**Алгоритм анализа**

CYK-таблица хранится в векторе-матрице размера N\*N, где N – количество лексических единиц в обрабатываемом предложении.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 |

Дополнительно хранится синтаксическое дерево tree:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| tree | 0 | 1 | 2 | 3 | … |
| from1 |  |  |  |  |  |
| from2 |  |  |  |  |  |
| pos |  |  |  |  |  |

На вход подаётся «x := 0 && 8». Лексический анализатор распознаёт следующие лексемы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 |
| X | := | N | OP | N |

Затем в игру вступает синтаксический анализатор. Для начала правила первого уровня обновляют последнюю, нижнюю строку таблицу:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 |
| X, E | := | N, E | OP | N, E |

После этого начинают работать правила 2 уровня. Первый шаг:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | S2 |  | E1 |  |
| X, E | := | N, E | OP | N, E |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| tree | 0 | 1 | 2 | 3 | … |
| from1 | 21 | 23 |  |  |  |
| from2 | 22 | 24 |  |  |  |
| pos | 16 | 18 |  |  |  |

Второй шаг:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
| S |  | E |  |  |
|  | S2 |  | E1 |  |
| X, E | := | N, E | OP | N, E |

Третий шаг:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | S2 |  |  |  |
| S |  | E |  |  |
|  | S2 |  | E1 |  |
| X, E | := | N, E | OP | N, E |

Четвертый шаг:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S |  |  |  |  |
|  | S2 |  |  |  |
| S |  | E |  |  |
|  | S2 |  | E1 |  |
| X, E | := | N, E | OP | N, E |

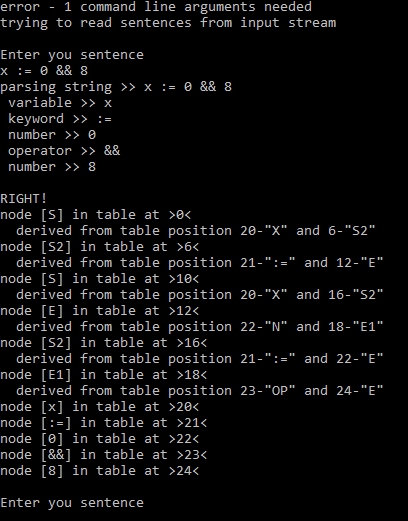
Таблица получена, вектор дерева сортируется:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| tree | 0 | 1 | 2 | 3 | … |
| from1 | 20 | 21 | 20 | 22 |  |
| from2 | 6 | 12 | 16 | 18 |  |
| pos | 0 | 6 | 10 | 12 |  |

Элементы дерева выводятся по порядку в консоль. Графическое отображение дерева выглядело бы так:

C:\Users\Admin\Downloads\Untitled Diagram (5).png

Практически же в консоль выводится список узлов:



**Работа программы**

Программа считывает значение поданного ей аргумента. Если аргумент не подан или файл по адресу аргумента не найден, считывает предложения построчно из окна консоли.

Командой

*C:\Users\Admin\Documents\build-custom\_parser-Release\release\custom\_parser.exe sentence.txt*

Считываем файл со следующим содержанием (в первой строке пробелы, во второй и третьей знаки табуляции).

if x == y

then read y

else skip;

read x; read x; x := 1000 && 1001; read x

Получаем следующий ответ

