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

Данилов П. Ю.

P4114

Преподаватель

Кореньков Ю. Д.

Санкт-Петербург, 2024 г.

Цель

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.

Задачи

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

Описание работы

1. Входная информация

Текстовый файл с исходным текстом разбираемой программы. Пример содержимого файла:

```
def main()
    int i = 100;
end
```

Пример простейшего исходного текста программы на разбираемом языке

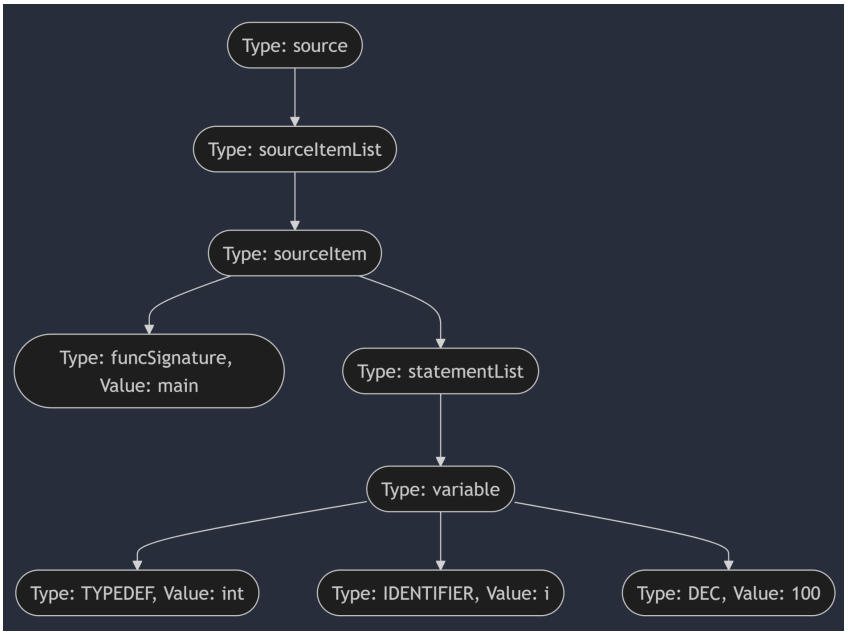
2. Выходная информация

Текстовый файл с описанием дерева разбора введенной программы в формате flowchart. Пример:

```
flowchart TB
node11([Type: source]) --> node10([Type: sourceItemList])
node10([Type: sourceItemList]) --> node9([Type: sourceItem])
node9([Type: sourceItem]) --> node3([Type: funcSignature, Value: main])
node9([Type: sourceItem]) --> node8([Type: statementList])
node8([Type: statementList]) --> node6([Type: variable])
node6([Type: variable]) --> node2([Type: TYPEDEF, Value: int])
node6([Type: variable]) --> node4([Type: IDENTIFIER, Value: i])
node6([Type: variable]) --> node5([Type: DEC, Value: 100])
```

Пример дерева синтаксического разбора для простейшего исходного текста в формате flowchart

Формат flowchart поддерживает визуализацию данных в дерево при помощи внешних утилит. Пример:



Пример визуализации дерева синтаксического разбора для простейшего исходного текста

3. Интерфейс реализуемого модуля

Функция, осуществляющая разбор исходного файла в структуру, содержащую дерево синтаксического анализа, а также список ошибок разбора:

```
ResultTree* parse(FILE *file);

struct ResultTree {
    int size;
    TreeNode **tree;
    char** errors;
    int errorsSize;
};
```

Функция разбора, а также структура результата

Аспекты реализации

1. Модель реализации синтаксического анализа

Анализ осуществляется с использованием лексера flex и парсера bison. Для конфигурации утилит составлены файлы lexems.l, parser.y.
При разборе исходного текста bison и flex инициируют создание узлов дерева синтаксического анализа.

```
[0-9]+ {
```

```
    yylval.node = createNode("DEC", NULL, 0, yytext);
    return DEC;
}
```

Фрагмент содержимого lexems.l

```
source: sourceItemList {{TreeNode* nodes[] = {$1}; $$ = createNode("source", nodes,
sizeof(nodes) / sizeof(nodes[0]), "");}};
```

Фрагмент содержимого parser.y

2. Модель внутреннего представления дерева анализа исходного текста программы

Дерево представлено как массив структур TreeNode. Каждая TreeNode ссылается на произвольный набор потомков TreeNode. На практике количество потомков варьируется от 0 до 3.

```
struct TreeNode {
    char *type;
    TreeNode **children;
    long childrenQty;
    char *value;
    int id;
};
```

Структура узла дерева

Результаты

В результате выполнения работы:

1. Было изучено средство синтаксического анализа bison: принципы его работы, спецификация API и его практическое использование.
2. Была разработана спецификация для flex и bison, позволяющая осуществлять синтаксический анализ исходного текста программ на заданном в варианте языке.
3. Под разработанную спецификацию был реализован модуль, реализующий представление разобранных конструкций в формате дерева синтаксического анализа.
4. Реализована тестовая программа, осуществляющая вызов интерфейса формирования дерева разбора по набору аргументов в формате [входной файл, выходной файл].
5. Был составлен исходный текст программы на языке из варианта, на основе которой было произведено тестирование разработанных модулей. Тестирование показало работоспособность модулей.

Исходный код разработанного решения: https://github.com/47iq/spo

Выводы

В итоге цель работы можно считать успешно выполненной, так как было разработано решение, осуществляющее разбор текста на исходном языке из варианта в формате дерева с узлами; при этом решение поддерживает вывод разбора в формате, поддерживающем визуализацию дерева. Ручное тестирование разработанных модулей с использованием составленных мной примеров программ на языке из варианта показало его работоспособность. Проблемы разбора в итоговой реализации не были обнаружены.

В ходе выполнения работы я изучил принципы проведения синтаксического анализа исходного текста программ, а также на практике реализовал подход к разбору исходного текста заданного формата при помощи flex и bison. Также в ходе изучения материалов по теме я ознакомился с историей развития направления синтаксического анализа, а также с альтернативными решениями для разбора.