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

Задание

Разработать транслятор для округления значения любой обыкновенной дроби с необязательным знаком до наименьшего целого числа, которое превосходит ее по величине. Записи обыкновенных дробей, в которых числитель и знаменатель разделены символом '/', должны передаваться транслятору строками потока стандартного ввода. Результаты трансляции должны отображаться строками целых чисел в потоке стандартного вывода.

Округление обыкновенных дробей

Введение

Настоящий документ определяет техническое задание на разработку программы округления значения любой обыкновенной дроби (далее по тексту - программа ROUND) с необязательным знаком до наименьшего целого числа, которое превосходит ее по величине.

Основания для разработки

Программа ROUND 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 ROUND предназначается для округления значения любой обыкновенной дроби.

Требования к программе

1. Требования к функциональным характеристикам

1.1. Программа ROUND должна обеспечивать интерактивную обработку для анализа обыкновенных дробей из потока стандартного ввода, в которых числитель и знаменатель разделены символом ‘/’.

1.2. Программа ROUND должна распознавать обыкновенные дроби, полученные из входной строки потока стандартного ввода.

1.3. Результат распознавания программой ROUND должен отображаться строками потока стандартного вывода.

1.4. С учетом требований пп. 1.2 и 1.3 информационное сообщение по результатам распознавания и преобразования обыкновенной дроби, например, 1/2, должно отображаться следующей строкой потока стандартного вывода:

1

1.5. Программа ROUND должна обнаруживать любые нарушения формата представления обыкновенных дробей, которые противоречат требованиям пп. 1.1.

1.6. Любые нарушения во входной строке из потока стандартного ввода должны сопровождаться информационным сообщением "syntax error" в потоке стандартного вывода.

1.7. Программа ROUND должна диагностировать стандартный ввод пустых строк или строк без дробей, отображая информационное сообщение "Empty line!" в потоке стандартного вывода и классифицируя их как ошибку формата ввода, но без указания позиции ошибки.

1.8. Программа ROUND должна завершать обработку входных строк при обнаружении конца потока стандартного ввода, отображая результат трансляции в поток стандартного вывода.

2. Условия эксплуатации

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

2.2. Программа ROUND должна быть реализована в виде выполняемого файла с именем round, по которому она должна вызываться средствами любого командного процессора OS UNIX.

2.3. Программа ROUND 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 ROUND должна реализовывать лексический и синтаксический анализ входных строк из потока стандартного ввода.

3.2. Лексический анализатор программы ROUND должен обеспечивать распознавание лексем, соответствующих формату обыкновенных дробей, в строках потока стандартного ввода.

3.3. Лексический анализатор программы ROUND должен обеспечивать выделение следующих типов лексем:

- DIGIT – положительное или отрицательное число.

Макроопределения значений типов лексем следует сосредоточить в заголовочном файле y.tab.h, который должен формироваться при разработке синтаксичекого анализатора программы ROUND (см. ниже).

3.4. Для сохранения значений лексем следует использовать внешнюю (extern) целочисленную (типа int) переменную yylval, которая содержит целочисленные значения числителя и знаменателя дроби.

3.5. Для разработки лексического анализатора программы ROUND, 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).

3.6. При разработке лексического анализатора программы ROUND необходимо составить Lex-файл, отражающий специфику лексического анализа обыкновенных дробей, и сохранить его под именем round.l в выбранном рабочем каталоге файловой системы OS UNIX.

3.7. Проектируемый Lex-файл round.l должен состоять из 2-х разделов: раздел деклараций и раздел правил. Разделы Lex-файла должны отделяться символической парой %%.

3.8. В разделе деклараций Lex-файла round.l необходимо:

- объявить макрошаблоны возможных чисел в числителе и знаменателе дроби;
- специфицировать блок внешних описаний, ограничив его директивами

%{ и %}, в котором нужно включить заголовочный файл y.tab.h макроопределений типов лексем директивой "#include" препроцессора системы программирования C.

3.9. В разделе правил Lex-файла round.l должны быть введены правила, которые обеспечивают распознавание лексем, с помощью шаблонов регулярных выражений и необходимую функциональную обработку в блоках действий правил.

3.10. Блоки действий правил Lex-файла round.l должны обеспечивать:

- сохранение значений числителя и знаменателя внешней переменной yylval, которые определяются по символическому представлению цифр, предоставляемому встроенным массивом yytext;
- 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.

3.11. Лексический анализатор программы ROUND должен быть реализован отдельным объектным модулем lex.yy.o с точкой входа yylex(), который должен включаться в состав выполняемого файла round редактором связей системы программирования C.

3.12. Вызов лексического анализатора должен осуществляться путем обращения к функции yylex() в синтаксическом анализаторе программы ROUND. Связь между синтаксическим и лексическим анализаторами должна осуществляться по типу лексемы, определяемому кодом возврата функции yylex(), и по значению лексемы, который передается через внешнюю переменную yylval.

3.13. Синтаксический анализатор программы ROUND должен обеспечивать грамматический разбор потока лексем от лексического анализатора с целью установить соответствие или несоответствие содержащих их строк потока стандартного ввода требуемому формату обыкновенных дробей.

3.14. Для разработки синтаксического анализатора программы ROUND, 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.

3.15. При разработке синтаксического анализатора программы ROUND необходимо составить Yacc-файл, отражающий специфику грамматического разбора обыкновенных дробей, и сохранить его под именем round.y в выбранном рабочем каталоге файловой системы OS UNIX.

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

3.17. Секция деклараций Yacc-файла round.y должна включать:

- перечисление терминалов грамматики констант, соответствующих по обозначениям типам лексем, выделяемым лексическим анализатором, с помощью директив(ы) %token;
- спецификацию блока внешних описаний, ограниченную директивами %{ и %}, в котором необходимо определить подключаемые библиотеки, определить функции yyerror, yylex, yywrap и Round, необходимую для округления дробей.

3.18. В секции правил Yacc-файла round.y должны быть приведены

описания продукций приведения нетерминалов грамматики констант.

3.19. Каждая продукция секции правил Yacc-файла round.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 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.

3.20. Продукции секции правил Yacc-файла round.y должны задаваться в форме лево-рекурсивных правил. Например, нетерминал input должен определяться следующим лево-рекурсивным правилом:

```
input : line |
        input line;
```

которое позволяет трактовать нетерминал input как одиночную лексему line, или (|) последовательность лексем line произвольной длины.

3.21. Продукции секции правил Yacc-файла round.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. Внутри блоков действий можно использовать любые конструкции и вызовы функций системы программирования C, а также операции с псевдо- переменными YACC ($$, $1, $2, ...), которые нужно применять для доступа к элементам продукций. Псевдо-переменная $$ должна использоваться для назначения желаемого численного значения нетерминалам в левых частях продукций. Псевдо-переменные $1 и $2 должны применяться для доступа к терминалам или нетерминалам в порядке их следования в правой части каждой продукции грамматики констант, где это необходимо. При формировании продукций грамматики констант следует учитывать, что по умолчанию, значение псевдо-переменной $$ равно значению псевдо-переменной $1, а значение псевдо-переменной для терминала равно значению переменной yylval, которая хранит значение соответствующей лексемы.

3.22. Значение распознанной обыкновенной дроби должно быть использовано в блоке действий продукции приведения начального нетерминала input секции правил Yacc-файла round.y для символического отображения округленной дроби.

3.23. Чтобы обеспечить требуемый формат отображения результатов грамматического разбора, в блоке действий продукции приведения нетерминала input секции правил Yacc- файла round.y, следует применить библиотечную функцию printf().

3.24. Секция функций Yacc-файла round.y 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 констант, обеспечивая аварийное прерывание грамматического разбора текущей входной строки потока стандартного ввода. Спецификация функции yyparse() должна соответствовать правилам оформления исходного кода функций в системе программирования C.

3.25. Функция yyerror(), специфицированная в Yacc-файле round.y, должна иметь единственный аргумент типа (char const *), который по умолчанию содержит адрес предопределенной символьной строки: "syntax error". Аргумент функции yyerror() следует использовать для формирования диагностического

сообщения.

3.26. Функция yyerror(), специфицированная в Yacc-файле round.y, должна обеспечивать целочисленный (типа int) возврат с кодом, равным 1.

3.27. Синтаксический анализатор, формируемый по Yacc-файлу round.y, должен быть реализован отдельным объектным модулем y.tab.o с точкой входа yyparse(), который должен включаться в выполняемый файл round программы ROUND редактором связей системы программирования C.

3.28. Вызов синтаксического анализатора должен осуществляться путем обращения к функции yyparse() в основной функции main() программы ROUND, исходный код которой необходимо составить на языке программирования C.

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

3.30. Основная функция main() программы ROUND 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, пока не исчерпан поток стандартного ввода.

Стадии и этапы разработки

Процесс разработки программы ROUND должен разделяться на следующие 3 стадии:

- разработка лексического анализатора средствами LEX;
- разработка синтаксического анализатора средствами YACC;
- сборка выполняемого файла программы ROUND.

1. При разработке лексического анализатора программы ROUND необходимо выполнить следующие этапы:

- составить Lex-файл round.l в выбранном рабочем каталоге, используя любой текстовый редактор OS UNIX, например, xedit;
- получить исходный код лексического анализатора в файле lex.yy.c, обработав Lex-файл round.l командой lex следующим образом:

```
$ lex round.l
```

2. Стадию разработки синтаксического анализатора программы ROUND необходимо разделить на следующие этапы:

- составить Yacc-файл round.y в выбранном рабочем каталоге, используя любой текстовый редактор OS UNIX, например, xedit;
- 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 round.y командой yacc, следующим образом:

```
$ yacc -d round.y
```

3. Для сборки выполняемого файла round из объектных модулей lex.yy.c, y.tab.c, необходимо применить редактор связей OS UNIX, реализовав его вызов следующей командой:

```
$ cc y.tab.c lex.yy.c
```

Порядок контроля и приёмки

Для проверки функционирования программы ROUND должен быть предложен контрольный пример, предусматривающий стандартный ввод корректных и некорректных обыкновенных дробей, для которого возможна простая визуальная оценка полученных результатов.

Приложение 1

Текст высокоуровневой спецификации для генератора LEX.

```
%{
   #include "y.tab.h"
%}

DIGIT       0|-?[1-9]+[0-9]*

%%
{DIGIT}   { yylval = atoi(yytext); return NUMBER; }
\/          { return '/'; }
\n          { return '\n'; }
.           { }
%%
```

Текст высокоуровневой спецификации для генератора YACC.

```
%{
   #include <stdio.h>
   void yyerror(char const *);
   int yylex(void);
   int yywrap() { return 1; }
   int Round(int, int);
%}

%token NUMBER

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

line: '\n' { printf("Empty line!\n"); }
   | error '\n' { yyerrok; }
   | expr
   ;

expr: NUMBER '/' NUMBER '\n'
   {
      if ($3 == 0)
         yyerror("syntax error");
      else
         printf("%d\n", Round($1, $3));
   }
   ;
%%

int main()
{
   yyparse();
   return 0;
}
```

```
void yyerror(char const *s)
{
   fprintf(stderr, "%s\n", s);
   return 1;
}

int Round(int numer, int denom)
{
   if (numer % denom != 0)
   {
      if (((numer < 0) && (denom > 0)) || ((numer > 0) && (denom <
0)))
         return (numer / denom);
      return (numer / denom + 1);
   }
   else
      return (numer / denom);
}
```

Приложение 2

При разработке лексических и синтаксических анализаторов в OS UNIX для проектирования программы ROUND рекомендуется использовать литературные источники, перечисленные ниже.

1. Рейуорд-Смит В.Дж. Теория формальных языков. Вводный курс, М.: Радио и связь, 1988.
2. Тихомиров В.П., Давидов М.И. Операционная система ДЕМОС: инструментальные средства программирования, М.: Финансы и статистика, 1988.
3. SCO XENIX, Development System, LEX Programmer Guide, SCO Inc., 1986. (Имеется русский перевод: Генератор лексических анализаторов LEX. Руководство Программиста)
4. SCO XENIX, Development System, YACC Programmer Guide, SCO Inc., 1986. (Имеется русский перевод: Генератор синтаксических анализаторов YACC. Руководство Программиста)
5. SCO XENIX, Development System, Make Programmer Guide, SCO Inc., 1986. (Имеется русский перевод: Интерпретатор make. Руководство Программиста)