Министерство науки и высшего образования Российской Федерации

Новосибирский государственный технический университет

Кафедра ТПИ

ЯЗЫКИ ПРОГРАММИРОВАНИЯ И МЕТОДЫ ТРАНСЛЯЦИИ

Лабораторная работа № 3

# Разработка и реализация блока синтаксического анализа

Факультет: ПМИ Преподаватель:

Еланцева И.Л.

Группа: ПМ-81

Студенты: Ефремов А. А.,

Ртищева К. С.

Бригада: 1

Вариант: 1

Новосибирск

2021

1. **Цель работы**

Изучить табличные методы синтаксического анализа. Получить представление о методах диагностики и исправления синтаксических ошибок. Научиться проектировать синтаксический анализатор на основе табличных методов.

1. **Условие задачи**

Подмножество языка С++ включает:

* данные типа int;
* инструкции описания переменных;
* операторы присваивания, if, if- else любой вложенности и в любой последовательности;
* операции +, – , \*, ==, != , < , /.

В соответствии с выбранным вариантом задания к лабораторным работам реализовать синтаксический анализатор с использованием одного из табличных методов (LL-, LR-метод, метод предшествования).

Этапы проектирования синтаксического анализатора:

1. Сконструировать КС-грамматику в соответствии с вариантом задания.
2. В случае несоответствия построенной грамматики требованиям выбранного табличного метода разбора следует провести эквивалентные преобразования грамматики либо выбрать другой метод разбора.
3. Построить таблицу разбора и запрограммировать драйвер, реализующий работу с этой таблицей.

Исходные данные – файл токенов, таблицы лексем.

Результатом работы синтаксического анализатора является:

* синтаксическое дерево или постфиксная запись;
* файл сообщений об ошибках. В лабораторной работе необходимо реализовать возможности табличного метода по диагностике и исправлению синтаксических ошибок в исходной программе.

1. **Вид, структура входных и выходных данных**

**Структура постфиксной записи условного оператора**

*if (A) { B } else { C }*

*ОПЗ: A m1 CJF B m2* UJ *m1: С m2:*

*if (A) { B }*

*ОПЗ: A m1 CJF B m1:*

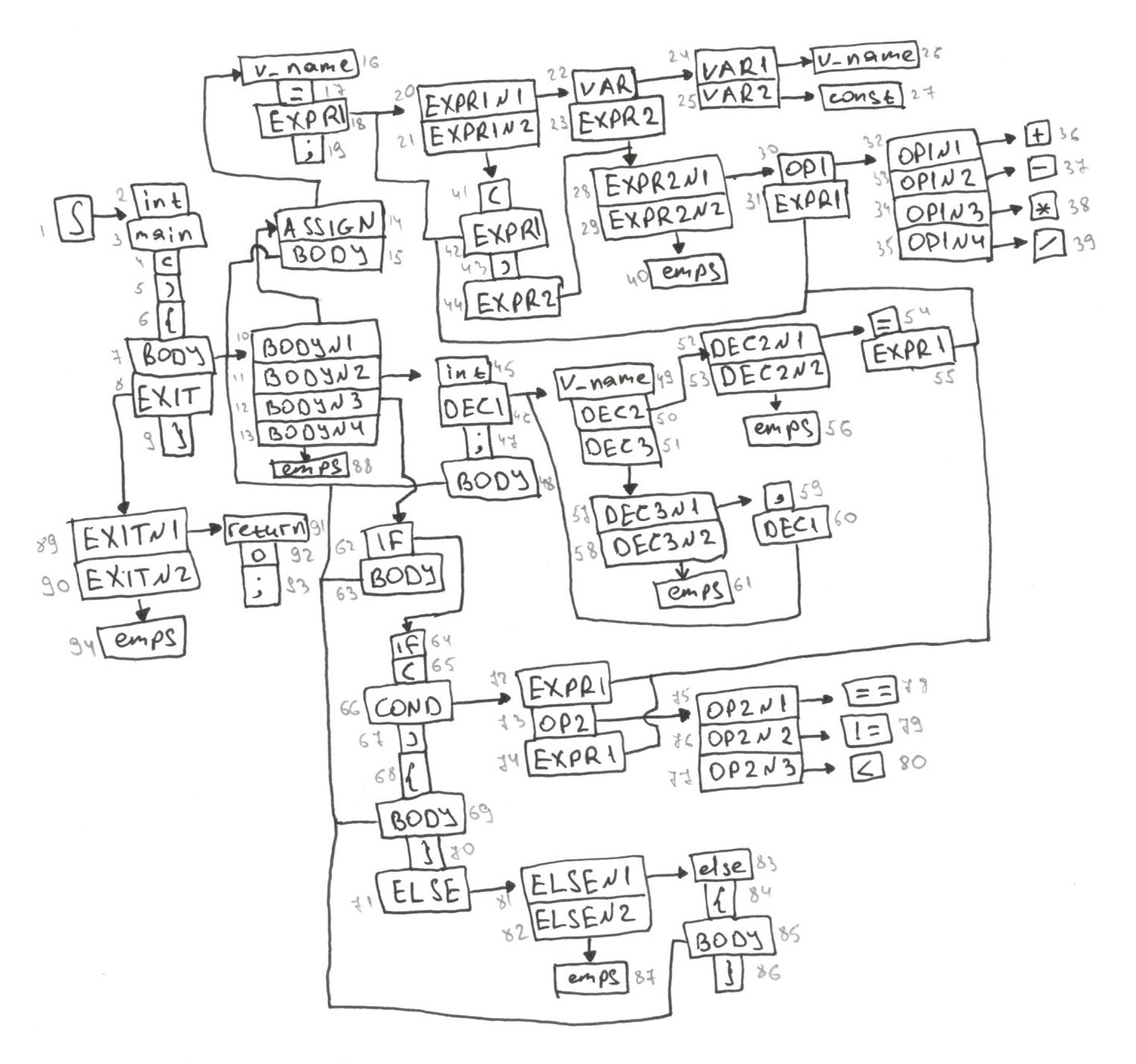
A – логическое выражение, B и C – операторы;

*CJF* – условный переход по значению ложь, UJ – безусловный переход.

1. **Стратегия разбора**
2. **Грамматика входного языка**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Начальный символ | | | | | |
| S -> int main ( ) { BODY EXIT } | | | | |  |
| Операторы | | | | | |
| Арифметические | | Логические | | |  |
| 1 | OP1 -> + | 1 | | OP2 -> == |  |
| 2 | OP1 -> - | 2 | | OP2 -> != |  |
| 3 | OP1 -> \* | 3 | | OP2 -> < |  |
| 4 | OP1 -> / |  | |  |  |
| Переменные и константы | | | | | |
| VAR -> VAR1 | | | | | Имя переменной |
| VAR -> VAR2 | | | | | Значение константы |
| VAR1 -> v\_name | | | VAR2 -> const | |  |
| Выражение | | | | | |
| 1 | EXPR1 -> VAR EXPR2 | | | | Переменная и константа и дальнейшая операция |
| 2 | EXPR1 -> (EXPR1) EXPR2 | | | | Выражение в скобках |
| 1 | EXPR2 -> OP1 EXPR1 | | | | Бинарный оператор и выражение |
| 2 | EXPR2 -> emps | | | | Конец выражения |
| Тело программы | | | | | |
| 1 | BODY -> ASSIGNMENT BODY | | | | Операции присваивания |
| 2 | BODY -> INT DEC1; BODY | | | | Операции объявления |
| 3 | BODY -> IF BODY | | | | Условный оператор |
| 4 | BODY -> emps | | | | Пустая строка |
| Объявление | | | | | |
| DEC1 -> v\_name DEC2 DEC3 | | | | | Имя переменной |
| 1 | DEC2 -> = EXPR | | | | Результат выражения |
| 2 | DEC2 -> emps | | | | Ничего |
| 1 | DEC3 -> , DEC1 | | | | Еще одна переменная |
| 2 | DEC3 -> emps | | | | Ничего |
| Присваивание | | | | | |
| ASSIGNMENT -> v\_name = EXPR ; | | | | | Имя, тип присваивания, выражение |
| Условный оператор | | | | | |
| IF -> if ( CONDITION ) { BODY } ELSE | | | | |  |
| Условное выражение | | | | | |
| CONDITION -> EXPR1 OP2 EXPR1 | | | | |  |
| Иначе | | | | | |
| 1 | ELSE -> else { BODY } | | | | Действие |
| 2 | ELSE -> emps | | | | Бездействие |
| Выход из программы | | | | | |
| 1 | EXIT -> return 0 ; | | | |  |
| 2 | EXIT -> emps | | | |  |

1. **Схема разбора**

****

1. **Таблица разбора**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Terminals** | **jump** | **accept** | **stack** | **return** | **error** |
| 1 | int | 2 | 0 | 0 | 0 | 1 |
| 2 | int | 3 | 1 | 0 | 0 | 1 |
| 3 | main | 4 | 1 | 0 | 0 | 1 |
| 4 | ( | 5 | 1 | 0 | 0 | 1 |
| 5 | ) | 6 | 1 | 0 | 0 | 1 |
| 6 | { | 7 | 1 | 0 | 0 | 1 |
| 7 | v\_name int if return } | 10 | 0 | 1 | 0 | 1 |
| 8 | return } | 89 | 0 | 1 | 0 | 1 |
| 9 | } | 0 | 1 | 0 | 1 | 1 |
| 10 | v\_name | 14 | 0 | 0 | 0 | 0 |
| 11 | int | 45 | 0 | 0 | 0 | 0 |
| 12 | if | 62 | 0 | 0 | 0 | 0 |
| 13 | return } | 88 | 0 | 0 | 0 | 1 |
| 14 | v\_name | 16 | 0 | 1 | 0 | 1 |
| 15 | v\_name int if return } | 10 | 0 | 0 | 0 | 1 |
| 16 | v\_name | 17 | 1 | 0 | 0 | 1 |
| 17 | = | 18 | 1 | 0 | 0 | 1 |
| 18 | v\_name const ( | 20 | 0 | 1 | 0 | 1 |
| 19 | ; | 0 | 1 | 0 | 1 | 1 |
| 20 | v\_name const | 22 | 0 | 0 | 0 | 0 |
| 21 | ( | 41 | 0 | 0 | 0 | 1 |
| 22 | v\_name const | 24 | 0 | 1 | 0 | 1 |
| 23 | + - \* / ; | 28 | 0 | 0 | 0 | 1 |
| 24 | v\_name | 26 | 0 | 0 | 0 | 0 |
| 25 | const | 27 | 0 | 0 | 0 | 1 |
| 26 | v\_name | 0 | 1 | 0 | 1 | 1 |
| 27 | const | 0 | 1 | 0 | 1 | 1 |
| 28 | + - \* / | 30 | 0 | 0 | 0 | 0 |
| 29 | ; | 40 | 0 | 0 | 0 | 1 |
| 30 | + - \* / | 32 | 0 | 1 | 0 | 1 |
| 31 | v\_name const ( | 20 | 0 | 0 | 0 | 1 |
| 32 | + | 36 | 0 | 0 | 0 | 0 |
| 33 | - | 37 | 0 | 0 | 0 | 0 |
| 34 | \* | 38 | 0 | 0 | 0 | 0 |
| 35 | / | 39 | 0 | 0 | 0 | 1 |
| 36 | + | 0 | 1 | 0 | 1 | 1 |
| 37 | - | 0 | 1 | 0 | 1 | 1 |
| 38 | \* | 0 | 1 | 0 | 1 | 1 |
| 39 | / | 0 | 1 | 0 | 1 | 1 |
| 40 | ; | 0 | 0 | 0 | 1 | 1 |
| 41 | ( | 42 | 1 | 0 | 0 | 1 |
| 42 | v\_name const ( | 20 | 0 | 1 | 0 | 1 |
| 43 | ) | 44 | 1 | 0 | 0 | 1 |
| 44 | + - \* / ; | 28 | 0 | 0 | 0 | 1 |
| 45 | int | 46 | 1 | 0 | 0 | 1 |
| 46 | v\_name | 49 | 0 | 1 | 0 | 1 |
| 47 | ; | 48 | 1 | 0 | 0 | 1 |
| 48 | v\_name int if return } | 10 | 0 | 0 | 0 | 1 |
| 49 | v\_name | 50 | 1 | 0 | 0 | 1 |
| 50 | = , ; | 52 | 0 | 1 | 0 | 1 |
| 51 | , ; | 57 | 0 | 0 | 0 | 1 |
| 52 | = | 54 | 0 | 0 | 0 | 0 |
| 53 | , ; | 56 | 0 | 0 | 0 | 1 |
| 54 | = | 55 | 1 | 0 | 0 | 1 |
| 55 | v\_name const ( | 20 | 0 | 0 | 0 | 1 |
| 56 | , ; | 0 | 0 | 0 | 1 | 1 |
| 57 | , | 59 | 0 | 0 | 0 | 0 |
| 58 | ; | 61 | 0 | 0 | 0 | 1 |
| 59 | , | 60 | 1 | 0 | 0 | 1 |
| 60 | v\_name | 49 | 0 | 0 | 0 | 1 |
| 61 | ; | 0 | 0 | 0 | 1 | 1 |
| 62 | if | 64 | 0 | 1 | 0 | 1 |
| 63 | v\_name int if return } | 10 | 0 | 0 | 0 | 1 |
| 64 | if | 65 | 1 | 0 | 0 | 1 |
| 65 | ( | 66 | 1 | 0 | 0 | 1 |
| 66 | v\_name const ( | 72 | 0 | 1 | 0 | 1 |
| 67 | ) | 68 | 1 | 0 | 0 | 1 |
| 68 | { | 69 | 1 | 0 | 0 | 1 |
| 69 | v\_name int if return } | 10 | 0 | 1 | 0 | 1 |
| 70 | } | 71 | 1 | 0 | 0 | 1 |
| 71 | else return } | 81 | 0 | 0 | 0 | 1 |
| 72 | v\_name const ( | 20 | 0 | 1 | 0 | 1 |
| 73 | == != < | 75 | 0 | 1 | 0 | 1 |
| 74 | v\_name const ( | 20 | 0 | 0 | 0 | 1 |
| 75 | == | 78 | 0 | 0 | 0 | 0 |
| 76 | != | 79 | 0 | 0 | 0 | 0 |
| 77 | < | 80 | 0 | 0 | 0 | 1 |
| 78 | == | 0 | 1 | 0 | 1 | 1 |
| 79 | != | 0 | 1 | 0 | 1 | 1 |
| 80 | < | 0 | 1 | 0 | 1 | 1 |
| 81 | else | 83 | 0 | 0 | 0 | 0 |
| 82 | return } | 87 | 0 | 0 | 0 | 1 |
| 83 | else | 84 | 1 | 0 | 0 | 1 |
| 84 | { | 85 | 1 | 0 | 0 | 1 |
| 85 | v\_name int if return } | 10 | 0 | 1 | 0 | 1 |
| 86 | } | 0 | 1 | 0 | 1 | 1 |
| 87 | return } | 0 | 0 | 0 | 1 | 1 |
| 88 | return } | 0 | 0 | 0 | 1 | 1 |
| 89 | return | 91 | 0 | 0 | 0 | 0 |
| 90 | } | 94 | 0 | 0 | 0 | 1 |
| 91 | return | 92 | 1 | 0 | 0 | 1 |
| 92 | 0 | 93 | 1 | 0 | 0 | 1 |
| 93 | ; | 0 | 1 | 0 | 1 | 1 |
| 94 | } | 0 | 0 | 0 | 1 | 1 |

1. **Тексты программ**

***Файл “VarTableRow.h”***

1. **Тестовые примеры**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Входные данные** | **Выходные данные** | **Назначение** |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |
| 10 |  |  |  |
| 11 |  |  |  |
| 12 |  |  |  |