Федеральное государственное автономное образовательное учреждение высшего образования

Университет ИТМО

Дисциплина: Системное программное обеспечение

Лабораторная работа 1

Вариант 1

Выполнили:
Кривоносов Егор Дмитриевич

Группа: P4114

Преподаватель:
Кореньков Юрий Дмитриевич

2024 г.

Санкт-Петербург

Оглавление

Задание

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 (3а)
   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

Грамматика

Вариант 1

```
source: sourceItem*;
typeRef: {
      |builtin: 'bool'|'byte'|'int'|'uint'|'long'|'ulong'|'char'|'string';
      |custom: identifier;
      |array: typeRef '[' (',')* ']';
};

funcSignature: typeRef? identifier '(' list<argDef> ')' {
      argDef: typeRef? identifier;
};

sourceItem: {
      |funcDef: funcSignature (statement.block|';');
};

statement: {
      |var: typeRef list<identifier ('=' expr)?> ';'; // for static typing
      |if: 'if' '(' expr ')' statement ('else' statement)?;
      |block: '{' statement* '}';
      |while: 'while' '(' expr ')' statement;
      |do: 'do' block 'while' '(' expr ')' ';';
      |break: 'break' ';';
      |expression: expr ';'
};

expr: { // присваивание через '='
      |binary: expr binOp expr; // где binOp - символ бинарного оператора
      |unary: unOp expr; // где unOp - символ унарного оператора
      |braces: '(' expr ')';
      |call: expr '(' list<expr> ')';
      |indexer: expr '[' list<expr> ']';
      |place: identifier;
      |literal: bool|str|char|hex|bits|dec;
};
```

Описание структуры данных

Для построен дерева была создана структура ASTNode. Ноды создаются для простых лексем, а также грамматических конструкций.

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

Вывод полученного дерева происходит в SVG-файл с помощью Graph-Easy.

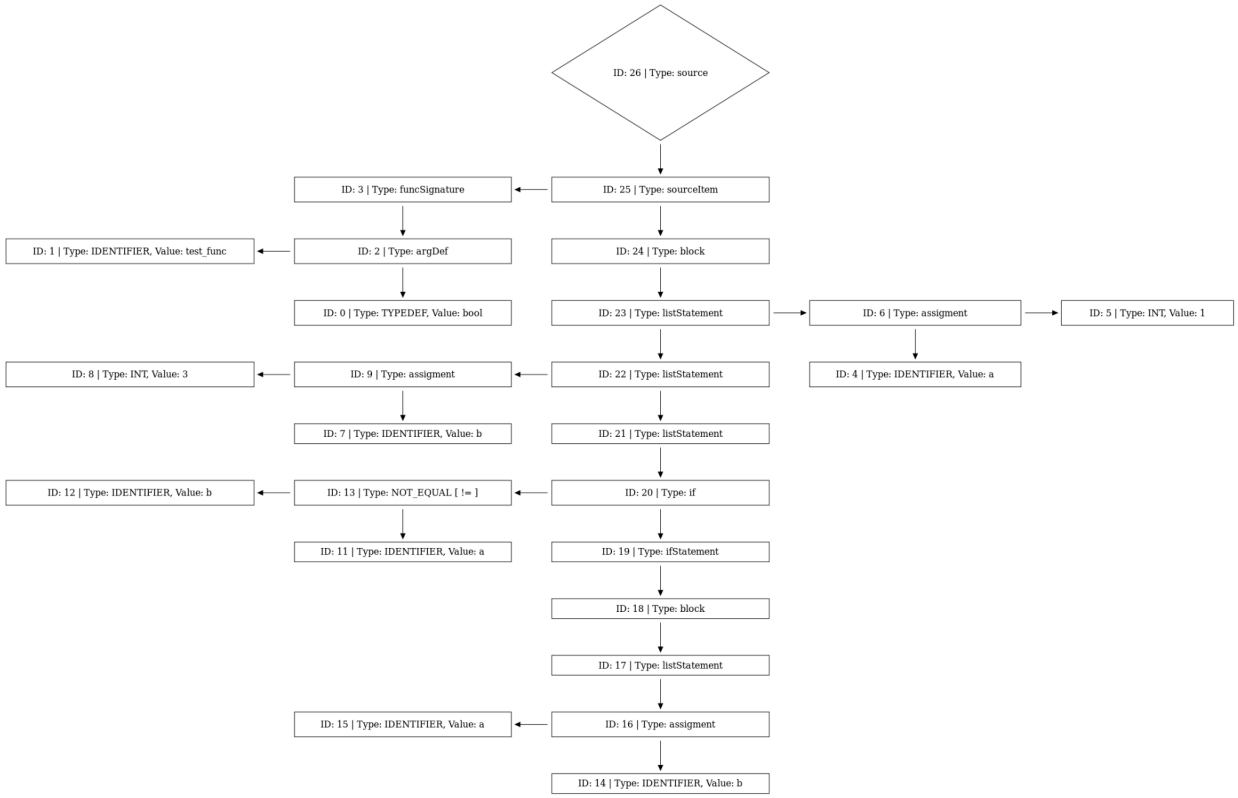
Примеры работы

Пример 1

Код:

```
bool test_func() {
    a = 1;
    b = 3;
    if (a != b) {
        b = a;
    }
}
```

Дерево:

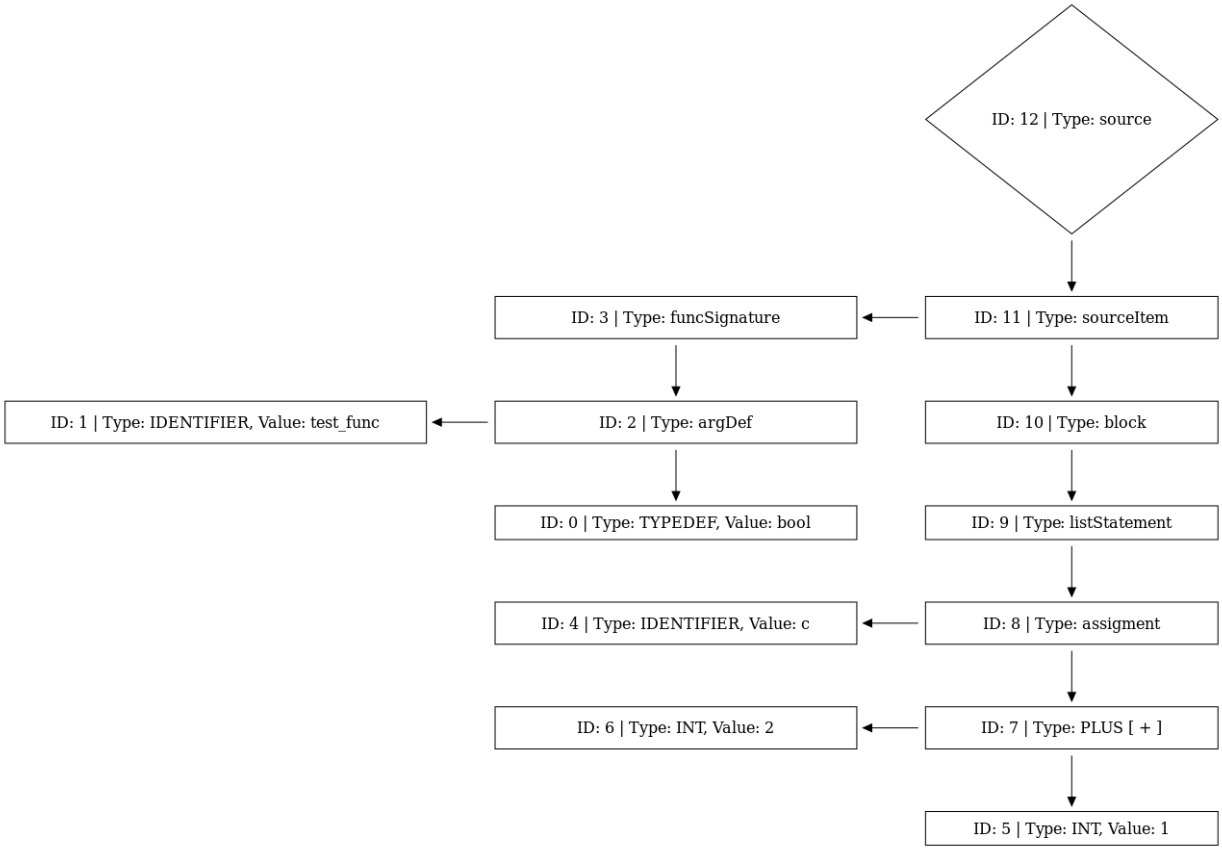


Пример 2

Код:

```
bool test_func() {
    c = 1 + 2;
}
```

Дерево:

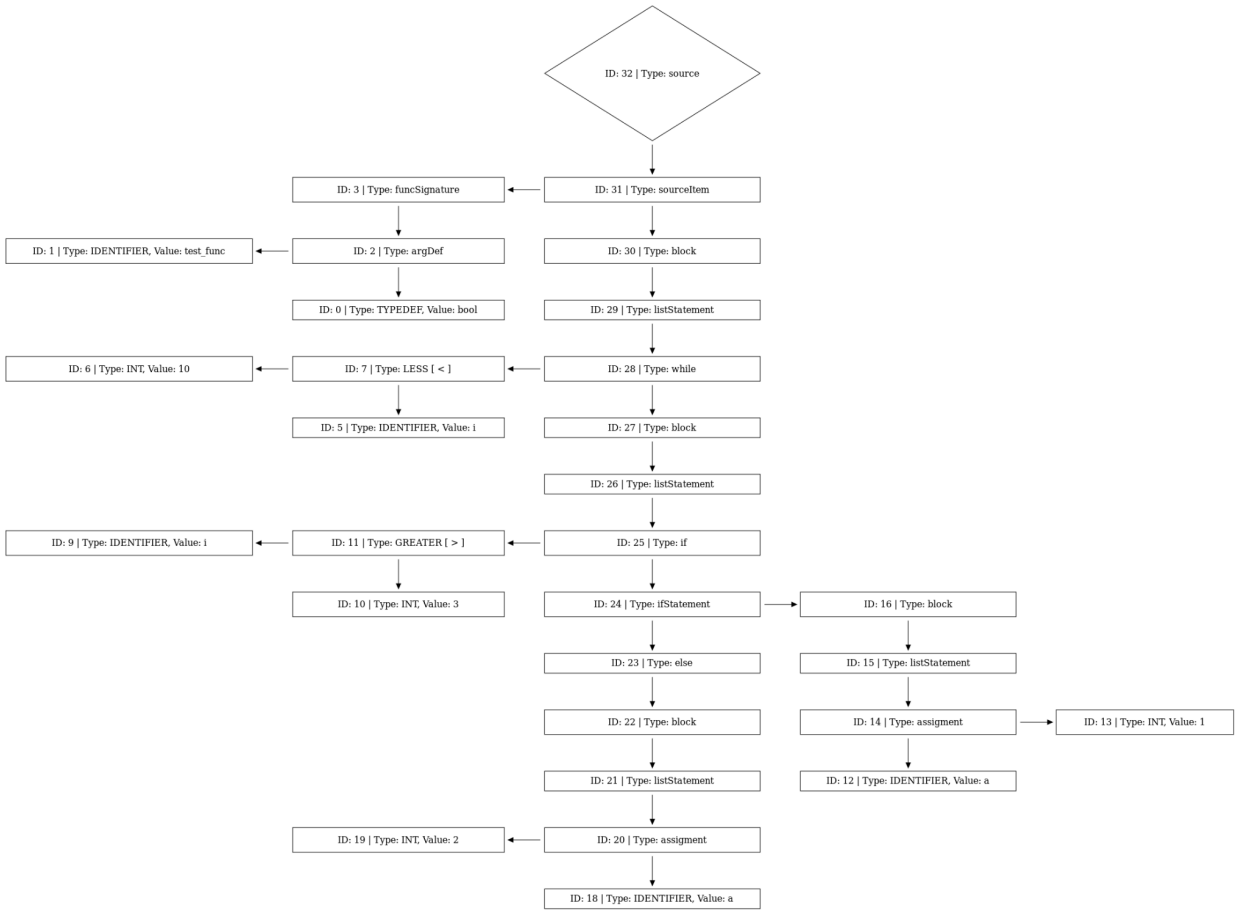


Пример 3

Код:

```
bool test_func() {
    while (i < 10) {
        if (i > 3) {
            a = 1;
        } else {
            a = 2;
        }
    }
}
```

Дерево:



Вывод

В ходе выполнения лабораторной работы, я повторил основы языка Си, работу с файлами сборки Makefile, изучил какие лексические анализаторы бывают, а так же поработал с одним из них. Научился строить графы через утилиту Graph-Easy.