

УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.04.04 Программная инженерия

Дисциплина «Системное программное обеспечение»

Лабораторная работа №1

Вариант 1

Студент
Алексеев М. Н.
Р4116

Преподаватель
Кореньков Ю. Д.

Санкт-Петербург, 2023 г.

Задание лабораторной работы

Задание 1

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 об 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

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

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

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

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

с. 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.

Пример:

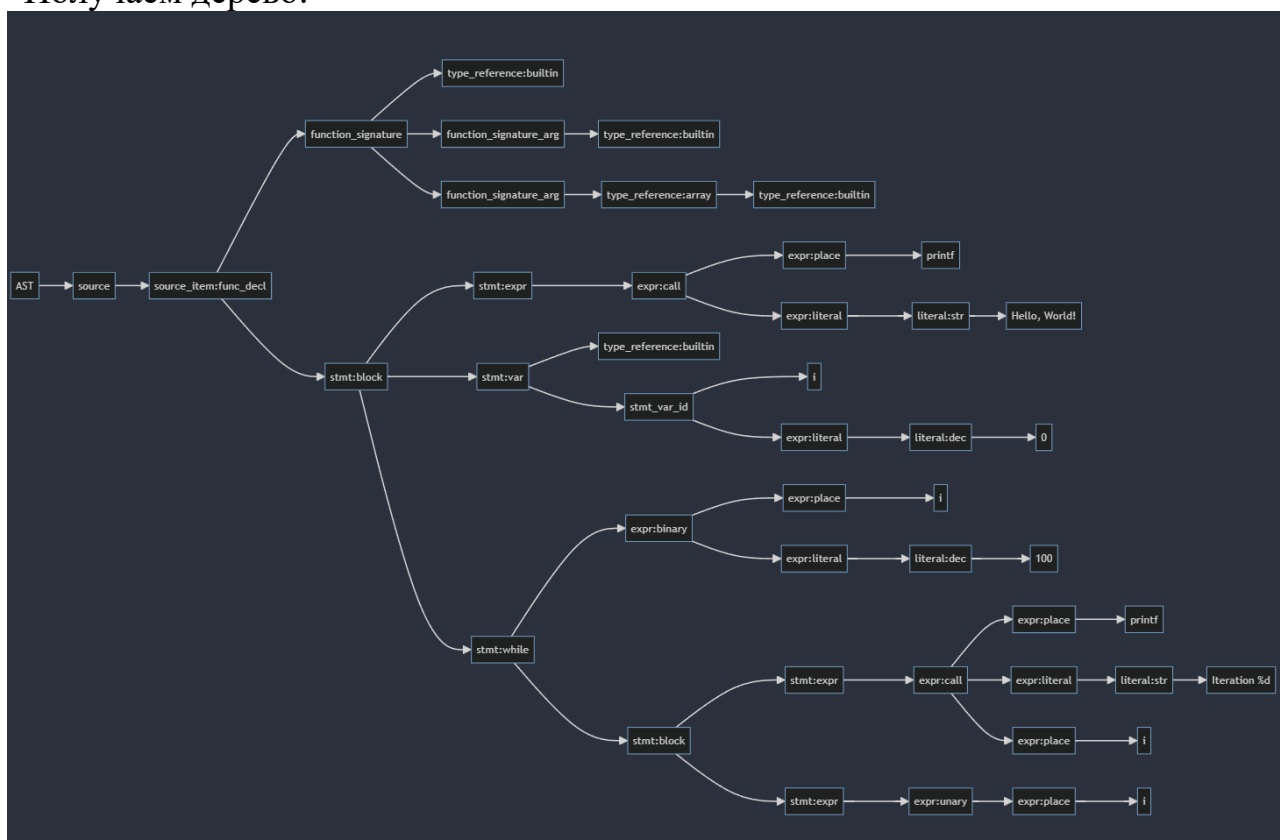
- AST -> <source>
- items[0] -> <source_item:func_decl>
 - signature -> <function_signature> main
 - return_type -> <type_reference:builtin> int
 - args[0] -> <function_signature_arg> argc
 - type -> <type_reference:builtin> int
 - args[1] -> <function_signature_arg> argv
 - type -> <type_reference:array> 1
 - type -> <type_reference:builtin> string
- body -> <stmt:block>
 - stmts[0] -> <stmt:expr>
 - expr -> <expr:call>
 - function -> <expr:place> printf
 - arguments[0] -> <expr:literal>
 - value -> <literal:str> "Hello, World!\n"
 - stmts[1] -> <stmt:var>
 - type -> <type_reference:builtin> int
 - ids[0] -> <stmt_var_id> i
 - value -> <expr:literal>
 - value -> <literal:dec> 0
 - stmts[2] -> <stmt:while>
 - condition -> <expr:binary> <
 - lhs -> <expr:place> i
 - rhs -> <expr:literal>
 - value -> <literal:dec> 100
 - body -> <stmt:block>
 - stmts[0] -> <stmt:expr>
 - expr -> <expr:call>
 - function -> <expr:place> printf
 - arguments[0] -> <expr:literal>
 - value -> <literal:str> "Iteration %d\n"
 - arguments[1] -> <expr:place> i
 - stmts[1] -> <stmt:expr>
 - expr -> <expr:unary> ++
 - expr -> <expr:place> i

Примеры входных данных и результаты обработки

Пример:

```
int main(int argc, string[] argv) {  
    printf("Hello, World!\n");  
  
    int i = 0;  
    while (i < 100) {  
        printf("Iteration %d\n", i);  
        ++i;  
    }  
}
```

Получаем дерево:



Вывод

В ходе данной лабораторной работы я ознакомился с основами языка Си, а также имплементировал парсер для синтаксического анализа.