебном портале edu.gstu.by.

7.3 Содержание работы и указания по ее выполнению

Дано общее описание конструкции некоторого языка программирования. Задана конкретная конструкция. Предполагается, что проведен лексический анализ этой конструкции, в результате которого составлен новый (упрощенный) алфавит исходного языка. Данный алфавит характерен тем, что в нем каждое служебное слово, каждый идентификатор и каждое число, входящие в конструкцию языка программирования, рассматриваются как отдельные символы.

Требуется:

1) составить ДКА, который анализирует синтаксическую правильность заданной конструкции, т. е. соответствие структуры конструкции общему описанию;

2) разработать программу (на одном из изученных языков программирования), которая моделирует работу ДКА;

3) проверить работу программы при анализе правильной и неправильной конструкции.

7.4 Варианты индивидуальных заданий

Вариант 1

Условный оператор языка программирования имеет следующую структуру:

Библиотека ГГТУ им. П.О.Сухого

if *логическое_выражение* ***then*** *оператор_присваивания* ***else*** *оператор_присваивания*;

Логическое выражение, входящее в условный оператор, имеет следующую структуру:

элемент операция_отношения элемент

В качестве элемента может быть *переменная* или *число*.

Операция отношения – это одна из шести операций: =, <, >, <=, >=, <>.

Оператор присваивания имеет одну из следующих структур:

*переменная***:=***выражение*

*переменная***:=***алгебраическое_сложение выражение*

*переменная***:=***выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***if***, ***then***, ***else***.

Знаки: **;**, **:=**, **=,** <, >, +, **-, *,** /, *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

if *a*2 <= *r*5 ***then*** *a*3:= *r*1-*a*2*r5 ***else*** *a*3:= *r*1/*a*4+*a*2;

Пример неправильной конструкции:

if *a*2 <= *r*5 ***then*** *a*3:= *r*1-*a*2+ *r*5 ***else*** *a*3:= *r*1/*a*4+*a*2;

Вариант 2

Оператор цикла с предусловием языка программирования имеет следующую структуру:

while *логическое_выражение* ***do begin*** *оператор_присваивания***;** *оператор_присваивания***; end;**

Логическое выражение, входящее в оператор цикла, имеет следующую структуру:

элемент операция_отношения элемент

В качестве элемента может быть *переменная* или *число*.

Библиотека ГГТУ им. П.О. Сухого

Операция отношения – это одна из шести операций: =, <, >, <=, >=, <>.

Оператор присваивания имеет одну из следующих структур:

*переменная***:=***выражение*

*переменная***:=***алгебраическое_сложение выражение*

*переменная***:=***выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или **-**.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***while***, ***do***, ***begin***, ***end***.

Знаки: **;**, **:=**, **=,** <, >, +, **-, *,** /, *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

while *a*2 <= *r*5 ***do begin*** *a*3:= *r*1-*a*2*r*5; *a*4:= *r*1/*a*4+*a*2**; *end***;

Пример неправильной конструкции:

while *a*2 <= *r*5 ***do begin*** *a*3:= *r*1-*a*2**; *end***;

Вариант 3

Арифметический оператор цикла языка программирования имеет следующую структуру:

for *переменная* **:=** *выражение* ***until*** *выражение* ***do begin*** *оператор_присваивания***;** *оператор_присваивания***;** *оператор_присваивания***; *end*;**

Оператор присваивания имеет одну из следующих структур:

переменная **:=** *выражение*

переменная **:=** *алгебраическое_сложение выражение*

переменная **:=** *выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или **-**.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Библиотека ГГТУ им. П.О. Сухого

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***for***, ***until***, ***do***, ***begin***, ***end***.

Знаки: **;**, **:=**, ***,** /, +, **-,** *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

for *a*2 := *r*5*a*1 ***until*** *a*3-*a*4/*r*3 ***do begin*** *a*3:= *r*1-*a*2; *a*4:= *r*1**a*2**;** *a*6:= *r*6**; *end***;

Пример неправильной конструкции:

for *a*2 := *r*5*a*1 ***until*** *a*3-*a*4/*r*3*a*1 ***do begin*** *a*3:= *r*1-*a*2; *a*4:= *r*1**a*2**;** *a*6:= *r*6**; *end***;

Вариант 4

Оператор выбора языка программирования имеет следующую структуру:

case *переменная* ***of*** *число* **:** *оператор_присваивания***;** *число* **:** *оператор_присваивания***;** *число* **:** *оператор_присваивания***; *end;***

Оператор присваивания имеет одну из следующих структур:

переменная **:=** *выражение*

переменная **:=** *алгебраическое_сложение выражение*

переменная **:=** *выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***case***, ***of***, ***end***.

Знаки: **;**, **:=**, **:**, ***,** /, +, **-,** *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

case *a*2 ***of*** *r*1**:** *a*3**:=** *r*1-*a*2**a*5**;** *r*2**:** *a*3**:=** *r*1+*a*2/*r*4; *r*3**:** *a*3**:=***a*6**; *end*;**

Пример неправильной конструкции:

case *a*2 ***of*** *r*1**:** *a*3**:=** *r*1-*a*2**a*5**;** *r*2**:** *a*3**:=** *r*1+*a*2/*r*4; *r*3**:** *a*3**:=***a*6 ***end*;**

Библиотека ГГТУ им. П.О. Сухого

Вариант 5

Конструкция обработки исключительных ситуаций языка программирования имеет следующую структуру:

try *оператор_присваивания*; *оператор_присваивания*;
except
on *идентификатор_класса_исключения*: *класс_исключения* ***do*** *оператор_присваивания*;
on *идентификатор_класса_исключения*: *класс_исключения* ***do*** *оператор_присваивания*;
else *оператор_присваивания*;
end;

Оператор присваивания имеет одну из следующих структур:

переменная:=*число*
переменная:=*алгебраическое_сложение число*
переменная:= *число алгебраическое_сложение число*

Алгебраическое сложение – это одна из операций + или -.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***try***, ***except***, ***on***, ***do***, ***else***, ***end***.

Знаки: ;, :=, :, +, -, *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Идентификаторы классов исключений: *i*1, *i*2, *i*3.

Классы исключений: *k*1, *k*2, *k*3.

Пример правильной конструкции:

try *a*2 := *r*5; *a*5 := -*r*4; ***except*** ***on*** *i*3: *k*1 ***do*** *a*6:= *r*3+*r*4; ***on*** *i*1: *k*2 ***do*** *a*6:= *r*3-*r*4; ***else*** *a*6:= *r*3; ***end***;

Пример неправильной конструкции:

try *a*2 := *r*5; *a*5 := -*r*4; ***excepton*** *i*3: *k*1 ***do*** *a*6:= *r*3+*r*4; ***on*** i1: *k*2 ***do*** *a*6:= *r*3-*r*4; ***else*** *a*6:= *r*3; ***end***;

Вариант 6

Оператор цикла с постусловием языка программирования имеет следующую структуру:

repeat *оператор_присваивания*; *оператор_присваивания*; *оператор_присваивания*; ***until*** *логическое_выражение*;

Библиотека ГГТУ им. П.О. Сухого

Логическое выражение, входящее в оператор цикла, имеет следующую структуру:

элемент операция_отношения элемент

В качестве элемента может быть *переменная* или *число*.

Операция отношения – это одна из шести операций: =, <, >, <=, >=, <>.

Оператор присваивания имеет одну из следующих структур:

переменная **:=** *выражение*

переменная **:=** *алгебраическое_сложение выражение*

переменная **:=** *выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***repeat***, ***until***.

Знаки: **;**, **:=**, **=,** <, >, +, -, *, /, *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

repeat *a*3**:=** *r*1**-***a*2***a*5**;** *a*2**:=** *r*1+*a*2/*r*4; *a*6**:=-***a*1**; *until*** *a*2 <= *r*5;

Пример неправильной конструкции:

repeat *a*3**:=** *r*1**-***a*2***a*5**;** *a*2**:=** *r*1+*a*2/*r*4; *a*6**:=-***a*1 ***until*** *a*2 <= *r*5;

Вариант 7

Составной оператор языка программирования имеет следующую структуру:

begin *оператор_процедуры***;** *оператор_ процедуры;* ***end*;**

Оператор процедуры имеет одну из следующих структур:

имя_процедуры(параметр_процедуры)

имя_процедуры(параметр_процедуры, параметр_процедуры)

В качестве параметра процедуры может использоваться одна из следующих конструкций:

выражение

алгебраическое_сложение выражение

Библиотека ГГТУ им. П.О.Сухого

выражение алгебраическое_сложение выражение

Алгебраическое сложение – это одна из операций **+** или **-**.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*****) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***begin***, ***end***.

Знаки: **;**, **,**, ***,** /, **+**, **-**, **(**, **)**, *пробел*.

Имена процедур: *p*1, *p*2, *p*3, *p*4.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

begin *p*2(*r*5**a*1+*a*4)**;** *p*4(*a*3-*a*4/*r*3**,** *r*1**r*1) **; *end*;**

Пример неправильной конструкции:

begin *p*2(*r*5**a*1+*a*4)**;** *p*4(*a*3-*a*4/*r*3**,** *r*1**r*1) ***end*;**

Вариант 8

Описание функции языка программирования имеет следующую структуру:

function *имя_функции*(*параметр_функции***;** *параметр_функции***)** ***begin*** *оператор_присваивания***;** *оператор_присваивания***;** *оператор_присваивания***; *end*;**

Оператор присваивания имеет одну из следующих структур:

переменная **:=** *выражение*

переменная **:=** *алгебраическое_сложение выражение*

переменная **:=** *выражение алгебраическое_сложение выражение*

В левой части оператора присваивания в качестве переменной может использоваться имя функции.

Алгебраическое сложение – это одна из операций **+** или **-**.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*****) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

В качестве параметра функции может использоваться только переменная.

Библиотека ГГТУ им.П.О.Сухого

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***function***, ***begin***, ***end***.

Знаки: **;**, ***,** /, +, **-**, **(**, **)**, *пробел*.

Имена функций: *f*1, *f*2, *f*3, *f*4.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

function *f*1**(***a*2**;** *a*5**)** ***begin*** *a*3**:=** *r*1**-***a*2*****a*5**;** *a*4**:=** *r*1**+***a*2**/***r*4; *f*1**:=-***a*4**;** ***end*;**

Пример неправильной конструкции:

function *f*1(*a*2**,** *a*5**)** ***begin*** *a*3**:=** *r*1**-***a*2*****a*5**;** *a*4**:=** *r*1**+***a*2**/***r*4; *f*1**:=-***a*4**;** ***end*;**

Вариант 9

Условный оператор языка программирования имеет следующую структуру:

if *логическое_выражение* ***then*** *оператор_присваивания* ***else*** *оператор_присваивания*;

Логическое выражение, входящее в условный оператор, имеет следующую структуру:

простое_логическое_выражение логическая_операция простое_логическое_выражение

Логическая операция – это одна из операций $\wedge$ (конъюнкция) или $\vee$ (дизъюнкция).

Простое логическое выражение имеет следующую структуру:

(*элемент операция_отношения элемент*)

В качестве элемента может быть *переменная* или *число*.

Операция отношения – это одна из шести операций: =, <, >, <=, >=, <>.

Оператор присваивания имеет одну из следующих структур:

*переменная***:=***выражение*

*переменная***:=***алгебраическое_сложение выражение*

*переменная***:=***выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

Выражение – это *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Библиотека ГГТУ им.П.О.Сухого

Алфавит языка после лексического анализа:
Служебные слова: ***if***, ***then***, ***else***.
Знаки: **;**, **:=**, ∧**,** ∨**, =, <**, **>**, +, **-,** *пробел*.
Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.
Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.
Пример правильной конструкции:
if (*a*2 <= *r*5) ∧ (*a*5>*r*3) ***then*** *a*3:= *r*1-*a*2 ***else*** *a*3:= *r*1+*a*2;
Пример неправильной конструкции:
if *a*2 <= *r*5 ***then*** *a*3:= *r*1-*a*2 ***else*** *a*3:= *r*1+*a*2;

Вариант 10

Оператор цикла с предусловием языка программирования имеет следующую структуру:

while *логическое_выражение* ***do begin*** *оператор_присваивания*; *оператор_присваивания*; ***end***;

Логическое выражение, входящее в оператор цикла, имеет следующую структуру:

простое_логическое_выражение логическая_операция простое_логическое_выражение

Логическая операция – это одна из операций ∧ (конъюнкция) или ∨ (дизъюнкция).

Простое логическое выражение имеет следующую структуру:

(*элемент операция_отношения элемент*)

В качестве элемента может быть *переменная* или *число*.

Операция отношения – это одна из шести операций: =, <, >, <=, >=, <>.

Оператор присваивания имеет одну из следующих структур:

переменная:=*выражение*

переменная:=*алгебраическое_сложение выражение*

переменная:=*выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

Выражение – это *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***while***, ***do***, ***begin***, ***end***.

Знаки: **;**, **:=**, **=,** <, >, +, **-,** ∧**,** ∨**,** *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Библиотека ГГТУ им. П.О. Сухого

Пример правильной конструкции:
while (*a*2 <= *r*5) ∨ (*a*6 > *r*4) ***do begin*** *a*3:= *r*1-*a*2; *a*4:= *r*1+*a*2**; *end***;
Пример неправильной конструкции:
while *a*2 <= *r*5 ***do begin*** *a*3:= *r*1-*a*2; *a*4:= *r*1+*a*2**; *end***;

Вариант 11

Арифметический оператор цикла языка программирования имеет следующую структуру:

for *переменная* **:=** *выражение* ***until*** *выражение* ***step*** *выражение* ***do begin*** *оператор_присваивания***;** *оператор_присваивания***; *end*;**

Оператор присваивания имеет одну из следующих структур:

переменная **:=** *выражение*

переменная **:=** *алгебраическое_сложение выражение*

переменная **:=** *выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций **+** или **-**.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*****) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***for***, ***until***, ***step***, ***do***, ***begin***, ***end***.

Знаки: **;**, **:=**, ***,** /, **+**, **-,** *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

for *a*2 := *r*5**a*1 ***until*** *a*3-*a*4/*r*3 ***step*** *r*6 ***do begin*** *a*3:= *r*1-*a*2; *a*4:= *r*1**a*2**; *end***;

Пример неправильной конструкции:

for *a*2 := *r*5**a*1 ***until*** *a*3-*a*4/*r*3**a*1 ***step*** *r*6 ***do begin*** *a*3:= *r*1-*a*2; *a*4:= *r*1**a*2**; *end***;

Вариант 12

Оператор выбора языка программирования имеет следующую структуру:

case *переменная* ***of*** *число* **:** *оператор_присваивания***;** *число* **:** *оператор_присваивания***;** ***else*** *оператор_присваивания***; *end;***

Библиотека ГГТУ им. П.О. Сухого

Оператор присваивания имеет одну из следующих структур:

переменная **:=** *выражение*

переменная **:=** *алгебраическое_сложение выражение*

переменная **:=** *выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*****) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***case***, ***of***, ***else***, ***end***.

Знаки: **;**, **:=**, **:**, ***,** /, +, **-,** *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

case *a*2 ***of*** *r*1**:** *a*3**:=** *r*1**-***a*2*****a*5**;** *r*2**:** *a*3**:=** *r*1**+***a*2**/***r*4; ***else*** *a*3**:=***a*6**; *end*;**

Пример неправильной конструкции:

case *a*2 ***of*** *r*1**:** *a*3**:=** *r*1**-***a*2*****a*5**;** *r*2**:** *a*3**:=** *r*1**+***a*2**/***r*4; ***else*** *a*3**:=***a*6 ***end*;**

Вариант 13

Конструкция обработки исключительных ситуаций языка программирования имеет следующую структуру:

try *оператор_присваивания***;** *оператор_присваивания***;**

except

on *идентификатор_класса_исключения***:** *класс_исключения* ***do*** *оператор_присваивания***;**

on *идентификатор_класса_исключения***:** *класс_исключения* ***do*** *оператор_присваивания***;**

***end*;**

Оператор присваивания имеет одну из следующих структур:

*переменная***:=***выражение*

*переменная***:=***алгебраическое_сложение выражение*

*переменная***:=** *выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций **+** или **-**.

В качестве выражения может выступать *переменная* или *число*.

Библиотека ГГТУ им. П.О. Сухого

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***try***, ***except***, ***on***, ***do***, ***end***.

Знаки: **;**, **:=**, **:**, +, **-,** *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Идентификаторы классов исключений: *i*1, *i*2, *i*3.

Классы исключений: *k*1, *k*2, *k*3.

Пример правильной конструкции:

try *a*2 **:=** *a*1+ *r*5; *a*5 **:=** **-***r*4-*a*2; ***except*** ***on*** *i*3**:** *k*1 ***do*** *a*6**:=** *r*3+*a*4; ***on*** *i*1**:** *k*2 ***do*** *a*6**:=** *r*3**-***a*4; ***end*;**

Пример неправильной конструкции:

try *a*2 **:=** *r*5; *a*5 **:=** **-***r*4; ***excepton*** *i*3**:** *k*1 ***do*** *a*6**:=** *r*3+*a*4; ***on*** *i*1**:** *k*2 ***do*** *a*6**:=** *r*3**-***a*4**; *end*;**

Вариант 14

Оператор цикла с постусловием языка программирования имеет следующую структуру:

repeat *оператор_присваивания*; *оператор_присваивания*; *оператор_присваивания*; ***until*** *логическое_выражение*;

Логическое выражение, входящее в оператор цикла, имеет следующую структуру:

простое_логическое_выражение логическая_операция простое_логическое_выражение

Логическая операция – это одна из операций $\wedge$ (конъюнкция) или $\vee$ (дизъюнкция).

Простое логическое выражение имеет следующую структуру:

(*элемент операция_отношения элемент*)

В качестве элемента может быть *переменная* или *число*.

Операция отношения – это одна из шести операций: =, <, >, <=, >=, <>.

Оператор присваивания имеет одну из следующих структур:

переменная **:=** *выражение*

переменная **:=** *алгебраическое_сложение выражение*

переменная **:=** *выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

В качестве выражения может выступать *переменная* или *число*.

Библиотека ГГТУ им.П.О.Сухого

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***repeat***, ***until***.

Знаки: **;**, **:=**, **=,** <, >, +, **-**, ∧, ∨, *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

repeat *a*3**:=** *r*1**-***a*2**;** *a*2**:=** *r*1**+***a*2; *a*6**:=-***a*1**; *until*** (*a*2 <= *r*5) ∧ (*a*1 > *r*6);

Пример неправильной конструкции:

repeat *a*3**:=** *r*1**-***a*2**;** *a*2**:=** *r*1**+***a*2; *a*6**:=-***a*1 ***until*** (*a*2 <= *r*5) ∧ (*a*1 > *r*6);

Вариант 15

Составной оператор языка программирования имеет следующую структуру:

begin *оператор_процедуры***;** *оператор_ процедуры; оператор_ процедуры;* ***end*;**

Оператор процедуры имеет одну из следующих структур:

имя_процедуры()

имя_процедуры(параметр_процедуры)

имя_процедуры(параметр_процедуры, параметр_процедуры)

В качестве параметра процедуры может использоваться одна из следующих конструкций:

выражение

алгебраическое_сложение выражение

выражение алгебраическое_сложение выражение

Алгебраическое сложение – это одна из операций **+** или **-**.

Выражение – это *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***begin***, ***end***.

Знаки: **;**, **,**, +, **-**, **(**, **)**, *пробел*.

Имена процедур: *p*1, *p*2, *p*3, *p*4.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

begin *p*2(*r*5*****a*1+*a*4)**;** *p*4(*a*3-*a*4/*r*3**,** *r*1*****r*1) **;** *p*1()**;*end*;**

Библиотека ГГТУ им.П.О.Сухого

Пример неправильной конструкции:
begin *p*2(*r*5**a*1+*a*4)**;** *p*4(*a*3-*a*4/*r*3**,** *r*1**r*1) **;** *p*1() ***end*;**

Вариант 16

Описание функции языка программирования имеет следующую структуру:

function *имя_функции***(***список_параметров_функции***)** ***begin*** *оператор_присваивания***;** *оператор_присваивания***;** *оператор_присваивания***; *end*;**

Оператор присваивания имеет одну из следующих структур:

переменная **:=** *выражение*

переменная **:=** *алгебраическое_сложение выражение*

переменная **:=** *выражение алгебраическое_сложение выражение*

В левой части оператора присваивания в качестве переменной может использоваться имя функции.

Алгебраическое сложение – это одна из операций **+** или **-**.

Выражение – это *переменная* или *число*.

Список параметров функции разделяется символом «**;**». Количество параметров – от одного до трех. В качестве параметра функции может использоваться только переменная.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***function***, ***begin***, ***end***.

Знаки: **;**, **+**, **-**, **(**, **)**, *пробел*.

Имена функций: *f*1, *f*2, *f*3, *f*4.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

function *f*1**(***a*2**;** *a*5**;** *a*1**)** ***begin*** *a*3**:=** *r*1**-***a*2**-***a*5**;** *a*4**:=** *r*1**+***a*2**;** *f*1**:=-***a*4**;** ***end*;**

Пример неправильной конструкции:

function *f*1(*a*2**,** *a*5**)** ***begin*** *a*3**:=** *r*1**-***a*2**;** *a*4**:=** *r*1**+***a*2**;** *f*1**:=-***a*4**;** ***end*;**

Вариант 17

Условный оператор языка программирования имеет следующую структуру:

Библиотека ГГТУ им. П.О.Сухого

if *логическое_выражение* ***then*** *оператор_присваивания* ***else*** *оператор_присваивания*;

Логическое выражение, входящее в условный оператор, имеет следующую структуру:

элемент операция_отношения элемент

В качестве элемента может быть *переменная* или *число*.

Операция отношения – это одна из шести операций: =, <, >, <=, >=, <>.

Оператор присваивания имеет одну из следующих структур:

переменная:=*выражение*

переменная:=*алгебраическое_сложение выражение*

переменная:=*выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***if***, ***then***, ***else***.

Знаки: **;**, **:=**, **=,** <, >, +, **-, *,** /, *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

if *a*2 <= *r*5 ***then*** *a*3:= *r*1-*a*2*r5 ***else*** *a*3:= *r*1/*a*4+*a*2;

Пример неправильной конструкции:

if *a*2 <= *r*5 ***then*** *a*3:= *r*1-*a*2+*r*5 ***else*** *a*3:= *r*1/*a*4+*a*2;

Вариант 18

Оператор цикла с предусловием языка программирования имеет следующую структуру:

while *логическое_выражение* ***do begin*** *оператор_присваивания*; *оператор_присваивания*; ***end***;

Логическое выражение, входящее в оператор цикла, имеет следующую структуру:

элемент операция_отношения элемент

В качестве элемента может быть *переменная* или *число*.

Библиотека ГГТУ им. П.О. Сухого

Операция отношения – это одна из шести операций: =, <, >, <=, >=, <>.

Оператор присваивания имеет одну из следующих структур:

*переменная***:=***выражение*

*переменная***:=***алгебраическое_сложение выражение*

*переменная***:=***выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***while***, ***do***, ***begin***, ***end***.

Знаки: **;**, **:=**, **=,** <, >, +, **-, *,** /, *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

while *a*2 <= *r*5 ***do begin*** *a*3:= *r*1-*a*2*r5; *a*4:= *r*1/*a*4+*a*2**; *end***;

Пример неправильной конструкции:

while *a*2 <= *r*5 ***do begin*** *a*3:= *r*1-*a*2**; *end***;

Вариант 19

Арифметический оператор цикла языка программирования имеет следующую структуру:

for *переменная* **:=** *выражение* ***until*** *выражение* ***do begin*** *оператор_присваивания***;** *оператор_присваивания***;** *оператор_присваивания***; *end*;**

Оператор присваивания имеет одну из следующих структур:

переменная **:=** *выражение*

переменная **:=** *алгебраическое_сложение выражение*

переменная **:=** *выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Библиотека ГГТУ им. П.О. Сухого

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***for***, ***until***, ***do***, ***begin***, ***end***.

Знаки: **;**, **:=**, ***,** /, +, **-,** *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

for *a*2 := *r*5**a*1 ***until*** *a*3-*a*4/*r*3 ***do begin*** *a*3:= *r*1-*a*2; *a*4:= *r*1**a*2**;** *a*6:= *r*6**; *end***;

Пример неправильной конструкции:

for *a*2 := *r*5**a*1 ***until*** *a*3-*a*4/*r*3**a*1 ***do begin*** *a*3:= *r*1-*a*2; *a*4:= *r*1**a*2**;** *a*6:= *r*6**; *end***;

Вариант 20

Оператор выбора языка программирования имеет следующую структуру:

case *переменная* ***of*** *число* **:** *оператор_присваивания***;** *число* **:** *оператор_присваивания***;** *число* **:** *оператор_присваивания***; *end;***

Оператор присваивания имеет одну из следующих структур:

переменная **:=** *выражение*

переменная **:=** *алгебраическое_сложение выражение*

переменная **:=** *выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*****) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***case***, ***of***, ***end***.

Знаки: **;**, **:=**, **:**, ***,** /, +, **-,** *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

case *a*2 ***of*** *r*1**:** *a*3**:=** *r*1-*a*2**a*5**;** *r*2**:** *a*3**:=** *r*1+*a*2/*r*4; *r*3**:** *a*3**:=***a*6**; *end*;**

Пример неправильной конструкции:

Библиотека ГГТУ им. П.О. Сухого

case *a*2 ***of*** *r*1**:** *a*3**:=** *r*1**-***a*2***** *a*5**;** *r*2**:** *a*3**:=** *r*1**+***a*2**/***r*4; *r*3**:** *a*3**:=***a*6 ***end*****;**

Вариант 21

Конструкция обработки исключительных ситуаций языка программирования имеет следующую структуру:

try *оператор_присваивания***;** *оператор_присваивания***;**
except
on *идентификатор_класса_исключения***:** *класс_исключения* ***do*** *оператор_присваивания***;**
on *идентификатор_класса_исключения***:** *класс_исключения* ***do*** *оператор_присваивания***;**
else *оператор_присваивания***;**
end**;**

Оператор присваивания имеет одну из следующих структур:

*переменная***:=***число*
*переменная***:=***алгебраическое_сложение число*
*переменная***:=** *число алгебраическое_сложение число*

Алгебраическое сложение – это одна из операций **+** или **-**.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***try***, ***except***, ***on***, ***do***, ***else***, ***end***.

Знаки: **;**, **:=**, **:**, **+**, **-,** *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Идентификаторы классов исключений: *i*1, *i*2, *i*3.

Классы исключений: *k*1, *k*2, *k*3.

Пример правильной конструкции:

try *a*2 **:=** *r*5; *a*5 **:= -***r*4; ***except*** ***on*** *i*3**:** *k*1 ***do*** *a*6**:=** *r*3**+***r*4; ***on*** *i*1**:** *k*2 ***do*** *a*6**:=** *r*3**-***r*4; ***else*** *a*6:= *r*3; ***end*****;**

Пример неправильной конструкции:

try *a*2 **:=** *r*5; *a*5 **:= -***r*4; ***excepton*** *i*3**:** *k*1 ***do*** *a*6**:=** *r*3**+***r*4; ***on*** *i*1**:** *k*2 ***do*** *a*6**:=** *r*3**-***r*4; ***else*** *a*6:= *r*3; ***end*****;**

Вариант 22

Оператор цикла с постусловием языка программирования имеет следующую структуру:

repeat *оператор_присваивания***;** *оператор_присваивания***;** *оператор_присваивания***;** ***until*** *логическое_выражение***;**

Библиотека ГГТУ им. П.О. Сухого

Логическое выражение, входящее в оператор цикла, имеет следующую структуру:

элемент операция_отношения элемент

В качестве элемента может быть *переменная* или *число*.

Операция отношения – это одна из шести операций: =, <, >, <=, >=, <>.

Оператор присваивания имеет одну из следующих структур:

переменная **:=** *выражение*

переменная **:=** *алгебраическое_сложение выражение*

переменная **:=** *выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***repeat***, ***until***.

Знаки: **;**, **:=**, **=,** <, >, +, **-**, *, /, *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

repeat *a*3**:=** *r*1**-***a*2***a*5**;** *a*2**:=** *r*1+*a*2/*r*4; *a*6**:=-***a*1**; *until*** *a*2 <= *r*5;

Пример неправильной конструкции:

repeat *a*3**:=** *r*1**-***a*2***a*5**;** *a*2**:=** *r*1+*a*2/*r*4; *a*6**:=-***a*1 ***until*** *a*2 <= *r*5;

Вариант 23

Составной оператор языка программирования имеет следующую структуру:

begin *оператор_процедуры***;** *оператор_ процедуры; оператор_ процедуры;* ***end*;**

Оператор процедуры имеет одну из следующих структур:

имя_процедуры(параметр_процедуры)

имя_процедуры(параметр_процедуры, параметр_процедуры)

В качестве параметра процедуры может использоваться одна из следующих конструкций:

выражение

Библиотека ГГТУ им. П.О.Сухого

алгебраическое_сложение выражение

выражение алгебраическое_сложение выражение

Алгебраическое сложение – это одна из операций **+** или **-**.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*****) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***begin***, ***end***.

Знаки: **;**, **,**, ***,** /, **+**, **-**, **(**, **)**, *пробел*.

Имена процедур: *p*1, *p*2, *p*3, *p*4.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

begin *p*2(*r*5**a*1+*a*4)**;** *p*4(*a*3-*a*4/*r*3**,** *r*1**r*1) **;** *p*3(*r*2-*r*1) **;*end*;**

Пример неправильной конструкции:

begin *p*2(*r*5**a*1+*a*4)**;** *p*4(*a*3-*a*4/*r*3**,** *r*1**r*1) ***end*;**

Вариант 24

Описание функции языка программирования имеет следующую структуру:

function *имя_функции*(*параметр_функции*; *параметр_функции*) ***begin*** *оператор_присваивания*; *оператор_присваивания*; *оператор_присваивания*; ***end*;**

Оператор присваивания имеет одну из следующих структур:

переменная **:=** *выражение*

переменная **:=** *алгебраическое_сложение выражение*

переменная **:=** *выражение алгебраическое_сложение выражение*

В левой части оператора присваивания в качестве переменной может использоваться имя функции.

Алгебраическое сложение – это одна из операций **+** или **-**.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*****) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Библиотека ГГТУ им. П.О. Сухого

В качестве параметра функции может использоваться только переменная.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***function***, ***begin***, ***end***.

Знаки: **;**, ***,** /, +, **-**, **(**, **)**, *пробел*.

Имена функций: *f*1, *f*2, *f*3, *f*4.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

function *f*1**(***a*2**;** *a*5**)** ***begin*** *a*3**:=** *r*1**-***a*2*****a*5**;** *a*4**:=** *r*1**+***a*2**/***r*4; *f*1**:=-***a*4**;** ***end*;**

Пример неправильной конструкции:

function *f*1**(***a*2**,** *a*5**)** ***begin*** *a*3**:=** *r*1**-***a*2*****a*5**;** *a*4**:=** *r*1**+***a*2**/***r*4; *f*1**:=-***a*4**;** ***end*;**

Вариант 25

Условный оператор языка программирования имеет следующую структуру:

if *логическое_выражение* ***then*** *оператор_присваивания* ***else*** *оператор_присваивания*;

Логическое выражение, входящее в условный оператор, имеет следующую структуру:

простое_логическое_выражение логическая_операция простое_логическое_выражение

Логическая операция – это одна из операций $\wedge$ (конъюнкция) или $\vee$ (дизъюнкция).

Простое логическое выражение имеет следующую структуру:

(*элемент операция_отношения элемент*)

В качестве элемента может быть *переменная* или *число*.

Операция отношения – это одна из шести операций: =, <, >, <=, >=, <>.

Оператор присваивания имеет одну из следующих структур:

*переменная***:=***выражение*

*переменная***:=***алгебраическое_сложение выражение*

*переменная***:=***выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

Выражение – это *переменная* или *число*.

Библиотека ГГТУ им. П.О.Сухого

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***if***, ***then***, ***else***.

Знаки: **;**, **:=**, ∧**,** ∨**, =,** <, >, +, **-,** *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

if (*a*2 <= *r*5) ∧ (*a*5>*r*3) ***then*** *a*3:= *r*1-*a*2 ***else*** *a*3:= *r*1+*a*2;

Пример неправильной конструкции:

if *a*2 <= *r*5 ***then*** *a*3:= *r*1-*a*2 ***else*** *a*3:= *r*1+*a*2;

Вариант 26

Оператор цикла с предусловием языка программирования имеет следующую структуру:

while *логическое_выражение* ***do begin*** *оператор_присваивания***;** *оператор_присваивания***; end;**

Логическое выражение, входящее в оператор цикла, имеет следующую структуру:

простое_логическое_выражение логическая_операция простое_логическое_выражение

Логическая операция – это одна из операций ∧ (конъюнкция) или ∨ (дизъюнкция).

Простое логическое выражение имеет следующую структуру:

(*элемент операция_отношения элемент*)

В качестве элемента может быть *переменная* или *число*.

Операция отношения – это одна из шести операций: =, <, >, <=, >=, <>.

Оператор присваивания имеет одну из следующих структур:

*переменная***:=***выражение*

*переменная***:=***алгебраическое_сложение выражение*

*переменная***:=***выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

Выражение – это *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***while***, ***do***, ***begin***, ***end***.

Знаки: **;**, **:=**, **=,** <, >, +, **-,** ∧**,** ∨**,** *пробел*.

Библиотека ГГТУ им.П.О.Сухого

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

while (*a*2 <= *r*5) ∨ (*a*6 > *r*4) ***do begin*** *a*3:= *r*1-*a*2; *a*4:= *r*1+*a*2**; *end***;

Пример неправильной конструкции:

while *a*2 <= *r*5 ***do begin*** *a*3:= *r*1-*a*2; *a*4:= *r*1+*a*2**; *end***;

Вариант 27

Арифметический оператор цикла языка программирования имеет следующую структуру:

for *переменная* **:=** *выражение* ***until*** *выражение* ***step*** *выражение* ***do begin*** *оператор_присваивания*; *оператор_присваивания*; ***end***;

Оператор присваивания имеет одну из следующих структур:

переменная **:=** *выражение*

переменная **:=** *алгебраическое_сложение выражение*

переменная **:=** *выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций **+** или **-**.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*****) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***for***, ***until***, ***step***, ***do***, ***begin***, ***end***.

Знаки: **;**, **:=**, ***,** /, +, **-,** *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

for *a*2 := *r*5**a*1 ***until*** *a*3-*a*4/*r*3 ***step*** *r*6 ***do begin*** *a*3:= *r*1-*a*2; *a*4:= *r*1**a*2**; *end***;

Пример неправильной конструкции:

for *a*2 := *r*5**a*1 ***until*** *a*3-*a*4/*r*3**a*1 ***step*** *r*6 ***do begin*** *a*3:= *r*1-*a*2; *a*4:= *r*1**a*2**; *end***;

Вариант 28

Оператор выбора языка программирования имеет следующую структуру:

Библиотека ГГТУ им. П.О. Сухого

case *переменная* ***of*** *число* **:** *оператор_присваивания***;** *число* **:** *оператор_присваивания***; *else*** *оператор_присваивания***; *end;***

Оператор присваивания имеет одну из следующих структур:

переменная **:=** *выражение*

переменная **:=** *алгебраическое_сложение выражение*

переменная **:=** *выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

Выражение – это единственный *множитель* или два *множителя*, между которыми стоит *операция умножения* (*) или *деления* (/).

В качестве множителя может выступать *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***case***, ***of***, ***else***, ***end***.

Знаки: **;**, **:=**, **:**, ***,** /, +, **-,** *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

case *a*2 ***of*** *r*1**:** *a*3**:=** *r*1-*a*2***a*5**;** *r*2**:** *a*3**:=** *r*1+*a*2/*r*4; ***else*** *a*3**:=***a*6**; *end*;**

Пример неправильной конструкции:

case *a*2 ***of*** *r*1**:** *a*3**:=** *r*1-*a*2***a*5**;** *r*2**:** *a*3**:=** *r*1+*a*2/*r*4; ***else*** *a*3**:=***a*6 ***end*;**

Вариант 29

Конструкция обработки исключительных ситуаций языка программирования имеет следующую структуру:

try *оператор_присваивания***;** *оператор_присваивания***;**

except

on *идентификатор_класса_исключения***:** *класс_исключения* ***do*** *оператор_присваивания***;**

on *идентификатор_класса_исключения***:** *класс_исключения* ***do*** *оператор_присваивания***;**

***end*;**

Оператор присваивания имеет одну из следующих структур:

*переменная***:=***выражение*

*переменная***:=***алгебраическое_сложение выражение*

*переменная***:=** *выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

Библиотека ГГТУ им. П.О.Сухого

В качестве выражения может выступать *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***try***, ***except***, ***on***, ***do***, ***end***.

Знаки: **;**, **:=**, **:**, +, **-,** *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Идентификаторы классов исключений: *i*1, *i*2, *i*3.

Классы исключений: *k*1, *k*2, *k*3.

Пример правильной конструкции:

try *a*2 **:=** *a*1+ *r*5; *a*5 **:=** **-***r*4-*a*2; ***except*** ***on*** *i*3**:** *k*1 ***do*** *a*6**:=** *r*3+*a*4; ***on*** *i*1**:** *k*2 ***do*** *a*6**:=** *r*3**-***a*4; ***end*;**

Пример неправильной конструкции:

try *a*2 **:=** *r*5; *a*5 **:=** **-***r*4; ***excepton*** *i*3**:** *k*1 ***do*** *a*6**:=** *r*3+*a*4; ***on*** *i*1**:** *k*2 ***do*** *a*6**:=** *r*3**-***a*4; ***end*;**

Вариант 30

Оператор цикла с постусловием языка программирования имеет следующую структуру:

repeat *оператор_присваивания***;** *оператор_присваивания***;** *оператор_присваивания***;** ***until*** *логическое_выражение***;**

Логическое выражение, входящее в оператор цикла, имеет следующую структуру:

простое_логическое_выражение логическая_операция простое_логическое_выражение

Логическая операция – это одна из операций $\wedge$ (конъюнкция) или $\vee$ (дизъюнкция).

Простое логическое выражение имеет следующую структуру:

(*элемент операция_отношения элемент*)

В качестве элемента может быть *переменная* или *число*.

Операция отношения – это одна из шести операций: =, <, >, <=, >=, <>.

Оператор присваивания имеет одну из следующих структур:

переменная **:=** *выражение*

переменная **:=** *алгебраическое_сложение выражение*

переменная **:=** *выражение алгебраическое_сложение выражение*

Алгебраическое сложение – это одна из операций + или -.

Библиотека ГГТУ им. П.О.Сухого

В качестве выражения может выступать *переменная* или *число*.

Пробелы могут стоять в конструкции в любом месте. Каждое служебное слово должно быть выделено пробелами.

Алфавит языка после лексического анализа:

Служебные слова: ***repeat***, ***until***.

Знаки: **;**, **:=**, **=**, **<**, **>**, +, **-**, $\wedge$, $\vee$, *пробел*.

Переменные: *a*1, *a*2, *a*3, *a*4, *a*5, *a*6.

Числа: *r*1, *r*2, *r*3, *r*4, *r*5, *r*6.

Пример правильной конструкции:

repeat *a*3**:=** *r*1**-***a*2**;** *a*2**:=** *r*1**+***a*2; *a*6**:=-***a*1**; *until*** (*a*2 <= *r*5) $\wedge$ (*a*1 > *r*6);

Пример неправильной конструкции:

repeat *a*3**:=** *r*1**-***a*2**;** *a*2**:=** *r*1**+***a*2; *a*6**:=-***a*1 ***until*** (*a*2 <= *r*5) $\wedge$ (*a*1 > *r*6);

7.5 Структура отчета по лабораторной работе

Отчет по лабораторной работе должен содержать следующие элементы:

- цель работы;
- задание;
- описание ДКА с представлением функции переходов в виде графа и табличной форме;
- описание структуры данных и алгоритма решения задачи;
- скриншоты результатов выполнения программы;
- рационально прокомментированный текст программы;
- вывод о правильности построенного автомата.

Библиотека ГГТУ им. П.О.Сухого

Кравченко Ольга Алексеевна

МЕТОДЫ ТРАНСЛЯЦИИ

Практикум
по выполнению лабораторных работ для студентов специальности 1-40 04 01 «Информатика и технологии программирования» дневной и заочной форм обучения

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 20.01.21.

Рег. № 10Е.
http://www.gstu.by

Библиотека ГГТУ им. П.О.Сухого