Переходы

<q0, x0, s0> -> <q1, s1>  
q0 - начальное состояние

x0 - символ (метка перехода)

s0 - стэковый элемент

x0 э Vt U { ^ }

s0, s1 э G U { ^ }

q0, q1 э Q

-------------------------------------

Ещё переходы (в стэк кладём следующее состояние)

<q0, x0> -> <q1, s1>

q0, q1 э Q

x0 э Vt U { ^ }

s1 э Q U { ^ }

Пример:

1 -> x1 -> 2 | <q1, x1> -> <q2, ^>

или

3 -> x3 -> подавтомат -> 4 | <q3, x3> -><подавтомат, q4>

--------------------------------------

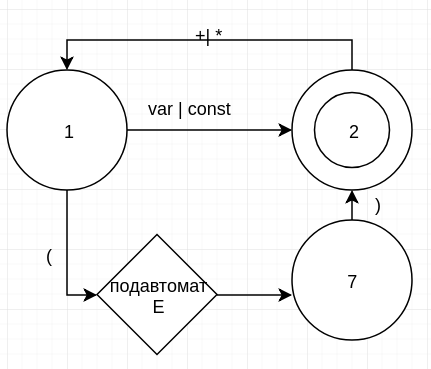
Примеееры

Автомат для арифметического выражения:

<MEXPR> ::= <T> | <E> + <T>

<T> ::= <F> | <T> \* <F>

<F> ::= var | const | (<E>)



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Состояние | Метка | Стэк | Следующее состояние | Семантическая подпрограмма |
| 1 | var |  | 2 |  |
|  | const |  | 2 |  |
|  | ( | 7 | подавтомат E | [!=] error |
| 2 | + |  | 1 |  |
|  | \* |  | 1 | [!=] exit |
| 7 | ) |  | 2 | [!=] error |

a + 5 \* (b + 4)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вход | a | + | 5 | \* | ( | b | + | 4 | ) |  |  |
| Состояние | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 1 | 2 | 7 | 2 |
| Стэк |  |  |  |  |  | 7 | 7 | 7 | 7 |  |  |

((x + 1)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вход | ( | ( | x | + | 1 | ) | ) | ) | errrrrrrrror |  |  |
| Состояние | 1 | 1 | 1 | 2 | 1 | 2 | 7 | 2 | 7 |  |  |
| Стэк |  | 7 | 7 7 | 7 7 | 7 7 | 7 7 | 7 | 7 |  |  |  |

(x + 1))

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вход | ( | x | + | 1 | ) | ) | ) | errrrrrrrrororororororoororo |  |  |  |
| Состояние | 1 | 1 | 2 | 1 | 2 | 7 | 2 |  |  |  |  |
| Стэк |  | 7 | 7 | 7 | 7 |  |  |  |  |  |  |

-----

автомат для языка

<ROOT> ::= <DECL\_LIST> { <IMPL\_LIST> }

<DECL\_LIST> ::= <DECL\_LIST>; <DECL> | <DECL>;

<IMPL\_LIST> ::= <IMPL\_LIST> <IMPL>; | <IMPL>;

<DECL> ::= <TYPE> <VARID\_LIST>

<TYPE> ::= int | single

<VARID\_LIST> ::= id | id, <VARID\_LIST>

<IMPL> ::= id = <EXPR> | <IN> | <OUT> | for id = <E> to <E> do { <IMPL\_LIST> } | if <LOGIC> { <IMPL\_LIST> }

<IN> ::= cin >> id | <IN> >> id

<OUT> ::= cout <<id | <OUT> << id

<LOGIC> ::= <E> <REL> <E>

<REL> ::= < | > | == | != | <= | >=

27

