УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.04.04 Программная инженерия

Дисциплина «Системное программное обеспечение»

Лабораторная работа №1

Вариант 4

Студент

Митрофанов Е. Ю.

P4116

Преподаватель

Кореньков Ю. Д.

Санкт-Петербург, 2023 г.

Задание лабораторной работы

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
   1.1. Средство должно поддерживать программный интерфейс, совместимый с языком Си
   1.2. Средство должно параметризоваться спецификацией, описывающей синтаксическую структуру разбираемого языка
   1.3. Средство может функционировать посредством кодогенерации и/или подключения необходимых для его работы дополнительных библиотек
   1.4. 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:
   2.2. Подпрограммы со списком аргументов и возвращаемым значением
   2.3. Операции контроля потока управления – простые ветвления if-else и циклы или аналоги
   2.4. В зависимости от варианта – определения переменных
   2.5. Целочисленные, строковые и односимвольные литералы
   2.6. Выражения численной, битовой и логической арифметики
   2.7. Выражения над одномерными массивами
   2.8. Выражения вызова функции
3. Реализовать модуль, использующий средство синтаксического анализа для разбора языка по варианту
   3.1. Программный интерфейс модуля должен принимать строку с текстом и возвращать структуру, описывающую соответствующее дерево разбора и коллекцию сообщений ошибке
   3.2. 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
   3.3. Реализовать тестовую программу для демонстрации работоспособности созданного модуля
   3.4. 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
   3.5. Сообщения об ошибке должны выводиться тестовой программной (не модулем, отвечающим за анализ!) в стандартный поток вывода ошибок
4. Результаты тестирования представить в виде отчета, в который включить:
   4.1. В части 3 привести описание структур данных, представляющих результат разбора текста (3а)

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

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

Описание структур данных

В процессе парсинга входного текста по лексемам и описанной грамматике формируется дерево, одна нода которого описано структурой:

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

Дерево содержит id, указатель на левую и правую связанную ноду, тип и значение, если у такого типа может быть значение. Сами ноды создаются при описании простых лексем:

```
0[bB][01]+ {
    char* buffer = malloc(256);
    sprintf(buffer, "%ld", strtol(yytext + 2, NULL, 2));
    yylval.node = createNode("BIN", NULL, NULL, buffer);
    return BIN;
}
```

А также при описании грамматических конструкций:

```
if: IF expr THAN statement optionalElseStatement {{$$ = createNode("if", $3,
createNode("ifStatements", $4, $5, ""), "");}};

optionalElseStatement: ELSE statement optionalElseStatement {{$$ =
createNode("else", $2, $3, "");}}
    | ELSE statement {{$$ = createNode("else", $2, NULL, "");}}
    | {{$$ = NULL;}};
```

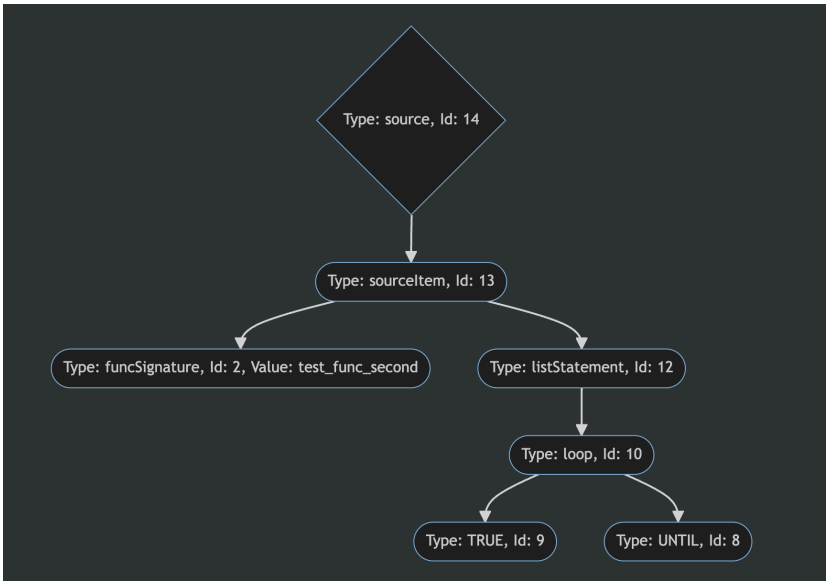
Вывод построенного дерева осуществляется в .md файл в формате построения *mermaid* – диаграмм.

Примеры входных данных и результаты обработки

Для примера:

```
def test_func_second()
   {
        test = false;
   } until true;
end
```

Получаем дерево:



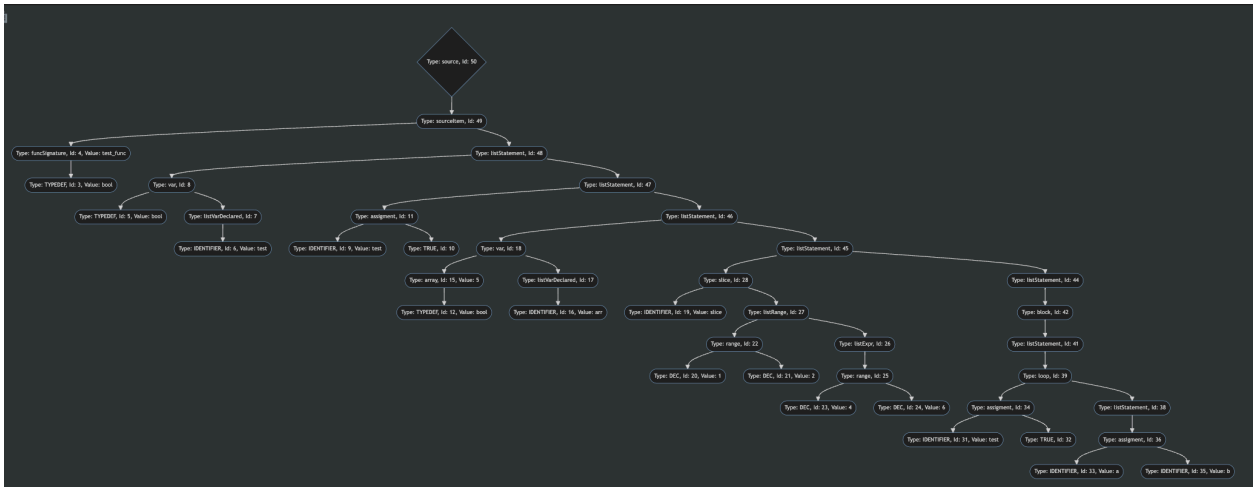
Для более сложного примера:

```
def test_func() (of bool)
    bool test;
    test = true;
    bool array[5] arr;

    slice[1..2, 4..6];
    //comment

    begin
        while test = true
            a = b;
        end
    end
end
```

Результат:



Вывод

В ходе данной лабораторной работы я ознакомился с основами языка Си, работой Makefaile-ов, понятием абстрактного синтаксического дерева, а также имплементирован собственный лексер и парсер в соответствии с описанным во варианте языковым синтаксисом. Разработал алгоритмы построения и вывода синтаксического дерева в формате *mermaid* диаграммы.