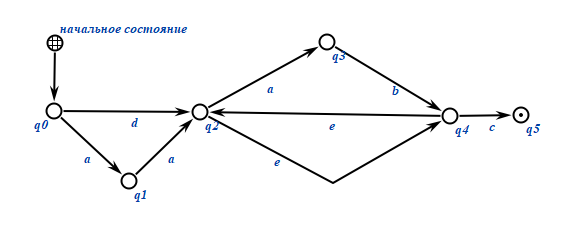
ПЗ 1

(d|a2)(ab)\*c

1. Диаграмма переходов



1. Конечный автомат без e-переходов

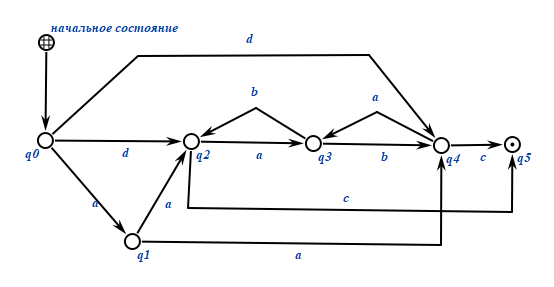
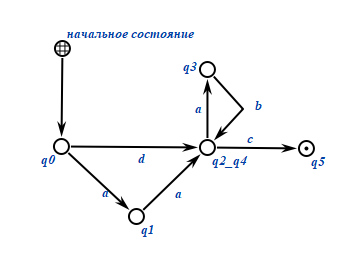


Таблица состояний:

|  | a | b | c | d |
| --- | --- | --- | --- | --- |
| q0 | q1 |  |  | {q2, q4} |
| q1 | {q2, q4} |  |  |  |
| q2 | q3 |  | q5 |  |
| q3 |  | {q2, q4} |  |  |
| q4 | q3 |  | q5 |  |
| q5 |  |  |  |  |

1. Минимизированный детерминированный конечный автомат



1. Примеры:

dc

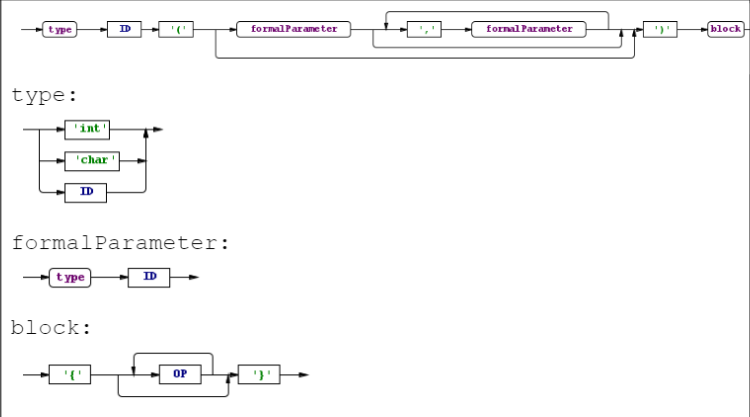
aaababababc

dabc

aac

e

**ПЗ 2**

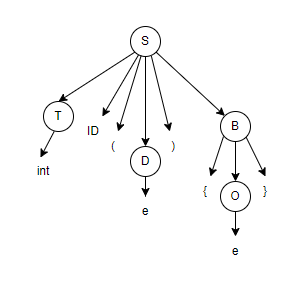


S -> T ID (D) B

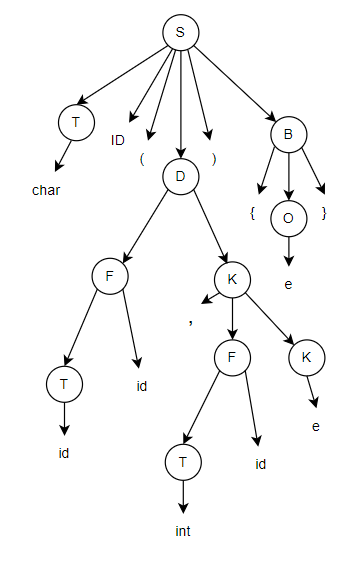
| T -> int  T -> char  T -> ID | D -> F K  D -> e | B -> {O} |
| --- | --- | --- |
|  | K -> e  K -> , F K | O -> e  O -> OP O |
|  | F -> T ID |  |

Примеры:

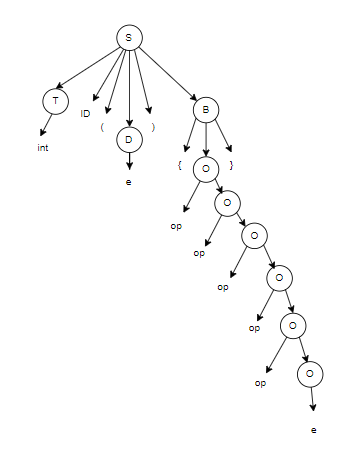
int id () {}



char id (id id , int id) {}



id id () {op op op op op}



1. Нет левой рекурсии
2. Грамматика левофакторизована

**ПЗ 3**

S -> T “ID” “(” D “)” B

| T -> “int”  T -> “char”  T -> “ID” | D -> F K  D -> e | B -> “{” O “}” |
| --- | --- | --- |
|  | K -> e  K -> “,” F K | O -> e  O -> “OP” O |
|  | F -> T “ID” |  |

| **FIRST** | **FOLLOW** |
| --- | --- |
| S = {“int”, “char”, “ID”}  T = {“int”, “char”, “ID”}  D = {“int”, “char”, “ID”, e}  B = {“{”}  K = {e, “,”}  F = {“int”, “char”, “ID”}  O = {e, “OP”} | S = {$}  T = {“ID”}  D = {“)”}  B = {$}  K = {“)”}  F = {“,”, “)”}  O = {“}”} |

|  | “int” | “char” | “ID” | ( | ) | { | } | OP | , | $ |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S | S->T ID (D) B | S->T ID (D)B | S->T ID (D)B |  |  |  |  |  |  |  |
| T | T -> int | T -> char | T -> ID |  |  |  |  |  |  |  |
| D | D -> F K | D -> F K | D -> F K |  | D->e |  |  |  |  |  |
| B |  |  |  |  |  | B->{O} |  |  |  |  |
| K |  |  |  |  | K->e |  |  |  | K->,FK |  |
| F | F -> T ID | F -> T ID | F -> T ID |  |  |  |  |  |  |  |
| O |  |  |  |  |  |  | O->e | O->OP O |  |  |

Примеры:

| int id () {}  $  S$  T ID(D)B$  int ID(D)B$  D)B$  B$  {O}$  O}$  }$  $ | char id (id id , int id) {}  $  S$  T ID(D)B$  char ID(D)B$  D)B$  FK)B$  T ID K)B$  ID ID K)B$  K)B$  ,FK)B$  T ID K)B$  int ID K)B$  K)B$  B$  {O}$  O}$  }$  $ | id id () {op,}  $  S$  T ID(D)B$  ID ID(D)B$  D)B$  B$  {O}$  O}$  op}$  ошибка - лишний терминал “,” |
| --- | --- | --- |

**ПЗ 4**

S -> AB

A -> aAb | Ab | ε

B -> bB | b

Устранение левой рекурсии:

A -> aA’b

A’-> bA’| ε

Левофакторизация:

B -> bB | b

B -> bD

D -> B | ε

Новая грамматика:

S -> AB

A -> aAb | bA | ε

B -> bD

D -> B | ε

| **FIRST** | **FOLLOW** |
| --- | --- |
| S = {a, b}  A = {a, e}  B = {b}  D = {b, e} | S = {$}  A = {b, $}  B = {$}  D = {$} |

Пронумеруем правила:

0)S’ -> S

1. S -> AB
2. A -> aAb
3. A -> bA
4. A -> ε
5. B -> bD
6. D -> B
7. D -> ε

Сформируем множество пунктов:

I0 = {S’ ->・S, S ->・AB, A ->・aAb, A ->・bA, A ->・}

I1 = goto(I0, S) = {S’->S・}

I2 = goto(I0, A) = {S->A・B, B ->・bD}

I3 = goto(I2, B) = {S->AB・}

I4 = goto(I2, b) = {B ->b・D, D ->・B, D ->・, B ->・bD}

I5 = goto(I4, D) = {B ->bD・}

I6 = goto(I4, B) = {D ->B・}

I7 = goto(I0, a) = {A->a・Ab, A ->・aAb, A ->・bA, A ->・}

I8 = goto(I7, A) = {A->aA・b}

I9 = goto(I8, b) = {A->aAb・}

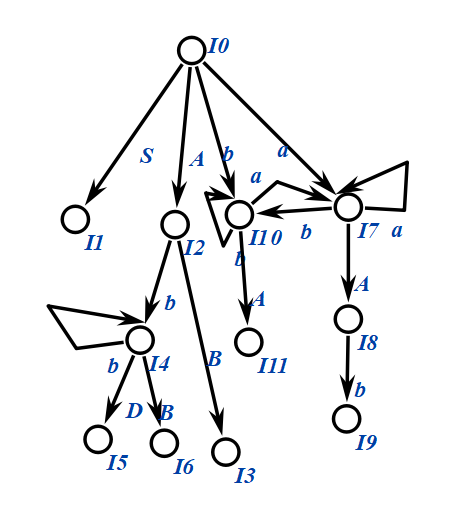
I10 = goto(I0, b) = {A -> b・A, A ->・aAb, A ->・bA, A ->・}

I11 = goto(I10, A) = {A -> bA・}

Таблица переходов

| Состояние | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 |  | S | A |  |  |  |  | a |  |  | b |  |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  | B | b |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  | b | D | B |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  | a | A |  | b |  |
| 8 |  |  |  |  |  |  |  |  |  | b |  |  |
| 9 |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  | a |  |  | b | A |
| 11 |  |  |  |  |  |  |  |  |  |  |  |  |

Диаграмма переходов



| Состояние | action | | | goto | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | $ | A | B | D | S |
| 0 | s7 | s10 | r4 | 2 |  |  | 1 |
| 1 |  |  | accept |  |  |  |  |
| 2 |  | s4 |  |  | 3 |  |  |
| 3 |  |  | r1 |  |  |  |  |
| 4 |  | s4 | r7 |  | 6 | 5 |  |
| 5 |  |  | r5 |  |  |  |  |
| 6 |  |  | r6 |  |  |  |  |
| 7 | s7 | r4 | r4 | 8 |  |  |  |
| 8 |  | s9 |  |  |  |  |  |
| 9 |  | r2 | r2 |  |  |  |  |
| 10 | s7 | r4 | r4 | 11 |  |  |  |
| 11 |  | s3 | r3 |  |  |  |  |

Примеры:

1. bb

| Стек | Входной поток | Комментарии |
| --- | --- | --- |
| $0 | bb$ | *action[I0, b] = S10* |
| $0b10 | b$ | *action[I10, b] = r4*  *goto[I10, A] = 11* |
| $0b10A11 | b$ | *action[I11, b] = s3* |
| $0A8b3 | $ | *action[I3, $] = r1* |
| $0S1 | $ | *goto[I0, S] = s1* |
| $0S1 | $ | *action[I1, $] = accept* |

1. aaa

| Стек | Входной поток | Комментарии |
| --- | --- | --- |
| $0 | aaa$ | *action[I0, a] = S7* |
| $0a7 | aa$ | *action[I7, a] = s7* |
| $0a7a7 | a$ | *action[I7, b] = s7* |
| $0a7a7a7 | $ | *action[I7, $] = r4*  *goto[I7, A] = 8* |
| $0a7a7A8 | $ | *action[I8, $] = нет перехода* |

1. bcbb

| Стек | Входной поток | Комментарии |
| --- | --- | --- |
| $0 | bcbb$ | *action[I0, c] = S10* |
| $0b10 | cbb$ | *action[I10, c] = ???* |

**ПЗ 5**

F1,C3 -> C1

F1:

int a, b, c = 1, d = c \* 3;

char \* s = “строка символов”;

if (s[c + d] == 0) {

a = c + 1;

b = c + 2;

} else {

a = d + 1;

b = d + 2;

}

printf(“Ответ: %d”, a + b);

C3:

Трехадресный код. Четверки

C1:

Синтаксическое дерево

| Номер ячейки | Значение |
| --- | --- |
| D01 | 0 |
| D02 | 1 |
| D03 | 2 |
| D04 | 3 |
| D05 | указатель на строку |
| D06 | “%d” |

Четверки:

|  | op | arg1 | arg2 | result |
| --- | --- | --- | --- | --- |
| (0) | assign | D02 |  | с |
| (1) | \* | с | D03 | t1 |
| (2) | assign | t1 |  | d |
| (3) | assign | D05 |  | s |
| (4) | + | с | d | t2 |
| (5) | [] | s | t2 | t3 |
| (6) | == | t3 | D01 | t4 |
| (7) | if | t4 | (9) |  |
| (8) | goto | (14) |  |  |
| (9) | + | c | D02 | t5 |
| (10) | assign | t5 |  | a |
| (11) | + | c | D03 | t6 |
| (12) | assign | t6 |  | b |
| (13) | goto | (18) |  |  |
| (14) | + | d | D02 | t7 |
| (15) | assign | t7 |  | a |
| (16) | + | d | D03 | t8 |
| (17) | assign | t8 |  | b |
| (18) | param | D06 |  |  |
| (19) | + | a | b | t9 |
| (20) | param | t9 |  |  |
| (21) | call | printf | 2 |  |

Команды:

1. assign – присваивание: аргумент – присваиваемое значение, результат – переменная, к которой присваивается значение;
2. \* – умножение: аргументы - множители;
3. + – сложение: аргументы – слагаемые;
4. [] – доступ к элементу массива: аргумент 1 – массив, аргумент 2 –индекс обращения;
5. == - сравнение. Определяет факт равенства/неравенства двух значений, передающихся в качестве аргументов.
6. if – проверка выполнения/невыполнения условия: аргумент – блок, к которому совершается переход при выполнении условия;
7. goto – безусловный переход;
8. param - параметр функции: аргумент – переменная, передаваемая в качестве параметра, или непосредственно значение;
9. call – вызов функции, аргумент 1 – вызываемая функция, аргумент 2 – список параметров функции;
10. printf – оператор вывода.

Оптимизация:

1.Распространение копий:

c = 1, значит d = c \* 3 можно заменить на d = 3, так как c всегда равно 1.

2. Условие s[c + d] == 0 можно упростить. Так как c = 1 и d = 3, условие станет s[4] == 0.

3. Удаление бесполезного кода в блоках if и else можно заменить переменные a и b на их вычисленные значения, так как c и d фиксированы.

| int a, b, c = 1, d = c \* 3;  char \* s = “строка символов”;  if (s[c + d] == 0) {  a = c + 1;  b = c + 2;  } else {  a = d + 1;  b = d + 2;  }  printf(“Ответ: %d”, a + b); | int a, b, c = 1;  char \* s = “строка символов”;  if (s[c \* 4] == 0) {  a = c + 1;  b = c + 2;  } else {  a = d + 1;  b = d + 2;  }  printf(“Ответ: %d”, a + b); |
| --- | --- |
| int a, b, c = 1;  char \* s = “строка символов”;  if (s[c \* 4] == 0) {  a = c + 1;  b = c + 2;  } else {  a = c \* 3 + 1;  b = c \* 3 + 2;  }  printf(“Ответ: %d”, a + b); |  |

Синтаксическое дерево:

