

**Министерство образования и науки Российской Федерации**

**Федеральное агентство по образованию**

**Государственное образовательное учреждение высшего профессионального образования**

**«Московский государственный технический университет имени Н.Э. Баумана»**

**(МГТУ им. Н.Э. Баумана)**

**Факультет «Робототехника и комплексная автоматизация» (РК)**

**Кафедра «Системы автоматизированного проектирования» (РК6)**

****

**Отчет по лабораторной работе \_\_\_\_\_Т14\_\_\_\_\_\_\_ по курсу**

**«САПР»**

**Студент: Петраков Станислав**

**Группа:** РК6-36Б

**Преподаватель:** Волосатова Т.М.

Проверил:

Дата:

**Отчет по лабораторным работам 5 и 6 по автоматизированной разработке трансляторов на основе разработки высокоуровневых спецификаций для LEX и YACC.**

**Задание:**

Разработать транслятор для преобразования записи любой конечной десятичной дроби с необязательным знаком в обыкновенную дробь. Все десятичные дроби должны передаваться транслятору в экспоненциальном формате записи вещественных чисел через строки потока стандартного ввода. Результаты трансляции должны отображаться строками потока стандартного вывода, где числители и знаменатели обыкновенных дробей разделены символом '/'.

**Введение**

Настоящий документ определяет техническое задание на разработку программы интерактивного интерпретатора вещественного числа (далее по тексту - программа EXP), заданной в экспоненциальной форме.

**Основания для разработки**

Программа EXP 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 EXP предназначается для реализации грамматического разбора и преобразования экспоненциальной формы записи вещественного числа в обыкновенную дробь.

**Требования к программе**

1. **Требования к функциональным характеристикам**
   1. Программа EXP должна обеспечивать интерактивную обработку для анализа вещественного числа из потока стандартного ввода, заданной в экспоненциальной форме.
   2. Корректная экспоненциальная форма записипредставляет собой такую запись вещественного числа, в которой сначала записывается вещественное число, а затем степень десятки: «е+01» или «е-12».
   3. Корректная форма записи обыкновенной дроби представляет собой такую запись вещественного числа, в которой она имеет вид , где m,n≠0.
   4. Программа EXP должна распознавать экспоненциальную форму записи вещественного числа, полученной из входной строки потока стандартного ввода.
   5. Результат распознавания программой EXP корректной обыкновенной дроби должен отображаться в потоке стандартного вывода информационным сообщением.
   6. Программа EXP должна обеспечивать преобразование экспоненциальной формы записи вещественного числа в обыкновенную дробь и отображение результата преобразования в потоке стандартного вывода.
   7. Программа EXP должна обнаруживать любые нарушения формата представления обыкновенной дроби, которые противоречат требованиям пп. 1.2.
   8. Любые нарушения допустимого формата записи обыкновенной дроби во входной строке из потока стандартного ввода должны сопровождаться информационным сообщением "error" в потоке стандартного вывода.
   9. Программа EXP должна диагностировать стандартный ввод пустых строк, отображая информационное сообщение "Empty line " в потоке стандартного вывода и классифицируя их как ошибку.
2. **Требования к надежности**

Программа EXP должна обеспечивать обработку вещественных чисел, внутреннее представление которых не превышает размер машинного слова, длиной 4 байта.

1. **Требования к составу и параметрам технических средств**

Программа EXP должна быть разработана исходя из возможности реализации на стандартном составе технических средств компьютеров любой архитектуры с классическим порядком байт в машинном слове и разрядов в байте, после соответствующей перетрансляции исходного кода.

1. **Условия эксплуатации**
   1. Программа EXP должна быть ориентирована на эксплуатацию в среде OS UNIX.
   2. Программа EXP должна быть реализована в виде выполняемого файла, по которому она должна вызываться средствами любого командного процессора OS UNIX.
   3. Программа EXP должна эксплуатироваться в интерактивном режиме, читая строки из потока стандартного ввода и отображая результаты их обработки в потоке стандартного вывода.
2. **Требования к информационной и программной совместимости**
   1. Чтобы обеспечить выполнение требуемых технических характеристик, программа EXP должна реализовывать лексический и синтаксический анализ входных строк из потока стандартного ввода.
   2. Лексический анализатор программы EXP должен обеспечивать распознавание лексем, соответствующих экспоненциальному формату вещественного числа, в строках потока стандартного ввода.
   3. Лексический анализатор программы EXP должен обеспечивать выделение следующих типов лексем: цифры, заглавная и строчная буква Е латинского алфавита, знак «+» и «-».
   4. Значение любой лексемы, выделяемой лексическим анализатором программы EXP, должно содержать номер ее позиции, считая с 0, в текущей входной строке потока стандартного ввода. Для сохранения значений лексем следует использовать внешнюю целочисленную переменную yylval.
   5. Для разработки лексического анализатора программы EXP, необходимо использовать генератор лексических анализаторов (далее по тексту - LEX) OS UNIX, инструментальные средства которого должны быть ориентированы на обработку файла спецификаций проектируемого лексического анализатора (далее по тексту - Lex-файл).
   6. При разработке лексического анализатора программы EXP необходимо составить Lex-файл, отражающий специфику лексического анализа постфиксной формы записи.
   7. Проектируемый Lex-файл должен состоять из 2-х разделов: раздел деклараций и раздел правил. Разделы Lex-файла должны отделяться символической парой %%.
   8. В разделе деклараций Lex-файла необходимо специфицировать блок внешних описаний, ограничев его дериктивами %{ и %}, в котором нужно включить заголовочный файл y.tab.h макроопределений типов лексем директивой "#include" препроцессора системы программирования С;
   9. В разделе правил Lex-файла должны быть введены правила, которые обеспечивают распознавание лексем, перечисленных в п. 5.3, с помощью шаблонов регулярных выражений и необходимую функциональную обработку переменных в блоках действий правил.
   10. Блоки действий правил Lex-файла должны обеспечивать:

* сохранение номеров позиций лексем в старших байтах внешней переменной yylval;
* сохранение величин распознанных цифр в младших байтах внешней переменной yylval, которые определяются по символическому представлению цифр, предоставляемому нулевым элементом встроенного массива yytext, вычитанием из него символов '0' или 'a' в зависимости от системы счисления;
* возврат типа распознанной лексемы оператором return, аргумент которого должен соответствовать макроопределению лексемы из заголовочного файла y.tab.h;

Каждый блок действий должен быть ограничен парой фигурных скобок, внутри которых допустимо использовать любые конструкции и вызовы библиотечных функций системы программирования C.

* 1. Синтаксический анализатор программы EXP должен обеспечивать грамматический разбор потока лексем от лексического анализатора с целью установить соответствие или несоответствие содержащих их строк потока стандартного ввода требуемому формату постфиксной форме записи.
  2. Для выполнения грамматического разбора синтаксический анализатор программы EXP должен реализовывать однозначную контекстно-свободную грамматику простого предшествования, которая ориентирована на построчную обработку цифр и букв латинского алфавита из потока стандартного ввода и далее по тексту называется грамматикой записей.
  3. Грамматику записей синтаксического анализатора программы EXP должны составлять следующие элементы: начальный нетерминал, нетерминалы, терминалы (лексемы, выделенные лексическим анализатором, см. п. 5.3), система продукций (правил вывода), которые должны обеспечивать вывод входной строки стандартного ввода путем приведения терминалов и нетерминалов к начальному нетерминалу грамматики записей.
  4. Приведение потока лексем к начальному нетерминалу с помощью продукций грамматики записей синтаксического анализатора программы EXP должно означать успешное завершение грамматического разбора, т.е. распознавание корректной цифры или буквы латинского алфавита. Альтернативный результат должен рассматриваться как синтаксическая ошибка формата введенной константы.
  5. Начальный нетерминал должен определяться продукциями грамматики записей синтаксического анализатора программы EXP как входная строка, где входная строка может соответствовать нетерминалам или быть пустой, чтобы распознать конец потока стандартного ввода (EOF).
  6. Для разработки синтаксического анализатора программы EXP, 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.
  7. При разработке синтаксического анализатора программы EXP необходимо составить Yacc-файл, отражающий специфику грамматического разбора цифр и букв латинского алфавита.
  8. Проектируемый Yacc-файл должен состоять из 3-х секций: секция деклараций, секция правил и секция функций. Разделителем секций должна быть символическая пара %%.
  9. Секция деклараций Yacc-файла должна включать:
* перечисление терминалов грамматики записей, соответствующих по обозначениям типам лексем, выделяемым лексическим анализатором с помощью директивы %token
* cпецификацию блока внешних описаний, ограниченную директивами, %{ и %};
  1. В секции правил Yacc-файла должны быть приведены описания продукций приведения нетерминалов грамматики записей.
  2. Каждая продукция секции правил Yacc-файла должна быть задана в нотации, где в левой части указывается приводимый нетерминал, а в правой - последовательность терминалов и нетерминалов грамматики констант, 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. Альтернативные продукции (с одинаковыми нетерминалами в левых частях правил), необходимые для приведения нетерминалов, должны быть объединены в секции правил Yacc-файла с помощью оператора ИЛИ, который обозначается символом вертикальной черты (|).
  4. Продукции секции правил Yacc-файла, приведение нетерминалов которых необходимо сопровождать функциональной обработкой, должны содержать блоки действий. Блоки действий должны располагаться в правых частях продукций и ограничиваться парой фигурных скобок. Внутри блоков действий можно использовать любые конструкции и вызовы функций системы программирования C, а также операции с псевдо- переменными YACC ($$, $1, $2, ... ), которые нужно применять для доступа к элементам продукций. Псевдопеременная $$ должна использоваться для назначения желаемого численного значения нетерминалам в левых частях продукций. Псевдопеременные должны применяться для доступа к терминалам или нетерминалам в порядке их следования в правой части каждой продукции грамматики записей, где это необходимо. При формировании продукций грамматики записей следует учитывать, что по умолчанию, значение псевдопеременной $$ равно значению псевдопеременной $1, а значение псевдопеременной для терминала равно значению переменной yylval, которая хранит значение соответствующей лексемы.
  5. Блоки действий с псевдопеременными в секции правил Yacc-файла следует использовать для динамического накопления значений лексем параллельно с грамматическим разбором входной строки стандартного ввода.
  6. Накопленные в нетерминалах значения должны передаваться нетерминалам через явное или неявное (принятое по умолчанию) присваивание псевдопеременных в блоках действий соответствующих продукций секции правил Yacc-файла.
  7. В блоках действий для альтернативных продукций приведения нетерминалов к нетерминалу грамматики записей в секции правил Yacc-файла должна быть обеспечена индентификация постфиксной формы записи путем присваивания глобальной статической переменной. Должно осуществляться через неявное переприсваивание псевдопеременных, которое выполняется по умолчанию.
  8. Чтобы обеспечить требуемый формат отображения результатов грамматического разбора инфиксной формы записи, в блоке действий продукции приведения нетерминала секции правил Yacc-файла, следует применить библиотечную функцию printf(). Для доступа к нетерминалу, содержащему значение распознанной лексемы, в блоке действий продукции приведения нетерминала необходимо использовать псевдо-переменную $1.
  9. Блок действия альтернативной продукции приведения начального нетерминала через пустую структурную единицу грамматики записей, которая необходима для обработки признака конца потока стандартного ввода, должен обеспечивать возможность отличить эту ситуацию от ситуации успешного завершения грамматического разбора в программе EXP и блокировать отображение соответствующего информационного сообщения.
  10. Блок действия альтернативной продукции приведения начального нетерминала через нетерминал, которая необходима для обработки пустой или неправильной строки, должен обеспечивать диагностику ошибки, используя библиотечную функцию printf() системы программирования C.
  11. Секция функций Yacc-файла должна содержать спецификацию функции обработки ошибок синтаксического анализа, с зарезервированным именем yyerror, которая автоматически вызывается, когда входной поток лексем не может быть приведен к начальному нетерминалу грамматики записей, обеспечивая аварийное прерывание грамматического разбора текущей входной строки потока стандартного ввода. Спецификация функции yyparse() должна соответствать правилам оформления исходного кода функций в системе программирования C.
  12. Функция yyerror(), специфицированная в Yacc-файле, должна иметь единственный аргумент типа (char \*). Аргумент функции yyerror() следует использовать для формирования диагностического сообщения в соответствии с требованием п. 1.10.
  13. Функция yyerror(), специфицированная в Yacc-файле, должна обеспечивать диагностику места ошибки грамматического разбора во входной строке потока стандартного ввода в соответствии с требованием п.1.10.
  14. Синтаксический анализатор, формируемый по Yacc-файлу, должен быть реализован отдельным объектным модулем y.tab.o с точкой входа yyparse(), который должен включаться в выполняемый файл программы EXP редактором связей системы программирования C.
  15. Вызов синтаксического анализатора должен осуществляться путем обращения к функции yyparse() в основной функции main() программы EXP, исходный код которой необходимо составить на языке программирования C.
  16. Функция yyparse() должна возвращать в основную функцию main() программы EXP код 0 при успехе грамматического разбора каждой входной строки потока стандартного ввода или в конце потока стандартного ввода и 1 - при обнаружении синтаксических ошибок.
  17. Основная функция main() программы EXP должна обеспечивать циклический вызов функции синтаксического анализатора yyparse() и анализ ее кода возврата для отображения информационных сообщений по результатам грамматического разбора каждой входной строки из потока стандартного ввода в соответствии с требованиями пп. 1.6 и 1.10, пока не исчерпан поток стандартного ввода.
  18. Основная функция main() программы EXP должна быть реализована в отдельном объектном модуле с точкой входа main(). Он должен включаться в выполняемый файл программы EXP редактором связей системы программирования C.

**Стадии и этапы разработки**

Процесс разработки программы EXP должен разделяться на следующие 4 стадии:

* разработка лексического анализатора средствами LEX;
* разработка синтаксического анализатора средствами YACC;
* разработка основной функции main для вызова анализаторов;
* сборка выполняемого файла программы EXP.

1. При разработке лексического анализатора программы EXP необходимо выполнить следующие этапы:
   * составить Lex-файл в выбранном рабочем каталоге, используя любой текстовый редактор OS UNIX;
   * получить исходный код лексического анализатора в файле lex.yy.c,

обработав Lex-файл командой lex следующим образом: lex exp.l

* + получить объектный код лексического анализатора в файле lex.yy.o, обработав его исходный код в файле lex.yy.c компилятором системы программирования C следующей командой OS UNIX: cc -c lex.yy.c

1. Стадию разработки синтаксического анализатора программы INFIX необходимо разделить на следующие этапы:
   * составить Yacc-файл в выбранном рабочем каталоге, используя любой текстовый редактор OS UNIX;
   * 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 командой yacc, следующим образом: yacc -vd exp.y;
   * получить объектный код синтаксического анализатора в файле y.tab.o, обработав его исходный код в файле y.tab.c компилятором системы программирования C следующей командой OS UNIX: cc -c y.tab.c.
2. На стадии разработки основной функции main для вызова анализаторов в программе EXP следует выполнить следующие этапы:
   * составить исходный код основной функции main в файле выбранного рабочего каталога, используя любой текстовый редактор OS UNIX;
   * получить объектный код основной функции main в файле, обработав ее исходный код в файле компилятором системы программирования C следующей командой OS UNIX: cc -c radix.c
3. Для сборки выполняемого файла из объектных модулей lex.yy.o, y.tab.o, radix.o и объектных модулей библиотеки LEX, необходимо применить редактор связей OS UNIX, реализовав его вызов следующей командой: cc -o exp exp.o y.tab.o lex.yy.o -ll.

**Порядок контроля и приемки**

Для проверки функционирования программы EXP должен быть предложен контрольный пример, предусматривающий стандартный ввод корректных и некорректных алгебраических выражений постфиксной формы записи, для которого возможна простая визуальная оценка полученных результатов.

**Тесты:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| **1.7e2** | **170/1** |
| **+1.7e2** | **170/1** |
| **-0.7e-1** | **-7/100** |
| **1.0E1** | **10/1** |

**Приложение 1:**

Lex-файл:

DOT "."

%{

#include"y.tab.h"

%}

%%

^[-]([1-9][0-9]\*|[0])/{DOT} {yylval=atoi(yytext+1); return CELM;}

^[+](([1-9][0-9]\*)|[0])/{DOT} {yylval=atoi(yytext+1); return CEL;}

^(([1-9][0-9]\*)|[0])/{DOT} {yylval=atoi(yytext); return CEL;}

\.([0-9]\*)/[eE] {yylval=atoi(yytext+1); return DRB;}

[eE][-+]?[0-9]{1,2}$ {yylval=atoi(yytext+1); return STP;}

([a-zA-z ]\*[0-9]\*)\* {return ER;}

\n |

. ;

Yacc-файл:

%{

#include <stdio.h>

#include <string.h>

int intPart=0, drbPart=0, ePart=0;

void convertP(int intPart, int drbPart, int ePart);

void convertM(int intPart, int drbPart, int ePart);

%}

%start list

%token CEL DRB CELM STP ER

%%

list:

|list drob

;

drob: CEL DRB STP

{

intPart=$1;

drbPart=$2;

ePart=$3;

convertP(intPart, drbPart, ePart);

}

|CELM DRB STP

{

intPart=$1;

drbPart=$2;

ePart=$3;

convertM(intPart, drbPart, ePart);

}

;

%%

int yyerror()

{

puts("ERROR");

}

int yywrap()

{

return 0;

}

int gcd(int a, int b) {

int c;

while (b) {

c = a % b;

a = b;

b = c;

}

if (a < 0)

return (-a);

else

return a;

}

void convertP(int intPart, int drbPart, int ePart)

{

//Calc length of drb part

int drbLen = 0;

int temp = drbPart;

do {

drbLen++;

temp /= 10;

} while (temp);

//calc numenator

int drbE = 1;

for (int i = 0; i < drbLen; i++)

drbE \*= 10;

int num = intPart \* drbE + drbPart;

//calc den 1.56e6

int den = 1;

//number = num \* 10 ^ (ePart - drbLen)

if (ePart - drbLen >= 0)

for (int i = 0; i < (ePart - drbLen); i++)

num \*= 10;

else

for (int i = 0; i < (-ePart + drbLen); i++)

den \*= 10;

int commonDivider = gcd(num, den);

num /= commonDivider;

den /= commonDivider;

printf("%d/%d\n", num, den);

}

void convertM(int intPart, int drbPart, int ePart)

{

printf("-");

convertP(intPart, drbPart, ePart);

}

int main()

{

yyparse();

}

**Приложение 2:**

При разработке лексических и синтаксических анализаторов в OS UNIX для проектирования программы INFIX были использованы литературные источники, перечисленные ниже.

* + 1. Рейуорд-Смит В.Дж. Теория формальных языков. Вводный курс, M.:

Радио и связь, 1988.

* + 1. Тихомиров В.П., Давидов М.И. Операционная система ДЕМОС: инструментальные средства программирования, М.: Финансы и статистика, 1988.
    2. SCO XENIX, Development System, LEX Programmer Guide, SCO Inc., 1986. ( Имеется русский перевод: Генератор лексических анализаторов LEX. Руководство Программиста )
    3. SCO XENIX, Development System, YACC Programmer Guide, SCO Inc., 1986. ( Имеется русский перевод: Генератор синтаксических анализаторов YACC. Руководство Программиста )
    4. SCO XENIX, Development System, Make Programmer Guide, SCO Inc., 1986. ( Имеется русский перевод: Интерпретатор make. Руководство Программиста )