Лекция 18
БГТУ, ФИТ, ПОИТ, 3 семестр
Конструирование программного обеспечения
Принцип реализации синтаксического анализатора

1. Вид сверху

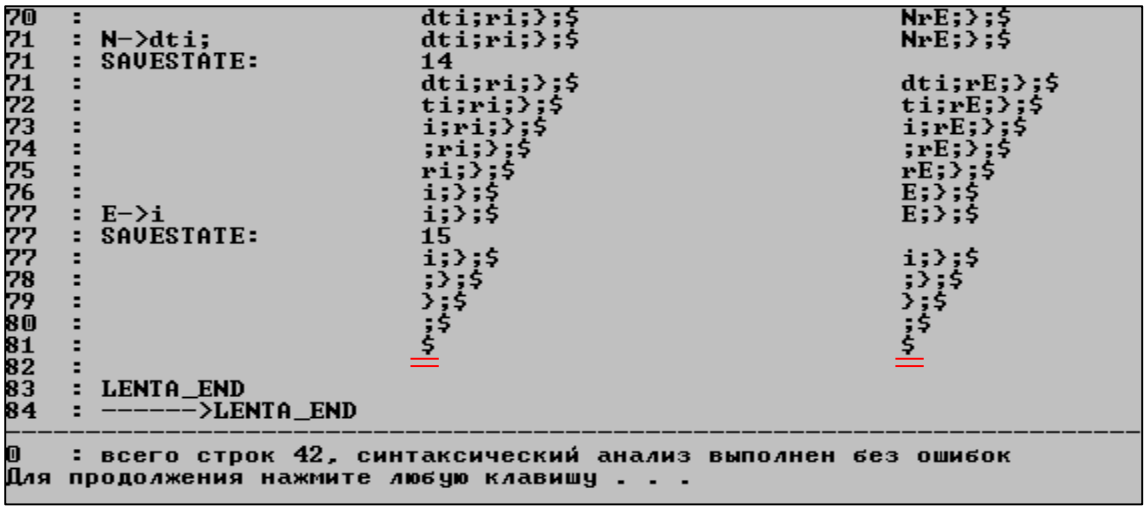
```
#include "stdafx.h"
#include <iostream>
#include <locale>
#include "MFST.h"  // магазинный автомат
int _tmain(int argc, _TCHAR* argv[])
{
    setlocale(LC_ALL,"rus");

    int s = 0;
    LEX::LEX lex;          // лексического анализа
    lex.lextable.table[ s]  = LT::Entry('t',1);      // LT::Entry( лексема , номер  исходной строки )
    lex.lextable.table[++s] = LT::Entry('i',1);
    lex.lextable.table[++s] = LT::Entry('f',1);
    //  и т.д. заполнение таблицы лексем
    // .........................................
    lex.lextable.table[++s] = LT::Entry(';',11);
    lex.lextable.table[++s] = LT::Entry('$',12);
    lex.lextable.size = ++s;
    MFST_TRACE_START                          // отладка
    MFST::Mfst mfst(lex, GRB::getGreibach()); // автомат
    mfst.start();                             // старт синтаксического анализа

    system("pause");
    return 0;
}
```

2. Ожидаемый результат

```
Шаг  : Правило                 Входная лента                  Стек
0    : S->tif(F)(NrE;);S       tif(ti,ti)(dti;i=iv(ivi);      S$
0    : SAVESTATE:              1
0    :                         tif(ti,ti)(dti;i=iv(ivi);      tif(F)(NrE;);S$
1    :                         if(ti,ti)(dti;i=iv(ivi);r      if(F)(NrE;);S$
2    :                         f(ti,ti)(dti;i=iv(ivi);ri      f(F)(NrE;);S$
3    :                         (ti,ti)(dti;i=iv(ivi);ri;      (F)(NrE;);S$
4    :                         ti,ti)(dti;i=iv(ivi);ri;)      F)(NrE;);S$
5    : F->ti                   ti,ti)(dti;i=iv(ivi);ri;)      F)(NrE;);S$
5    : SAVESTATE:              2
5    :                         ti,ti)(dti;i=iv(ivi);ri;)      ti)(NrE;);S$
6    :                         i,ti)(dti;i=iv(ivi);ri;);      i)(NrE;);S$
7    :                         ,ti)(dti;i=iv(ivi);ri;);m      )(NrE;);S$
8    : TS_NOK/NS_NORULECHAIN
8    : RESSTATE
8    :                         ti,ti)(dti;i=iv(ivi);ri;)      F)(NrE;);S$
9    : F->ti,F                 ti,ti)(dti;i=iv(ivi);ri;)      F)(NrE;);S$
9    : SAVESTATE:              2
9    :                         ti,ti)(dti;i=iv(ivi);ri;)      ti,F)(NrE;);S$
10   :                         i,ti)(dti;i=iv(ivi);ri;);      i,F)(NrE;);S$
11   :                         ,ti)(dti;i=iv(ivi);ri;);m      ,F)(NrE;);S$
12   :                         ti)(dti;i=iv(ivi);ri;);m(      F)(NrE;);S$
13   : F->ti                   ti)(dti;i=iv(ivi);ri;);m(      F)(NrE;);S$
13   : SAVESTATE:              3
13   :                         ti)(dti;i=iv(ivi);ri;);m(      ti)(NrE;);S$
14   :                         i)(dti;i=iv(ivi);ri;);m(d      i)(NrE;);S$
15   :                         )(dti;i=iv(ivi);ri;);m(dt      )(NrE;);S$
16   :                         (dti;i=iv(ivi);ri;);m(dti      (NrE;);S$
17   :                         dti;i=iv(ivi);ri;);m(dti;      NrE;);S$
18   : N->dti;                 dti;i=iv(ivi);ri;);m(dti;      NrE;);S$
18   : SAVESTATE:              4
18   :                         dti;i=iv(ivi);ri;);m(dti;      dti;rE;);S$
19   :                         ti;i=iv(ivi);ri;);m(dti;r      ti;rE;);S$
20   :                         i;i=iv(ivi);ri;);m(dti;ri      i;rE;);S$
21   :                         ;i=iv(ivi);ri;);m(dti;ri;      ;rE;);S$
22   :                         i=iv(ivi);ri;);m(dti;ri;)      rE;);S$
23   : TS_NOK/NS_NORULECHAIN
23   : RESSTATE
23   :                         dti;i=iv(ivi);ri;);m(dti;      NrE;);S$
24   : N->dtfi(F);             dti;i=iv(ivi);ri;);m(dti;      NrE;);S$
24   : SAVESTATE:              4
```



```
70   :                         dti;ri;);$                     NrE;);$
71   : N->dti;                 dti;ri;);$                     NrE;);$
71   : SAVESTATE:              14
71   :                         dti;ri;);$                     dti;rE;);$
72   :                         ti;ri;);$                      ti;rE;);$
73   :                         i;ri;);$                       i;rE;);$
74   :                         ;ri;);$                        ;rE;);$
75   :                         ri;);$                         rE;);$
76   :                         i;);$                          E;);$
77   : E->i                    i;);$                          E;);$
77   : SAVESTATE:              15
77   :                         i;);$                          i;);$
78   :                         ;);$                           ;);$
79   :                         );$                            );$
80   :                         ;$                             ;$
81   :                         $                              $
82   :
83   : LENTA_END
84   : ------>LENTA_END
-------------------------------------------------------------------------------
0    : всего строк 42, синтаксический анализ выполнен без ошибок
Для продолжения нажмите любую клавишу . . .
```

3. Грамматика (Грейбах)

Правила грамматики Грейбах:

S→ m{NrE;};|tfi(F){NrE;};S|m{NrE;};S|tfi(F){NrE;};

N→ dti;|rE;|i=E;|dtfi(F);|dti;N|rE;N|i=E;N|dtfi(F);N

E→i|l|(E)|i(W)|iM|lM|(E)M|i(W)M

M→vE|vEM

F→ti|ti,F

W→i|l|i,W|l,W

```
#include "GRB.h"
#define  GRB_ERROR_SERIES 600
namespace GRB
{
    #define NS(n) Rule::Chain::N(n)
    #define TS(n) Rule::Chain::T(n)
    Greibach  greibach( NS('S'), TS('$'),    // стартовый символ, дно стека
                        6,                   // количество правил
                        Rule(NS('S'), GRB_ERROR_SERIES + 0,   // Неверная  структура программы
                                    3,     // S->m{NrE;}; | tfi(F){NrE;};S | m{NrE;};S | tfi(F){NrE;}
                                    Rule::Chain(8,  TS('m'),  TS('{'),  NS('N'), TS('r'), NS('E'),TS(';'), TS('}'), TS(';')),
                                    Rule::Chain(14, TS('t'),  TS('i'),  TS('f'), TS('('), NS('F'),TS(')'), TS('{'), NS('N'), TS('r'), NS('E'), TS(';'),TS('}'),
                                    Rule::Chain(9,  TS('m'),  TS('{'),  NS('N'), TS('r'), NS('E'),TS(';'), TS('}'), TS(';'), NS('S'))
                           ),
                        Rule(NS('N'), GRB_ERROR_SERIES + 1, // Ошибочный  оператор
                                    8,    //N->dti;|rE;|i=E;|dtfi(F);|dtiN|rE;N|i=E;N|dtfi(F);N
                                    Rule::Chain(4,  TS('d'),  TS('t'),  TS('i'), TS(';')),
                                    Rule::Chain(3,  TS('r'),  NS('E'),  TS(';')),
                                    Rule::Chain(4,  TS('i'),  TS('='),  NS('E'), TS(';')),
                                    Rule::Chain(8,  TS('d'),  TS('t'),  TS('f'), TS('i'), TS('('),NS('F'), TS(')'), TS(';')),
                                    Rule::Chain(5,  TS('d'),  TS('t'),  TS('i'), TS(';'), NS('N')),
                                    Rule::Chain(4,  TS('r'),  NS('E'),  TS(';'), NS('N')),
                                    Rule::Chain(5,  TS('i'),  TS('='),  NS('E'), TS(';'), NS('N')),
                                    Rule::Chain(9,  TS('d'),  TS('t'),  TS('f'), TS('i'), TS('('),NS('F'), TS(')'), TS(';'), NS('N'))
                          ),
                        Rule(NS('E'), GRB_ERROR_SERIES + 2, // Ошибка в выражении
                                    8,    // E->i|l|(E)|i(W)|iM|lM|(E)M|i(W)M
                                    Rule::Chain(1,  TS('i')),
                                    Rule::Chain(1,  TS('l')),
                                    Rule::Chain(3,  TS('('),  NS('E'),  TS(')')),
                                    Rule::Chain(4,  TS('i'),  TS('('),  NS('W'), TS(')')),
                                    Rule::Chain(2,  TS('i'),  NS('M')),
                                    Rule::Chain(2,  TS('l'),  NS('M')),
                                    Rule::Chain(4,  TS('('),  NS('E'),  TS(')'), NS('M')),
                                    Rule::Chain(5,  TS('i'),  TS('('),  NS('W'), TS(')'), NS('M'))
                          ),
```

```
#pragma once
#include "Error.h"
typedef short  GRBALPHABET;   // символы алфавита грамматики терминалы > 0,нетерминалы < 0
namespace GRB
{
    struct Rule     //правило в грамматике  Грейбах
    {
        GRBALPHABET  nn;     // нетерминал (левый символ правила)   < 0
        int iderror;         // идентификатор диагностического сообщения
        short size;          // количество цепочек - правых частей правила
        struct Chain         //  цепочка (правая часть правила)
        {
              short size;                 //  длина цепочки
              GRBALPHABET* nt;            //  цепочка терминалов (>0) и  нетерминалов (<0)
              Chain() {size = 0; nt = 0;};
              Chain(
                    short psize,          // количество символов в цепочке
                    GRBALPHABET s, ...    // символы  (терминал или нетерминал)
                    );
              char* getCChain(char* b);    // получить правую сторону правила
              static GRBALPHABET T(char t) {return  GRBALPHABET(t);};  // терминал
              static GRBALPHABET N(char n) {return -GRBALPHABET(n);};  // не терминал
              static bool isT(GRBALPHABET s) {return s > 0;};   // терминал?
              static bool isN(GRBALPHABET s) {return !isT(s);}  // нетерминал?
              static char alphabet_to_char(GRBALPHABET s) {return isT(s)?char(s):char(-s);}; // GRBALPHABET->char
        }* chains;                        //  массив цепочек - правых частей правила
        Rule(){nn = 0x00; size = 0;}
        Rule(
             GRBALPHABET pnn,             // нетерминал  (< 0)
             int iderror,                 // идентификатор диагностического сообщения (Error)
             short psize,                 // количество цепочек - правых частей правила
             Chain c, ...                 // можество цепочек - правых частей правила
             );
```

```
        char* getCRule(              // получить правило в виде N->цепочка (для рапечатки)
                    char* b,       // буфер
                    short nchain   // номер цепочки (правой части) в правиле
                   );
        short  Rule::getNextChain(   // получить следующую за j подходящую цепочку, вернуть ее номер или -1
                                  GRBALPHABET t,        // первый симол цепочки
                                  Rule::Chain& pchain,  // возвращаемая цепочка
                                  short j               // номер цепочки
                                  );
    };

    struct Greibach  // грамматика Грейбах
    {
     short size;       // количество правил
     GRBALPHABET startN;     // стартовый символ
     GRBALPHABET  stbottomT;  // дно стека
     Rule* rules;      // множество правил
     Greibach() {short size = 0; startN  = 0;  stbottomT  = 0;  rules = 0; };
     Greibach(
              GRBALPHABET pstartN,     // стартовый символ
              GRBALPHABET pstbottomT,  // дно стека
              short psize,   // количество правил
              Rule r, ...    // правила
              );
     short getRule(     // получить  правило, возвращается номер правила или -1
              GRBALPHABET pnn,     // левый символ правила
              Rule& prule          // возвращаемое правило грамматики
              );
     Rule getRule(short n);  // получить правило по номеру
     };

     Greibach getGreibach();  // получить грамматику
};
```

```
ERROR errors[ERROR_MAX_ENTRY] =  //таблица ошибок
{
   ERROR_ENTRY(0, "Недопустимый код ошибки"),    // код ошибки вне диапазона 0 - ERROR_MAX_ENTRY
   ERROR_ENTRY(1, "Системный сбой"),
   ERROR_ENTRY_NODEF(2), ERROR_ENTRY_NODEF(3), ERROR_ENTRY_NODEF(4), ERROR_ENTRY_NODEF(5),
   ERROR_ENTRY_NODEF(6), ERROR_ENTRY_NODEF(7), ERROR_ENTRY_NODEF(8), ERROR_ENTRY_NODEF(9),
   ERROR_ENTRY_NODEF10(10), ERROR_ENTRY_NODEF10(20), ERROR_ENTRY_NODEF10(30), ERROR_ENTRY_NODEF10(40), ERROR_ENTRY_NODEF10(50),
   ERROR_ENTRY_NODEF10(60), ERROR_ENTRY_NODEF10(70), ERROR_ENTRY_NODEF10(80), ERROR_ENTRY_NODEF10(90),
   ERROR_ENTRY(100, "Параметр -in должен быть задан"),
   ERROR_ENTRY_NODEF(101), ERROR_ENTRY_NODEF(102),ERROR_ENTRY_NODEF(103),
   ERROR_ENTRY(104, "Превышена длина входного параметра"),
   ERROR_ENTRY_NODEF(105), ERROR_ENTRY_NODEF(106),  ERROR_ENTRY_NODEF(107),
   ERROR_ENTRY_NODEF(108), ERROR_ENTRY_NODEF(109),
   ERROR_ENTRY(110, "Ошибка при открытии файла с исходным кодом (-in)"),
   ERROR_ENTRY(111, "Недопустимый символ в исходном файле (-in)"),
   ERROR_ENTRY(112, "Ошибка при создании файла протокола(-log)"),
   ERROR_ENTRY_NODEF(113), ERROR_ENTRY_NODEF(114), ERROR_ENTRY_NODEF(115),
   ERROR_ENTRY_NODEF(116), ERROR_ENTRY_NODEF(117), ERROR_ENTRY_NODEF(118), ERROR_ENTRY_NODEF(119),
   ERROR_ENTRY_NODEF10(120), ERROR_ENTRY_NODEF10(130), ERROR_ENTRY_NODEF10(140), ERROR_ENTRY_NODEF10(150),
   ERROR_ENTRY_NODEF10(160), ERROR_ENTRY_NODEF10(170), ERROR_ENTRY_NODEF10(180), ERROR_ENTRY_NODEF10(190),
   ERROR_ENTRY_NODEF100(200), ERROR_ENTRY_NODEF100(300), ERROR_ENTRY_NODEF100(400), ERROR_ENTRY_NODEF100(500),
   ERROR_ENTRY(600, "Неверная  структура программы"),
   ERROR_ENTRY(601, "Ошибочный оператор"),
   ERROR_ENTRY(602, "Ошибка в выражении"),
   ERROR_ENTRY(603, "Ошибка в параметрах функции"),
   ERROR_ENTRY(604, "Ошибка в параметрах вызваемой функции"),
   ERROR_ENTRY_NODEF(605), ERROR_ENTRY_NODEF(606), ERROR_ENTRY_NODEF(607),ERROR_ENTRY_NODEF(608),ERROR_ENTRY_NODEF(609),
   ERROR_ENTRY_NODEF10(610),ERROR_ENTRY_NODEF10(620),ERROR_ENTRY_NODEF10(630),ERROR_ENTRY_NODEF10(640),
   ERROR_ENTRY_NODEF10(650),ERROR_ENTRY_NODEF10(660),ERROR_ENTRY_NODEF10(670),ERROR_ENTRY_NODEF10(680),
   ERROR_ENTRY_NODEF10(690),
   ERROR_ENTRY_NODEF100(700), ERROR_ENTRY_NODEF100(800), ERROR_ENTRY_NODEF100(900)
};
```

4. Грамматика (Грейбах): конструкторы

```
Rule::Chain::Chain(short psize,  GRBALPHABET s, ... )
{
 nt = new GRBALPHABET[size = psize];
 int* p = (int*)&s;
 for (short i = 0; i < psize; ++i) nt[i] =(GRBALPHABET)p[i];
};
Rule::Rule( GRBALPHABET pnn,  int piderror,  short psize, Chain c, ... )
{
  nn = pnn;
  iderror = piderror;
  chains = new Chain[size = psize];
  Chain* p = &c;
  for (int i = 0; i < size; i++) chains[i] = p[i];
};
Greibach::Greibach(GRBALPHABET pstartN,   GRBALPHABET pstbottom,  short psize,   Rule r, ... )
{
     startN = pstartN;
     stbottomT = pstbottom;
     rules = new Rule[size = psize];
     Rule* p = &r;
     for (int i = 0; i < size; i++) rules[i] = p[i];
 };
```

5. Грамматика (Грейбах): функции и методы

```
Greibach getGreibach() {return greibach;};

short Greibach::getRule(GRBALPHABET pnn, Rule& prule)
 {
        short rc = -1;
        short k = 0;
        while(k < size &&  rules[k].nn != pnn) k++;
        if (k < size) prule = rules[rc=k];
        return rc;
 };

Rule Greibach::getRule(short n)
{
    Rule rc;
    if (n < size) rc = rules[n];
    return rc;
};

char*  Rule::getCRule(char* b , short nchain)  // получить правило в виде N->цепочка
{
    char bchain[200];
    b[0] =   Chain::alphabet_to_char(nn); b[1]='-'; b[2]='>'; b[3]=0x00;
    chains[nchain].getCChain(bchain);
    strcat_s(b, sizeof(bchain)+5, bchain);
    return b;
};
```

```
short  Rule::getNextChain(GRBALPHABET t, Rule::Chain& pchain, short j)
{
   short rc = -1;
   while(j < size && chains[j].nt[0] != t) ++j;
   rc = (j < size? j: -1);
   if (rc >= 0)  pchain  = chains[rc];
   return rc;
 };

char* Rule::Chain::getCChain(char* b)  // получить правую сторону правила
{
    for (int i = 0; i < size; i++) b[i] = Chain::alphabet_to_char(nt[i]);
     b[size]=0x00;
   return b;
 };
```

6. Магазинный автомат

```
#define  MFST_DIAGN_MAXSIZE  2*ERROR_MAXSIZE_MESSAGE
#define  MFST_DIAGN_NUMBER 3
 class MFSTSTSTACK :public std::stack<short> {      // стек автомата
 public:
    using std::stack<short>::c;
 };

namespace MFST
{
    struct MfstState                        // состояние автомата (для сохранения)
    {

      short  lenta_position;                //  позиция на ленте
      short  nrule;                         //  номер текущего правила
      short  nrulechain;                    //  номер текущей цепочки, текущего правила
      MFSTSTSTACK  st;                      //  стек автомата
      MfstState();
      MfstState(
               short pposition,             //  позиция на ленте
               MFSTSTSTACK   pst,           //  стек автомата
               short pnrulechain            //  номер текущей цепочки, текущего правила
               );
       MfstState(
               short pposition,             //  позиция на ленте
               short pnrule,                //  номер текущего правила
               short pnrulechain            //  номер текущей цепочки, текущего правила
               );


    };
    class MFSTSTATE :public std::stack<MfstState> {  // стек автомата
    public:
        using std::stack<MfstState>::c;
    };
```

```
struct Mfst                         // магазинный автомат
{
  enum RC_STEP {                        // код возврата  функции step
                NS_OK,                  // найдено правило и цепочка, цепочка записана в стек
                NS_NORULE,              // не найдено правило грамматики (ошибка в грамматике)
                NS_NORULECHAIN,         // не найдена походящая  цепочка правила (ошибка в исходном коде)
                NS_ERROR,               // неизвесный нетерминальный символ грамматики
                TS_OK,                  // тек. символ ленты == вершине стека, продвинулась лента, pop стека
                TS_NOK,                 // тек. символ ленты != вершине стека, восстановленно состояние
                LENTA_END,              // теущая позиция ленты >= lenta_size
                SURPRISE                // неожиданный код возврата (ошибка в step)
               };

  struct MfstDiagnosis              // диагностика
  {
   short     lenta_position;          // позиция на ленте
   RC_STEP   rc_step;                 // код завершения шага
   short     nrule;                   // номер правила
   short     nrule_chain;             // номер цепочки правила
   MfstDiagnosis();
   MfstDiagnosis(                   // диагностика
                short  plenta_position, // позиция на ленте
                RC_STEP prc_step ,      // код завершения шага
                short   pnrule,         // номер правила
                short   pnrule_chain    // номер цепочки правила
                );
  } diagnosis[MFST_DIAGN_NUMBER];       //  последние самые глубокие сообщения

  GRBALPHABET* lenta;               // перекодированная (TS/NS) лента (из LEX)
  short  lenta_position;            // текущая позиция на ленте
  short  nrule;                     // номер текущего правила
  short  nrulechain;                // номер текущей цепочки, текущего правила
  short  lenta_size;                // размер ленты
  GRB::Greibach grebach;            // грамматика Грейбах
  LEX::LEX lex;                     // результат работы лексического анализатора
  MFSTSTSTACK  st;                      // стек автомата
  MFSTSTATE storestate;             // стек для сохранения состояний
```

```
Mfst();
Mfst(
    LEX::LEX plex,                  // результат работы лексического анализатора
    GRB::Greibach pgrebach          // грамматика Грейбах
    );
char* getCSt(char* buf);            // получить содержимое стека
char* getCLenta(char* buf, short pos,  short n = 25);   //  лента: n символов с pos
char* getDiagnosis(short n, char* buf);   // получить n-ую строку диагностики или 0x00
bool savestate();                   // сохранить состояние автомата
bool reststate();                   // восстановить состояние автомата
    bool push_chain(                    // поместить цепочку правила в стек
                GRB::Rule::Chain chain   // цепочка правила
                );
RC_STEP step();                     // выполнить шаг автомата
bool start();                       // запустить автомат
bool savediagnosis(
                  RC_STEP pprc_step  // код завершения шага
                  );
void printrules();                  //  вывести последовательность правил

struct Deducation                  // вывод
{
 short size;              //  количество шагов в выводе
 short* nrules;           //  номера правил грамматики
 short* nrulechains;      //  номера цепочер правил грамматики (nrules)
 Deducation() { size = 0;  nrules = 0; nrulechains = 0;};
} deducation;
bool savededucation();   // сохранить дерево вывода
};
```

7. Магазинный автомат: конструкторы

```
MfstState::MfstState()
{
     lenta_position = 0;
     nrule = -1;
     nrulechain = -1;
};
MfstState::MfstState(short pposition, MFSTSTSTACK  pst, short pnrulechain)
{
   lenta_position = pposition;
   st = pst;
   nrulechain = pnrulechain;
};
MfstState::MfstState(short pposition, MFSTSTSTACK  pst, short pnrule, short pnrulechain)
{
   lenta_position = pposition;
   st = pst;
   nrule = pnrule;
   nrulechain = pnrulechain;
};
```

```
Mfst::MfstDiagnosis::MfstDiagnosis()
{
    lenta_position = -1;
    rc_step = SURPRISE;
    nrule = -1;
    nrule_chain = -1;
};
Mfst::MfstDiagnosis::MfstDiagnosis(short plenta_position, RC_STEP prc_step, short  pnrule, short  pnrule_chain )
{
    lenta_position = plenta_position;
    rc_step = prc_step;
    nrule = pnrule;
    nrule_chain = pnrule_chain;
};
```

```
Mfst::Mfst(){ lenta = 0; lenta_size = lenta_position = 0;};
Mfst::Mfst(LEX::LEX plex, GRB::Greibach pgrebach)
 {
     grebach = pgrebach;
     lex = plex;
     lenta = new short[lenta_size =lex.lextable.size];
     for(int k = 0; k < lenta_size; k++)  lenta[k] =  TS(lex.lextable.table[k].lexema);
     lenta_position = 0;
     st.push(grebach.stbottomT);
     st.push(grebach.startN);
     nrulechain = -1;
 };
```

8. Магазинный автомат: методы и функции

```
Mfst::RC_STEP Mfst::step()
{
     RC_STEP rc = SURPRISE;
     if(lenta_position < lenta_size)
     {
       if (ISNS(st.top()))
       {
            GRB::Rule rule;
            if ((nrule = grebach.getRule(st.top(), rule)) >= 0)
            {
                GRB::Rule::Chain chain;
                if ((nrulechain = rule.getNextChain(lenta[lenta_position], chain, nrulechain+1)) >= 0)
                {
                   MFST_TRACE1
                   savestate();  st.pop();  push_chain(chain); rc = NS_OK;
                   MFST_TRACE2
                }
                else
                {
                  MFST_TRACE4("TNS_NORULECHAIN/NS_NORULE")
                  savediagnosis(NS_NORULECHAIN); rc = reststate()?NS_NORULECHAIN: NS_NORULE;
                };
            }
            else rc = NS_ERROR;
       }
       else if ((st.top() == lenta[lenta_position]))
       {
         lenta_position++;  st.pop(); nrulechain  = -1; rc = TS_OK;
        MFST_TRACE3
       }
       else { MFST_TRACE4("TS_NOK/NS_NORULECHAIN") rc = reststate()?TS_NOK:NS_NORULECHAIN;};
     }
     else { rc = LENTA_END;   MFST_TRACE4("LENTA_END") };
    return rc;
   };
```

```
bool  Mfst::push_chain(GRB::Rule::Chain chain)
{
  for (int k = chain.size - 1; k >= 0; k--) st.push(chain.nt[k]);
  return true;
};
```

```
bool  Mfst::savestate()
{
    storestate.push(MfstState(lenta_position, st, nrule, nrulechain));
    MFST_TRACE6("SAVESTATE:", storestate.size());
    return true;
};
bool  Mfst::reststate()
{
    bool rc = false;
    MfstState  state;
    if (rc = (storestate.size() > 0))
    {
        state = storestate.top();
        lenta_position = state.lenta_position;
        st = state.st;
        nrule = state.nrule;
        nrulechain = state.nrulechain;
        storestate.pop();
        MFST_TRACE5("RESSTATE")
        MFST_TRACE2
    };

    return rc;
};
```

```
bool  Mfst::savediagnosis(RC_STEP prc_step)
{
  bool rc = false;
  short k = 0;
  while (k < MFST_DIAGN_NUMBER && lenta_position <= diagnosis[k].lenta_position) k++;
  if(rc=(k < MFST_DIAGN_NUMBER))
  {
     diagnosis[k] = MfstDiagnosis(lenta_position,prc_step, nrule, nrulechain);
     for (short j=k+1; j < MFST_DIAGN_NUMBER; j++) diagnosis[j].lenta_position = -1;
  };
  return rc;
};
```

```
bool  Mfst::start()
{
      bool rc = false;
      RC_STEP rc_step  = SURPRISE;
      char  buf[MFST_DIAGN_MAXSIZE];
      rc_step = step();
      while (rc_step == NS_OK || rc_step == NS_NORULECHAIN || rc_step == TS_OK || rc_step == TS_NOK ) rc_step = step();

      switch (rc_step)
      {
       case LENTA_END:      MFST_TRACE4("------>LENTA_END")
         std::cout<<"------------------------------------------------------------------------  ----"<<std::endl;
         sprintf_s(buf, MFST_DIAGN_MAXSIZE  ,"%d: всего строк %d,  синтаксический анализ выполнен без ошибок", 0, lenta_size);
         std::cout<<std::setw(4)<<std::left<<0<<": всего строк "<<lenta_size<< ", синтаксический анализ выполнен без ошибок" <<std::endl;
         rc = true;
         break;
       case NS_NORULE:      MFST_TRACE4("------>NS_NORULE")
         std::cout<<"------------------------------------------------------------------------  ----"<<std::endl;
         std::cout<<getDiagnosis(0, buf)<<std::endl;
         std::cout<<getDiagnosis(1, buf)<<std::endl;
         std::cout<<getDiagnosis(2, buf)<<std::endl;
         break;
       case NS_NORULECHAIN: MFST_TRACE4("------>NS_NORULENORULECHAIN") break;
       case NS_ERROR:       MFST_TRACE4("------>NS_ERROR") break;
       case SURPRISE:       MFST_TRACE4("------>SURPRISE") break;

      };
      return rc;
};
```

```
char*Mfst::getCSt(char*buf) {
    for (int k = (signed)st.size() - 1; k >= 0; --k)
    {
        short p = st.c[k];
        buf[st.size() - 1 - k] = GRB::Rule::Chain::alphabet_to_char(p);
    }
    buf[st.size()] = 0x00;
    return buf;
}
```

```
char* Mfst::getCLenta(char* buf, short pos, short n)
{
   short i, k = (pos+n < lenta_size)?pos+n: lenta_size;
   for (i = pos; i < k; i++) buf[i-pos] = GRB::Rule::Chain::alphabet_to_char(lenta[i]) ;
   buf[i-pos] = 0x00;
   return buf;

};

char* Mfst::getDiagnosis(short n, char* buf)
{
  char *rc = "";
  int errid = 0;
  int lpos  = -1;
  if (n < MFST_DIAGN_NUMBER &&  (lpos = diagnosis[n].lenta_position) >= 0)
  {
     errid = grebach.getRule(diagnosis[n].nrule).iderror;
     Error::ERROR err = Error::geterror(errid);
     sprintf_s(buf, MFST_DIAGN_MAXSIZE  ,"%d: строка %d,  %s", err.id, lex.lextable.table[lpos].sn,err.message);
     rc = buf;
  };
  return rc;
};
```

```
void Mfst::printrules() {
    MfstState state;
    GRB::Rule rule;
    for (unsigned short k = 0; k < storestate.size(); k++)
    {
        state = storestate.c[k];
        rule = grebach.getRule(state.nrule);
        MFST_TRACE7
}
```

```
bool Mfst::savededucation() {
    MfstState state;
    GRB::Rule rule;
    deducation.nrules = new short[deducation.size = storestate.size()];
    deducation.nrulechains = new short[deducation.size];
    for (unsigned short k = 0; k < storestate.size(); k++)
    {
        state = storestate.c[k];
        deducation.nrules[k] = state.nrule;
        deducation.nrulechains[k] = state.nrulechain;
    }
    return true;
};
```

9. Подготовка к генерации кода

```
    lex.lextable.table[++s] = LT::Entry('1',10);    // 37
    lex.lextable.table[++s] = LT::Entry(';',10);    // 38
    lex.lextable.table[++s] = LT::Entry('}',11);    // 39
    lex.lextable.table[++s] = LT::Entry(';',11);    // 40
    lex.lextable.table[++s] = LT::Entry('$',12);    // 41
    lex.lextable.size = ++s;
    MFST_TRACE_START                                // отладка
    MFST::Mfst mfst(lex, GRB::getGreibach());       // автомат
    mfst.start();                                   // старт синтаксического анализа

    mfst.savededucation();                          // сохранить вывести правила вывода


    mfst.printrules();                              // отладка: вывести правила вывода


    system("pause");
    return 0;
}
```

```
68   : S->m(NrE;);          m(dti;ri;);$                  S$
68   : SAVESTATE:           13
68   :                      m(dti;ri;);$                  m(NrE;);$
69   :                      (dti;ri;);$                   (NrE;);$
70   :                      dti;ri;);$                    NrE;);$
71   : N->dti;              dti;ri;);$                    NrE;);$
71   : SAVESTATE:           14
71   :                      dti;ri;);$                    dti;rE;);$
72   :                      ti;ri;);$                     ti;rE;);$
73   :                      i;ri;);$                      i;rE;);$
74   :                      ;ri;);$                       ;rE;);$
75   :                      ri;);$                        rE;);$
76   :                      i;);$                         E;);$
77   : E->i                 i;);$                         E;);$
77   : SAVESTATE:           15
77   :                      i;);$                         i;);$
78   :                      ;);$                          ;);$
79   :                      );$                           );$
80   :                      ;$                            ;$
81   :                      $                             $
82   :
83   : LENTA_END
84   : ------>LENTA_END
-------------------------------------------------------------------------------
8    : Всего строк 12, синтаксический анализ выполнен без ошибок
0    : S->tif(F)(NrE;);S
4    : F->ti,F
7    : F->ti
11   : N->dti;N
15   : N->i=E;
17   : E->iM
18   : M->vE
19   : E->(E)
20   : E->iM
21   : M->vE
22   : E->i
26   : E->i
30   : S->m(NrE;);
32   : N->dti;
37   : E->i
```

10. Диагностика

```
Шаг  : Правило                Входная лента                  Стек
0    : S->tif(F){NrE;};S      tif(ti,ti){dtii=iv(ivi);r      S$
0    : SAVESTATE:             1
0    :                        tif(ti,ti){dtii=iv(ivi);r      tif(F){NrE;};S$
1    :                        if(ti,ti){dtii=iv(ivi);ri      if(F){NrE;};S$
2    :                        f(ti,ti){dtii=iv(ivi);ri;      f(F){NrE;};S$
3    :                        (ti,ti){dtii=iv(ivi);ri;}      (F){NrE;};S$
4    :                        ti,ti){dtii=iv(ivi);ri;};      F){NrE;};S$
5    : F->ti                  ti,ti){dtii=iv(ivi);ri;};      F){NrE;};S$
5    : SAVESTATE:             2
5    :                        ti,ti){dtii=iv(ivi);ri;};      ti){NrE;};S$
6    :                        i,ti){dtii=iv(ivi);ri;};m      i){NrE;};S$
7    :                        ,ti){dtii=iv(ivi);ri;};m{      ){NrE;};S$
8    : TS_NOK/NS_NORULECHAIN
8    : RESSTATE
8    :                        ti,ti){dtii=iv(ivi);ri;};      F){NrE;};S$
9    : F->ti,F                ti,ti){dtii=iv(ivi);ri;};      F){NrE;};S$
9    : SAVESTATE:             2
9    :                        ti,ti){dtii=iv(ivi);ri;};      ti,F){NrE;};S$
10   :                        i,ti){dtii=iv(ivi);ri;};m      i,F){NrE;};S$
11   :                        ,ti){dtii=iv(ivi);ri;};m{      ,F){NrE;};S$
12   :                        ti){dtii=iv(ivi);ri;};m{d      F){NrE;};S$
13   : F->ti                  ti){dtii=iv(ivi);ri;};m{d      F){NrE;};S$
13   : SAVESTATE:             3
13   :                        ti){dtii=iv(ivi);ri;};m{d      ti){NrE;};S$
14   :                        i){dtii=iv(ivi);ri;};m{dt      i){NrE;};S$
15   :                        ){dtii=iv(ivi);ri;};m{dti      ){NrE;};S$
16   :                        {dtii=iv(ivi);ri;};m{dti;      {NrE;};S$
17   :                        dtii=iv(ivi);ri;};m{dti;r      NrE;};S$
18   : N->dti;                dtii=iv(ivi);ri;};m{dti;r      NrE;};S$
18   : SAVESTATE:             4
18   :                        dtii=iv(ivi);ri;};m{dti;r      dti;rE;};S$
19   :                        tii=iv(ivi);ri;};m{dti;ri      ti;rE;};S$
20   :                        ii=iv(ivi);ri;};m{dti;ri;      i;rE;};S$
21   :                        i=iv(ivi);ri;};m{dti;ri;}      ;rE;};S$
22   : TS_NOK/NS_NORULECHAIN
22   : RESSTATE
22   :                        dtii=iv(ivi);ri;};m{dti;r      NrE;};S$
23   : N->dtfi(F);            dtii=iv(ivi);ri;};m{dti;r      NrE;};S$
23   : SAVESTATE:             4
23   :                        dtii=iv(ivi);ri;};m{dti;r      dtfi(F);rE;};S$
24   :                        tii=iv(ivi);ri;};m{dti;ri      tfi(F);rE;};S$
25   :                        ii=iv(ivi);ri;};m{dti;ri;      fi(F);rE;};S$
```

```
40   : TS_NOK/NS_NORULECHAIN
40   : RESSTATE
40   :                      ti){dtii=iv(ivi);ri;};m{d      F){NrE;};S$
41   : TNS_NORULECHAIN/NS_NORULE
41   : RESSTATE
41   :                      ti,ti){dtii=iv(ivi);ri;};      F){NrE;};S$
42   : TNS_NORULECHAIN/NS_NORULE
42   : RESSTATE
42   :                      tif(ti,ti){dtii=iv(ivi);r      S$
43   : TNS_NORULECHAIN/NS_NORULE
44   : ------>NS_NORULE
--------------------------------------------------------------------------------
501: строка 3,  Ошибочный оператор
503: строка 1,  Ошибка в параметрах функции
503: строка 1,  Ошибка в параметрах функции
Для продолжения нажмите любую клавишу . . .
```

11. Отладка: трассировка

```
#define MFST_TRACE_START std::cout<<std::setw( 4)<<std::left<<"Шаг" <<":" \
                                 <<std::setw(20)<<std::left<<" Правило" \
                                 <<std::setw(30)<<std::left<<" Входная лента" \
                                 <<std::setw(20)<<std::left<<" Стек" \
                                 <<std::endl;
```

```
int  FST_TRACE_n = -1;
char rbuf[205], sbuf[205], lbuf[1024];  // печать
#define NS(n)    GRB::Rule::Chain::N(n)
#define TS(n)    GRB::Rule::Chain::T(n)
#define ISNS(n)  GRB::Rule::Chain::isN(n)
#define MFST_TRACE1      std::cout<<std::setw(4)<<std::left<<++FST_TRACE_n <<": " \
                                 <<std::setw(20)<<std::left<<rule.getCRule(rbuf,nrulechain) \
                                 <<std::setw(30)<<std::left<<getCLenta(lbuf, lenta_position) \
                                 <<std::setw(20)<<std::left<<getCSt(sbuf) \
                                 <<std::endl;
#define MFST_TRACE2 std::cout<<std::setw(4)<<std::left<< FST_TRACE_n<<": " \
                                 <<std::setw(20)<<std::left<<" " \
                                 <<std::setw(30)<<std::left<<getCLenta(lbuf, lenta_position) \
                                 <<std::setw(20)<<std::left<<getCSt(sbuf) \
                                 <<std::endl;
#define MFST_TRACE3 std::cout<<std::setw(4)<<std::left<< ++FST_TRACE_n<<": " \
                                 <<std::setw(20)<<std::left<<" " \
                                 <<std::setw(30)<<std::left<<getCLenta(lbuf, lenta_position) \
                                 <<std::setw(20)<<std::left<<getCSt(sbuf) \
                                 <<std::endl;
#define MFST_TRACE4(c) std::cout<<std::setw(4)<<std::left<< ++FST_TRACE_n<<": "<<std::setw(20)<<std::left<<c<<std::endl;
#define MFST_TRACE5(c) std::cout<<std::setw(4)<<std::left<<  FST_TRACE_n<<": "<<std::setw(20)<<std::left<<c<<std::endl;
#define MFST_TRACE6(c,k) std::cout<<std::setw(4)<<std::left<<  FST_TRACE_n<<": "<<std::setw(20)<<std::left<<c<<k<<std::endl;
#define MFST_TRACE7      std::cout<<std::setw(4)<<std::left<<state.lenta_position<<": " \
                                 <<std::setw(20)<<std::left<<rule.getCRule(rbuf,state.nrulechain) \
                                 <<std::endl;
```