Санкт-петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Факультет программной инженерии и компьютерной техники

Лабораторная работа № 1

Вариант 2

По дисциплине «Системное программное обеспечение»

Выполнила: Носкова Е.Е.
Группа № P4114
Проверил: Кореньков Ю. Д.

Санкт-Петербург
2023

Использовать средство синтаксического анализа по выбору, реализовать модуль для разбора текста в соответствии с языком по варианту. Реализовать построение по исходному файлу с текстом синтаксического дерева с узлами, соответствующими элементам синтаксической модели языка. Вывести полученное дерево в файл в формате, поддерживающем просмотр графического представления.

Порядок выполнения:

1 Изучить выбранное средство синтаксического анализа

a. Средство должно поддерживать программный интерфейс, совместимый с языком Си

b. Средство должно параметризоваться спецификацией, описывающей синтаксическую структуру разбираемого языка

c. Средство может функционировать посредством кодогенерации и/или подключения необходимых для его работы дополнительных библиотек

d. Средство может быть реализовано с нуля, в этом случае оно должно использовать обобщённый алгоритм, управляемый спецификацией

2 Изучить синтаксис разбираемого по варианту языка и записать спецификацию для средства синтаксического анализа, включающую следующие конструкции:

a. Подпрограммы со списком аргументов и возвращаемым значением

b. Операции контроля потока управления – простые ветвления if-else и циклы или аналоги

c. В зависимости от варианта – определения переменных

d. Целочисленные, строковые и односимвольные литералы

e. Выражения численной, битовой и логической арифметики

f. Выражения над одномерными массивами

g. Выражения вызова функции

3 Реализовать модуль, использующий средство синтаксического анализа для разбора языка по варианту

a. Программный интерфейс модуля должен принимать строку с текстом и возвращать структуру, описывающую соответствующее дерево разбора и коллекцию сообщений ошибке

b. Результат работы модуля – дерево разбора – должно содержать иерархическое представление для всех синтаксических конструкций, включая выражения, логически представляющие собой иерархически организованные данные, даже если на уровне средства синтаксического анализа для их разбора было использовано линейное представление

4 Реализовать тестовую программу для демонстрации работоспособности созданного модуля

a. Через аргументы командной строки программа должна принимать имя входного файла для чтения и анализа, имя выходного файла записи для дерева, описывающего синтаксическую структуру разобранного текста

b. Сообщения об ошибке должны выводиться тестовой программной (не модулем, отвечающим за анализ!) в стандартный поток вывода ошибок

5 Результаты тестирования представить в виде отчета, в который включить:

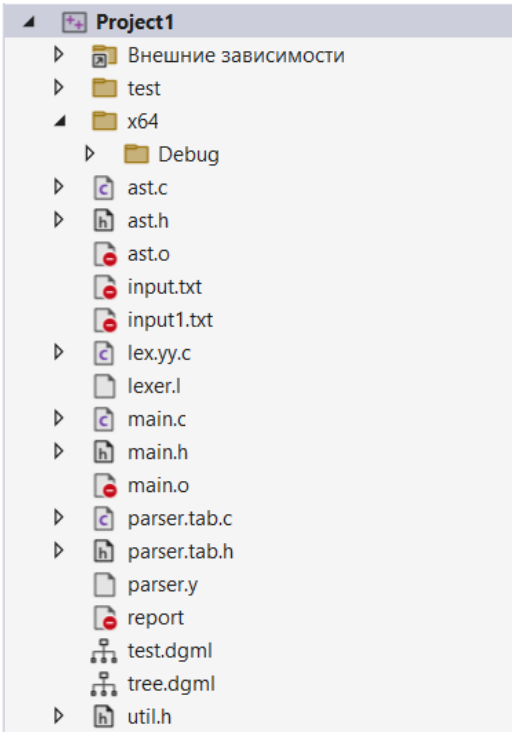
a. В части 3 привести описание структур данных, представляющих результат разбора текста (3a)

b. В части 4 описать, какая дополнительная обработка потребовалась для результата разбора, предоставляемого средством синтаксического анализа, чтобы сформировать результат работы созданного модуля

c. В части 5 привести примеры исходных анализируемых текстов для всех синтаксических конструкций разбираемого языка и соответствующие результаты разбора

Для выполнения задания используется Flex и Bison. Flex используется для лексического анализа, а bison – для синтаксического анализа. Лексический анализ разбивает ввод на значимые фрагменты, называемые token. Анализ синтаксиса определяет, как эти блоки слов связаны друг с другом (с использованием выражения синтаксического дерева).

На основе заданного варианта синтаксической модели языка была сформированная грамматика, сгенерированы парсер и лексер.



Узел абстрактного синтаксического дерева.

```
typedef struct ASTNode {
    char* type;
    struct ASTNode* left;
    struct ASTNode* right;
    char* value;
    int id;
} ASTNode;
```

parser.y:

```
%start translation_unit


%%

primary_expression  : IDENTIFIER {$$ = createNode("IDENTIFIER", NULL,NULL, $1);}
 | CONSTANT {$$ = createNode("CONSTANT", NULL,NULL, $1);}
 | STRING_LITERAL {$$ = createNode("STRING_LITERAL", NULL,NULL, $1);}
 | '(' expression ')' {$$ = $2; }
 ;

postfix_expression
 : primary_expression {$$ = $1;}
 | postfix_expression '[' expression ']' {$$ = createNode("postfix_expression", $1, $3, "");}
 | postfix_expression '(' ')' {$$ = createNode("postfix_expression", $1, NULL, ""); }
 | postfix_expression '(' argument_expression_list ')' {$$ = createNode("postfix_expression", $1, $3, "");}
    ;

argument_expression_list
 : assignment_expression {$$ = createNode("argument_expression_list", $1, NULL, "");}
 | argument_expression_list ',' assignment_expression {$$ = createNode("argument_expression_list", $1, $3, "");}
    ;

unary_expression
    : postfix_expression {$$ = $1;}
    | unary_operator primary_expression {$$ = createNode("unary_expression", $1, $2, "");}
    ;

unary_operator
    : '&' {$$ = NULL}
    | '*' {$$ = NULL}
    | '+' {$$ = NULL}
    | '-' {$$ = NULL}
    | '~' {$$ = NULL}
    | '!' {$$ = NULL}
    ;
```

lexer.l:

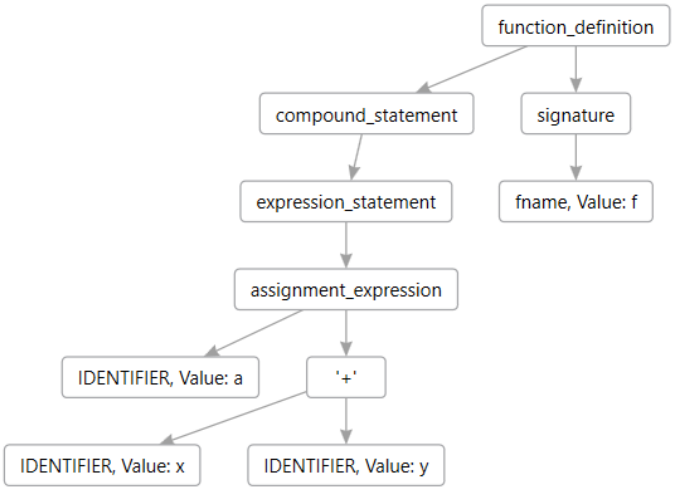
```
D [0-9]
L [a-zA-Z_]
H [a-fA-F0-9]
E ([Ee][+-]?{D}+)
P ([Pp][+-]?{D}+)
FS (f|F|l|L)
IS ((u|U)|(u|U)?(l|L|ll|LL)|(l|L|ll|LL)(u|U))

%{
#include <stdlib.h>
#include <stdio.h>
#include "util.h"
#include "ast.h"
#include "parser.tab.h"

void count(void);
void comment(void);
%}

%%
"/*" { comment(); }
"//"[^\n]* { /* consume //-comment */ }


"var" { count(); return(VAR); }
"then" { count(); return(THEN); }
"begin" { count(); return(T_BEGIN); }
"end" { count(); return(T_END); }
"of" { count(); return(OF); }
"array" { count(); return(ARRAY); }
"repeat" { count(); return(REPEAT); }
"until" { count(); return(UNTIL); }
"auto" { count(); return(AUTO); }
"bool" { count(); return(BOOL); }
"byte" { count(); return(BYTE); }
"break" { yylval.node = createNode("BREAK", NULL, NULL, ""); return(BREAK); }
"case" { count(); return(CASE); }
"char" { count(); return(CHAR); }
"_Complex" { count(); return(COMPLEX); }
"const" { count(); return(CONST); }
"continue" { count(); return(CONTINUE); }
"default" { count(); return(DEFAULT); }
"do" { count(); return(DO); }
"double" { count(); return(DOUBLE); }
"else" { count(); return(ELSE); }
"enum" { count(); return(ENUM); }
"extern" { count(); return(EXTERN); }
"method" { count(); return(METHOD); }
"float" { count(); return(FLOAT); }
"for" { count(); return(FOR); }
```

Примеры входных и выходных данных:

Входные данные 1.

```
method f()
begin
   a = x+y;
end;
```

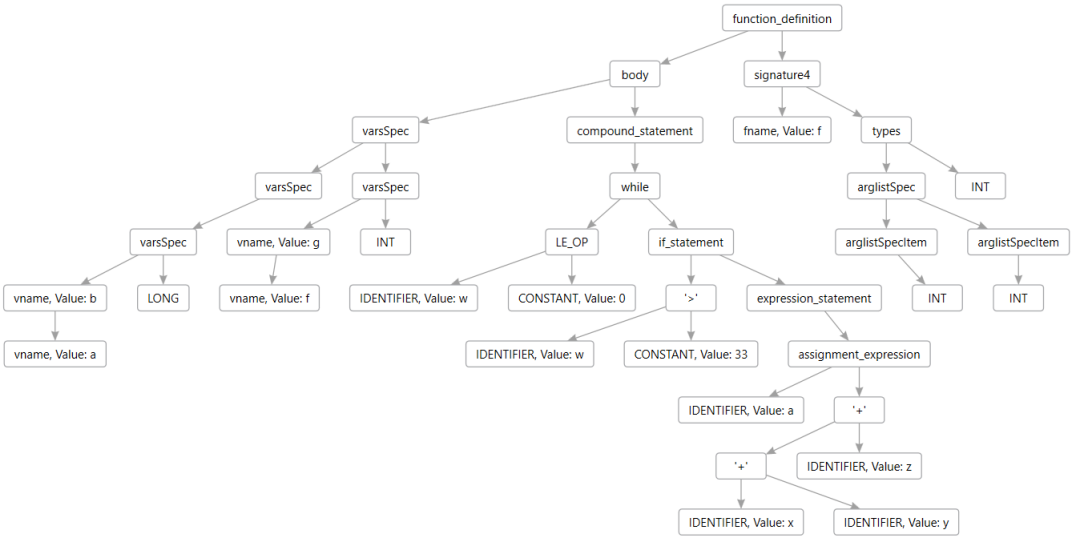
Результат 1.



Входные данные 2.

```
method f(x:int,y:int) :int
var
a,b:long;
f,g:int;
begin

    while w<=0 do
        if w>33 then
        a = x+y+z;


end;
```

Результат 2.



Вывод:

В ходе выполнения лабораторной работы был реализован модуль для разбора текста. Изучены средства синтаксического анализа. Реализовано построение по исходному файлу с текстом синтаксического дерева с узлами, соответствующими элементам синтаксической модели языка. Полученное дерево выведено в файл в формате dgml.