|  |  |
| --- | --- |
| **Изображение выглядит как текст, эмблема, герб, нашивка  Автоматически созданное описание** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Робототехника и комплексная автоматизация (РК)

КАФЕДРА Системы автоматизированного проектирования (РК-6)

**ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**по курсу «Программное обеспечение систем автоматизированного проектирования»**

Студент Долженко Анастасия Тимофеевна

Группа РК6-32Б

Тип задания Лабораторная работа №6

Вариант T23

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Долженко А.Т.**

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Волосатова Т.М.**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2023 г**.*

**Задание к лабораторной работе**

**Постановка задачи**

Разработать транслятор для преобразования записей линейных алгебраических неравенств (ЛАН) с целочисленными коэффициентами в эквивалентный формат линейных алгебраических уравнений (ЛАУ). При этом все переменные ЛАН и ЛАУ должны быть заданы одинаковыми латинскими буквами с различными числовыми индексами. Входные записи ЛАН должны передаваться транслятору строками стандартного ввода. Результаты их трансляции в ЛАУ должны отображать строки стандартного вывода.

**Требования к программе**

1. Требования к функциональным характеристикам

1.1. Программа должна в интерактивном режиме распознавать и преобразовывать записи ЛАН в эквивалентный формат ЛАУ.

1.2. ЛАН должны передаваться строками стандартного ввода без разделителей и иных символов, кроме знаков ‘-‘, ’+’, ‘(’, ’)’, ‘>’, ‘<’, ‘>=’, ‘<=’, ‘!=’ описания переменных и десятичных цифр.

1.3. Программа обнаруживает соответствие или несоответствие символьной

2. Условия эксплуатации

2.1. Программа должна быть ориентирована на эксплуатацию в среде OS UNIX.

2.2. Программа должна эксплуатироваться в интерактивном режиме, читая строки из потока стандартного ввода и отображая результаты их обработки в потоке стандартного вывода.

3. Требования к информационной и программной совместимости

3.1. Чтобы обеспечить выполнение требуемых технических характеристик, программа должна реализовывать синтаксический и лексический анализ любой входной строки, содержащей символьные наборы, из потока стандартного ввода.

3.2. Для разработки синтаксического анализатора программы необходимо использовать генератор синтаксических анализаторов (далее по тексту - YACC) из состава OS UNIX, инструментальные средства которого ориентированы на обработку файла спецификаций (далее по тексту Yacc-файл) проектируемого синтаксического анализатора.

3.3. Проектируемый Yacc-файл LDE.y должен состоять из 3-х секций: секция деклараций, секция правил и секция функций. Разделителем секций должна быть символическая пара %%.

3.4. Секция деклараций Yacc-файла LDE.y должна включать: перечисление терминалов грамматики ЛДУ с помощью директивы %token, cпецификацию блока внешних описаний, ограниченную директивами %{ и %}, в котором необходимо определить функцию yywrap() , а также подключить библиотеку ввода-вывода stdio.h.

3.5. В секции правил Yacc-файла LDE.y должны быть приведены описания продукций приведения нетерминалов грамматики ЛДУ.

3.6. Каждая продукция секции правил Yacc-файла LDE.y должна быть задана в нотации, близкой к форме Бэкуса-Наура, где в левой части указывается приводимый нетерминал, а в правой - последовательность терминалов и нетерминалов грамматики ЛДУ, которые перечисляются через пробел. В частном случае правая часть может быть пустой, если необходимо построить пустую продукцию. Для разделения частей продукции должен использоваться символ двоеточия (:). Каждую продукцию нужно начинать с новой строки и завершать либо символом точки с запятой (;), либо блоком действий в фигурных скобках.

3.7. Продукции секции правил Yacc-файла LDE.y, приведение нетерминалов которых необходимо сопровождать функциональной обработкой, должны содержать блоки действий. Блоки действий должны располагаться в правых частях продукций и ограничиваться парой фигурных скобок.

3.8. Блок действия альтернативной продукции приведения начального нетерминала input через терминал error может содержать макрос yyerrok для сброса состояния ошибки.

3.9. Вызов синтаксического анализатора должен осуществляться путем обращения к функции yyparse() в основной функции main() программы.

3.10. Функция yyparse() должна возврашать в основную функцию main() программы код 0 при успехе грамматического разбора каждой входной строки потока стандартного ввода и 1 - при обнаружении синтаксических ошибок.

**Стадии и этапы разработки**

При разработке лексического анализатора программы lab5 необходимо выполнить следующие этапы:

* получить исходный код синтаксического анализатора в файле y.tab.c и заголовочный файл макроопределений типов лексем для лексического анализатора в файле y.tab.h, обработав Yacc-файл lab5\_6.y командой yacc, следующим образом: $ yacc -d lab5\_6.y
* получить исходный код лексического анализатора в файле lex.yy.c, обработав Lex-файл lab5\_6.l командой lex следующим образом: $ lex lab5\_6.l
* Для сборки выполняемого файла lab5 из объектных модулей lex.yy.o, y.tab.o, lab5.o, полученных на предыдущих стадиях, необходимо применить редактор связей OS UNIX, реализовав его вызов следующей командой:  
  $ cc y.tab.с lex.yy.с

**Результаты тестирования**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Содержимое файла спецификации yacc**

%{

#include <stdio.h>

#include <stdlib.h>

int yylex();

void yyerror();

%}

%token DIGIT VAR SIGN LBRACE RBRACE PN\_TOKEN

%start input

%union{

int digit;

char var\_;

char pn\_;

char lbrace;

char rbrace;

}

%%

input : ; | input line ;

line: exp sign exp '\n' {printf("\n");}

| error '\n' {yyerrok;}

|'\n' {printf("Empty line.\n");};

exp: l\_brace exp r\_brace

| exp pn exp

| pn num var

| num var

| num

| var

l\_brace: pn num LBRACE {printf("%c", $<lbrace>3);}

| num LBRACE {printf("%c", $<lbrace>2);}

| LBRACE {printf("%c", $<lbrace>1);};

r\_brace: RBRACE {printf("%c", $<rbrace>1);};

pn: PN\_TOKEN {printf("%c", $<pn\_>1);};

num : DIGIT {printf("%d", $<digit>1);};

var: VAR DIGIT {printf("%c%d", $<var\_>1, ++$<digit>2); };

sign: SIGN {printf("=");}

%%

void yyerror() { printf("Incorrect\n"); return; }

int main() { yyparse(); }

**Источники**

1. Родионов C.В., Волосатова Т.М. “Автоматизация проектирования лексических анализаторов”
2. Тихомиров В.П., Давидов М.И. Операционная система ДЕМОС: инструментальные средства программирования, М.: Финансы и статистика, 1988
3. База и Генератор Образовательных Ресурсов (bigor.bmstu.ru)
4. Генератор программ синтаксического анализа yacc. Производственно- внедренческий кооператив "И Н Т Е Р Ф Е Й С": Демос/P 2.1, 1988