**Лабораторная работа 16 (6 часов)**

**Языки программирования**

**Разработка синтаксического анализатора**

1. Используйте материал лекций № 16-18.
2. Используйте результаты лабораторных работ № 13-15.
3. Создайте проект(VS2012, C++, консольное приложение) с именем **LPLab16.**
4. Разработайте синтаксический анализатор для языка SVV-2015 (описан в лекциях и задании к лабораторной работе 14).
5. В контрольном примере в качестве входных данных используйте таблицу лексем, полученную в лабораторной работе 14.
6. Для представления грамматики языка (в форме Грейбах) SVV-2015 создайте структуры следующей спецификации (рис.1). На рис. 2 представлен пример (фрагмент) представления грамматики для языка SVV-2015 с помощью структуры **Greibach.**

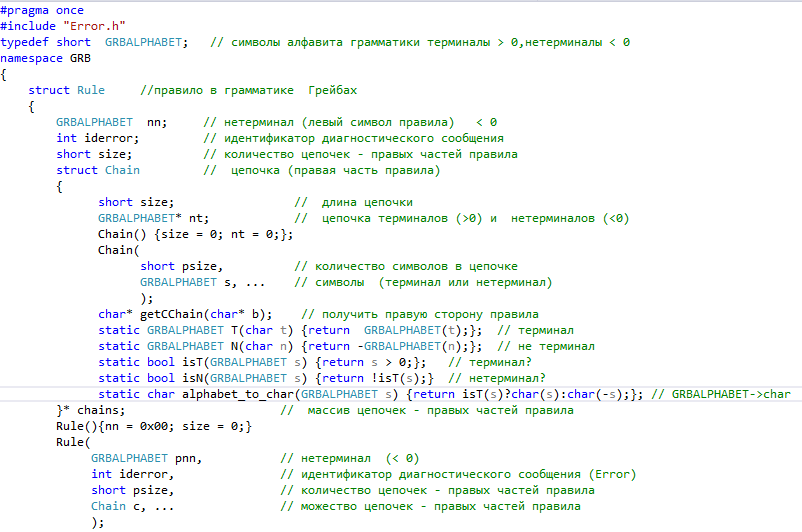


Рис.1. Спецификация структуры для представления грамматики в форме Грейбах.

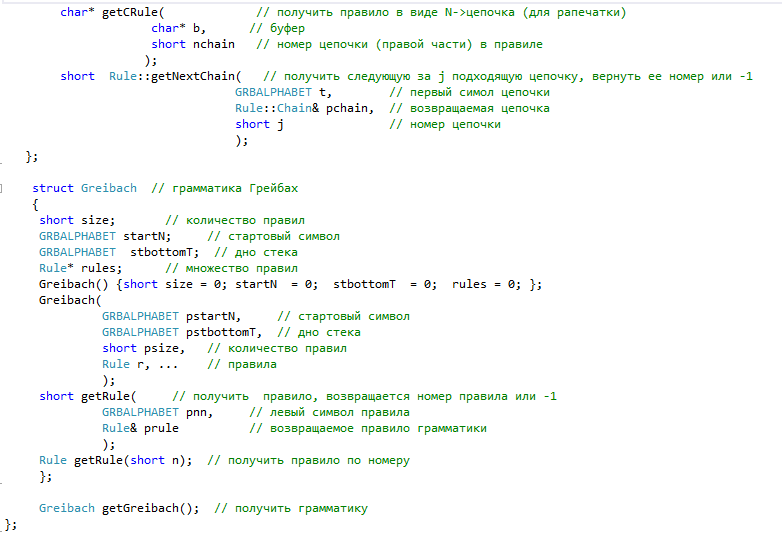


Рис.1. Спецификация структуры для представления грамматики в форме Грейбах (продолжение).

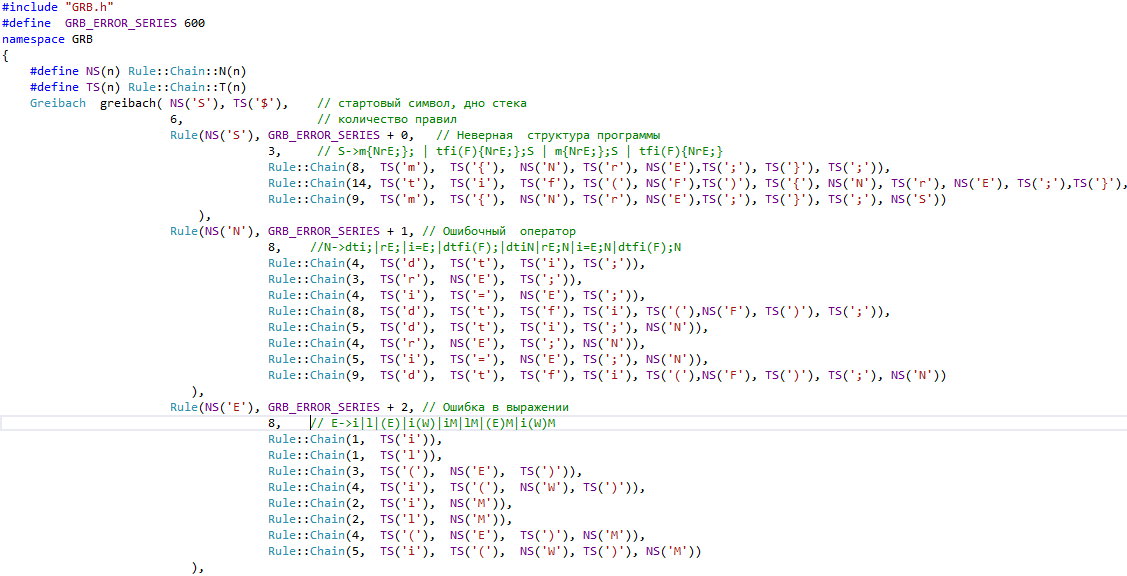


Рис.2. Фрагмент программы, демонстрирующий представление грамматики языка SVV-2015 с помощью структуры **Greibach**.

1. Табл. 1 описывает назначение структур, приведенных на рис.1 и 2.

Таблица 1. Описание структур, для представления грамматики

|  |  |
| --- | --- |
| Структура | Описание |
| **Greibach** | Структура: представление грамматики. Все символы (алфавит) грамматики представляются в формате **GRBALPHABET** (short). Причем терминалы – положительные значения, нетерминалы – отрицательные.  Структура включает:  - множество правил: переменная **rules(**типа структура **Rule);**  - количество правил: переменная **size** (short);  - стартовый символ грамматики: **startN** (**GRBALPHABET**);  - служебный символ (дно стека и последняя лексема таблице лексем): **stbottomT** (**GRBALPHABET**);  - два конструктора;  - методы **getRule**:1) позволяет получить номер правила или -1 (к точке возврата) и правило (второй параметр типа Rule) по левому символу правила (первый параметр); 2) позволяет получить правило (возвращает к точке вызова параметр типа Rule) по его номеру. |
| **Rule** | Структура: представление одного правила, имеющего вид A->xxx|yyy|….  Структура включает:   * нетерминал – левый символ правила : **nn** (**GRBALPHABET**); * идентификатор ошибки, связанной с правилом: **iderror** (int) – код ошибки в подсистеме **Error**; * количество цепочек в правой стороне правила: **size** (short); * цепочки-правые стороны правила: **chains** (типа **Rule::Chain**); * два конструктора; * метод **getCRule**: позволяет получить правило в виде строки вида N->цепочка (в символьном ASCII-виде, для отображения); * метод **getNextChain**:позволяет найти следующую за заданным номером (3й параметр **j** типа short)цепочку (параметр **pchain** типа **Rule::Chain**) и ее номер (к точке возврата типа short) |
| **Rule::Chain** | Структура: представление цепочки – правой стороны правила.  Структура включает:   * размер цепочки: **size** (short) в символах; * цепочка: **nt** (**GRBALPHABET**); * два конструктора; * метод **getCChain**: позволяет получить строку-цепочку в символьном виде для отображения; * методы **T** и **N**:преобразовывают ASCI-символы в **GRBALPHABET-**символы (терминалы и нетерминалы); * методы **isT** и **isN**: проверяют является **GRBALPHABET-**символ терминалом или нетерминалом; * метод **aplphabet\_to\_char**: преобразует заданный (параметр) **GRBALPHABET-**символ в ASCII-символ. |

1. Настройте таблицу сообщений (подсистема **Error**) так, чтобы диагностические сообщения, связанные с грамматикой языка SVV-2015содержались в таблице сообщений подсистемы **Error** в диапазоне 600 - 699. На рис. 3 приведен пример описания (в подсистеме Error) таких диагностических сообщений.



Рис.3. Фрагмент таблицы диагностических сообщений об ошибках, используемых синтаксическим анализатором

1. Разработайте структуры: **Rule, Rule::Chain** и **Greibach** для представления грамматики языка SVV-2015. Опишите грамматику с помощью структур примерно так, как это сделано на рис. 2.
2. Для моделирования конечного магазинного автомата создайте структуры по следующей спецификации (рис.4).

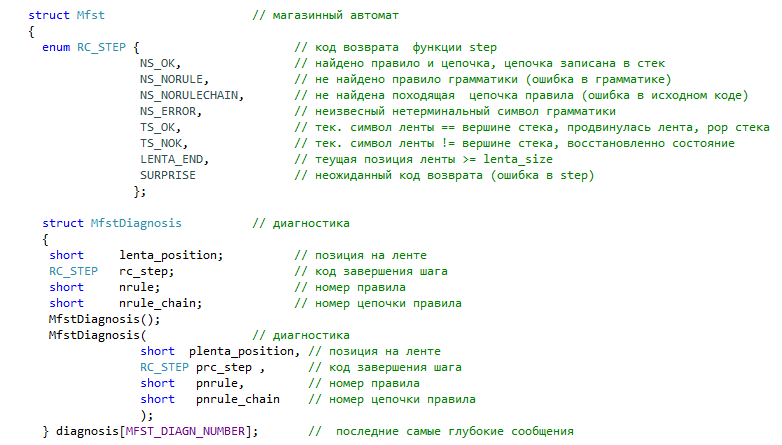
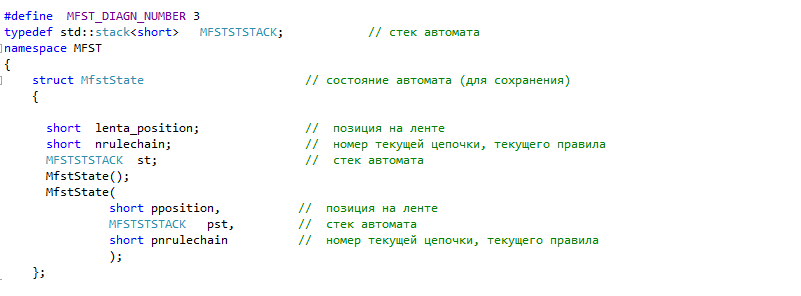


Рис.4. Спецификация структуры **Mfst** для моделирования магазинного конченого автомата

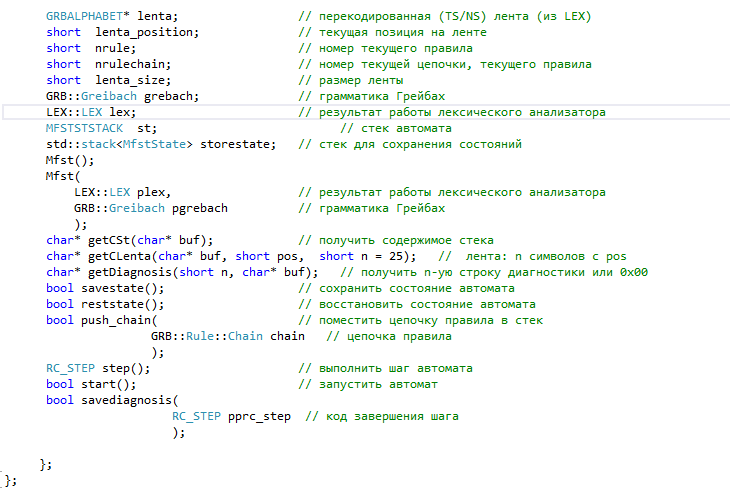


Рис.4. Спецификация структуры **Mfst** для моделирования магазинного конченого автомата (продолжение)

1. Табл. 2 описывает назначение структур, приведенных на рис.4.

Таблица 2. Описание структур, для представления магазинного конечного автомата

|  |  |
| --- | --- |
| Структура | Описание |
| **MfstState** | Структура: для сохранения состояния автомата; сохранять состояние автомата необходимо для того, чтобы иметь возможность к этому состоянию вернуться и осуществить альтернативный вариант синтаксического разбора (в силу недетерминированности автомата).  Структура включает:   * текущую позицию на входной ленте автомата: **lenta\_position** (short); * номер текущей цепочки, текущего правила: **nrulechain** (short); * стек автомата с содержимым на момент сохранения **st** (**MFSTSTACK**); * два конструктора. |
| **Mfst** | Структура: представление магазинного конечного автомата.  Структура включает:   * перечисление, содержащее возможные коды возврата метода step: **RC\_STEP**; * массив структур для строк диагностики: **diagnosis (MstDiagnosis**,описание ниже**)**; * входную ленту: **lenta (GRBALPHABET\*)**; * текущая позиция на входной ленте: **lenta\_position** (short); * номер текущего правила грамматики: **nrule** (short); * номер текущей цепочки текущего правила грамматики: **nrulechain** (short); * количество символов на ленте: **lenta\_size** (short); * грамматика языка: **grebach (**GRB::Greibach); * результат, предварительно выполненного лексического анализа (таблицы лексем и идентификаторов): **lex** (**LEX::LEX**); * стек автомата: **st** (**MFSSTATACK**); * стек для хранения состояний (структур **MfstState**)автомата: **storestate** (std::stack< **MfstState**>); * два конструктора; * функция **getCSt:** принимает один параметр – буфер; заполняет буфер содержимым стека (в формате ASCII-строки) для отображения, в конце 0x00; возвращает к точке вызова указатель на буфер; * функция **getCLenta:** заполняет буфер (первый параметр) содержимым ленты с заданной позиции (второй параметр) заданное количество символов (третий параметр) в формате ASCII-строки для отображения, в конце строки 0x00; возвращает к точке вызова указатель на строки буфер; * функция **getDiagnosis:** по заданному номеру (первый параметр) строки диагностики записывает строку в буфер (второй параметр) в формате ASCII-строки для отображения и возвращает указатель на буфер; * функция **savestate:** сохраняет текущее состояние автомата в **storestate,** всегдавозвращает **true**; * функция **reststate:** восстанавливает последнее сохраненное состояние автомата из **storestate**, возвращает **true**, если восстановление выполнено (есть данные для восстановления); * функция **push\_chain:** помещает реверс цепочки (единственный параметр) в стек автомата**,** всегдавозвращает **true**; * функция **step:** выполняет такт работы автомата, формирует диагностические сообщения, осуществляет отладочный вывод на консоль; * функция **start:** запускает работу автомата, в цикле выполняет функцию **step**, осуществляет вывод диагностических сообщений; * функция **savediagnosis:** сохраняет в массиве **diagnosis** строку диагностики; в массиве **diagnosis** сохраняются диагностические сообщения в порядке убывания позиции ленты (вызвавшей диагностику) и только в равным длине массива (макрос **MFST\_DIAGN\_NUMBER**). |
| **MfstDiagnosis** | Структура (внутренняя для **Mfst**): представление строки диагностики.  Структура включает:   * позиция входной ленты: **lenta\_position** (short); * код возврата, сформированный функцией **step**; * номер действующего на момент диагностики правила грамматики: **nrule** (short); * номер текущей цепочки действующего на момент диагностики правила грамматики: **nrule\_chain**(short). * два конструктора. |

1. Вызов лексического анализатора выполните в следующем виде (рис.5).

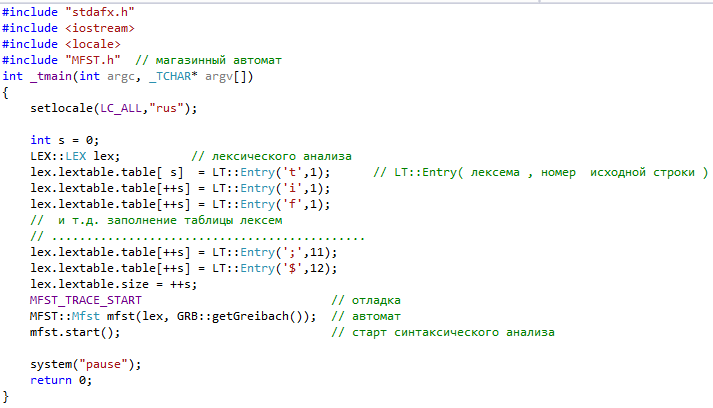
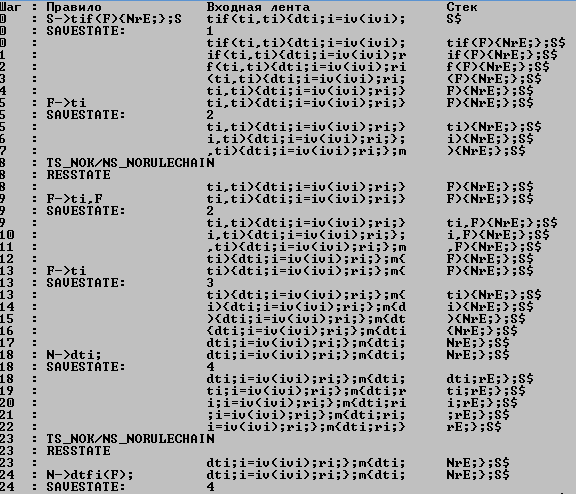


Рис.5. Подготовка таблицы лексем и вызов синтаксического анализатора

1. В результате работы синтаксического анализа на консоль должна выводиться трассировка каждого шага содержащая: номер шага, действующее правило (левые и правые части),состояние ленты и стека. Кроме того, в трассировке должны быть отражены операции сохранения и восстановления состояния автомата. В случае обнаружения ошибки должны быть отражены соответствующие диагностические сообщения (см. рис. 6 и 7).



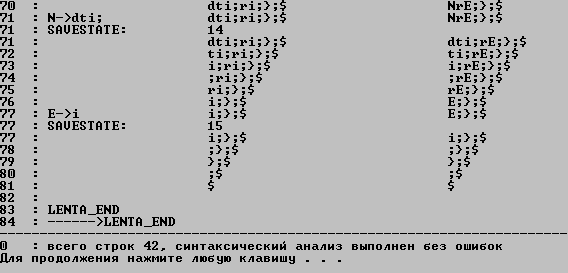
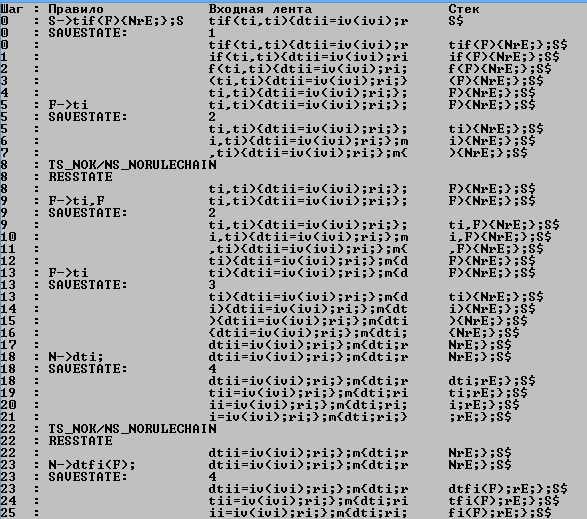


Рис.6. Пример отладочного вывода трассировки синтаксического разбора и диагностики



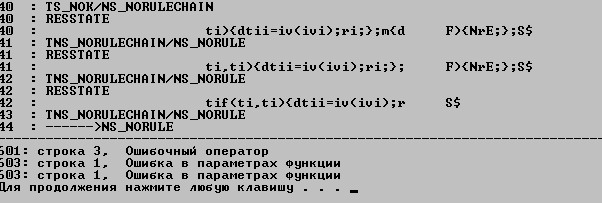


Рис.7. Пример отладочного вывода трассировки синтаксического разбора и диагностики