4. EXEMPLO – ENTRADA PARA O SIC

```
%%COMPILER PROGRAM exemplo;   %%BATCH


%%TOKENS

    "id"            = xid ;
    "cte"           = xcte ;
    "program "      = xprogram
    "end"           = xend
    "begin"         = xbegin
    "integer"       = xinteger
    "procedure"     = xprocedure
    "if"            = xif
    "then"          = xthen
    "else"          = xelse
    "while"         = xwhile
    "do"            = xdo
    ";"             = xpv
    ":"             = xdp
    ":="            = xatr
    "("             = xap
    ")"             = xfp
    "+"             = xmais
    "-"             = xmenos
    "eof"           = YYEOF
    "*"             = xvezes
    "/"             = xdiv
    "**"            = xpot
    "="             = xigual
    "error"         = xerror
    "boolean"       = xboolean


%%STACK 50  OF ATTRIBUTES


    exp       = (r,tipo:int)  { exp.r = 0; exp.tipo = INTEIRO; };
    "id"      = (valor :int)  { "id".valor = 1; }
    "cte"     = (valor :int)
    cond      = (quad  :int)
    n         = (quad  :int)
    marca     = (quad  :int)
    prochead  = (valor :int; inicio : int)
```

```
%%SCOPEMAP

    "begin" : "end" ;
    "("     : ")"   ;

%%NTMAP

    exp , dcl , cmd ;



(*=================================================================*)
(*                        MODULO DE TRATAMENTO DE ERRO             *)
(*=================================================================*)

%%CONSTANTS

        #define MAXMSG 20
        #define MAXERROR 11

%%TYPES

        typedef char erros[MAXMSG+1];
        typedef struct { erros msg; } mensg;

%%VARIABLES

        FILE *out;

%%PROCEDURES

    void ERRO(int n,int l,int p)
        {
                mensg message;

                fseek(out,sizeof(mensg)*n,SEEK_SET);
                fread(&message,sizeof(mensg),1,out);
                fprintf(YYSAIDA,"\n");
                fprintf(YYSAIDA,"%5c%s%s%d%s%d\n",'+',message.msg,"na linha", l,
                        "posicao", p);
        } /* ERRO */
```

```
/*=============================================================*/
/*                    MODULO TABELA DE SIMBOLOS                */
/*=============================================================*/


%%CONSTANTS

        /* tipo tabela de simbolos */

        #define INTEIRO 1
        #define LOGICO 2

        /*  classe tabela de simbolos */

        #define NAO_DECL 0
        #define VARIAVEL 1
        #define PAR 3
        #define PROC 4
        #define NMAX 29
        #define MAX 1000
        #define ALPHA1 9
        #define NPC 10
        #define IDTAM 15        /* no. caracteres significativos ident */
                                /* a posicao 0 de ident nao contem nada.*/

%%TYPES

        typedef char alfa[IDTAM+1];
        typedef char alfa1[ALPHA1+1];

        typedef struct {    alfa nome;
                            int classe;
                            int tipotam;
                            int endoff;
                            int nivel;
                            int col;
                       } tscampos;

%%VARIABLES

        /* tabela de simbolos */

        int nivel,l;
        tscampos ts[MAX+1];
        int thash[91];
        int escopo[NMAX+1];
```

```
%%PROCEDURES

        int HASH(alfa simb)
        {
         int i,h;

         h = 0;
         i = 1;
         while ( (simb[i] != ' ') && (i < IDTAM))
         {
                 h = h + simb[i];
                 i = i + 1;
         }
         return h % 91;
        }  /*HASH*/


        int INSTALA(alfa simb)
        {
         int n,k;

         n = HASH(simb);
         k = thash[n];
         while (k >= escopo[nivel])
         {
                 if (strcmp(simb,ts[k].nome)==0)
                 {
                  ERRO(1,YYLINHA,YYPOS);
                  return k;
                 }
                 else k = ts[k].col;
         }
         if (l == MAX + 1)
         {
                 ERRO(2,YYLINHA,YYPOS);              /* estouro da TS */
                 return l;
         }
         strcpy(ts[l].nome,simb);
         ts[l].nivel = nivel;
         ts[l].classe = NAO_DECL;
         ts[l].col = thash[n];
         thash[n] = l;
         l++;
         return l-1;
        } /* INSTALA*/
```

```
int PROCURAID(alfa simb)
{
  int n,k;

  n = HASH(simb);
  k = thash[n];
  while (k != 0)
  {
        if (strcmp(simb,ts[k].nome) == 0) return k;
        k = ts[k].col;
  }
  return INSTALA(simb);
} /* PROCURAID */

void ABLOCO(void)
{
  nivel++;
  if (nivel > NMAX)
        ERRO(3,YYLINHA,YYPOS);  /* estouro limite de niveis */
  else escopo[nivel] = l;
} /*ABLOCO*/

void FBLOCO(void)
{
  int s,b,k;

  s = l;
  b = escopo[nivel];
  while (s > b)               /* desfaz hash */
  {
        s--;
        k = HASH(ts[s].nome);
        thash[k] = ts[s].col;
  }
  nivel--;
} /*FBLOCO*/

void TAMANHO(int proced,int addr)
{
  ts[proced].endoff = addr;
} /* TAMANHO */

void DECLARA(int k,int class,int tipot,int offset)
{
  ts[k].classe = class;
  ts[k].tipotam = tipot;
  ts[k].endoff = offset;
} /* DECLARA */
```

```
        int FOIDECLARADO(int k)
        {
         if (ts[k].classe != NAO_DECL)
                return 1;               /* FOIDECLARADO = true  */
         else return 0;                 /* FOIDECLARADO = false */
        } /*FOIDECLARADO*/


/*=================================================================*/
/*                          MODULO   OFFSET                        */
/*=================================================================*/


%%VARIABLES

        int poff[8];
        int toff;
        int offset;

%%PROCEDURES

        void POPOFF(int offst)
        {
                if (toff < 1)
                  ERRO(4,YYLINHA,YYPOS);
                else {
                        offst = poff[toff];
                        toff = toff-1;
                      }
        } /* POPOFF */

        void PUSHOFF(int offst)
        {
         if (toff > 7)
                 ERRO(5,YYLINHA,YYPOS);
         else {
                        toff = toff +1;
                        poff[toff] = offst;
                        offst = 1;
                      }
        } /* PUSHOFF */

        int TEMP(void)
        {
         temps = temps - 1;
         return temps + 1;
        } /* TEMP */
```

```
/*=================================================================*/
/*                    MODULO   DE ANALISE LEXICA                   */
/*=================================================================*/


%%CONSTANTS

        /**** GETCHAR ****/

        #define LINHAMAX 66
        #define SPACELINE 1

        /***** YYSCAN ******/

        #define ENDFILE EOF
        #define NEWLINE 13
        #define HTAB 9
        #define LINEFEED 10
        #define CTLZ 26
        #define BACKSPACE 8
        #define VT 11
        #define FF 12
        #define TLE 72

%%VARIABLES

        int errofatal;  /* Variavel para indicar um erro fatal */
        FILE *FONTE;

        /*YYSCAN*/

        int ii;
        int ctx;
        alfa key[12];
        int tipo_pal_res[12];

        /***** GETCHAR ******/

        char ch;
        int ll,cc;
        char linha[TLE + 1];
        int pagina;
        int linenum;
        int pagcomp;
        int marginf;
        int margem;
        int margsup1;
```

```
        int margsup2;
        int margsup;

%%PROCEDURES

        void MONTATABELAS(void)
        {
          /* tabela de palavras reservadas */
          strcpy(key[1]," begin         ");
          strcpy(key[2]," boolean       ");
          strcpy(key[3]," do            ");
          strcpy(key[4]," else          ");
          strcpy(key[5]," end           ");
          strcpy(key[6]," if            ");
          strcpy(key[7]," integer       ");
          strcpy(key[8]," procedure     ");
          strcpy(key[9]," program       ");
          strcpy(key[10]," then         ");
          strcpy(key[11]," while        ");

          tipo_pal_res[1] = xbegin;
          tipo_pal_res[2] = xboolean;
          tipo_pal_res[3] = xdo;
          tipo_pal_res[4] = xelse;
          tipo_pal_res[5] = xend;
          tipo_pal_res[6] = xif;
          tipo_pal_res[7] = xinteger;
          tipo_pal_res[8] = xprocedure;
          tipo_pal_res[9] = xprogram;
          tipo_pal_res[10] = xthen;
          tipo_pal_res[11] = xwhile;
        } /* MONTATABELAS */

        void INICIA(void)
        {
          ch = ' ';
          ll = 0;  cc = 0;
          pagina  = 0;
          linenum = 0;
          pagcomp = LINHAMAX;
          margem  = 4;
          margsup1 = 2;
          margsup2 = 2;
          margsup = margsup1 + margsup2 + 3;
          marginf = pagcomp - margem;
          ctx     = 1;
          l       = 2;
```

```
  strcpy(ts[1].nome,"               ");
  ts[1].nivel   = 1;
  ts[1].classe  = NAO_DECL;
  ts[1].tipotam = INTEIRO;
  ts[1].endoff  = 0;
  for (ii = 1; ii <= 1000; ii++)
          ts[ii].col = 0;
  for (ii = 0; ii <= 90;ii++)
          thash[ii] = 0;
  MONTATABELAS();
  temps = -1;
  nivel = 0;
  ABLOCO();
} /* INICIA */

void WRITEFONTE(void)
{
  int i;

  if (linenum > marginf)
  {
        fprintf(YYSAIDA,"%4s"," ");
        for (i = 1; i <= 15; i++)
                fprintf(YYSAIDA,"----+");
        for (i = 1; i <= margem-1; i++)
                fprintf(YYSAIDA,"\n");
        linenum = 0;
  }
  if (linenum == 0)
  {             /* comeca nova pagina */
        pagina++;
        for (i = 1; i <= margsup1; i++)
                fprintf(YYSAIDA,"\n");
        fprintf(YYSAIDA,"%4s"," ");
        for (i = 1; i <= 15; i++)
                fprintf(YYSAIDA,"----+");
        fprintf(YYSAIDA,"\n");
        fprintf(YYSAIDA,"%s%8s%s%40s%s%3d\n","LINHA"," ","TEXTO", " ",
                "PAGINA ",pagina);
        for (i = 1; i <= margsup2; i++)
                fprintf(YYSAIDA,"\n");
        linenum = margsup + 3;
        fprintf(YYSAIDA,"%1d%3s",YYLINHA," ");
        for (i = 1; i <= ll-1; i ++)
                fprintf(YYSAIDA,"%c",linha[i]);
        linenum = linenum + SPACELINE;
        for (i = 1; i<= SPACELINE; i++)
```

```
                fprintf(YYSAIDA,"\n");
  }
  else {
                fprintf(YYSAIDA,"%1d%3s",YYLINHA," ");
                for (i = 1; i <= ll-1; i ++)
                        fprintf(YYSAIDA,"%c",linha[i]);
                linenum = linenum + SPACELINE;
                for (i = 1; i<= SPACELINE; i++)
                        fprintf(YYSAIDA,"\n");
               }
}  /* WRITEFONTE */

void GETCHAR(char *c)
{
  if (cc == ll)
  {
        if (feof(FONTE))
        {
          *c = ENDFILE;
          return;
        }
        ll = 1;
        cc = 0;
        fscanf(FONTE,"%c",&linha[ll]);
        while (linha[ll] != '\n')
        {
          ll++;
          fscanf(FONTE,"%c",&linha[ll]);
        }
        ll++;
        linha[ll] = NEWLINE;
        fscanf(FONTE,"\n");
        YYLINHA++;

        /* impressao do texto de entrada */
        WRITEFONTE();
  }
  cc++;
  *c = linha[cc];
} /*GETCHAR*/

void YYSCAN(void)
{
  int inum,i,j,k,kk;
  int tam;
  alfa ident;                   /* a posicao 0 de ident contem branco. */
  int PermiteLoop;              /* permite primeira entrada no loop while*/
```

```
  for (k = 0; k <= IDTAM; k++)
          ident[k] = ' ';
  ident[IDTAM] = '\0';
UM:while ((ch == ' ')  || (ch == HTAB) || (ch == NEWLINE) ||
                  (ch == CTLZ) || (ch == BACKSPACE) || (ch == VT) ||
                  (ch == FF) || (ch == LINEFEED))
                 GETCHAR(&ch);
  YYPOS = cc;
  if (ch == ENDFILE)
  {
        YYSIMB = YYEOF;
        goto DOIS;
  }
  else if (ch == '{')
                {
                  GETCHAR(&ch);
                  while (ch != '}')
                                GETCHAR(&ch);
                  GETCHAR(&ch);
                  goto UM;
                }
                else if (isalpha(ch))       /* palavra */
                        {
                                tam = 0;
                                PermiteLoop = 1;
                                while ((PermiteLoop) || (isalnum(ch)))
                                {
                                  PermiteLoop = 0;
                                  if (tam < IDTAM)
                                  {
                                  /* converte carater lido para minuscula */
                                        if (isupper(ch))
                                                ch = ch + 32;
                                        tam++;
                                        ident[tam] = ch;
                                  }
                                  GETCHAR(&ch);
                                }
                                /* procura por palavra chave */
                                i = 1;
                                j = NPC+1;
                                PermiteLoop = 1;
                                while ((PermiteLoop) || (i<=j))
                                {
                                  PermiteLoop = 0;
                                  k = (i+j) / 2;
                                  if (strcmp(ident,key[k]) <= 0)
```

```
                j = k-1;
            if (strcmp(ident,key[k]) >= 0)
                i = k+1;
        }
        if (i-1>j)
          YYSIMB = tipo_pal_res[k];
        else {
                YYSIMB = xid;
                if (ctx == 1)
                  "id".valor = INSTALA(ident);
                else "id".valor = PROCURAID(ident);
              }
}
else if (isdigit(ch))    /* numero */
        {
            inum = 0;
            YYSIMB = xcte;
            PermiteLoop = 1;
            while ((PermiteLoop) || (isdigit(ch)))
            {
             PermiteLoop = 0;
             kk = ch - '0';
             if ((inum > 3276) || ((inum == 3276) && (kk >
                    7)))
                 ERRO(6,YYLINHA,YYPOS);
             else {
                        inum = inum * 10 + kk;
                        GETCHAR(&ch);
                  }
            }
            "cte".valor = inum;
        }
        else switch (ch)
             {
                case ':':{      GETCHAR(&ch);
                                if (ch == '=')
                                {
                                YYSIMB = xatr;
                                GETCHAR(&ch);
                                }
                                else YYSIMB = xdp;
                                break;
                           }
                case ';': {     GETCHAR(&ch);
                                YYSIMB = xpv;
                                break;
                          }
```

```
                case '(': {     GETCHAR(&ch);
                                YYSIMB = xap;
                                break;
                          }
                case ')': {     GETCHAR(&ch);
                                YYSIMB = xfp;
                                break;
                          }
                case '+': {     GETCHAR(&ch);
                                YYSIMB = xmais;
                                break;
                           }
                case '-': {     GETCHAR(&ch);
                                YYSIMB = xmenos;
                                break;
                          }
                case '*': {     GETCHAR(&ch);
                                if (ch == '*')
                                {
                                YYSIMB = xpot;
                                GETCHAR(&ch);
                                }
                                else YYSIMB = xvezes;
                                break;
                          }
                case '/': {     GETCHAR(&ch);
                                YYSIMB = xdiv;
                                break;
                          }
                case '=': {     GETCHAR(&ch);
                                YYSIMB = xigual;
                                break;
                           }
                default : {     fprintf(YYSAIDA,
                                "CHAR=%d%c", ch,ch);
                                YYSIMB = xerror;  /* 40 */
                                GETCHAR(&ch);
                                fprintf(YYSAIDA,
                                "char=%1d%1c",ch,ch);
                                break;
                          }
            } /* switch */
DOIS:;} /*YYSCAN*/
```

```
/*=================================================================*/
/*                    MODULO CODIGO INTERMEDIARIO                  */
/*=================================================================*/


%%CONSTANTS

        /* operadores para opquad */

        #define CPROCEND   1
        #define CPROCBEGIN 2
        #define CATRIB     3
        #define CPARAM     4
        #define CCALL      5
        #define CIGUAL     6
        #define CGOTO      7
        #define CMAIS      8
        #define COU        13
        #define CVEZES     9
        #define CE         14
        #define CMENOS     10
        #define CINVERTE   15
        #define CNEGA      16
        #define CDIV       11
        #define CPOT       12
        #define CDSVF      17
        #define CPROGEND   18
        #define CPROGBEGIN 19

%%TYPES

        /* gen */

        typedef struct {
                          int opquad,opn1quad,opn2quad,opn3quad;
                        } tquad;

%%VARIABLES

        /* gen */
        int op;
        tquad quad[501];
        FILE *CODE;
        int proxq;
        alfa opnome[20];
```

```
        int temps;
%%PROCEDURES

        void GEN(int operador,int opn1,int opn2,int opn3)
        {
         quad[proxq].opquad   = operador;
         quad[proxq].opn1quad = opn1;
         quad[proxq].opn2quad = opn2;
         quad[proxq].opn3quad = opn3;
         proxq++;
        } /* GEN */

        void CONSERTA(int cond,int addr)
        {
         quad[cond].opn2quad = addr;
        } /* CONSERTA */

        void SALVA(int i,int k)
        {
         int j;
         for (j = i; j <= k; j++)
                 fwrite(&quad[j],sizeof(tquad),1,CODE);
        } /* SALVA */

        void MONTAOPNOME(void)
        {
         strcpy(opnome[1 ]," PROCEND      ");
         strcpy(opnome[2 ]," PRCBEGIN     ");
         strcpy(opnome[3 ]," :=           ");
         strcpy(opnome[4 ]," PARAM        ");
         strcpy(opnome[5 ]," CALL         ");
         strcpy(opnome[6 ]," =            ");
         strcpy(opnome[7 ]," GOTO         ");
         strcpy(opnome[8 ]," +            ");
         strcpy(opnome[9 ]," *            ");
         strcpy(opnome[10]," -            ");
         strcpy(opnome[11]," /            ");
         strcpy(opnome[12]," **           ");
         strcpy(opnome[13]," OU           ");
         strcpy(opnome[14]," E            ");
         strcpy(opnome[15]," -            ");
         strcpy(opnome[16]," NEGA         ");
         strcpy(opnome[17]," DSVF         ");
         strcpy(opnome[18]," PROGEND      ");
         strcpy(opnome[19]," PRGBEGIN     ");
        } /* MONTAOPNOME */
```

```
void IMPRIMECODIGO(void)
{
 int i,j;
 tquad quad;

 MONTAOPNOME();
 fprintf(YYSAIDA,"%4s%s\n"," ",
              "==========CODIGO GERADO==========");
 fprintf(YYSAIDA,"\n");
 fprintf(YYSAIDA,"%6s%s%2s%s%s%s"," ","opquad"," ","opn1quad ",
              "opn2quad ","opn3quad","\n");
 j = 1;
 while (!feof(CODE))
 {
        fread(&quad,sizeof(tquad),1,CODE);
        fprintf(YYSAIDA,"\n");
        if (!feof(CODE))
        {
        fprintf(YYSAIDA,"%4d%2s%s",j," ",opnome[quad.opquad]);
        switch(quad.opquad)
        {
              case CPROCEND    : { break; }

              case CPROCBEGIN  : {    fprintf(YYSAIDA,"%4s%d"," ",
                                      quad.opn1quad);
                                       break;
                                 }

              case CATRIB      : { fprintf(YYSAIDA,"%4s%d%6s%d"," ",
                                   quad.opn1quad," ",quad.opn2quad);
                                   break;
                                 }

              case CPARAM      : {fprintf(YYSAIDA,"%4s%d"," ",quad.opn1quad);
                                   break;
                                   }

              case CCALL       : { fprintf(YYSAIDA,"%4s%d"," ",quad.opn1quad);
                                   break;
                                 }

              case CGOTO       : { fprintf(YYSAIDA,"%12s%d"," ",quad.opn2quad);
                                    break;
                                 }
```

```
case CMAIS      : { fprintf(YYSAIDA,"%4s%d%6s%d%8s%d"," ",
                        quad.opn1quad," ",quad.opn2quad," ",
                        quad.opn3quad);
                    break;
                  }

case COU        : {   fprintf(YYSAIDA,"%4s%d%6s%d%8s%d"," ",
                        quad.opn1quad," ", quad.opn2quad," ",
                        quad.opn3quad);
                    break;
                  }

case CIGUAL     : {fprintf(YYSAIDA,"%4s%d%6s%d%8s%d"," ",
                        quad.opn1quad," ",quad.opn2quad," ",
                        quad.opn3quad);
                    break;
                  }

case CDIV       : {   fprintf(YYSAIDA,"%4s%d%6s%d%8s%d"," ",
                        quad.opn1quad," ",quad.opn2quad," ",
                        quad.opn3quad);
                    break;
                  }

case CVEZES     : {fprintf(YYSAIDA,"%4s%d%6s%d%8s%d"," ",
                    quad.opn1quad," ",quad.opn2quad," ",
                    quad.opn3quad);
                    break;
                  }

case CE         : {     fprintf(YYSAIDA,"%4s%d%6s%d%8s%d"," ",
                        quad.opn1quad," ",
                    quad.opn2quad," ",quad.opn3quad);
                    break;
                  }

case CPOT       : {   fprintf(YYSAIDA,"%4s%d%6s%d%8s%d"," ",
                    quad.opn1quad," ",
                    quad.opn2quad," ",quad.opn3quad);
                    break;
                  }

case CMENOS     : {fprintf(YYSAIDA,"%4s%d%6s%d%8s%d"," ",
                        quad.opn1quad," ",
                    quad.opn2quad," ",quad.opn3quad);
                    break;
                   }
```

```
                case CINVERTE   : {     fprintf(YYSAIDA,"%4s%d%6s%d"," ",
                                        quad.opn1quad," ",quad.opn2quad);
                                        break;
                                  }

                case CNEGA      : { fprintf(YYSAIDA,"%4s%d%6s%d"," ",
                                    quad.opn1quad," ",quad.opn2quad);
                                    break;
                                  }

                case CDSVF      : { fprintf(YYSAIDA,"%4s%d%6s%d"," ",
                                    quad.opn1quad," ",quad.opn2quad);
                                    break;
                                          }
                case CPROGEND   : { break; }

                case CPROGBEGIN  : { break; }

            } /* switch */
            j++;
            } /* if */
        } /* while */

        /* imprime tabela de simbolos */
        fprintf(YYSAIDA,"\n");
        fprintf(YYSAIDA,"%4s%s\n"," ",
                    "==========TABELA DE SIMBOLOS==========");
        fprintf(YYSAIDA,"%s%4s%s%4s%s%4s%s%4s%s\n","nome"," ","classe"," ",
                     "tipotam"," ","endoff"," ","nivel");
        for (i = 1; i <= l-1; i++)
              fprintf(YYSAIDA,"%6s%4d%8d%12d%12d\n",ts[i].nome,ts[i].classe,
                       ts[i].tipotam,ts[i].endoff,ts[i].nivel);
    } /* IMPRIMECODIGO */

/*=========================================================================*/
/*                          MODULO ROTINAS SEMANTICAS                      */
/*=========================================================================*/


%%VARIABLES

    int temporario;
    int classe;
```

```
%%GRAMMAR  program  AND  SEMANTICS


program  = proghead dcls cmdc
                {
                        SALVA(1,proxq-1);
                        GEN(CPROGEND,0,0,0);
                };

proghead = "program"
                {
                        toff    = 0;
                        offset  = 0;
                        proxq   = 1;
                        GEN(CPROGBEGIN,0,0,0);
                };

dcls  = dcl
                {    }
          |  dcls ";" dcl
                {    } ;

dcl  = "id" ":" "integer"
           {
                DECLARA("id".valor,VARIAVEL,INTEIRO,offset);
                offset++;
           };

dcl  = "id" ":" "boolean"
           {
                DECLARA("id".valor,VARIAVEL,LOGICO,offset);
                offset++;
           };

dcl  = prochead "(" par ")" dcls cmdc
           {
                TAMANHO(prochead.valor,offset);
                POPOFF(offset);
                ctx = 1;
                GEN(CPROCEND,0,0,0);
                FBLOCO();
                SALVA(prochead.inicio,proxq-1);
                proxq = prochead.inicio;
           };
```

```
par  = "id" ":" "integer"
          {
               DECLARA("id".valor,PAR,INTEIRO,offset);
               offset++;
          };

par  = "id" ":" "boolean"
          {
               DECLARA("id".valor,PAR,LOGICO,offset);
               offset++;
          };

prochead  = "id" ":" "procedure"
                    {
                     DECLARA("id".valor,PROC,0,proxq);
                     prochead.inicio = proxq;
                     GEN(CPROCBEGIN,"id".valor,0,0);
                     prochead.valor = "id".valor;
                     ABLOCO();
                     PUSHOFF(offset);
                    };

cmdc  = c "begin" cmds "end"
               {    };

c  =
               { ctx = 2; /* CONTEXTO DE COMANDOS  */ };

cmds  = cmd
               {    }
          | cmds ";" cmd
               {    };

cmd  = "id" ":=" exp
          {
               classe = ts["id".valor].classe;
               if ((FOIDECLARADO("id".valor)) && ((classe==VARIAVEL) ||
                    ( classe == PAR)))
                    GEN(CATRIB,"id".valor,exp.r,0);
               else ERRO(7,"id".YYLINHA,"id".YYPOS);
          };

cmd  = "id" "(" exp ")"
          {
               if ((FOIDECLARADO("id".valor)) && (ts["id".valor].classe == PROC))
               {
                    GEN(CCALL,"id".valor,0,0);
```

```
                    GEN(CPARAM,exp.r,0,0);
                }
                else ERRO(8,"id".YYLINHA,"id".YYPOS);
        };

cmd  = "if" cond "then" cmd  "else" n cmd
        {
                CONSERTA(cond.quad,n.quad+1);
                CONSERTA(n.quad,proxq);
        };

cmd  = "while" marca cond "do"  cmd
        {
                GEN(CGOTO,marca.quad,0,0);
                CONSERTA(cond.quad,proxq);
        };

cmd  = cmdc
        {    };

cond  = exp
                {
                 if (exp.tipo == LOGICO)
                        GEN(CDSVF,exp.r,0,0);
                 else ERRO(10,exp.YYLINHA,exp.YYPOS);
                 cond.quad = proxq;
                };

n  =
                {
                 n.quad = proxq;
                 GEN(CGOTO,0,0,0);
                };

marca  =
                { marca.quad = proxq; };

exp  = exp "+" exp
        {
                if (exp[2].tipo != exp[3].tipo)
                        ERRO(9,exp[2].YYLINHA,exp[2].YYPOS);
                if (exp[2].tipo == INTEIRO)
                        op = CMAIS;
                else op = COU;
                temporario = TEMP();
                GEN(op,temporario,exp[2].r,exp[3].r);
                exp.r = temporario;
```

```
                exp.tipo = exp[2].tipo;
        };

exp  = exp "=" exp
        {
                if ((exp[2].tipo != exp[3].tipo) || (exp[2].tipo != INTEIRO))
                        ERRO(9,exp[2].YYLINHA,exp[2].YYPOS);
                temporario = TEMP();
                GEN(CIGUAL,temporario,exp[2].r,exp[3].r);
                exp.r = temporario;
                exp.tipo = LOGICO;
        };

exp  = exp "/" exp
        {
                if ((exp[2].tipo != exp[3].tipo) || (exp[2].tipo != INTEIRO))
                        ERRO(9,exp[2].YYLINHA,exp[2].YYPOS);
                temporario = TEMP();
                GEN(CDIV,temporario,exp[2].r,exp[3].r);
                exp.r = temporario;
                exp.tipo = exp[2].tipo;
        };

exp  = exp "*" exp
        {
                if (exp[2].tipo != exp[3].tipo)
                        ERRO(9,exp[2].YYLINHA,exp[2].YYPOS);
                if (exp[2].tipo == INTEIRO)
                        op = CVEZES;
                else op = CE;
                temporario = TEMP();
                GEN(op,temporario,exp[2].r,exp[3].r);
                exp.r  = temporario;
                exp.tipo = exp[2].tipo;
        };

exp  = exp "**" exp
        {
                if ((exp[2].tipo != exp[3].tipo) || (exp[2].tipo != INTEIRO))
                         ERRO(9,exp[2].YYLINHA,exp[2].YYPOS);
                temporario = TEMP();
                GEN(CPOT,temporario,exp[2].r,exp[3].r);
                exp.r = temporario;
                exp.tipo = exp[2].tipo;
        };
```

```
 exp = exp "-" exp
          {
                if ((exp[2].tipo != exp[3].tipo) || (exp[2].tipo != INTEIRO))
                      ERRO(9,exp[2].YYLINHA,exp[2].YYPOS);
                temporario = TEMP();
                GEN(CMENOS,temporario,exp[2].r,exp[3].r);
                exp.r = temporario;
                exp.tipo = exp[2].tipo;
          };

exp  = "-" exp   %%PREC "*"
          {
                temporario = TEMP();
                if (exp[2].tipo == INTEIRO)
                      op = CINVERTE;
                else op = CNEGA;
                GEN(op,temporario,exp[2].r,0);
                exp.r = temporario;
                exp.tipo = exp[2].tipo;
          }

         | "(" exp ")"
          {
                exp.r = exp[2].r;
                exp.tipo = exp[2].tipo;
          };

exp  = "id"
          {
                classe = ts["id".valor].classe ;
                exp.r  = "id".valor;
                if (FOIDECLARADO("id".valor) && ((classe == VARIAVEL) ||
                      (classe == PAR)))
                      exp.tipo = ts["id".valor].tipotam;
                else {
                            ERRO(7,"id".YYLINHA,"id".YYPOS) ;
                            exp.tipo = INTEIRO;
                       }
          };

exp  = "cte"
          {
                exp.r ="cte".valor;
                exp.tipo = INTEIRO;
          };
```

```
%%CONFLICTS

        %%RIGHT   "=" ;
        %%LEFT    "+","-" ;
        %%LEFT    "*","/"
        %%RIGHT   "**"

(*=================================================================*)
(*                          MODULO PRINCIPAL                       *)
(*=================================================================*)


%%PROGRAM

        if ((FONTE=fopen("entrada.008","r"))==NULL)
        {
                printf("ARQUIVO FONTE NAO PODE SER ABERTO\n");
                exit(1);
        }
        else if ((YYSAIDA = fopen("saida.008","w"))==NULL)
                 {
                        printf("ARQUIVO DE SAIDA NAO PODE SER ABERTO\n");
                        exit(1);
                 }
                 else if ((CODE =fopen("EX.COD","wb"))==NULL)
                          {
                                printf("ARQUIVO PARA ESCRITA DAS QUADRUPLAS
                                        NAO PODE SER ABERTO","\n");
                                exit(1);
                          }
                          else { INICIA();
                                        YYPARSER();
                                        fclose(CODE);
                                        if ((CODE=fopen("EX.COD","rb"))==NULL)
                                        {
                                                printf("ARQUIVO PARA LEITURA DAS
                                                QUADRUPLAS NAO PODE SER
                                                ABERTO","\n");
                                                exit(1);
                                        }
                                        else {IMPRIMECODIGO();
                                                switch(YYRESULTADO)
                                                {
                                                case YYSINTAXEOK   : { fprintf(YYSAIDA,
                                                "PARSE COMPLETED -- SINTAXE OK!\n");
                                                break;
                                                }
```

```
                                        case YYERROFATAL   : { fprintf(YYSAIDA,
                                        "ERRO FATAL: PROGRAMA
                                        CANCELADO.\n");
                                        break;
                                    }
                                        case YYERROSINTAXE : { fprintf(YYSAIDA,
                                        "ERRO DE SINTAXE.\n");
                                        break;
                                    }
                            }
                            fclose(YYSAIDA);
                            fclose(CODE);
                        }
                }
%%END
```