**Вариант 4**

**Для пункта A**

1. <целое\_знаковое\_число> ::= <знак><цифры> | <цифры>

<знак> ::= ‘+’ | ‘-‘

<цифры> ::= <цифра> | <цифра><цифры>

<цифра> ::= ‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9

**Пример вывода правильной цепочки языка**

Рассмотрим цепочку **-123**.

1. **<целое\_знаковое\_число>**
2. **<знак><цифры>**
3. **-<цифры>**
4. **-<цифра><цифры>**
5. **-1<цифры>**
6. **-1<цифра><цифры>**
7. **-12<цифры>**
8. **-12<цифра>**
9. **-123**
10. **Дерево вывода для цепочки -123**

**Левосторонний вывод**

**<целое\_знаковое\_число>**

**<знак><цифры>**

**-<цифры>**

**-<цифра><цифры>**

**-1<цифры>**

**-1<цифра><цифры>**

**-12<цифры>**

**-12<цифра>**

**-123**

**Правосторонний вывод**

**<целое\_знаковое\_число>**

**<знак><цифры>**

**<знак><цифра><цифры>**

**<знак><цифра><цифра><цифры>**

**<знак><цифра><цифра><цифра>**

**-<цифра><цифры>**

**-<цифра><цифра>**

**-<цифра><цифра>**

**-1<цифра><цифра>**

**-12<цифра>**

**-123**

<целое\_знаковое\_число>

|

+- <знак> -> -

|

+- <цифры>

|

+- <цифра> -> 1

|

+- <цифры>

|

+- <цифра> -> 2

|

+- <цифры>

|

+- <цифра> -> 3

1. Грамматика, определяющая понятие «целое\_знаковое\_число», относится к Контекстно-свободной грамматике (CFG) (Тип 2 по иерархии Хомского).

Обоснование:

Правила продукций имеют вид A -> α, где A — это нетерминальный символ, а α — строка из терминальных и/или нетерминальных символов.

Каждое правило продукции в нашей грамматике соответствует этому виду:

<целое\_знаковое\_число> ::= <знак><цифры> | <цифры>

<знак> ::= '+' | '-'

<цифры> ::= <цифра> | <цифра><цифры>

<цифра> ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

Контекстно-свободные грамматики используются для описания синтаксиса большинства языков программирования и естественных языков, так как они достаточно мощные, чтобы выразить большинство структур, и достаточно простые для эффективного анализа.

**Для пункта b**

a) <функция> ::= <тип> <идентификатор> '(' <параметры> ')' '{' <тело> '}'

<тип> ::= 'int' | 'float' | 'double' | 'char' | 'void'

<идентификатор> ::= <буква> <идентификатор\_символы>

<идентификатор\_символы> ::= <буква> <идентификатор\_символы> | <цифра> <идентификатор\_символы> | ε

<буква> ::= 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z' | 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z' | '\_'

<цифра> ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

<параметры> ::= <параметр> | <параметр> ',' <параметры> | ε

<параметр> ::= <тип> <идентификатор>

<тело> ::= <операторы>

<операторы> ::= <оператор> <операторы> | ε

<оператор> ::= <тип> <идентификатор> ';' | <идентификатор> '=' <выражение> ';' | <выражение> ';'

<выражение> ::= <идентификатор> | <число>

<число> ::= <цифра> <число\_символы>

<число\_символы> ::= <цифра> <число\_символы> | ε

Пример вывода правильной цепочки языка

int sum(int a, int b) {

int result;

result = a + b;

}

<функция>

int <идентификатор> '(' <параметры> ')' '{' <тело> '}'

int sum '(' <параметры> ')' '{' <тело> '}'

int sum '(' <параметр> ',' <параметр> ')' '{' <тело> '}'

int sum '(' int <идентификатор> ',' int <идентификатор> ')' '{' <тело> '}'

int sum '(' int a ',' int b ')' '{' <тело> '}'

int sum '(' int a ',' int b ')' '{' <операторы> '}'

int sum '(' int a ',' int b ')' '{' <оператор> <операторы> '}'

int sum '(' int a ',' int b ')' '{' int <идентификатор> ';' <операторы> '}'

int sum '(' int a ',' int b ')' '{' int result ';' <операторы> '}'

int sum '(' int a ',' int b ')' '{' int result ';' <оператор> <операторы> '}'

int sum '(' int a ',' int b ')' '{' int result ';' <идентификатор> '=' <выражение> ';' <операторы> '}'

int sum '(' int a ',' int b ')' '{' int result ';' result '=' <выражение> ';' <операторы> '}'

int sum '(' int a ',' int b ')' '{' int result ';' result '=' a '+' b ';' ε '}'

b) Дерево вывода для цепочки

Левосторонний вывод

<функция>

int <идентификатор> '(' <параметры> ')' '{' <тело> '}'

int sum '(' <параметры> ')' '{' <тело> '}'

int sum '(' <параметр> ',' <параметр> ')' '{' <тело> '}'

int sum '(' int <идентификатор> ',' int <идентификатор> ')' '{' <тело> '}'

int sum '(' int a ',' int b ')' '{' <тело> '}'

int sum '(' int a ',' int b ')' '{' <операторы> '}'

int sum '(' int a ',' int b ')' '{' <оператор> <операторы> '}'

int sum '(' int a ',' int b ')' '{' int <идентификатор> ';' <операторы> '}'

int sum '(' int a ',' int b ')' '{' int result ';' <операторы> '}'

int sum '(' int a ',' int b ')' '{' int result ';' <оператор> <операторы> '}'

int sum '(' int a ',' int b ')' '{' int result ';' <идентификатор> '=' <выражение> ';' <операторы> '}'

int sum '(' int a ',' int b ')' '{' int result ';' result '=' <выражение> ';' <операторы> '}'

int sum '(' int a ',' int b ')' '{' int result ';' result '=' a '+' b ';' ε '}'

Правосторонний вывод

<функция>

<тип> <идентификатор> '(' <параметры> ')' '{' <тело> '}'

int <идентификатор> '(' <параметры> ')' '{' <тело> '}'

int sum '(' <параметры> ')' '{' <тело> '}'

int sum '(' <параметр> ',' <параметры> ')' '{' <тело> '}'

int sum '(' int <идентификатор> ',' <параметр> ')' '{' <тело> '}'

int sum '(' int a ',' int <идентификатор> ')' '{' <тело> '}'

int sum '(' int a ',' int b ')' '{' <тело> '}'

int sum '(' int a ',' int b ')' '{' <операторы> '}'

int sum '(' int a ',' int b ')' '{' <оператор> <операторы> '}'

int sum '(' int a ',' int b ')' '{' int <идентификатор> ';' <операторы> '}'

int sum '(' int a ',' int b ')' '{' int result ';' <операторы> '}'

int sum '(' int a ',' int b ')' '{' int result ';' <оператор> '}'

int sum '(' int a ',' int b ')' '{' int result ';' <идентификатор> '=' <выражение> ';' '}'

int sum '(' int a ',' int b ')' '{' int result ';' result '=' <выражение> ';' '}'

int sum '(' int a ',' int b ')' '{' int result ';' result '=' a '+' b ';' '}'

<функция>

|

+- <тип> -> int

|

+- <идентификатор> -> sum

|

+- '('

|

+- <параметры>

| |

| +- <параметр>

| | |

| | +- <тип> -> int

| | |

| | +- <идентификатор> -> a

| |

| +- ','

| |

| +- <параметр>

| |

| +- <тип> -> int

| |

| +- <идентификатор> -> b

|

+- ')'

|

+- '{'

|

+- <тело>

| |

| +- <операторы>

| |

| +- <оператор>

| | |

| | +- <тип> -> int

| | |

| | +- <идентификатор> -> result

| | |

| | +- ';'

| |

| +- <операторы>

| |

| +- <оператор>

| | |

| | +- <идентификатор> -> result

| | |

| | +- '='

| | |

| | +- <выражение> -> a + b

| | |

| | +- ';'

| |

| +- ε

|

+- '}'

c)

Определение типа грамматики по иерархии Хомского

Грамматика, определяющая понятие «функция языка С++», относится к Контекстно-свободной грамматике (CFG) (Тип 2 по иерархии Хомского).

Обоснование:

Правила продукций имеют вид A -> α, где A — это нетерминальный символ, а α — строка из терминальных и/или нетерминальных символов.

Каждое правило продукции в нашей грамматике соответствует этому виду:

<функция> ::= <тип> <идентификатор> '(' <параметры> ')' '{' <тело> '}'

<тип> ::= 'int' | 'float' | 'double' | 'char' | 'void'

<идентификатор> ::= <буква> <идентификатор\_символы>

<идентификатор\_символы> ::= <буква> <идентификатор\_символы> | <цифра> <идентификатор\_символы> | ε

<параметры> ::= <параметр> | <параметр> ',' <параметры> | ε

<параметр> ::= <тип> <идентификатор>

<тело> ::= <операторы>

<операторы> ::= <оператор> <операторы> | ε

<оператор> ::= <тип> <идентификатор> ';' | <идентификатор> '=' <выражение> ';' | <выражение> ';'

<выражение> ::= <идентификатор> | <число>

<число> ::= <цифра> <число\_символы>

<число\_символы> ::= <цифра> <число\_символы> | ε

Контекстно-свободные грамматики используются для описания синтаксиса большинства языков программирования, поскольку они достаточно мощные, чтобы выразить большинство структур, и достаточно простые для эффективного анализа.