```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

#define MAXNISXTXT 50
#define NSINT     201
#define NCEL       20
#define NDST      500
#define NVXOD      53
#define NSTROKA   200
#define NNETRM     16
#define MAXLTXT    50
#define MAXFORMT   30

#define NSYM      100

/*
***** Б а з а  данных компилятора
*/

/*
***** Б л о к  об'явления массива с исходным текстом
*/

int  NISXTXT;                          /* длина массива         */
char ISXTXT [MAXNISXTXT][80];          /* тело массива          */

/*
***** Б л о к  об'явления рабочих переменных
*/

int I1,I2,I3,I4;                       /* счетчики циклов       */

char PREDSYM = ' ';                    /*последний обработанный  */
                                       /*символ в уплотненном    */
                                       /*исходном тексте         */

char STROKA [ NSTROKA ];               /*место хранения уплотнен-*/
                                       /*ного исходного текста   */

int I,J,K,L;                           /*текущие индексы соответ-*/
                                       /*ственно в:              */
                                       /* - уплотненном тексте;  */
                                       /* - табл.грамм.правил;   */
                                       /* - стеке поставленных   */
                                       /*целей;                  */
                                       /* - стеке достигнутых    */
                                       /*целей;                  */

union                                  /*шаблон для генерации    */
 {                                     /*записи выходного файла  */
  char    BUFCARD [80];                /*на АССЕМБЛЕРЕ IBM 370   */
  struct
   {
    char METKA   [8];
    char PROB1;
    char OPERAC  [5];
    char PROB2;
    char OPERAND [12];
    char PROB3;
    char COMM    [52];
   } _BUFCARD;
 } ASS_CARD ;

char ASSTXT [ MAXLTXT ][80];           /*массив для хранения     */
                                       /*выходного текста на     */
                                       /*АССЕМБЛЕРЕ IBM 370      */

int  IASSTXT;                          /*индекс выходного массива*/

char FORMT [MAXFORMT] [9];             /*массив для форматирован-*/
                                       /*ного  (в виде последова-*/
```

```
                                    /*тельности 9-ти позицион-*/
                                    /*ных строк-лексем) пред- */
                                    /*ставления интерпретиру- */
                                    /*емого фрагмента исходно-*/
                                    /*го плотного текста      */

int  IFORMT;                          /*индекс форматированного */
                                    /*массива              */
/*
***** Б л о к  об'явления таблиц базы данных
*/

/*
***** Т а б л и ц а, используемая как магазин (стек) достижений
*/

struct
 {
  char DST1 [ 4 ];
  int  DST2;
  int  DST3;
  int  DST4;
  int  DST5;
 } DST [ NDST ];

/*
***** Т а б л и ц а, используемая как магазин (стек) целей
*/

struct
 {
  char CEL1 [ 4 ];
  int  CEL2;
  int  CEL3;
 } CEL [ NCEL ];

/*
***** Т а б л и ц а, синтаксических правил, записанных в форме распознавания,
***** сгруппированных в "кусты" и представленных в виде двухнапрвленного
***** списка с альтернативными разветвлениями
*/

struct
 {
  int  POSL;
  int  PRED;
  char DER[4];
  int  ALT;
 } SINT [ NSINT ] =
/*  _________ _________ _______ _______ ______
    | NN    :   посл : пред  :  дер  : альт |
    |_________:_________:_______:_______:______|                    */
{
 {/*.  0   .*/  -1 ,  -1 , "***" ,  -1 },
  /*                          вход с символа - 0    */
 {/*.  1   .*/   2 ,   0 , "0  " ,   0 },
 {/*.  2   .*/   3 ,   1 , "CIF" ,   0 },
 {/*.  3   .*/   0 ,   2 , "*  " ,   0 },
  /*                          вход с символа - 1    */
 {/*.  4   .*/   5 ,   0 , "1  " ,   0 },
 {/*.  5   .*/   6 ,   4 , "CIF" ,   7 },
 {/*.  6   .*/   0 ,   5 , "*  " ,   0 },

 {/*.  7   .*/   8 ,   4 , "MAN" ,   0 },
 {/*.  8   .*/   0 ,   7 , "*  " ,   0 },
  /*                          вход с символа - 2    */
 {/*.  9   .*/  10 ,   0 , "2  " ,   0 },
 {/*.  10  .*/  11 ,   9 , "CIF" ,   0 },
 {/*.  11  .*/   0 ,  10 , "*  " ,   0 },
  /*                          вход с символа - 3    */
 {/*.  12  .*/  13 ,   0 , "3  " ,   0 },
 {/*.  13  .*/  14 ,  12 , "CIF" ,   0 },
 {/*.  14  .*/   0 ,  13 , "*  " ,   0 },
```

```
 /*                          вход с символа - 4    */
{/*. 15  .*/  16,   0,"4 ",   0},
{/*. 16  .*/  17,  15,"CIF",   0},
{/*. 17  .*/   0,  16,"* ",   0},
 /*                          вход с символа - 5    */
{/*. 18  .*/  19,   0,"5 ",   0},
{/*. 19  .*/  20,  18,"CIF",   0},
{/*. 20  .*/   0,  19,"* ",   0},
 /*                          вход с символа - 6    */
{/*. 21  .*/  22,   0,"6 ",   0},
{/*. 22  .*/  23,  21,"CIF",   0},
{/*. 23  .*/   0,  22,"* ",   0},
 /*                          вход с символа - 7    */
{/*. 24  .*/  25,   0,"7 ",   0},
{/*. 25  .*/  26,  24,"CIF",   0},
{/*. 26  .*/   0,  25,"* ",   0},
 /*                          вход с символа - 8    */
{/*. 27  .*/  28,   0,"8 ",   0},
{/*. 28  .*/  29,  27,"CIF",   0},
{/*. 29  .*/   0,  28,"* ",   0},
 /*                          вход с символа - 9    */
{/*. 30  .*/  31,   0,"9 ",   0},
{/*. 31  .*/  32,  30,"CIF",   0},
{/*. 32  .*/   0,  31,"* ",   0},
 /*                          вход с символа - A    */
{/*. 33  .*/  34,   0,"A ",   0},
{/*. 34  .*/  35,  33,"BUK",   0},
{/*. 35  .*/   0,  34,"* ",   0},
 /*                          вход с символа - B    */
{/*. 36  .*/  37,   0,"B ",   0},
{/*. 37  .*/  38,  36,"BUK",   0},
{/*. 38  .*/   0,  37,"* ",   0},
 /*                          вход с символа - C    */
{/*. 39  .*/  40,   0,"C ",   0},
{/*. 40  .*/  41,  39,"BUK",   0},
{/*. 41  .*/   0,  40,"* ",   0},
 /*                          вход с символа - D    */
{/*. 42  .*/  43,   0,"D ",   0},
{/*. 43  .*/  44,  42,"BUK",  45},
{/*. 44  .*/   0,  43,"* ",   0},

{/*. 45  .*/  46,  42,"C ",   0},
{/*. 46  .*/  47,  45,"L ",   0},
{/*. 47  .*/  48,  46," ",    0},
{/*. 48  .*/  49,  47,"IDE",   0},
{/*. 49  .*/  50,  48," ",    0},
{/*. 50  .*/  51,  49,"B ", 187},
{/*. 51  .*/  52,  50,"I ",   0},
{/*. 52  .*/  53,  51,"N ",   0},
{/*. 53  .*/  54,  52," ",    0},
{/*. 54  .*/  55,  53,"F ",   0},
{/*. 55  .*/  56,  54,"I ",   0},
{/*. 56  .*/  57,  55,"X ",   0},
{/*. 57  .*/  58,  56,"E ",   0},
{/*. 58  .*/  59,  57,"D ",   0},
{/*. 59  .*/  60,  58,"( ",   0},
{/*. 60  .*/  61,  59,"RZR",   0},
{/*. 61  .*/  62,  60,") ",   0},
{/*. 62  .*/  63,  61,"; ",  65},
{/*. 63  .*/  64,  62,"ODC",   0},
{/*. 64  .*/  65,  63,"* ",   0},

{/*. 65  .*/  66,  61,"I ",   0},
{/*. 66  .*/  67,  65,"N ",   0},
{/*. 67  .*/  68,  66,"I ",   0},
{/*. 68  .*/  69,  67,"T ",   0},
{/*. 69  .*/  70,  68,"( ",   0},
{/*. 70  .*/  71,  69,"LIT",   0},
{/*. 71  .*/  72,  70,") ",   0},
{/*. 72  .*/  73,  71,"; ",   0},
{/*. 73  .*/ 186,  72,"ODC",   0},
 /*                          вход с символа - E    */
{/*. 74  .*/  75,   0,"E ",   0},
```

```
{/*.  75   .*/   76 ,   74 , "N " ,  82 },
{/*.  76   .*/   77 ,   75 , "D " ,   0 },
{/*.  77   .*/   78 ,   76 , " " ,   0 },
{/*.  78   .*/   79 ,   77 , "IPR" ,   0 },
{/*.  79   .*/   80 ,   78 , "; " ,   0 },
{/*.  80   .*/   81 ,   79 , "OEN" ,   0 },
{/*.  81   .*/    0 ,   80 , "* " ,   0 },

{/*.  82   .*/   83 ,   74 , "BUK" ,   0 },
{/*.  83   .*/    0 ,   82 , "* " ,   0 },
 /*                          вход с символа - M     */
{/*.  84   .*/   85 ,    0 , "M " ,   0 },
{/*.  85   .*/   86 ,   84 , "BUK" ,   0 },
{/*.  86   .*/    0 ,   85 , "* " ,   0 },
 /*                          вход с символа - P     */
{/*.  87   .*/   88 ,    0 , "P " ,   0 },
{/*.  88   .*/   89 ,   87 , "BUK" ,   0 },
{/*.  89   .*/    0 ,   88 , "* " ,   0 },
 /*                          вход с символа - X     */
{/*.  90   .*/   91 ,    0 , "X " ,   0 },
{/*.  91   .*/   92 ,   90 , "BUK" ,   0 },
{/*.  92   .*/    0 ,   91 , "* " ,   0 },
 /*                          вход с символа - BUK   */
{/*.  93   .*/   94 ,    0 , "BUK" ,   0 },
{/*.  94   .*/   95 ,   93 , "IDE" ,   0 },
{/*   95   .*/    0 ,   94 , "* " ,   0 },
 /*                          вход с символа - IDE   */
{/*.  96   .*/   97 ,    0 , "IDE" ,   0 },
{/*.  97   .*/   98 ,   96 , "BUK" , 100 },
{/*.  98   .*/   99 ,   97 , "IDE" ,   0 },
{/*.  99   .*/    0 ,   98 , "* " ,   0 },

{/*. 100   .*/  101 ,   96 , "CIF" , 103 },
{/*. 101   .*/  102 ,  100 , "IDE" ,   0 },
{/*. 102   .*/    0 ,  101 , "* " ,   0 },

{/*. 103   .*/  104 ,   96 , "IPE" , 105 },
{/*. 104   .*/    0 ,  103 , "* " ,   0 },

{/*. 105   .*/  106 ,   96 , "IPR" ,   0 },
{/*. 106   .*/    0 ,  105 , "* " ,   0 },
 /*                          вход с символа - +     */
{/*. 107   .*/  108 ,    0 , "+ " ,   0 },
{/*. 108   .*/  109 ,  107 , "ZNK" ,   0 },
{/*. 109   .*/    0 ,  108 , "* " ,   0 },
 /*                          вход с символа - -     */
{/*. 110   .*/  111 ,    0 , "- " ,   0 },
{/*. 111   .*/  112 ,  110 , "ZNK" ,   0 },
{/*. 112   .*/    0 ,  111 , "* " ,   0 },
 /*                          вход с символа - IPR   */
{/*. 113   .*/  114 ,    0 , "IPR" ,   0 },
{/*. 114   .*/  115 ,  113 , ": " ,   0 },
{/*. 115   .*/  116 ,  114 , "P " ,   0 },
{/*. 116   .*/  117 ,  115 , "R " ,   0 },
{/*. 117   .*/  118 ,  116 , "O " ,   0 },
{/*. 118   .*/  119 ,  117 , "C " ,   0 },
{/*. 119   .*/  120 ,  118 , "  " ,   0 },
{/*. 120   .*/  121 ,  119 , "O " ,   0 },
{/*. 121   .*/  122 ,  120 , "P " ,   0 },
{/*. 122   .*/  123 ,  121 , "T " ,   0 },
{/*. 123   .*/  124 ,  122 , "I " ,   0 },
{/*. 124   .*/  125 ,  123 , "O " ,   0 },
{/*. 125   .*/  126 ,  124 , "N " ,   0 },
{/*. 126   .*/  127 ,  125 , "S " ,   0 },
{/*. 127   .*/  128 ,  126 , "( " ,   0 },
{/*. 128   .*/  129 ,  127 , "M " ,   0 },
{/*. 129   .*/  130 ,  128 , "A " ,   0 },
{/*. 130   .*/  131 ,  129 , "I " ,   0 },
{/*. 131   .*/  132 ,  130 , "N " ,   0 },
{/*. 132   .*/  133 ,  131 , ") " ,   0 },
{/*. 133   .*/  134 ,  132 , "; " ,   0 },
{/*. 134   .*/  135 ,  133 , "OPR" ,   0 },
{/*. 135   .*/    0 ,  134 , "* " ,   0 },
```

```
/*                          вход с символа - CIF   */
{/*. 136   .*/  137 ,   0 , "CIF" ,   0 },
{/*. 137   .*/  138 ,  136 , "RZR" ,   0 },
{/*. 138   .*/   0 ,   0 , "* " ,   0 },
 /*                          вход с символа - RZR   */
{/*. 139   .*/  140 ,   0 , "RZR" ,   0 },
{/*. 140   .*/  141 ,  139 , "CIF" ,   0 },
{/*. 141   .*/  142 ,  140 , "RZR" ,   0 },
{/*. 142   .*/   0 ,  141 , "* " ,   0 },
 /*                          вход с символа - MAN   */
{/*. 143   .*/  144 ,   0 , "MAN" ,   0 },
{/*. 144   .*/  145 ,  143 , "B " , 147 },
{/*. 145   .*/  146 ,  144 , "LIT" ,   0 },
{/*. 146   .*/   0 ,  145 , "* " ,   0 },

{/*. 147   .*/  148 ,  143 , "0 " , 150 },
{/*. 148   .*/  149 ,  147 , "MAN" ,   0 },
{/*. 149   .*/   0 ,  148 , "* " ,   0 },

{/*. 150   .*/  151 ,  143 , "1 " ,   0 },
{/*. 151   .*/  152 ,  150 , "MAN" ,   0 },
{/*. 152   .*/   0 ,  151 , "* " ,   0 },
 /*                          вход с символа - IPE   */
{/*. 153   .*/  154 ,   0 , "IPE" ,   0 },
{/*. 154   .*/  155 ,  153 , "= " , 159 },
{/*. 155   .*/  156 ,  154 , "AVI" ,   0 },
{/*. 156   .*/  157 ,  155 , "; " ,   0 },
{/*. 157   .*/  158 ,  156 , "OPA" ,   0 },
{/*. 158   .*/   0 ,  157 , "* " ,   0 },

{/*. 159   .*/  160 ,  153 , "AVI" ,   0 },
{/*. 160   .*/   0 ,  159 , "* " ,   0 },
 /*                          вход с символа - LIT   */
{/*. 161   .*/  162 ,   0 , "LIT" ,   0 },
{/*. 162   .*/  163 ,  161 , "AVI" ,   0 },
{/*. 163   .*/   0 ,  162 , "* " ,   0 },
 /*.                         вход с символа - AVI   */
{/*. 164   .*/  165 ,   0 , "AVI" ,   0 },
{/*. 165   .*/  166 ,  164 , "ZNK" ,   0 },
{/*. 166   .*/  167 ,  165 , "LIT" , 168 },
{/*. 167   .*/  197 ,  166 , "AVI" ,   0 },

{/*. 168   .*/  169 ,  165 , "IPE" ,   0 },
{/*. 169   .*/  170 ,  168 , "AVI" ,   0 },
{/*. 170   .*/   0 ,  169 , "* " ,   0 },
 /*                          вход с символа - OPR   */
{/*. 171   .*/  172 ,   0 , "OPR" ,   0 },
{/*. 172   .*/  173 ,  171 , "TEL" ,   0 },
{/*. 173   .*/  174 ,  172 , "OEN" ,   0 },
{/*. 174   .*/  175 ,  173 , "PRO" ,   0 },
{/*. 175   .*/   0 ,  174 , "* " ,   0 },
 /*.                         вход с символа - ODC   */
{/*. 176   .*/  177 ,   0 , "ODC" ,   0 },
{/*. 177   .*/  178 ,  176 , "TEL" ,   0 },
{/*. 178   .*/   0 ,  177 , "* " ,   0 },
 /*.                         вход с символа - TEL   */
{/*. 179   .*/  180 ,   0 , "TEL" ,   0 },
{/*. 180   .*/  181 ,  179 , "ODC" , 183 },
{/*. 181   .*/  182 ,  180 , "TEL" ,   0 },
{/*. 182   .*/   0 ,  181 , "* " ,   0 },

{/*. 183   .*/  184 ,  179 , "OPA" ,   0 },
{/*. 184   .*/  185 ,  183 , "TEL" ,   0 },
{/*. 185   .*/   0 ,  184 , "* " ,   0 },

{/*. 186   .*/   0 ,   73 , "* " ,   0 },


{/*. 187   .*/  188 ,   49 , "C " ,   0 },
{/*. 188   .*/  189 ,  187 , "H " ,   0 },
{/*. 189   .*/  190 ,  188 , "A " ,   0 },
{/*. 190   .*/  191 ,  189 , "R " ,   0 },
{/*. 191   .*/  192 ,  190 , "( " ,   0 },
```

```
 {/*. 192    .*/  193 ,  191 , "RZR" ,   0 },
 {/*. 193    .*/  194 ,  192 , ") " ,    0 },
 {/*. 194    .*/  195 ,  193 , "; " ,    0 },
 {/*. 195    .*/  196 ,  194 , "ODC" ,   0 },
 {/*. 196    .*/    0 ,  195 , "* " ,    0 },

 {/*. 197    .*/    0 ,  166 , "* " ,    0 },

 {/*. 198    .*/  199 ,    0 , "* " ,    0 },
 {/*. 199    .*/  200 ,  198 , "ZNK" ,   0 },
 {/*. 200    .*/    0 ,  199 , "* " ,    0 }
};

/*
***** Т а б л и ц а  входов в "кусты" ( корней )грамматических правил,
***** содержащая тип ( терминальность или нетерминальность ) корневых
***** символов
*/

struct
 {
  char SYM [4];
  int  VX;
  char TYP;
 } VXOD [ NVXOD ] =
/*   _________ ___________ _____ ______
    | NN     |    символ | вход| тип  |
    |_________|___________|_____|______|                          */

{
 {/*.  1    .*/  "AVI" , 164 , 'N' },
 {/*.  2    .*/  "BUK" ,  93 , 'N' },
 {/*.  3    .*/  "CIF" , 136 , 'N' },
 {/*.  4    .*/  "IDE" ,  96 , 'N' },
 {/*.  5    .*/  "IPE" , 153 , 'N' },
 {/*.  6    .*/  "IPR" , 113 , 'N' },
 {/*.  7    .*/  "LIT" , 161 , 'N' },
 {/*.  8    .*/  "MAN" , 143 , 'N' },
 {/*.  9    .*/  "ODC" , 176 , 'N' },
 {/*. 10    .*/  "OEN" ,   0 , 'N' },
 {/*. 11    .*/  "OPA" ,   0 , 'N' },
 {/*. 12    .*/  "OPR" , 171 , 'N' },
 {/*. 13    .*/  "PRO" ,   0 , 'N' },
 {/*. 14    .*/  "RZR" , 139 , 'N' },
 {/*. 15    .*/  "TEL" , 179 , 'N' },
 {/*. 16    .*/  "ZNK" ,   0 , 'N' },
 {/*. 17    .*/  "A  " ,  33 , 'T' },
 {/*. 18    .*/  "B  " ,  36 , 'T' },
 {/*. 19    .*/  "C  " ,  39 , 'T' },
 {/*. 20    .*/  "D  " ,  42 , 'T' },
 {/*. 21    .*/  "E  " ,  74 , 'T' },
 {/*. 22    .*/  "M  " ,  84 , 'T' },
 {/*. 23    .*/  "P  " ,  87 , 'T' },
 {/*. 24    .*/  "X  " ,  90 , 'T' },
 {/*. 25    .*/  "0  " ,   1 , 'T' },
 {/*. 26    .*/  "1  " ,   4 , 'T' },
 {/*. 27    .*/  "2  " ,   9 , 'T' },
 {/*. 28    .*/  "3  " ,  12 , 'T' },
 {/*. 29    .*/  "4  " ,  15 , 'T' },
 {/*. 30    .*/  "5  " ,  18 , 'T' },
 {/*. 31    .*/  "6  " ,  21 , 'T' },
 {/*. 32    .*/  "7  " ,  24 , 'T' },
 {/*. 33    .*/  "8  " ,  27 , 'T' },
 {/*. 34    .*/  "9  " ,  30 , 'T' },
 {/*. 35    .*/  "+  " , 107 , 'T' },
 {/*. 36    .*/  "-  " , 110 , 'T' },
 {/*. 37    .*/  ": " ,   0 , 'T' },
 {/*. 38    .*/  "I  " ,   0 , 'T' },
 {/*. 39    .*/  "R  " ,   0 , 'T' },
 {/*. 40    .*/  "N  " ,   0 , 'T' },
 {/*. 41    .*/  "O  " ,   0 , 'T' },
 {/*. 42    .*/  "T  " ,   0 , 'T' },
 {/*. 43    .*/  "S  " ,   0 , 'T' },
```

```
 {/*. 44   .*/  "( " ,  0 , 'T' },
 {/*. 45   .*/  ") " ,  0 , 'T' },
 {/*. 46   .*/  "  " ,  0 , 'T' },
 {/*. 47   .*/  "; " ,  0 , 'T' },
 {/*. 48   .*/  "L " ,  0 , 'T' },
 {/*. 49   .*/  "F " ,  0 , 'T' },
 {/*. 50   .*/  "= " ,  0 , 'T' },
 {/*. 51   .*/  "H " ,  0 , 'T' },
 {/*. 52   .*/  "* " , 198 , 'T' }
};

/*
***** Т а б л и ц а  матрицы смежности - основа построения матрицы
***** преемников
*/

char TPR [ NVXOD ] [ NNETRM ] =
 {
/*
  ___________ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___
  |      AVI:BUK:CIF:IDE:IPE:IPR:LIT:MAN:ODC:OEN:OPA:OPR:PRO:RZR:TEL:ZNK|
  |___________:___:___:___:___:___:___:___:___:___:___:___:___:___:___:___| */
 {/*AVI*/ 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*BUK*/ 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*CIF*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 },
 {/*IDE*/ 0 , 0 , 0 , 1 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*IPE*/ 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 },
 {/*IPR*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 },
 {/*LIT*/ 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*MAN*/ 0 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*ODC*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 },
 {/*OEN*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*OPA*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*OPR*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 },
 {/*PRO*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*RZR*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 },
 {/*TEL*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 },
 {/*ZNK*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
/*
  ___________ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___
  |      AVI:BUK:CIF:IDE:IPE:IPR:LIT:MAN:ODC:OEN:OPA:OPR:PRO:RZR:TEL:ZNK|
  |___________:___:___:___:___:___:___:___:___:___:___:___:___:___:___:___| */
 {/*  A*/ 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  B*/ 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  C*/ 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  D*/ 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  E*/ 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  M*/ 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  P*/ 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  X*/ 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  0*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  1*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  2*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  3*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  4*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  5*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 },
 {/*  6*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 },
 {/*  7*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
/*
  ___________ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___
  |      AVI:BUK:CIF:IDE;IPE:IPR:LIT:MAN:ODC:OEN:OPA:OPR:PRO:RZR:TEL:ZNK|
  |___________:___:___:___:___:___:___:___:___:___:___:___:___:___:___:___| */
 {/*  8*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  9*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  +*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 },
 {/*  -*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 },
 {/*  :*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  I*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  R*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  N*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  O*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  T*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  S*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
```

```
 {/*  (*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  )*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  ;*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*   */ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  L*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
/*
  ___________ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___
 |      AVI:BUK:CIF:IDE:IPE:IPR:LIT:MAN:ODC:OEN;OPA:OPR:PR0:RZR:TEL:ZNK|
 |___________:___:___:___:___:___:___:___:___:___:___:___:___:___:___:___:___:___| */
 {/*  F*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  =*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  H*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
 {/*  **/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 }
/*|_____________________________________________________________________________| */
 };
/*.....................................................................*/

/*
****** Н А Ч А Л О  обработки исходного текста
*/

/*.....................................................................*/

void compress_ISXTXT()                  /* Программа уплотнения   */
                                        /* исходного текста путем */
                                        /* удаления "лишних"     */
                                        /* пробелов, выполняющая  */
                                        /* роль примитивного лек- */
                                        /* сического анализатора  */
{
 I3 = 0;
 for ( I1 = 0 ; I1 < NISXTXT ; I1++ )
  {
   for ( I2 = 0 ; I2 < 80 ; I2++ )
    if ( ISXTXT [ I1 ][ I2 ] != '\x0' )
     {
      if ( ISXTXT [ I1 ][ I2 ] == ' ' &&
           ( PREDSYM == ' ' || PREDSYM == ';' ||
             PREDSYM == ')' || PREDSYM == ':' ||
             PREDSYM == '('
           )
          )
       {
        PREDSYM = ISXTXT [ I1 ][ I2 ];
        goto L2;
       }

      if
        (
         ( ISXTXT [ I1 ][ I2 ] == '+' ||
           ISXTXT [ I1 ][ I2 ] == '-' ||
           ISXTXT [ I1 ][ I2 ] == '=' ||
           ISXTXT [ I1 ][ I2 ] == '(' ||
           ISXTXT [ I1 ][ I2 ] == ')' ||
           ISXTXT [ I1 ][ I2 ] == '*'
         )
          &&
         PREDSYM == ' '
        )
       {
        I3-- ;
        goto L1;
       }


      if ( ISXTXT [ I1 ][ I2 ] == ' ' &&
           ( PREDSYM == '+' || PREDSYM == '-' ||
             PREDSYM == '=' || PREDSYM == '*'
           )
          )
       {
        goto L2;
       }
```

```
L1:
      PREDSYM = ISXTXT [ I1 ][ I2 ];
      STROKA [ I3 ] = PREDSYM;
      I3++ ;

L2:   continue;
     }
    else
     break;
  }
 STROKA [I3] = '\x0';
}

/*.....*/

void build_TPR ()                    /* Построение таблицы     */
                                     /* преемников из матрицы  */
                                     /* смежности по алгоритму */
                                     /* Варшалла               */
{
 for ( I1 = 0; I1 < NNETRM; I1++ )
  {
   for ( I2 = 0; I2 < NVXOD; I2++ )
    {
     if ( TPR [ I2 ][ I1 ] & ( I1 != I2 ) )
      {
       for ( I3 = 0; I3 < NNETRM; I3++ )
        TPR [ I2 ][ I3 ] |= TPR [ I1 ][ I3 ];
      }
    }
  }
}

/*.....*/

void mcel ( char* T1, int T2, int T3 )          /* программа заполнения   */
{                                    /* ячейки стека поставлен-*/
                                     /* ных целей              */
 strcpy ( CEL [ K ].CEL1, T1 );
 CEL [ K ].CEL2 = T2;
 CEL [ K ].CEL3 = T3;
 K++;
}

/*.....*/

void mdst ( char* T1, int T2, int T3, int T4, int T5 )
{                                    /* программа заполнения   */
 strcpy ( DST [ L ].DST1, T1 );              /* ячейки стека достигну- */
 DST [ L ].DST2 = T2;                /* тых целей              */
 DST [ L ].DST3 = T3;
 DST [ L ].DST4 = T4;
 DST [ L ].DST5 = T5;
 L++;
}


/*.....*/
                                     /* п р о г р а м м а      */
int numb ( char* T1, int T2 )                /* вычисления порядкового */
                                     /* номера строки в табл.  */
                                     /* VXOD, соответствующей  */
                                     /* строке-параметру функц.*/
{
 int k;

 for ( I1 = 0; I1 < NVXOD; I1++ )
  {
   for ( k = 0; k < T2; k++ )
    {
     if ( (*(T1+k) != VXOD [ I1 ].SYM [k] ) )
      goto numb1;
```

```
    }
  if ( (VXOD [ I1 ].SYM [k] == '\x0') ||
      (VXOD [ I1 ].SYM [k] == ' ' )
    )
   return ( I1 );
numb1:
   continue;
  }
 return -1;
}

/*.....................................................................*/
                                        /*  п р о г р а м м а  */
int sint_ANAL ()                            /*  построения дерева  */
                                        /*синтаксического разбора,*/
{                                      /*выполняющая роль синтак-*/
                                        /*сического анализатора  */
 I4 = 0;

L1:

 K = 0;
 L = 0;
 I = 0;
 J = 1;
 mcel ( "PRO" , I , 999 );

 if (!TPR [numb ( &STROKA [I], 1 )][numb ( "PRO", 3 )])
  return 1;

L2:

 J = VXOD [ numb ( &STROKA [ I ], 1 ) ].VX ;

L3:

 J = SINT [ J ].POSL;

L31:

 I++;

 if ( I > I4 )


  I4 = I;

 if (VXOD [ numb ( SINT [ J ].DER, 3 ) ].TYP == 'T')
 {

  if ( STROKA [ I ] == SINT [ J ].DER [ 0 ] )
   goto L3;
  else
   goto L8;

 }

L4:

 if ( SINT [ SINT [ J ].POSL ].DER [ 0 ] == '*' )
 {
  I--;

  if ( !strcmp (SINT [J].DER, CEL [K-1].CEL1 ) )
   {
    mdst ( CEL[K-1].CEL1,CEL[K-1].CEL2,CEL[K-1].CEL3,I,J );

    if ( !strcmp( CEL[K-1].CEL1 , "PRO" ) )
     return 0;

L5:

    if (TPR [numb (CEL[K-1].CEL1, 3)] [numb (CEL[K-1].CEL1, 3)])
```

```
      {
       J = VXOD [ numb ( CEL[K-1].CEL1, 3 ) ].VX;
       goto L3;
      }

L6:

      J = CEL[K-1].CEL3;
      K--;
      goto L3;
     }

    if (!TPR [numb (SINT[J].DER, 3)] [numb (CEL[K-1].CEL1, 3)])
     goto L9;

    mdst ( SINT[J].DER, CEL[K-1].CEL2,0,I,J );
    J = VXOD [numb (SINT[J].DER, 3)].VX;
    goto L3;
   }

  if (!TPR [numb (&STROKA [I], 1)] [numb (SINT[J].DER, 3)])
   goto L8;

  mcel ( SINT[J].DER,I,J );
  goto L2;

L8:

  I--;

L9:

  if (SINT[J].ALT != 0)
   {
    J = SINT[J].ALT;
    goto L31;
   }

  J = SINT[J].PRED;

  if
   (
    ( VXOD [numb (SINT[J].DER, 3)].TYP == 'N' )
    &&
    ( SINT[J].PRED > 0 )
   )
   {
    mcel (DST[L-1].DST1, DST[L-1].DST2, DST[L-1].DST3);

L10:

    J = DST[L-1].DST5;
    L--;
    goto L9;
   }

  if
   (
    ( VXOD [numb (SINT[J].DER, 3)].TYP == 'N' )
    &&
    ( SINT[J].PRED == 0 )
   )
   {
    if (!strcmp ( CEL[K-1].CEL1, DST[L-1].DST1 ) )
     goto L6;
    else
     goto L10;
   }

  if ( SINT[J].PRED > 0 )
   goto L8;

  J = CEL[K-1].CEL3;
```

```
 K--;

 if ( J == 999 )
  return 2;
 else
  goto L8;

 }

/*.....................................................................*/

struct                                  /* таблица имен меток и   */
 {                                     /* переменных, заполняемая*/
  char NAME [8];                             /* на первом проходе се-  */
  char TYPE;                               /* мантического вычисления*/
  char RAZR [5];                            /* и используемая на вто- */
  char INIT [50];                           /* ром проходе семантичес-*/
 } SYM [ NSYM ];                            /* кого вычисления       */

int ISYM = 0;                              /* текущий индекс таблицы */
                                          /* имен                  */

char NFIL [30]="\x0";                             /* хранилище имени транс- */
                                          /* лируемой программы    */

/*.....................................................................*/

long int VALUE ( char* s )                        /* п р о г р а м м а     */
 {                                       /* перевода двоичной     */
  long int S;                               /* константы из ASCIIz-ви-*/
  int i;                                  /* да во внутреннее пред- */
                                          /* ставление типа long int*/
  i = 0;
  S = 0;
  while ( *(s + i) != 'B' )

   {

    S <<= 1;
    if ( *(s + i) == '1' )
     S++;
    i++;
   }

  return (S);
 }



/*.....................................................................*/
void FORM ()                                    /* п р о г р а м м а     */
 {                                       /* представления фрагмента*/
                                          /* плотного текста в виде */
                                          /* массива 9-ти символьных*/
                                          /* лексем                */
  int i,j;

  for ( IFORMT = 0; IFORMT < MAXFORMT; IFORMT++ )
   memcpy ( FORMT [IFORMT], "\x0\x0\x0\x0\x0\x0\x0\x0\x0", 9 );

  IFORMT = 0;
  j = DST [I2].DST2;

FORM1:

  for ( i = j; i <= DST [I2].DST4+1; i++ )
   {
    if ( STROKA [i] == ':' || STROKA [i] == ' ' ||
         STROKA [i] == '(' || STROKA [i] == ')' ||
         STROKA [i] == ';' || STROKA [i] == '+' ||
         STROKA [i] == '-' || STROKA [i] == '=' ||
         STROKA [i] == '*'
       )
```

```
      {
       FORMT [IFORMT] [i-j] = '\x0';
       IFORMT ++;
       j = i+1;
       goto FORM1;
      }
   else
       FORMT [IFORMT][i-j] = STROKA [i];

  }

 return;
}

/*.....................................................................*/
                                    /* п р о г р а м м а     */
void ZKARD ()                           /* записи очередной сгене-*/
{                                  /* рированной записи вы-  */
                                    /* ходного файла в массив */
                                    /* ASSTXT                 */
 char i;
 memcpy ( ASSTXT [ IASSTXT++ ],
                    ASS_CARD.BUFCARD, 80 );

 for ( i = 0; i < 79; i++ )
  ASS_CARD.BUFCARD [i] = ' ';
 return;
}

/*.....................................................................*/
                                    /* п р о г р а м м а     */
                                    /* семантич. вычисления   */
                                    /* нетерминала AVI на пер-*/
                                    /* вом проходе.   Здесь   */
                                    /* AVI -   "арифм.выраж." */
int AVI1 ()
{
 return 0;
}

/*.....................................................................*/

                                    /* п р о г р а м м а     */
                                    /* семантич. вычисления   */
                                    /* нетерминала BUK на пер-*/
                                    /* вом проходе.   Здесь   */
                                    /* BUK -   "буква"        */
int BUK1 ()
{
 return 0;
}

/*.....................................................................*/

                                    /* п р о г р а м м а     */
                                    /* семантич. вычисления   */
                                    /* нетерминала CIF на пер-*/
                                    /* вом проходе.   Здесь   */
                                    /* CIF -   "цифра"        */
int CIF1 ()
{
 return 0;
}

/*.....................................................................*/

                                    /* п р о г р а м м а     */
                                    /* семантич. вычисления   */
                                    /* нетерминала IDE на пер-*/
                                    /* вом проходе.   Здесь   */
                                    /* IDE -   "идентификатор"*/
int IDE1 ()
{
```

```
 return 0;
}

/*.....................................................................*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала IPE на пер-*/
                                /* вом проходе.   Здесь   */
                                /* IPE - "имя переменной" */
int IPE1 ()
{
 return 0;
}

/*.....................................................................*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала IPR на пер-*/
                                /* вом проходе.   Здесь   */
                                /* IPR -   "имя программы" */
int IPR1 ()
{
 return 0;
}

/*.....................................................................*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала LIT на пер-*/
                                /* вом проходе.   Здесь   */
                                /* LIT -   "литерал"     */
int LIT1 ()
{
 return 0;
}

/*.....................................................................*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала MAN на пер-*/
                                /* вом проходе.   Здесь   */
                                /* MAN -   "мантисса"    */
int MAN1 ()
{
 return 0;
}

/*.....................................................................*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала ODC на пер-*/
                                /* вом проходе.   Здесь   */
                                /* ODC - "операт.ПЛ1- DCL"*/
int ODC1 ()
{
 int i;
 FORM ();                         /* форматирование ПЛ1-опе-*/
                                /* ратора DCL            */

 for ( i = 0; i < ISYM; i++ )              /* если фиксируем повтор- */
 {                              /* повторное объявление   */
  if (  !strcmp ( SYM [i].NAME, FORMT [1] ) &&  /* второго терма оператора*/
       strlen ( SYM [i].NAME ) ==           /* DCL, то               */
                  strlen ( FORMT [1] )
    )
   return 6;                        /* завершение программы   */
                                /* по ошибке             */
 }
```

```
  strcpy ( SYM [ISYM].NAME, FORMT [1] );         /* при отсутствии повтор- */
  strcpy ( SYM [ISYM].RAZR, FORMT [4] );         /* ного объявления иденти-*/
                                                 /* фикатора запоминаем его*/
                                                 /* вместе с разрядностью в*/
                                                 /* табл.SYM             */

  if ( !strcmp ( FORMT [2], "BIN" ) &&           /* если идентификатор оп- */
               !strcmp ( FORMT [3], "FIXED" ) )/* ределен как bin fixed, */
   {
    SYM [ISYM].TYPE = 'B';                       /* то устанавливаем тип   */
                                                 /* идентификатора = 'B' и */
    goto ODC11;                                  /* идем на продолжение об-*/
                                                 /* работки, а           */
   }
  else                                           /* иначе                */
   {
    SYM [ISYM].TYPE = 'U';                       /* устанавливаем тип иден-*/
                                                 /* тификатора = 'U'  и    */
    return 2;                                    /* завершаем программу    */
                                                 /* по ошибке            */
   }

ODC11:                                           /* если идентификатор     */
                                                 /* имеет начальную иници- */
  if ( !strcmp ( FORMT [5], "INIT" ) )           /* ализацию, то запомина- */
    strcpy ( SYM [ISYM++].INIT, FORMT [6] );     /* ем в табл. SYM это на- */
                                                 /* чальное значение, а    */
  else                                           /* иначе                */
    strcpy ( SYM [ISYM++].INIT, "0B" );          /* инициализируем иденти- */
                                                 /* фикатор нулем        */

    return 0;                                    /* успешное завешение     */
                                                 /* программы            */
 }

/*.....................................................................*/

                                                 /* п р о г р а м м а     */
                                                 /* семантич. вычисления   */
                                                 /* нетерминала OEN на пер-*/
                                                 /* вом проходе.   Здесь   */
                                                 /* OEN - "операт.ПЛ1-END" */
int OEN1 ()
 {
  char i = 0;
  FORM ();                                       /* форматирование ПЛ1-опе-*/
                                                 /* ратора END           */

  for ( i = 0; i < ISYM; i++ )                   /* если вторй терм опера- */
                                                 /* тора END записан в табл*/
   {                                             /* SYM и его тип = "P",то:*/
    if ( !strcmp ( SYM [i].NAME, FORMT [1] ) &&
                 (SYM [i].TYPE == 'P') &&
                 strlen (SYM [i].NAME) ==
                  strlen ( FORMT [1] ) )
     return 0;                                   /* успешное завершение    */
                                                 /* программы            */
   }

  return 1;                                      /* иначе завершение прог- */
                                                 /* раммы по ошибке        */
 }

/*.....................................................................*/

                                                 /* п р о г р а м м а     */
                                                 /* семантич. вычисления   */
                                                 /* нетерминала OPA на пер-*/
                                                 /* вом проходе.   Здесь   */
                                                 /* OPA - "операт.присваи- */
                                                 /* вания арифметический   */
int OPA1 ()
```

```
{
 return 0;
}

/*.....................................................................*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала OPR на пер-*/
                                /* вом проходе.  Здесь   */
                                /* OPR - "операт.ПЛ1-PROC"*/
int OPR1 ()
{
 FORM ();                        /* форматируем оператор   */
                                /* ПЛ1 PROC              */

 strcpy ( SYM [ISYM].NAME, FORMT [0] );        /* перепишем имя ПЛ1-прог-*/
                                /* раммы в табл. SYM,    */

 SYM [ISYM].TYPE   = 'P';                  /* установим тип этого    */
                                /* имени = 'P'           */
 SYM [ISYM++].RAZR [0] = '\x0';              /* установим разрядность  */
                                /* равной 0              */

 return 0;                       /* успешное завершение    */
                                /* программы             */
}

/*.....................................................................*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала PRO на пер-*/
                                /* вом проходе.  Здесь   */
                                /* PRO - "программа"     */

int PRO1 ()
{
 return 0;
}

/*.....................................................................*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала RZR на пер-*/
                                /* вом проходе.  Здесь   */
                                /* RZR - "разрядность"    */

int RZR1 ()
{
 return 0;
}

/*.....................................................................*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала TEL на пер-*/
                                /* вом проходе.  Здесь   */
                                /* TEL - "тело программы" */

int TEL1 ()
{
 return 0;
}

/*.....................................................................*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала ZNK на пер-*/
                                /* вом проходе.  Здесь   */
```

```
                                    /* ZNK - "знак операции"  */

int ZNK1 ()
 {
  return 0;
 }

/*.....................................................................*/

                                    /* п р о г р а м м а     */
                                    /* семантич. вычисления   */
                                    /* нетерминала AVI на вто-*/
                                    /* ром проходе.   Здесь   */
                                    /* AVI -   "арифм.выраж." */


int AVI2 ()
 {
  char i;
  FORM ();                           /*форматируем правую часть*/
                                    /*арифметического ПЛ1-опе-*/
                                    /*ратора присваивания     */

  if ( IFORMT == 1 )                      /* если правая часть одно-*/
    {                               /* термовая, то:         */
   for ( i = 0; i < ISYM; i++ )              /* ищем этот терм в табли-*/
    {                               /* це имен  и            */
     if ( !strcmp ( SYM [i].NAME, FORMT [0] )  &&/* если находим, то:     */
          strlen ( SYM [i].NAME ) ==
                      strlen ( FORMT [0] )
        )
      {
         if ( SYM [i].TYPE == 'B' )           /* в случае типа=bin fixed*/
          {

          if ( strcmp ( SYM [i].RAZR, "15" )    /* и разрядности <= 15    */
                                    <= 0 )
           memcpy ( ASS_CARD._BUFCARD.OPERAC,   /* формируем код ассембле-*/
                               "LH", 2 );/* ровской операции LH,   */
          else
           memcpy ( ASS_CARD._BUFCARD.OPERAC,   /* а при разрядности >15  */
                                "L", 1 );/* формируем код ассембле-*/
                                    /* ровской операции L     */

          strcpy ( ASS_CARD._BUFCARD.OPERAND,   /*      формируем       */
                                "RRAB," );/*       первый  и       */
          strcat ( ASS_CARD._BUFCARD.OPERAND,   /* второй операнды ассемб-*/
                               FORMT [0]);/* леровской операции     */

          ASS_CARD._BUFCARD.OPERAND [ strlen    /* вставляем разделитель  */
           ( ASS_CARD._BUFCARD.OPERAND ) ] = ' ';

          memcpy ( ASS_CARD._BUFCARD.COMM,      /* и построчный коментарий*/
           "Загрузка переменной в регистр", 29 );

          ZKARD ();                          /* запомнить операцию ас- */
                                    /* семблера  и           */
          return 0;                          /* завершить программу    */
         }
         else
          return 3;                          /* если тип терма не bin  */
                                    /* fixed,то выход по ошиб-*/
                                    /* ке                    */
      }
     }
   return 4;                           /* если терм-идентификатор*/
                                    /* неопределен, то выход  */
                                    /* по ошибке             */
  }
  else                                 /* если правая часть ариф-*/
                                    /* метического выражения  */
                                    /* двухтермовая, то:      */
  {
```

```
for ( i = 0; i < ISYM; i++ )                /* если правый терм ариф- */
 {                                  /* метического выражения  */
  if ( !strcmp ( SYM [i].NAME,              /*определен в табл.SYM,то:*/
                 FORMT [IFORMT-1] )  &&
       strlen ( SYM [i].NAME ) ==
                strlen ( FORMT [IFORMT-1] )
     )
  {
     if ( SYM [i].TYPE == 'B' )            /* если тип правого опе-  */
     {                                  /* ранда bin fixed, то:   */

      if ( STROKA [ DST [I2].DST4 -        /* если знак опер."+",то: */
       strlen( FORMT [IFORMT-1] ) ] == '+' )
       {
        if ( strcmp ( SYM [i].RAZR, "15" )  /* если разрядность прав. */
                                  <= 0 )/* операнда <= 15, то:    */
         memcpy ( ASS_CARD._BUFCARD.OPERAC,
                                 "AH", 2 );/* формируем код ассембле-*/
        else                             /* ровской операции "AH",a*/
         memcpy ( ASS_CARD._BUFCARD.OPERAC,
                                 "A", 1 );/* иначе - "A"           */
       }

      else

       {
        if ( STROKA [ DST [I2].DST4 -      /* если же знак операции  */
           strlen ( FORMT [IFORMT-1] ) ] == /* арифметического выра-  */
                                  '-' )/* жения "-", то:        */

         {
          if ( strcmp ( SYM [i].RAZR, "15" )/* при разрядности ариф-  */
                                  <= 0 )/* метич.выраж.<= 15      */
           memcpy( ASS_CARD._BUFCARD.OPERAC,/* формируем код ассембле-*/
                                 "SH", 2 );/* ровской операции "SH",F*/
          else
           memcpy( ASS_CARD._BUFCARD.OPERAC,/* иначе - "S"           */
                                 "S", 1 );
         }

        else

         return 5;                     /* если знак операции не  */
                                       /* "+" и не "-", то завер-*/
                                       /* шение  программы  по   */
                                       /* ошибке                 */
      }
                                       /* формируем:             */
     strcpy ( ASS_CARD._BUFCARD.OPERAND,   /* - первый операнд ассем-*/
                                 "RRAB," );/*блеровской операции;    */
     strcat ( ASS_CARD._BUFCARD.OPERAND,   /* - второй операнд ассем-*/
                    FORMT [IFORMT-1] );/*блеровской операции;    */
     ASS_CARD._BUFCARD.OPERAND [ strlen
          ( ASS_CARD._BUFCARD.OPERAND )] =/* - разделяющий пробел;  */
                                    ' ';
     memcpy ( ASS_CARD._BUFCARD.COMM,
     "Формирование промежуточного значения",/* - построчный коментарий*/
                                  36 );
     ZKARD ();                        /* запоминание ассембле-  */
                                      /* ровской операции       */

     return 0;                        /* успешное завершение    */
                                      /* пограммы               */
     }
     else
     return 3;                        /* если тип правого опе-  */
                                      /* ранда арифметического  */
                                      /* выражения не bin fixed,*/
                                      /* то завершение програм- */
                                      /* мы по ошибке           */
  }
 }
return 4;                             /* если правый операнд    */
```

```
                                /* арифметического выраже-*/
                                /*ния не определен в табл.*/
                                /* SYM, то завершить про- */
                                /* грамму по ошибке      */
  }

 }

/*.....*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала BUK на вто-*/
                                /* ром проходе.  Здесь   */
                                /* BUK -   "буква"       */

int BUK2 ()
 {
  return 0;
 }

/*.....*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала CIF на вто-*/
                                /* ром проходе.  Здесь   */
                                /* CIF -   "цифра"       */

int CIF2 ()
 {
  return 0;
 }

/*.....*/


                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала IDE на вто-*/
                                /* ром проходе.  Здесь   */
                                /* IDE -   "идентификатор"*/

int IDE2 ()
 {
  return 0;
 }


/*.....*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала IPE на вто-*/
                                /* ром проходе.  Здесь   */
                                /* IPE - "имя переменной" */

int IPE2 ()
 {
  return 0;
 }

/*.....*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала IPR на вто-*/
                                /* ром проходе.  Здесь   */
                                /* IPR -   "имя программы" */

int IPR2 ()
 {
  return 0;
```

```
}

/*.....................................................................*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала LIT на вто-*/
                                /* ром проходе.  Здесь   */
                                /* LIT -   "литерал"     */

int LIT2 ()
 {
  return 0;
 }

/*.....................................................................*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала MAN на вто-*/
                                /* ром проходе.  Здесь   */
                                /* MAN -   "мантисса"    */

int MAN2 ()
 {
  return 0;
 }

/*.....................................................................*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала ODC на вто-*/
                                /* ром проходе.  Здесь   */
                                /* ODC - "операт.ПЛ1- DCL"*/

int ODC2 ()
 {
  return 0;
 }

/*.....................................................................*/

                                /* п р о г р а м м а     */
                                /* семантич. вычисления   */
                                /* нетерминала OEN на вто-*/
                                /* ром проходе.  Здесь   */
                                /* OEN - "операт.ПЛ1-END" */

                                /* программа    формирует */
                                /* эпилог ассемблеровского*/
                                /* эквивалента ПЛ1-прог-  */
                                /* раммы                  */
int OEN2 ()
 {
  char RAB [20];
  char i = 0;
  FORM ();                       /* форматируем ПЛ1-опера- */
                                /* тор END               */

  memcpy ( ASS_CARD._BUFCARD.OPERAC, "BCR", 3 );  /* формируем код безуслов-*/
                                /*ного возврата управления*/
                                /* в вызывающую программу */

  memcpy ( ASS_CARD._BUFCARD.OPERAND,"15,14", 5 );/* операнды команды и     */

  memcpy ( ASS_CARD._BUFCARD.COMM,              /* поле построчного комен-*/
                "Выход из программы", 18 );/* тария                 */

  ZKARD ();                       /* запомнить опреацию     */
                                /* Ассемблера            */

                                /* далее идет блок форми- */
```

```
                                    /* рования декларативных  */
                                    /* псевдоопераций DC для  */
                                    /* каждого идентификатора,*/
                                    /* попавшего в табл.SYM   */
 for ( i = 0; i < ISYM; i++ )
  {                                 /* если строка табл.SYM   */
   if ( isalpha ( SYM [i].NAME [0] ) )         /* содержит идентификатор,*/
                                    /* т.е.начинается с буквы,*/
    {                               /* то:                    */
     if ( SYM [i].TYPE == 'B' )               /* если тип оператора bin */
                                    /* fixed, то:             */
      {
       strcpy ( ASS_CARD._BUFCARD.METKA,       /* пишем идентификатор в  */
                          SYM [i].NAME ); /* поле метки псевдоопера-*/
                                    /* ции DC                 */
       ASS_CARD._BUFCARD.METKA [ strlen
          ( ASS_CARD._BUFCARD.METKA ) ] = ' '; /* пишем разделитель полей*/

       memcpy ( ASS_CARD._BUFCARD.OPERAC,      /* пишем код псевдоопера- */
                               "DC", 2 ); /* ции DC               */

       if ( strcmp ( SYM [i].RAZR, "15" ) <= 0 ) /* формируем операнды псе-*/
                                    /* вдооперации DC         */
        strcpy ( ASS_CARD._BUFCARD.OPERAND,    /* для случая полуслова   */
                                  "H\'" );
       else                                /* или                    */

        strcpy ( ASS_CARD._BUFCARD.OPERAND,    /* для случая слова       */                                "F\'" );

 //Dos command
 //     strcat ( ASS_CARD._BUFCARD.OPERAND,      /* формируем цифровую     */
 //              ltoa ( VALUE (SYM [i].INIT),     /* часть операнда псевдо- */
 //                           &RAB [0], 10) ); /* операции,              */
 //let's do that in Unix!
       strcat(ASS_CARD._BUFCARD.OPERAND, gcvt(VALUE(SYM[i].INIT), 10, &RAB[0]));
       ASS_CARD._BUFCARD.OPERAND [ strlen       /* замыкающий апостроф    */
        ( ASS_CARD._BUFCARD.OPERAND ) ] = '\'';  /*         и              */

       memcpy ( ASS_CARD._BUFCARD.COMM,        /* поле построчного комен-*/
              "Определение переменной", 22 );  /* тария                  */

       ZKARD ();                          /* запомнить операцию     */
                                    /*    Ассемблера          */
      }
     }
    }
                                    /* далее идет блок декла- */
                                    /* ративных ассемблеровс- */
                                    /* ких EQU-операторов, оп-*/
                                    /* ределяющих базовый и   */
                                    /* рабочий регистры общего*/
                                    /* назначения             */

 memcpy ( ASS_CARD._BUFCARD.METKA, "RBASE", 5 ); /* формирование EQU-псев- */
 memcpy ( ASS_CARD._BUFCARD.OPERAC, "EQU",3 );   /* дооперации определения */
 memcpy ( ASS_CARD._BUFCARD.OPERAND, "15", 2 );  /* номера базового регист-*/
                                    /* ра общего назначения   */
                                    /*         и              */
 ZKARD ();                            /* запоминание ее         */

 memcpy ( ASS_CARD._BUFCARD.METKA, "RRAB", 4 );  /* формирование EQU-псев- */
 memcpy ( ASS_CARD._BUFCARD.OPERAC, "EQU",3 );   /* дооперации определения */
 memcpy ( ASS_CARD._BUFCARD.OPERAND, "5", 1 );   /* номера базового регист-*/
                                    /* ра общего назначения   */
                                    /*         и              */
 ZKARD ();                            /* запоминание ее         */

 memcpy ( ASS_CARD._BUFCARD.OPERAC, "END", 3 );  /* формирование кода ас-  */
                                    /* семблеровской псевдо-  */
                                    /* операции END,          */
 i = 0;

 while ( FORMT [1][i] != '\x0' )             /* ее операнда            */
```

```
 ASS_CARD._BUFCARD.OPERAND [i] = FORMT [1][i++];/*       и           */

 memcpy ( ASS_CARD._BUFCARD.COMM,             /* построчного коментария */
                  "Конец программы", 15 );

 ZKARD ();                              /* запоминание псевдоопе- */
                                       /* рации              */

 return 0;                              /* завершение программы   */
}

/*.....................................................................*/

                                       /* п р о г р а м м а     */
                                       /* семантич. вычисления   */
                                       /* нетерминала OPA на вто-*/
                                       /* ром проходе.  Здесь   */
                                       /* OPA - "операт.присваи- */
                                       /* вания арифметический   */

int OPA2 ()
 {
 int i;

 FORM ();                               /*форматируем ПЛ1-оператор*/
                                       /*присваивания арифметич. */

 for ( i = 0; i < ISYM; i++ )
 {                                  /* если идентификатор пра-*/
                                       /* вой части оператора оп-*/
  if ( !strcmp ( SYM [i].NAME, FORMT [0] )  &&  /* ределен ранее через    */
      strlen ( SYM [i].NAME ) ==             /* оператор DCL, то:     */
                  strlen ( FORMT [0] )
    )
    {
      if ( SYM [i].TYPE == 'B' )           /* если этот идентификатор*/
       {                              /* имеет тип bin fixed,то:*/

        if ( strcmp ( SYM [i].RAZR, "15" )    /* если bin fixed (15),то:*/
                                  <= 0 )
         memcpy ( ASS_CARD._BUFCARD.OPERAC,   /* сформировать команду   */
                        "STH", 3 );/* записи полуслова      */

        else                           /* иначе:               */
         memcpy ( ASS_CARD._BUFCARD.OPERAC,   /* команду записи слова   */
                         "ST", 2 );

        strcpy ( ASS_CARD._BUFCARD.OPERAND,   /*      доформировать    */
                         "RRAB," );/*        операнды     */

        strcat ( ASS_CARD._BUFCARD.OPERAND,   /*         команды     */
                        FORMT [0]) ;

        ASS_CARD._BUFCARD.OPERAND [ strlen    /*           и       */
        ( ASS_CARD._BUFCARD.OPERAND ) ] = ' ';

        memcpy ( ASS_CARD._BUFCARD.COMM,     /* построчный коментарий  */
        "Формирование значения арифм.выражения",
                        37 );
        ZKARD ();                      /* запомнить операцию     */
                                       /* Ассемблера  и        */
        return 0;                      /* завершить программу    */
       }

       else                           /* если идентификатор не  */
                                       /* имеет тип bin fixed,то:*/
        return 3;                      /* завершение с диагности-*/
                                       /* кой ошибки           */
    }
 }
 return 4;                              /* если идентификатор ра- */
                                       /* нее не определен через */
                                       /* ПЛ1-оператор DCL,то за-*/
```

```
                                    /* вершение с диагностикой*/
                                    /* ошибки                 */

 }

/*.....................................................................*/

                                    /* п р о г р а м м а      */
                                    /* семантич. вычисления   */
                                    /* нетерминала OPR на вто-*/
                                    /* ром проходе.   Здесь   */
                                    /* OPR - "операт.ПЛ1-PROC"*/

                                    /* программа    формирует */
                                    /* пролог ассемблеровского*/
                                    /* эквивалента  исходной  */
                                    /* ПЛ1-программы          */
int OPR2 ()
 {
  char i = 0;
  FORM ();                           /* форматируем оператор   */
                                    /* ПЛ1 - "начало процедур-*/
                                    /* ного блока"            */
  while ( FORMT [0][i] != '\x0' )
   ASS_CARD._BUFCARD.METKA [i++] = FORMT [0][i];  /* нулевой терм используем*/
                                    /* как метку в START-псев-*/
                                    /* дооперации Ассемблера  */

  memcpy ( ASS_CARD._BUFCARD.OPERAC, "START", 5 );/* достраиваем код и опе- */
  memcpy ( ASS_CARD._BUFCARD.OPERAND, "0", 1 );   /* ранды  в  START-псевдо-*/
  memcpy ( ASS_CARD._BUFCARD.COMM,              /* операции Ассемблера    */
             "Начало программы", 16 );
  ZKARD ();                          /* запоминаем карту Ассем-*/
                                    /* блера                  */

  memcpy ( ASS_CARD._BUFCARD.OPERAC, "BALR", 4 ); /* формируем BALR-операцию*/
  memcpy ( ASS_CARD._BUFCARD.OPERAND,           /* Ассемблера             */
                    "RBASE,0", 7 );
  memcpy ( ASS_CARD._BUFCARD.COMM,
           "Загрузить регистр базы", 22 );
  ZKARD ();                          /* и запоминаем ее        */

  memcpy ( ASS_CARD._BUFCARD.OPERAC, "USING", 5 );/* формируем USING-псевдо-*/
  memcpy ( ASS_CARD._BUFCARD.OPERAND,           /* операцию Ассемблера    */
                     "*,RBASE", 7 );
  memcpy ( ASS_CARD._BUFCARD.COMM,
           "Назначить регистр базой", 23 );
  ZKARD ();                          /* и запоминаем ее        */

  return 0;                          /* завершить подпрограмму */
 }

/*.....................................................................*/

                                    /* п р о г р а м м а      */
                                    /* семантич. вычисления   */
                                    /* нетерминала PRO на вто-*/
                                    /* ром проходе.   Здесь   */
                                    /* PRO - "программа"      */

int PRO2 ()                          /*прогр.формирует выходной*/
 {                                 /*файл                   */

  FILE *fp;                          /*набор                  */
                                    /*рабочих                */
                                    /*переменных             */

  strcat ( NFIL , "ass" );                /*сформировать имя выход- */
                                    /*ного файла             */

  if ( (fp = fopen ( NFIL , "wb" )) == NULL )    /*при неудачн.открыт.ф-ла */
   return (7);                       /* сообщение об ошибке    */
```

```
 else                                /*иначе:              */
  fwrite (ASSTXT, 80 , IASSTXT , fp);          /* формируем тело об.файла*/
 fclose ( fp );                       /*закрываем об'ектный файл*/
 return ( 0 );                        /*завершить полдпрограмму */
}

/*.....................................................................*/

                                     /* п р о г р а м м а     */
                                     /* семантич. вычисления   */
                                     /* нетерминала RZR на вто-*/
                                     /* ром проходе.   Здесь   */
                                     /* RZR - "разрядность"    */

int RZR2 ()
 {
  return 0;
 }

/*.....................................................................*/

                                     /* п р о г р а м м а     */
                                     /* семантич. вычисления   */
                                     /* нетерминала TEL на вто-*/
                                     /* ром проходе.   Здесь   */
                                     /* TEL - "тело программы" */

int TEL2 ()
 {
  return 0;
 }

/*.....................................................................*/

                                     /* п р о г р а м м а     */
                                     /* семантич. вычисления   */
                                     /* нетерминала ZNK на вто-*/
                                     /* ром проходе.   Здесь   */
                                     /* ZNK - "знак операции"  */

int ZNK2 ()
 {
  return 0;
 }

/*.....................................................................*/

                                     /*  п р о г р а м м а    */
                                     /* управления абстрактной */
                                     /* ЭВМ  -  семантического */
                                     /* вычислителя, интерпре- */
                                     /* тирующего абстрактную  */
                                     /* программу, сформирован-*/
                                     /* ную синтаксическим ана-*/
                                     /* лизатором в стеке дос- */
                                     /* тигнутых целей.        */

                                     /* Суть алгоритма управле-*/
                                     /*ния  в  последовательной*/
int gen_COD ()                            /*интерпретации строк сте-*/
 {                                  /*ка достижений  в направ-*/
  int NOSH;                              /*лении от дна к вершине. */

  int (* FUN [NNETRM][2]) () =              /*При этом каждая строка  */
   {                                 /*воспринимается как кома-*/
    {/*   1  */   AVI1, AVI2 },           /*нда абстрактной ЭВМ со  */
    {/*   2  */   BUK1, BUK2 },             /*следующими полями:     */
    {/*   3  */   CIF1, CIF2 },
    {/*   4  */   IDE1, IDE2 },           /* - DST.DST1 - код опера-*/
    {/*   5  */   IPE1, IPE2 },           /*ции;                */
    {/*   6  */   IPR1, IPR2 },
    {/*   7  */   LIT1, LIT2 },           /* - DST.DST2 - левая гра-*/
    {/*   8  */   MAN1, MAN2 },             /*ница интерпретируемого  */
```

```
   {/*   9  */   ODC1, ODC2 },         /*фрагмента исх.текста;   */
   {/*   10  */   OEN1, OEN2 },
   {/*   11  */   OPA1, OPA2 },         /* - DST.DST4 -правая гра-*/
   {/*   12  */   OPR1, OPR2 },         /*ница интерпретируемого  */
   {/*   13  */   PRO1, PRO2 },         /*фрагмента исх.текста.   */
   {/*   14  */   RZR1, RZR2 },
   {/*   15  */   TEL1, TEL2 },
   {/*   16  */   ZNK1, ZNK2 }
  };

  for ( I2 = 0; I2 < L; I2++ )                /* организация первого    */
   if ( ( NOSH = FUN [                        /* прохода семантического */
                 numb ( DST [I2].DST1, 3 )   /* вычисления             */
                ][0] ()
       ) != 0
     )
    return (NOSH);                           /* выход из программы     */
                                             /* по ошибке              */

  for ( I2 = 0; I2 < L; I2++ )                /* организация второго    */
   if ( ( NOSH = FUN [                        /* прохода семантического */
                 numb ( DST [I2].DST1, 3 )   /* вычисления             */
                ][1] ()
       ) != 0
     )
    return (NOSH);                           /* выход из программы     */
                                             /* по ошибке              */

  return 0;                                   /* успешное завершение    */
                                             /* программы              */
 }

/*.....................................................................*/

                                             /*  п р о г р а м м а,   */
                                             /* организующая последова-*/
                                             /* тельную обработку ис-  */
                                             /* ходного текста:        */
                                             /* - лексич.анализатором; */
                                             /* - синтаксич.анализат.; */
                                             /* - семантич.вычислителем*/
int main (int argc, char **argv )
 {                                           /* рабочие переменные:    */
  FILE *fp;                                   /* - указатель на файл;   */
  char *ptr=argv[1];                          /* - указатель на первый  */
                                             /*параметр командной стр. */

  strcpy ( NFIL, ptr );                       /*изъять имя транслируемой*/
                                             /*программы из командной  */
                                             /*строки в рабочее поле   */

                                             /*проверяем корректность  */
                                             /*командной строки        */
  if ( argc != 2 )

   {                                         /* по ошибке в командн.стр*/
    printf ("%s\n", "Ошибка в командной строке"); /* выдать диагностику и   */
    return;                                  /* завершить трансляцию   */
   }

                                             /* проверка типа исх.файла*/
  if
   (
    strcmp ( &NFIL [ strlen ( NFIL )-3 ], "pli" ) /* если тип не "pli", то: */
   )

   {
    printf ( "%s\n",                         /* выдать диагностику и   */
     "Неверный тип файла с исходным текстом" );
    return;                                  /* завершить трансляцию   */
   }
```

```
 else                                  /* если тип файла "pli",то*/

 {                                    /*пытаемся открыть файл и */
  if ( (fp = fopen ( NFIL , "rb" )) == NULL )   /*при неудачн.открыт.ф-ла */
                                       /* сообщение об ошибке и  */
   {
    printf ( "%s\n",
     "Не найден файл с исходным текстом" );
    return;                            /* завершение трансляции  */
   }

  else                                /* иначе:               */
                                       /* пишем файл в массив    */
                                       /*  ISXTXT              */
   {
    for ( NISXTXT = 0; NISXTXT <= MAXNISXTXT; NISXTXT++ )

     {
      if ( !fread ( ISXTXT [NISXTXT], 80, 1, fp ) )
       {
        if ( feof ( fp ) )             /* в конце файла идем на  */
         goto main1;                   /* метку  main1          */

        else                           /* при сбое чтения       */
         {                            /* выдаем диагностику     */
          printf ( "%s\n",
           "Ошибка при чтении фыйла с исх.текстом" );
          return;                      /* и завершаем трансляцию */
         }
       }
     }

    printf ( "%s\n",                   /*при пеерполнении массива*/
     "Переполнение буфера чтения исх.текста" ); /* ISXTXT выдать диагн.   */
    return;                            /* и завершить трансляцию */
   }

 }

main1:                                 /* по завершении чтения   */
                                       /* исх.файла формируем    */
  fclose ( fp );                       /* префикс имени выходного*/
  NFIL [ strlen ( NFIL )-3 ] = '\x0';          /* Ассемблеровского файла */

 memset ( ASS_CARD.BUFCARD, ' ', 80 );         /* чистка буфера строки   */
                                       /* выходного ассемблеров- */
                                       /* ского файла          */

 compress_ISXTXT ();                        /* лексический анализ     */
                                       /* исходного текста      */

 build_TPR ();                          /* построение матрицы     */
                                       /* преемников           */

 if ( (sint_ANAL ()) )                   /* синтаксический анализ  */
 {                                    /* исходного текста      */
  STROKA [I4 +20] = '\x0';
  printf                               /* если найдены ошибки    */
   (                                  /* синтаксиса, то :      */
    "%s%s%s%s\n",
    "ошибка синтаксиса в исх.тексте -> ",     /* выдаем диагностику и   */
    "\"...",&STROKA [I4], "...\""
   );
  printf
   (
    "%s\n", "трансляция прервана"
   );
  return;                              /* завершаем трансляцию   */
 }
 else                                 /* иначе делаем          */
 {
  switch ( gen_COD () )                    /* семантическое вычислен.*/
   {
```

```
   case  0:                          /*если код завершения = 0,*/
                                     /* то:                 */

    printf ( "%s\n",                 /* - диагностич.сообщение;*/
      "трансляция завершена успешно" );
    return;                          /* - завершить трансляцию */


   case  1:                          /*если код завершения = 1,*/
                                     /* то:                 */
    printf ( "%s\n",                 /* - диагностич.сообщение;*/
      "несовпадение имени процедуры в прологе-эпилоге" );
    break;                           /* - выйти на обобщающую  *//* - диагностич.сообщение;*/
                                     /*диагностику          */

   case  2:                          /*если код завершения = 2,*/
                                     /* то:                 */
    STROKA [ DST [I2].DST2 +20 ] = '\x0';     /* - диагностич.сообщение;*/
    printf ( "%s%s\n%s%s%s\n",
      "недопустимый тип идентификатора: ",
       &FORMT [1], " в исх.тексте -> \"...",
        &STROKA [ DST [I2].DST2 ], "...\"" );
    break;                           /* - выйти на обобщающую  */
                                     /*диагностику          */

   case  3:                          /*если код завершения = 3,*/
                                     /* то:                 */
    STROKA [ DST [I2].DST2 + 20 ] = '\x0';    /* - диагностич.сообщение;*/
    printf ( "%s%s\n%s%s%s\n",
      "недопустимый тип идентификатора: ",
        &FORMT [IFORMT-1], " в исх.тексте -> \"...",
         &STROKA [ DST [I2].DST2 ], "...\"" );
    break;                           /* -выйти на обобщающую   */
                                     /*диагностику          */

   case  4:                          /*если код завершения = 4,*/
                                     /* то:                 */
    STROKA [ DST [I2].DST2 + 20 ] = '\x0';    /* - диагностич.сообщение;*/
    printf ( "%s%s\n%s%s%s\n",
      "неопределенный идентификатор: ",
       &FORMT [IFORMT-1], " в исх.тексте -> \"...",
        &STROKA [ DST [I2].DST2 ], "...\"" );
    break;                           /* - выйти на обобщающую  */
                                     /*диагностику          */

   case  5:                          /*если код завершения = 5,*/
                                     /* то:                 */
    STROKA [ DST [I2].DST2 + 20 ] = '\x0';    /* - диагностич.сообщение;*/
    printf ( "%s%c\n%s%s%s\n",
      "недопустимая операция: ",
       STROKA [ DST [I2].DST4 - strlen ( FORMT [IFORMT-1] ) ],
       " в исх.тексте -> \"...", &STROKA [ DST [I2].DST2 ], "...\"");
    break;                           /* - выйти на обобщающую  */
                                     /*диагностику          */

   case  6:                          /*если код завершения = 6 */
                                     /* то:                 */
    STROKA [ DST [I2].DST2 + 20 ] = '\x0';    /* - диагностич.сообщение;*/
    printf ( "%s%s\n%s%s%s\n",
      "повторное объявление идентификатора: ",
       &FORMT [1], " в исх.тексте -> \"...",
        &STROKA [ DST [I2].DST2 ], "...\"" );
    break;                           /* - выйти на обобщающую  */
                                     /*диагностику          */

  }

 }

printf ( "%s\n", "трансляция прервана" );      /* обобщающая диагностика */

return 0;
}
```

```
/*.....*/
```