Projet Compilation

Binôme : Sagno Félix & Cissé Mamadou

Nouvelle fonctionnalité :
-Boucle for,
-Boucle while
-Verification de l'existance des variable dans un tableau avant leur utilisation
-Verification de ne pas avoir utiliser un mot clé du langage comme nom de variable

Analyseur Sémantique :

```
%{
        #include <stdio.h>
        #include <stdlib.h>
  #include <string.h>
        extern FILE* yyin;
        extern int valEntier;
  extern char valIdentif[256];
  extern char valChaine[255];
        int yylex(void); // defini dans progL.cpp, utilise par yyparse()
        void yyerror(char * msg);
        extern unsigned int lineNumber; // notre compteur de lignes

  typedef struct {
    char *identif;
    int type;
  } ENTREE_DICO;

#define TAILLE_INITIALE_DICO 50
#define INCREMENT_TAILLE_DICO 25

char TOKENS[5][8]={"Gui","nea","write","read", "CONST"};
ENTREE_DICO *dico;
int maxDico, sommet, base;

void creerDico(void) {
  maxDico = TAILLE_INITIALE_DICO;
  dico = malloc(maxDico * sizeof(ENTREE_DICO));
  if (dico == NULL)
    yyerror("Erreur interne (pas assez de mémoire)");
  sommet = base = 0;
}

void agrandirDico(void) {
  maxDico = maxDico + INCREMENT_TAILLE_DICO;
  dico = realloc(dico, maxDico);
  if (dico == NULL)
    yyerror("Erreur interne (pas assez de mémoire)");
}
int isToken(char *s)
```

```
{
        for(int i=0;i<3;i++)
        {
                if(strcmp(s,TOKENS[i])==0)
                {
                        return 1;
                }else{
                  return 0;
                }

        }
}
int isDeclared(char *s)
{
        for(int i=0;i<sommet;i++){
                if(strcmp(s,dico[i].identif)==0) {
      return 1;
                }
        }
        return 0;
}

void AjoutIdentif(char *identif,int type) {
        if(isToken(identif)==1) {
                printf("ERREUR --- linge %d :  Nom de variable est un mot cle\n",lineNumber-1);
                exit(-1);
        }
  if(isDeclared(identif)==1) {
                printf("ERREUR --- linge %d : Variable deja declaree\n", lineNumber-1);
                exit(-2);
        }
  if (sommet >= maxDico)
     agrandirDico();

  dico[sommet].identif = malloc(strlen(identif) + 1);
  if (dico[sommet].identif == NULL){
                printf("Erreur interne pas assez de mémoire)");
                exit(-1);
        }

  strcpy(dico[sommet].identif, identif);
  dico[sommet].type = type;
  sommet++;
}

%}
%union{
        char var[255];
        int entier;
}

%token GUI  <var>IDENTIF NEA <entier>ENTIER CRO_O CRO_F SI EGAL DIFF SINON SINONSI
%token WRITE READ INFEG SUPEG INF SUP FOR CHAINE POINT
%token INLINE EGALCONDI GRIF PLUS MOINS MUL MODULO DIV          COMMENT VIRGUL FUNCTION
VIDE

%start program
```

```
%%
program : listRetour
                    |
                    listDeclaration  GUI INLINE listInstruction NEA listRetour
                            ;
listDeclaration :   declaration
                                        |
                                        listDeclaration declaration
                                        ;


declaration :       IDENTIF INLINE
            {
                                        AjoutIdentif(valIdentif,0);
                                    }
                                    |
                                    FUNCTION CHAINE INLINE
                                    ;

listInstruction :instruction
                                    |
                                    listInstruction instruction
                                    ;

instruction :       affichage
                                    |
                                    affectation
                                    |
                                    condition
                                    |
                                    boucle
                                    |
                                    lecture
                                    |
                                    commentaire
                                    |
                                    INLINE
                                    ;
boucle :            for
                                    |
                                    while
                                    ;
for                 :               FOR IDENTIF CRO_O ENTIER VIRGUL ENTIER CRO_F INLINE
listInstruction POINT INLINE
                                    ;
while     :                 FOR CRO_O comparaison CRO_F INLINE listInstruction POINT INLINE
                                    ;

commentaire :       COMMENT CHAINE INLINE
                                    ;
lecture :           READ CRO_O expression CRO_F INLINE
                                    ;
affichage :                 WRITE CRO_O contenu CRO_F INLINE
                                    ;
contenu: simple_affichage
                    |
```

```
                texte
                |
                concat
    ;
concat: texte POINT simple_affichage
                |
                simple_affichage POINT texte
                |
                texte POINT simple_affichage POINT texte
                ;
texte:    GRIF list_chaine GRIF
                ;
list_chaine:  CHAINE
                        |
                        list_chaine CHAINE
                        ;
simple_affichage:ENTIER
                                |
                                IDENTIF
                                ;

affectation :       IDENTIF EGAL expression INLINE
        {
                                        if(isDeclared(valIdentif)==0){
                                                printf("ERREUR --- linge %d : %s non declaree\n",
lineNumber-1,valIdentif);
                        exit(-2);
                                        }
                                }
                                ;
condition:              condition_si POINT INLINE
        {
                                        if(isDeclared(valIdentif)==0){
                                                printf("ERREUR --- linge %d : %s non declaree\n",
lineNumber-1,valIdentif);
                        exit(-2);
                                        }
                                }
        |
        condition_si POINT INLINE condition_sinon
        |
        condition_si POINT INLINE list_condition_sinon_si
        ;
list_condition_sinon_si: condition_sinon_si POINT INLINE
                                                        |
                                                        condition_sinon_si POINT INLINE
condition_sinon
                                                        |
                                                        list_condition_sinon_si condition_sinon_si
POINT INLINE
            ;
condition_sinon_si:SINONSI CRO_O comparaison CRO_F INLINE listInstruction
                                ;
condition_si  :SI CRO_O comparaison CRO_F INLINE listInstruction
                                ;
condition_sinon: SINON INLINE listInstruction POINT INLINE
                                ;
```

```
comparaison: IDENTIF signe_comparaison expression
                                        ;
signe_comparaison: EGALCONDI
                                         |
                                         SUPEG
                                         |
                                         INFEG
                                         |
                                         INF
                                         |
                                         SUP
                                         |
                                         DIFF
                                         ;
expression :        IDENTIF
        {
                                                 if(isDeclared(valIdentif)==0){
                                                         printf("ERREUR --- linge %d : %s non declaree\n",
lineNumber-1,valIdentif);
                                exit(-2);
                                                 }
                                        }
                                        |
                                        ENTIER
                                        ;
listRetour: Retour
                                |
                                listRetour Retour
                                ;
Retour : INLINE;
%%

void yyerror(char * msg){
        printf("\t Ligne %d : %s\n", lineNumber, msg);
}
int main(int argc,char ** argv){
        if(argc>1) yyin=fopen(argv[1],"r"); // check result !!!
        lineNumber=1;
        if(!yyparse())
        printf("Expression correct\n");
        return(0);
}
```

Exemple D'algorithme :

```
@val4
@val2
@val1
Gui
    @val2=10
    write["Donner une entier"]
    read[@val1]
    #[@val1<=0]
        @val1=@val2
```

```
    .
    ##@val[1,5]
        //debut de la boucle for
        read[@val1]
        @val = @val1
        #[@val1<=0]
            @val1=@val2
        .
    .
    ##[@val1<=0]
        @val1=@val2
    .
nea
```