МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине

“Низкоуровневое программирование”

Вариант № 5

AQL

Студент:

Чухно Матвей
Романович

Группа P33312

Преподаватель:

Кореньков Юрий Дмитриевич

Санкт-Петербург, 2023

Задание:

Использовать средство синтаксического анализа по выбору, реализовать модуль для разбора некоторого достаточного подмножества языка запросов по выбору в соответствии с вариантом формы данных. Должна быть обеспечена возможность описания команд создания, выборки, модификации и удаления элементов данных.

- Изучить выбранное средство синтаксического анализа
    - Средство должно поддерживать программный интерфейс совместимый с языком C
    - Средство должно параметризоваться спецификацией, описывающий синтаксическую структуру разбираемого языка
    - Средство может функционировать посредством кодогенерации и/или подключения необходимых для его работы дополнительных библиотек
    - Средство может быть реализовано с нуля, в этом случае оно должно быть основано на обобщённом алгоритме, управляемом спецификацией
- Изучить синтаксис языка запросов и записать спецификацию для средства синтаксического анализа
    - При необходимости добавления новых конструкций в язык, добавить нужные синтаксические конструкции в спецификацию
    - Язык запросов должен поддерживать возможность описания следующих конструкций: порождение нового элемента данных, выборка, обновление и удаление существующих элементов данных по условию
        - Условия
            - На равенство и неравенство для чисел, строк и булевских значений
            - На строгие и нестрогие сравнения для чисел
            - Существование подстроки
        - Логическую комбинацию произвольного количества условий и булевских значений
        - В качестве любого аргумента условий могут выступать литеральные значения (константы) или ссылки на значения, ассоциированные с элементами данных (поля, атрибуты, свойства)
        - Разрешение отношений между элементами модели данных любых условий над сопрягаемыми элементами данных
        - Поддержка арифметических операций и конкатенации строк не обязательна

- Реализовать модуль, использующий средство синтаксического анализа для разбора языка запросов
    - Программный интерфейс модуля должен принимать строку с текстом запроса и возвращать структуру, описывающую дерево разбора запроса или сообщение о синтаксической ошибке
    - Результат работы модуля должен содержать иерархическое представление условий и других выражений, логически представляющие

собой иерархически организованные данные, даже если на уровне средства синтаксического анализа для их разбора было использовано линейное представление

Описание:

- Запрос

```
struct query_node {
    enum query_node_type query_node_type;
    void *statement;
};
```

- Тип запроса

```
enum query_node_type {
    FOR_QUERY_STATEMENT,
    INSERT_QUERY_STATEMENT,
    CREATE_QUERY_STATEMENT,
    DROP_QUERY_STATEMENT
};
```

- Подзапрос

```
struct subquery_node {
    enum subquery_node_type subquery_type;
    void *statement;
    struct subquery_node *next;
    struct subquery_node *prev;
};
```

- Тип подзапроса

```
enum subquery_node_type {
    FOR_STATEMENT,
    FILTER_STATEMENT,
    TERMINAL_STATEMENT
};
```

- Терминальное выражение

```
struct terminal_statement_node {
    enum AST_NODE_TYPE terminal_stmt_type;
    void *terminal_statement;
};
```

- Типы узлов синтаксического дерева

```
enum AST_NODE_TYPE {
```

```
    AST_NODE_FOR,
    AST_NODE_SELECT,
    AST_NODE_INSERT,
    AST_NODE_UPDATE,
    AST_NODE_REMOVE,
    AST_NODE_CONDITION,
    AST_NODE_FILTER,
    AST_NODE_RETURN,
};
```

Запросы

- Выборка с джоином

```
FOR c IN cars
FILTER c.brand == "AUDI"
RETURN {"model" : c.model, "brand" : c.brand};
```

```
node_type: for_query
table: cars
var: c
subqueries:
   subquery:
      node_type: filter
         condition_type: simple_condition
         operation: ==
            left_value:
               node_type: constant
               const_data_type: string
               value: c.brand
            right_value:
               node_type: constant
               const_data_type: string
               value: "AUDI"
   subquery:
      node_type: return
         return_value:
            node_type: map
            entries:
               entry:
                  key: "model"
                  node_type: constant
                  const_data_type: string
                  value: c.model
               entry:
                  key: "brand"
                  node_type: constant
                  const_data_type: string
                  value: c.brand
```

```
                  entry:
                     key: "model"
                     node_type: constant
                     const_data_type: string
                     value: c.model
                  entry:
                     key: "name"
                     node_type: constant
                     const_data_type: string
                     value: p.name
```

- Выборка с условием

```
FOR c IN cars
FILTER c.brand == "AUDI"
RETURN {"model" : c.model, "brand" : c.brand};
```

```
node_type: for_query
table: cars
var: c
subqueries:
   subquery:
      node_type: filter
         condition_type: simple_condition
         operation: ==
            left_value:
               node_type: constant
               const_data_type: string
               value: c.brand
            right_value:
               node_type: constant
               const_data_type: string
               value: "AUDI"
   subquery:
      node_type: return
         return_value:
            node_type: map
            entries:
               entry:
                  key: "model"
                  node_type: constant
                  const_data_type: string
                  value: c.model
               entry:
                  key: "brand"
                  node_type: constant
                  const_data_type: string
                  value: c.brand
```

- Вставка

```
INSERT { "id": 2, "brand": "AUDI", "model": "RS6", "is_release":
true } INTO cars;
```

```
node_type: insert_query
table: cars
values:
   node_type: map
   entries:
      entry:
         key: "id"
         node_type: constant
         const_data_type: int
         value: 2
      entry:
         key: "brand"
         node_type: constant
         const_data_type: string
         value: "AUDI"
      entry:
         key: "model"
         node_type: constant
         const_data_type: string
         value: "RS6"
      entry:
         key: "is_release"
         node_type: constant
         const_data_type: bool
         value: true
```

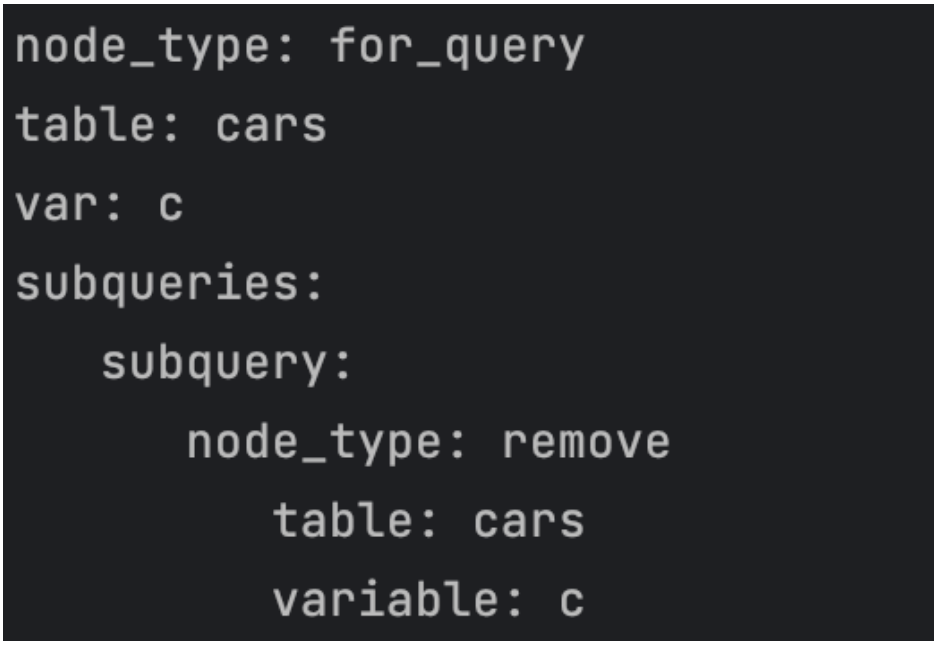
- Обновление

```
FOR c IN cars
UPDATE c WITH {"id": 1, "brand": "BMW"}
IN cars;
```

```
node_type: for_query
table: cars
var: c
subqueries:
   subquery:
      node_type: update
         table: cars
         variable: c
            node_type: map
            entries:
               entry:
                  key: "id"
                  node_type: constant
                  const_data_type: int
                  value: 1
               entry:
                  key: "brand"
                  node_type: constant
                  const_data_type: string
                  value: "BMW"
```

- Удаление

```
FOR c IN cars
REMOVE c IN cars;
```



- Создание таблицы

```
CREATE TABLE cars {"id": int, "brand": string, "model": string,
"is_release": bool };
```

```
node_type: create_table_query
table: cars
schema:
   node_type: map
   entries:
      entry:
         node_type: map_entry
         key: "id"
         value:
            node_type: constant
            const_type: reference
            const_data_type: int
      entry:
         node_type: map_entry
         key: "brand"
         value:
            node_type: constant
            const_type: reference
            const_data_type: string
      entry:
         node_type: map_entry
         key: "model"
         value:
            node_type: constant
            const_type: reference
            const_data_type: string
      entry:
         node_type: map_entry
         key: "is_release"
         value:
            node_type: constant
            const_type: reference
            const_data_type: bool
```

- Удаление таблицы

```
DROP TABLE cars;
```

```
node_type: drop_table_query
table: cars
```

Выводы:

В ходе выполнения данной лабораторной работы были изучены инструменты bison и flex для лексического разбора языка, его токенизации и дальнейшего парсинга. Была создана грамматика для разбора языка, а также была сделана структура абстрактного синтаксического дерева, которая содержит всю информацию о полученном запросе.