|  |  |
| --- | --- |
|  | **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 дисциплине: «Программное обеспечение систем автоматизированного проектирования»

Студент Камалов Антон Павлович

Группа РК6-35Б

Тип задания Лабораторная работа №6

Вариант T23 (YACC)

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Камалов А.П.**

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Родионов С.В.**

*подпись, дата фамилия, и.о.*

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

*Москва, 2023 г.*

**Оглавление**

[Задание 3](#_Toc1)

[Введение 3](#_Toc2)

[Основания для разработки 3](#_Toc3)

[Назначение разработки 3](#_Toc4)

[Требования к программе 4](#_Toc5)

[1. Требования к функциональным характеристикам 4](#_Toc6)

[2. Требования к надежности 4](#_Toc7)

[3. Требования к составу и параметрам технических средств 4](#_Toc8)

[4. Условия эксплуатации 4](#_Toc9)

[5. Требования к информационной и программной совместимости 5](#_Toc10)

[Стадии и этапы разработки 7](#_Toc11)

[Порядок контроля и приема 8](#_Toc12)

[Список литературы 8](#_Toc13)

[Приложение №1. Результаты разработки 9](#_Toc14)

[Приложение №2. Содержимое файла спецификации 10](#_Toc15)

# Задание

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.

# Введение

Настоящий документ определяет техническое задание на разработку синтаксического анализатора транслятора (далее по тексту - программа T23\_Y) для распознания заданных двоичных наборов. Программа T23\_Y разрабатывается в рамках лабораторной работы по курсу «Программное обеспечение систем автоматизированного проектирования» для практического изучения этапов лексического и синтаксического анализа в процедурах трансляции формальных языков.

# Основания для разработки

Программа T23\_Y разрабатывается в рамках лабораторной работы по курсу «Программное обеспечение систем автоматизированного проектирования» для практического изучения этапов лексического и синтаксического анализа в процедурах трансляции формальных языков.

# Назначение разработки

Программа T23\_Y предназначается для для преобразования записей ЛАН с целочисленными коэффициентами в эквивалентный формат ЛАУ.

# Требования к программе

## Требования к функциональным характеристикам

1.1. Программа T23\_Y должна в интерактивном режиме распознавать и преобразовывать ЛАН в ЛАУ.

1.2. Уравнение на вывод должно представляться в эквивалентном виде.

1.3. Неравенство должно передаваться строками стандартного ввода без разделителей или других символов, кроме знаков ‘+’ и ‘-‘, описания переменных и десятичных чисел. При этом все переменные неравенства должны быть заданы одинаковыми латинскими буквами с различными числовыми индексами.

1.4. Программа T23\_Y обнаруживает соответствие или несоответствие ЛАН заданным правилам и ошибку ввода.

1.5. При нарушении ввода или несоответствии последовательности заданным правилам программа T23\_Y должна отображать диагностическое сообщение «WRONG» в потоке стандартного вывода.

1.6. Результат распознавания программой T23\_Y неравенства, соответствующего заданным правилам, должен сопровождаться отображением эквивалентного неравенства.

## Требования к надежности

Программа T23\_Y не должна иметь каких-либо ограничений по числу символов в анализируемой строке, кроме внутренних ограничений инструментальных средств, использованных для её реализации.

## Требования к составу и параметрам технических средств

Программа T23\_Y должна быть разработана исходя из возможности реализации на стандартном составе технических средств компьютеров любой архитектуры после соответствующей трансляции исходного кода.

## Условия эксплуатации

4.1. Программа T23\_Y должна быть ориентирована на эксплуатацию в среде OS UNIX.

4.2. Программа T23\_Y должна быть реализована в виде выполняемого файла с именем T23\_Y, по которому она должна вызываться средствами любого командного процессора OS UNIX.

4.3. Программа T23\_Y должна эксплуатироваться в интерактивном режиме, читая строки из потока стандартного ввода и отображая результаты их обработки в потоке стандартного вывода.

## Требования к информационной и программной совместимости

5.1. Синтаксический анализатор программы T23\_Y должен обеспечивать трансляцию потока лексем от лексического анализатора в необходимый формат.

5.2. Для выполнения трансляции разбора синтаксический анализатор программы T23\_Y должен реализовывать однозначную контекстно-свободную грамматику, которая ориентирована на построчную обработку ЛАН из потока стандартного ввода и далее по тексту называется грамматикой ЛАН.

5.3. Грамматику ЛАН синтаксического анализатора программы T23\_Y должны составлять следующие элементы: начальный нетерминал (input), нетерминалы ineq, result, exp, терминалы (лексемы, выделенные лексическим анализатором), система продукций (правил вывода), которые должны обеспечивать вывод входной строки стандартного ввода путем приведения терминалов и нетерминалов к аксиоме грамматики.

5.4. Приведение потока лексем к начальному нетерминалу с помощью продукций грамматики ЛАН синтаксического анализатора программы должно означать успешное завершение грамматического разбора, т.е. распознавание корректного ЛАН. Альтернативный результат должен рассматриваться как синтаксическая ошибка формата введенного ЛАН.

5.5. Нетерминалы должны определяться продукциями грамматики ЛАН синтаксического анализатора программы T23\_Y следующим образом:

• input: начальный символ грамматики;

• line: возможные варианты ввода;

• ineq: возможные варианты ввода;

• exp: левая часть ЛАН

5.6. Начальный нетерминал input должен определяться продукциями грамматики ЛАН синтаксического анализатора программы T23\_Y как входная строка, где входная строка может соответствовать нетерминалу line или терминалу error. Приведение по 1-й альтернативе (input line) означает успех грамматического разбора.

5.7. Для разработки синтаксического анализатора программы T23\_Y, 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.

5.8. При разработке синтаксического анализатора программы T23\_Y необходимо составить Yacc-файл, отражающий специфику грамматического разбора ЛАН, и сохранить его под именем t23.y в выбранном рабочем каталоге файловой системы OS UNIX.

5.9. Проектируемый Yacc-файл t23.y должен состоять из 3-х секций: секция деклараций, секция правил и секция функций. Разделителем секций должна быть символическая пара %%.

5.10. Секция деклараций Yacc-файла t23.y должна включать:

• перечисление терминалов грамматики ЛАН 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.

5.11. В секции правил Yacc-файла t23.y должны быть приведены описания продукций приведения нетерминалов грамматики ЛАН.

5.12. Каждая продукция секции правил Yacc-файла t23.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 ЛАН, 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.

5.13. Альтернативные продукции (с одинаковыми нетерминалами в левых частях правил), необходимые для приведения нетерминалов, должны быть объединены в секции правил Yacc-файла t23.y с помощью оператора ИЛИ, который обозначается символом вертикальной черты (|). Например, альтернативную продукцию приведения нетерминала input рекомендуется реализовать путем сочетания пустого правила и правила, определяющего нетерминал input через нетерминал line, следующим образом:

input:

| input line;

Пустая альтернатива в этой продукции указывает на необязательность наличия в нетерминале input последовательности лексем expr.

5.14. Продукции секции правил Yacc-файла t23.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.

5.15. Внутри блока действий продукции приведения ЛАН к ЛАУ необходимо использовать функцию форматного вывода printf и псевдопеременные $1..$n, содержащие семантические значения терминалов, для вывода строки в требуемом формате.

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

5.17. Вызов синтаксического анализатора должен осуществляться путем обращения к функции yyparse() в основной функции main() программы T23\_Y.

5.18. Функция yyparse() должна возврашать в основную функцию main() программы T23\_Y код 0 при успехе грамматического разбора каждой входной строки потока стандартного ввода и 1 - при обнаружении синтаксических ошибок.

# Стадии и этапы разработки

Процесс разработки программы T23 целесообразно разделить на следующие 3 стадии:

* составить Yacc-файл в выбранном рабочем каталоге файловой системы OS

UNIX, используя любой текстовый редактор;

* получить исходный код синтаксического анализатора в файле y.tab.c текущего каталога файловой системы OS UNIX, обработав Yacc-файл командой yacc следующим образом:

**$ yacc t23.y**

* сформировать выполняемый модуль в файле текущего каталога файловой системы OS UNIX, компилируя исходный код синтаксического анализатора следующей командой:

**$ cc y.tab.c -o T23\_Y**

Результаты разработки программы T23\_Y должны содержать описание грамматики и файл спецификаций для генератора синтаксических анализаторов YACC.

# Порядок контроля и приема

1. Для проверки функционирования программы T23\_Y должны быть предложены контрольные примеры, предусматривающие стандартный ввод корректных и некорректных неравенств.
2. Для приёмки программы T23\_Y должен быть организован вызов выполняемого файла T23\_Y в консольном режиме работы OS UNIX.

# Список литературы

1. Родионов C.В., Волосатова Т.М. Автоматизация проектирования лексических анализаторов. Москва: Издательство МГТУ им. Н. Э. Баумана, 2005.

2. Волосатова Т.М., Родионов С.В. Лингвистическое обеспечение САПР. Режим доступа: http://bigor.bmstu.ru/?cnt/?doc=LO-SAPR/base.cou

## Приложение №1. Результаты разработки

**Описание структуры программы**

Секция объявлений:

* stdio.h - для работы с функциями стандартного ввода/вывода;
* free\_coef – для хранения свободного коэффициента;
* var\_index, char\_var, – массивы, хранящие индексы и символы переменных, а также коэффициенты при переменных;
* len\_var\_index, len\_char\_var, len\_a – длины массивов var\_index, char\_var и a соответственно;
* token X SIGN GLT COEF - терминалы, обозначающие переменные, знаки ‘+’ или ‘-‘, знаки ‘>’, ‘<’, ‘>=’, ‘<=’ и свободные коэффициенты.

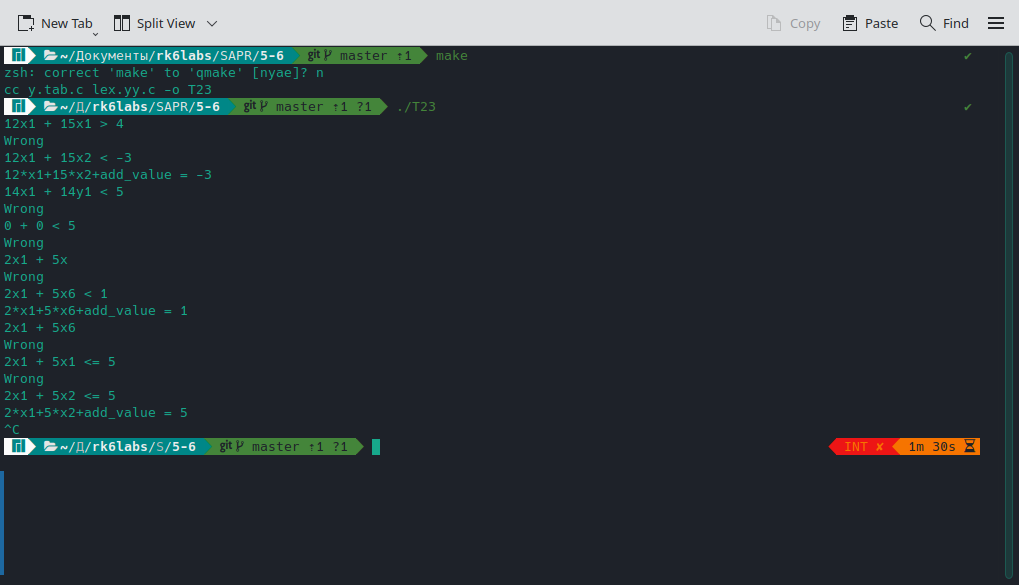
Секция правил:

* input - описывает входную строку вместе с приглашением ко вводу;
* line - содержит различные варианты входной строки и действия при этих вариантах. Для строки, не соответствующей правилу, будет сброшен флаг ошибки, чтобы программа предложила повторный ввод. Для верной строки будет выведено соответствующее сообщение;
* ineq – описывает корректный вид неравенства;
* exp - правило, рекурсивно описывающее требуемую левую часть ЛАН.

Секция программ:

* int yylex() - функция, реализующая лексический анализатор. Возвращает номер нетерминала, если он является нужной буквой, или символ в другом случае;
* int yyerror() - функция, вызываемая при неправильном вводе. Выводит на экран сообщение об ошибке;
* int main() - основная функция программы. Вызывает функцию yyparse(), которая реализует синтаксический анализатор;
* int check\_letters() – функция, проверяющая нахождение в неравенстве переменной с символом, отличающимся от остальных;
* int check\_indexes() – функция, проверяющая нахождение в неравенстве переменной с символом, индекс которого повторяется;
* void out() – функция, выводящая левую часть ЛАУ;

**Результаты лексического и синтаксического анализа**



## Приложение №2. Содержимое файла спецификации

%{

#include <stdio.h>

int free\_coef = 0;

int yyerror();

int check\_indexes();

int check\_letters();

void out();

extern int yylex();

int var\_index[512];

int len\_var\_index = 0;

char char\_var[512];

int len\_char\_var = 0;

int a[512];

int len\_a = 0;

%}

%union{

char cval;

int ival;

struct {

char letter;

int index;

} var;

}

%token <var> X

%token <cval>SIGN

%token GLT

%token <ival>COEF

%%

input:

| input line

;

line : ineq '\n' {

if (check\_letters() && check\_indexes()){

out();

printf("+add\_value = %d\n",free\_coef);

}

//else {printf("%d %d ", check\_letters(), check\_indexes()); }

else yyerror();

len\_a = 0;

len\_char\_var = 0;

len\_var\_index = 0;

free\_coef = 0;

}

| error '\n' {

yyerrok;

len\_a = 0;

len\_char\_var = 0;

len\_var\_index = 0;

free\_coef = 0;

}

;

ineq: exp GLT COEF {

free\_coef +=$3;

}

| exp GLT SIGN COEF{

if ($3 == '-') free\_coef -= $4;

else free\_coef += $4;

}

;

exp: exp SIGN COEF {

free\_coef = $2 == '+' ? free\_coef - $3: free\_coef + $3;

}

| exp SIGN COEF X {

char\_var[len\_char\_var++] = $4.letter;

var\_index[len\_var\_index++] = $4.index;

if ($2 == '-') a[len\_a++] = -$3;

else {a[len\_a++] = $3;}

}

| exp SIGN X {

char\_var[len\_char\_var++] = $3.letter;

var\_index[len\_var\_index++] = $3.index;

if ($2 == '-') a[len\_a++] = -1;

else {a[len\_a++] = 1;}

}

| SIGN COEF {

free\_coef = $1 == '+' ? free\_coef - $2: free\_coef + $2;

}

| SIGN COEF X {

char\_var[len\_char\_var++] = $3.letter;

var\_index[len\_var\_index++] = $3.index;

if ($1 == '-') a[len\_a++] = -$2;

else {a[len\_a++] = $2;}

}

| SIGN X {

char\_var[len\_char\_var++] = $2.letter;

var\_index[len\_var\_index++] = $2.index;

if ($1 == '-') a[len\_a++] = -1;

else {a[len\_a++] = 1;}

}

| COEF {

free\_coef -=$1;

}

| COEF X {

char\_var[len\_char\_var++] = $2.letter;

var\_index[len\_var\_index++] = $2.index;

a[len\_a++] = $1;

}

| X {

char\_var[len\_char\_var++] = $1.letter;

var\_index[len\_var\_index++] = $1.index;

a[len\_a++] = 1;

}

;

%%

int check\_letters(){

if (len\_char\_var < 2) return 1;

char start = char\_var[0];

for (int i = 1; i < len\_char\_var; i++){

if (char\_var[i] != start) return 0;

}

return 1;

}

int check\_indexes(){

if (len\_var\_index < 2) return 1;

for (int i = 0; i < len\_var\_index; i++){

for (int j = i+1; j < len\_var\_index; j++){

if (var\_index[i] == var\_index[j]) return 0;

}

}

return 1;

}

int main() {

return yyparse();

}

void out(){

for (int i = 0; i < len\_char\_var; i++){

if (i == 0) printf("%d\*%c%d", a[i], char\_var[i], var\_index[i]);

else if (a[i] < 0) printf("%d\*%c%d", a[i], char\_var[i], var\_index[i]);

else printf("+%d\*%c%d", a[i], char\_var[i], var\_index[i]);

}

}

int yyerror() {

printf("Wrong\n");

return 0;

}

int yywrap() {

return 1;

}