Министерство образования и науки 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)

**Отчёт**

**по лабораторной работе №3 по дисциплине**

**«Программное обеспечение САПР»**

**Вариант Y5**

Выполнил: студент группы РК6-32Б:Журавлев Н.В.

Проверил: доцент, к.т.н., Волосатова Т.М.

Москва, 2020 г.

**Техническое задание**

## Основания для разработки

Программа разрабатывается в рамках лабораторной работы по курсу "*Лингвистическое обеспечение САПР*" для практического изучения этапов лексического и синтаксического анализа в процедурах трансляции формальных языков.

## Назначение разработки

Программа предназначается для реализации грамматического разбора, заключающегося в распознавании необходимых двоичных наборов чисел, в системах и модулях обработки двоичных последовательностей и бинарных файлов .

## Требования к программе

1. ***Требования к функциональным характеристикам***
   1. Программа должна в интерактивном режиме распознавать записей двоичных наборов, которые соответствуют формальному языку L={0n1m | n, m>0}.
   2. Двоичные наборы должны передаваться строками стандартного.
   3. Программа обнаруживает соответствие или несоответствие бинарной последовательности заданным правилам, и ошибки ввода
   4. При нарушении ввода или несоответствии последовательности заданным правилам, программа должна отображать диагностическое сообщение “syntax error” в потоке стандартного вывода.
   5. Результат распознавания программой бинарной последовательности, соответствующей заданным правилам, должен сопровождаться отображением в потоке стандартного вывода информационным сообщением "Correct".

## Требования к надежности

Программа не должна иметь каких-либо ограничений по числу символов в анализируемой бинарной последовательности, кроме внутренних ограничений инструментальных средств, использованных для ее реализации.

## Требования к составу и параметрам технических средств

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

## Условия эксплуатации

* 1. Программа должна быть ориентирована на эксплуатацию в среде OS UNIX.
  2. Программа должна быть реализована в виде выполняемого файла с именем prog, по которому она должна вызываться средствами любого командного процессора OS UNIX.
  3. Программа 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

* 1. Чтобы обеспечить выполнение требуемых технических характеристик, программа должна реализовывать синтаксический анализ любой входной строки скобочного выражения из потока стандартного ввода.
  2. Синтаксический анализатор программы должен обеспечивать грамматический разбор двоичных наборов с целью установить соответствие или несоответствие содержащей их строки потока стандартного ввода заданным правилам.
  3. Для выполнения грамматического разбора синтаксический анализатор программы должен реализовывать однозначную контекстно-свободную грамматику, которая ориентирована на обработку строки заданной бинарной последовательности из потока стандартного ввода, и далее по тексту называется грамматикой БП.
  4. Грамматику БП синтаксического анализатора программы должны составлять следующие элементы:
* терминальные символы (терминалы), соответствующие структурным единицам (лексемам) входного выражения;
* начальный нетерминальный символ (начальный нетерминал), к которому приводится входное выражение;
* нетерминальные символы (нетерминалы), обозначающие допустимые варианты комбинации лексем во входном скобочном выражении;
* система продукций (правил вывода), обеспечивающая грамматический разбор входного скобочного выражения.
  1. Терминальные символы грамматики БП синтаксического анализатора программы должны представляться лексемами, специфицированными следующими литералами:

## ‘ZERO’,’ONE’,’\n’

которые обозначают коды ASCII нуля, единицы и перевода строки.

* 1. Начальный нетерминал грамматики БП синтаксического анализатора программы должен обозначаться именем input. Он должен выводиться из любой корректной или ошибочной последовательности.
  2. Система продукций грамматики БП синтаксического анализатора программы должна обеспечивать грамматический разбор произвольной входной строки потока стандартного ввода путем приведения терминалов и нетерминалов к начальному нетерминалу методом LR(1) анализа.
  3. 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.
  4. При разработке синтаксического анализатора программы необходимо составить Yacc-файл, отражающий специфику грамматического разбора скобочного выражения, и сохранить его под именем prog.y в выбранном доступном рабочем каталоге файловой системы OS UNIX.
  5. Проектируемый Yacc-файл prog.y должен состоять из 3-х секций: деклараций, правил и функций. Разделителем секций должны быть символические пары %%, расположенные в начальных позициях содержащих их строк Yacc-файла.
  6. Секция деклараций Yacc-файла prog.y должна включать:
     + cпецификацию блока внешних описаний, ограниченную директивами %{ и %}, в которой необходимо подключить библиотеку ввода/вывода stdio.h, а также необходимо явно объявить функции yyerror и yylex .
     + объявление с помощью директивы %token терминальных символов ZERO и ONE, задающих нуль и единицу соответственно.
  7. В секции правил Yacc-файла prog.y должны быть приведены описания продукций приведения нетерминалов грамматики БП.
  8. Каждая продукция секции правил Yacc-файла prog.y должна быть задана в нотации, близкой к форме Бэкуса-Наура, где в левой части указывается приводимый нетерминал, а в правой - последовательность терминалов и/или нетерминалов грамматики БП, которые перечисляются через пробел. 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.
  9. Продукции секции правил Yacc-файла prog.y, приведение нетерминалов которых необходимо сопровождать функциональной обработкой, должны содержать блоки действий. Блоки действий должны располагаться в правых частях продукций и ограничиваться парой

фигурных БП. Внутри блоков действий можно использовать любые конструкции и вызовы функций системы программирования C.

* 1. Продукции секции правил Yacc-файла prog.y, необходимые для приведения нетерминалов, должны быть специфицированы с помощью альтернативных правил. В каждом из них альтернативы свертки различных нетерминалов правой части должны быть объединены с помощью оператора ИЛИ, который обозначается символом вертикальной черты (|).
  2. Секция функций Yacc-файла prog.y должна содержать исходный код, оформленный по правилам системы программирования C, для функции yylex() которая должна иметь целочисленный код возврата.
  3. Для достижения целей лексического анализа, указанных в п. 5.17, исходный код функции yylex() должен предусматривать:
* побайтное чтение любой заданной строки входного скобочного выражения из потока стандартного ввода с помощью библиотечной функции getchar();
* распознавание символов во входной строке;
* возврат лексемы ONE, если была считана единица, и лексемы ZERO – если был считан нуль;
  1. Исходный код секции функций Yacc-файла prog.y образует исходный код программы , который должен формироваться генератором синтаксических анализаторов YACC в файле с предопределенным именем prog.c в текущем рабочем каталоге файловой системы OS UNIX. Выполняемый модуль программы должен быть создан по файлу исходного кода prog.c в выполняемом файле prog средствами компилирующей системы программирования C.

## Стадии и этапы разработки

Процесс разработки программы целесообразно разделяться на следующие 3 стадии:

* составить Yacc-файл prog.y в выбранном рабочем каталоге файловой системы OS UNIX, используя любой текстовый редактор;
* получить исходный код синтаксического анализатора в файле prog.c текущего каталога файловой системы OS UNIX, обработав Yacc-файл prog.y командой yacc, следующим образом:

**$ yacc -d prog.y**

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

# $ cc -o prog y.tab.c

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

## Порядок контроля и приемки

1. Для проверки функционирования программы должны быть предложены контрольные примеры, предусматривающие стандартный ввод корректных и некорректных скобочных выражений с учетом вложенности и конкатенации блоков скобок различных типов.
2. Для приемки программы должен быть организован вызов выполняемого файла prog в консольном режиме работы OS UNIX или режиме эмуляции терминала операционной среды X Window System.

**Результаты разработки**

# Описание грамматики

Терминальный алфавит: Σ = {'\n',ZERO, ONE,error} Нетерминальный алфавит: { exp0, exp1 };

Правила:

* 1. input -> {printf("Enter the line:\n");} – выводит на экран сообщение: "Enter the line: ";
  2. input -> input line – считывает строку;
  3. line -> '\n' {printf("Empty line\n");} – в случае, если на вход подана пустая строка, выводит диагностическое сообщение: "Empty line";
  4. line -> exp0 exp1 exp0 exp1 '\n' {printf ("Correct\n");} – в случае, если на вход подана бинарная последовательность, которая удовлетворяет указанному правилу, выводит диагностическое сообщение “Correct”;
  5. line -> error '\n' { yyerrok; } - в случае, если на вход подана бинарная последовательность, которая не удовлетворяет указанному правилу, выводит диагностическое сообщение об ошибке “syntax error”;
  6. exp0 -> ZERO – cимвол „0‟;
  7. exp0 -> exp0 ZERO - свёртка двух символов “0”;
  8. exp1 -> exp1 ONE - свёртка двух символов “1”;
  9. exp1-> ONE – cимвол „1‟;

Начальный нетерминал: input

# Результаты синтаксического анализа

01

Correct

001 01

Correct

01111

Correct 000110

syntax error 10 101

syntax error abc

syntax error

# Содержимое файла спецификации

%{

#include <stdio.h>

int yylex(void);

void yyerror(char const \*);

%}

%token ZERO

%token ONE

%token SPACE

%start input

%%

input: {printf("Enter the line:\n");}

| input line;

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

| error SPACE {yyerrok;}

| exp0 exp1 SPACE {

if ($2 > 0 && $1 > 0) {

printf("Correct\n");

} else{

yyerror("syntax error");

}

};

exp0: ZERO {$$++;}

| exp0 ZERO {$$++;}

;

exp1: ONE {$$++;}

| exp1 ONE {$$++;}

;

%%

int yylex(void) {

int c;

c = getchar();

switch (c) {

case ' ': case '\t': case '\n': return SPACE;

case '0': return ZERO;

case '1': return ONE;

case EOF: return 0;

default: return c;

}

}

**Список литературы**

База и Генератор Образовательных Ресурсов (bigor.bmstu.ru)

Генератор программ синтаксического анализа yacc. Производственно-внедренческий кооператив "И Н Т Е Р Ф Е Й С": Демос/P 2.1, 1988

Рейуорд-Смит В.Дж. Теория формальных языков. Вводный курс, M.: Радио и связь, 1988