Программа: главная программа языка C++. Допускается описание функций без параметров, функции возвращают значение.

Типы данных: целые разной длинны (в том числе int, short int, long int, long long int).

Операции: арифметические, сравнения.

Операторы: присваивания и do{} while().

Операнды: простые переменные и константы.

Константы: целые в 10 с/с и 16 с/с.

<S> → < S ><описание функций>|< S ><описание переменных>|< S ><main>| ε

<main>→<тип>main()<состовной оператор>

<описание переменных>→<тип><список переменных>;

<тип>→int|short int|long int|long long int

<список переменных>→<список переменных>,<переменная>|<переменная>

<переменная>→<идентификатор>|<идентификатор>=<выражение>

<описание функций>→<тип><идентификатор>()<состовной оператор>

<состовной оператор>→{<операторы и описания>}

<операторы и описания>→<операторы и описания><оператор>|<операторы и описания>< описание переменных >| ε

<оператор>→<состовной оператор>|<присваивание>|<do{}while()>|<вызов функции>|<return>|;

<do{}while()>→do<состовной оператор>while(<выражение>);

<return>→return<выражение>;

<присваивание>→<идентификатор>=<выражение>;

< выражение>→< выражение> > < слагаемое>|< выражение> >= < слагаемое> >| < выражение> < < слагаемое>|< выражение> <= < слагаемое>|< выражение> == < слагаемое>|< выражение> != < слагаемое>|+ < слагаемое>|- < слагаемое>|< слагаемое>

< слагаемое>→< слагаемое> + <множитель>|< слагаемое> - <множитель>|<множитель>

<множитель>→<множитель>\*<эл.выражение>|<множитель>/<эл.выражение>|<эл. выражение>

<эл. выражение>→<константа>|<вызов функции>|(<выражение>)

<вызов функции>→<идентификатор>()

<идентификатор>→<идентификатор><буква>|< идентификатор ><цифра>|<буква>

<константа>→<целая константа в 10 с/с>|<целая константа в 16 с/с>

<целая константа в 10 с/с>→<цифры не 0><цифры>|<цифра>

<целая константа в 16 с/с>→< целая константа в 16 с/с><цифры>|< целая константа в 16 с/с><буквы в 16 с/с>|0x

<цифры>→><цифры><цифра>| ε

<буквы в 16 с/с>→a|b|c|d|e|f

<цифры не 0>→1|2|…|9

<буква>→A|B|C|…|Z|a|b|…|z

<цифра>→0|1|2|…|9

|  |  |  |
| --- | --- | --- |
| Лексема | Тип лексемы | Ограничитель лексемы |

|  |  |  |
| --- | --- | --- |
| Ключевые слова | | |
| do | TDo = 1 | Не цифра, не буква |
| while | TWhile = 2 | Не цифра, не буква |
| Int | Tint = 3 | Не цифра, не буква |
| Short Int | TSInt = 4 | Не цифра, не буква |
| Long Int | TLInt = 5 | Не цифра, не буква |
| Long Long Int | TLLInt = 6 | Не цифра, не буква |
| return | TReturn = 7 | Не цифра, не буква |

|  |  |  |
| --- | --- | --- |
| Идентификаторы | TIdent = 10 | Не цифра, не буква |

|  |  |  |
| --- | --- | --- |
| Целая константа 10 с/с | TConsIntDec = 20 | Не цифра |
| Целая константа 16 с/с | TConsIntHex = 21 | Не цифра, не буква – a,b,c,d,e,f |

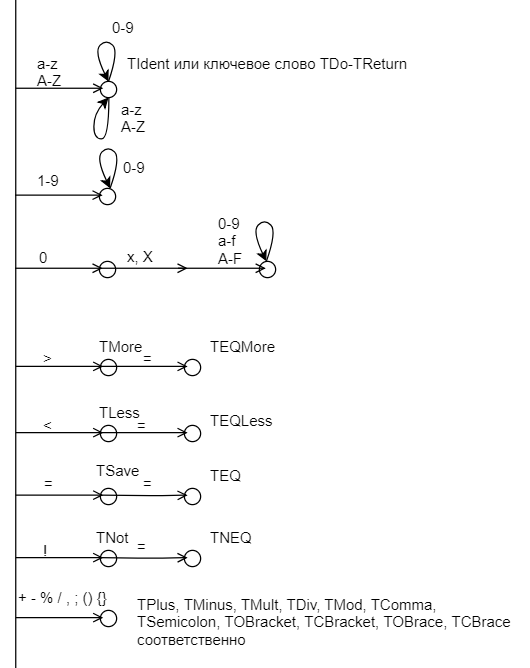
|  |  |  |
| --- | --- | --- |
| Специальные знаки | | |
| Знак «.» | TDot = 30 | Любой символ |
| Знак «,» | TComma = 31 | Любой символ |
| Знак «;» | TSemicolon = 32 | Любой символ |
| Знак «(» | TOBracket = 33 | Любой символ |
| Знак «)» | TCBracket = 34 | Любой символ |
| Знак «{» | TOBrace = 35 | Любой символ |
| Знак «}» | TCBrace = 36 | Любой символ |

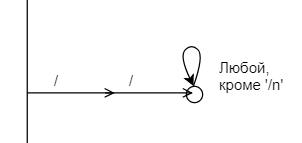
|  |  |  |
| --- | --- | --- |
| Знаки операций | | |
| Знак «+», «-» | TPlus = 40, TMinus = 41 | Любой символ |
| Знак «\*» | TMult = 42 | Любой символ |
| Знак «/» | Tdiv = 43 |  |
| Знак «%» | TMod = 44 |  |
| Знак «=» | TSave = 45 | Не знак = |
| Знак «<» | TLess = 46 | Не знак = |
| Знак «>» | TMore = 47 | Не знак = |
| Знак «<=» | TEQLess = 48 | Любой символ |
| Знак «>=» | TEQMore = 49 | Любой символ |
| Знак «==» | TEQ = 50 | Любой символ |
| Знак «!=» | TNEQ = 51 | Любой символ |

|  |  |  |
| --- | --- | --- |
| Ошибочный тип | Terr = 200 |  |
| Конец текста | Tend = 100 |  |

Игнорируемые символы

1. пробел
2. перевод строки
3. табуляция
4. комментарии от знака // до конца строки





|  |  |
| --- | --- |
| Понятие | Новое обозначение |
| <программа> | S |
| <описание функций> | F |
| <описание переменных> | D |
| <main> | M |
| <список переменных> | Z |
| <переменная> | I |
| < выражение> | V |
| <состовной оператор> | Q |
| <операторы и описания> | K |
| <оператор> | O |
| <вызов функции> | B |
| <return> | R |
| <присваивание> | P |
| <do{}while()> | W |
| < слагаемое> | A |
| <множитель> | H |
| <эл. выражение> | E |
| <константа> | C |
| <тип> | T |

S → SF | SD | SM | ε

F → T a() Q

D → TZ

M → T main() Q

Z → Z, I|I

I → a|a=V

V → V>A|V<A|V>=A|V <= A|V == A| V != A|+A|-A|A

Q → {K}

K → KO|KD| ε

O → P;|Q|W|B|R|;

B → a();

R → return V;

P → a=V

W → do Q while(V);

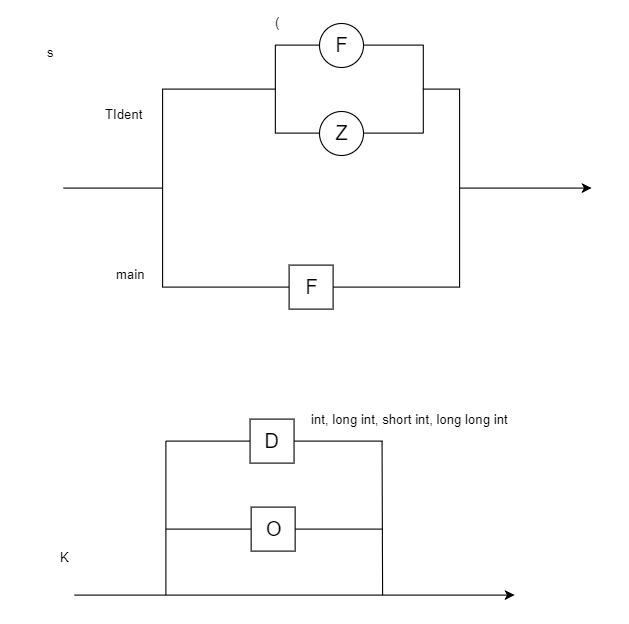
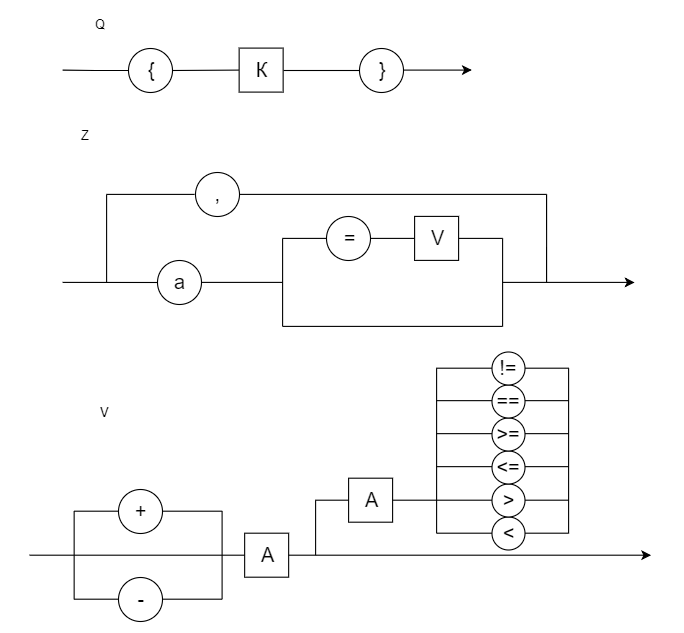
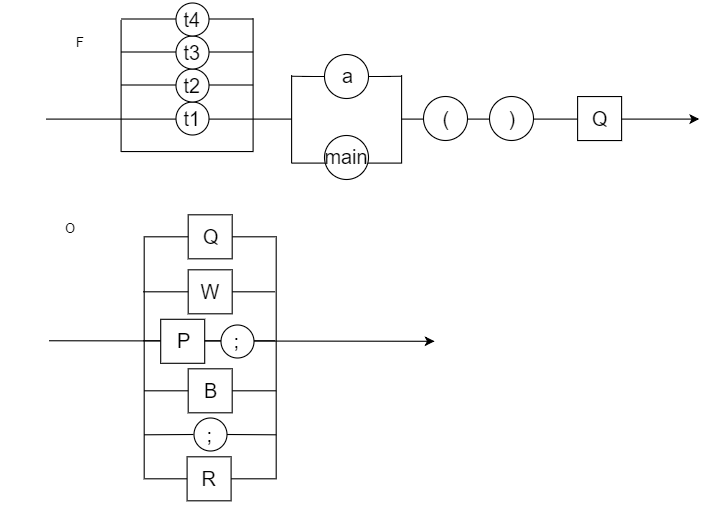
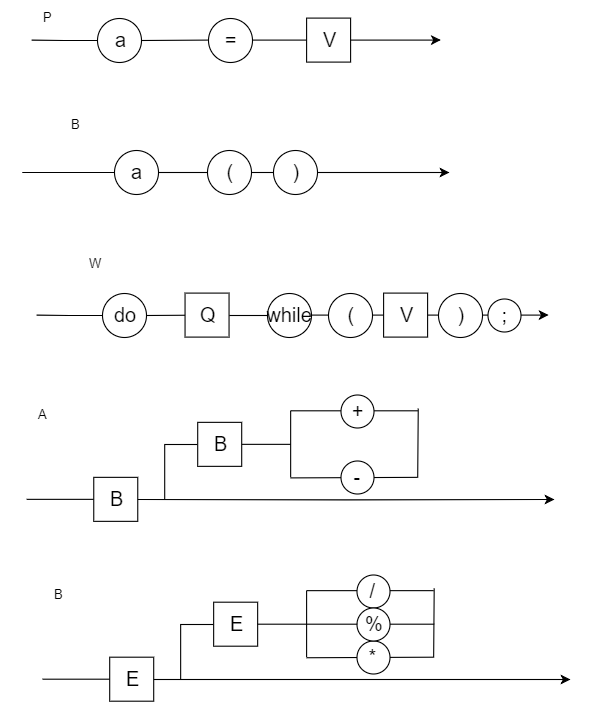
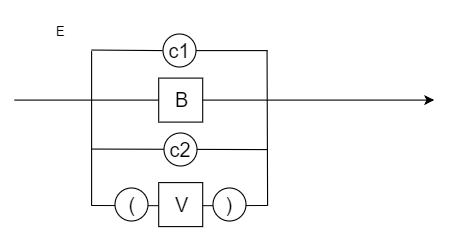
A → A+H|A-H|H

H → H\*E|H/E|H%E|E

E → C|B(V)

C →c1|c2

T → t1|t2|t3|t4

enum TypeObject {ObjConst=1, // константа

ObjVar, // простая переменная ObjTypeVar, // простой тип

ObjFunct}; // функция

enum DATA\_TYPE {TYPE\_UNKNOWN=1, TYPE\_INTEGER,

TYPE\_SINTEGER, TYPE\_LINTEGER, TYPE\_LLINTEGER, TYPE\_FUNCT

};

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| обязательные | | могут отсутствовать | | |
| идентификатор  переменной | тип | признак  инициализации | значение | адрес в оттранслированной  программе |

Описание соответствующей структуры в программе имеет вид

struct DataIdent

{

TypeObject t; // для переменной t=ObjVar LEX id; // идентификатор переменной int DataType; // тип значения

... // необязательные данные

};

Любая константа полностью представима следующими данными:

|  |  |  |  |
| --- | --- | --- | --- |
| обязательные | | | может отсутствовать |
| идентификатор  константы | тип | изображение  константы | адрес в оттранслированной  программе |

struct DataIdent

{

TypeObject t; // для переменной и константы t=ObjVar LEX id; // идентификатор переменной или константы int DataType; // тип значения

int FlagConst; // признак константы

char \* Data; // ссылка на значение константы или NULL

// необязательные данные:

int FlagInit; // Флаг начальной инициализации

char \* Addr; // адрес в оттранслированной программе

};

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| обязательные | | | | | |
| идентификатор  функции | тип  возвращаемого значения | число  параметров | тип  параметра 1 | тип  параметра 2 | ... |

|  |  |
| --- | --- |
| могут отсутствовать | |
| объем локальных  данных | адрес в оттранслированной  программе |

struct DataFunc

{

TypeObject t; // для функции t=ObjFunct

LEX id; // идентификатор функции

int DataType; // тип возвращаемого функцией значения

int Param; // количество параметров

};

struct TDataType

{

TypeObject t; // для типа значение определяется конструкцией типа:

// t=ObjTypeVar, t=ObjTypeArray, t=ObjTypeStruct

LEX id; // идентификатор типа

int FlagType; // признак типа

... // позиции, соответствующие данному типу

};

struct Node

{

// данные, общие для всех типов объектов

LEX id; // идентификатор объекта

DATA\_TYPE DataType; // тип значения ( в том числе и признак типа)

// обязательные данные для некоторых типов объектов

int FlagConst; // признак константы

char \* Data; // ссылка на значение константы или NULL

int Param; // количество параметров функции

// необязательные данные:

int FlagInit; // Флаг начальной инициализации

char \* Addr; // адрес в оттранслированной программе

};

1. все переменные должны быть описаны в пределах того блока, в котором они встречаются;
2. все функции должны быть описаны;
3. число фактических параметров функций должно совпадать с числом формальных параметров;
4. контроль приведения типов