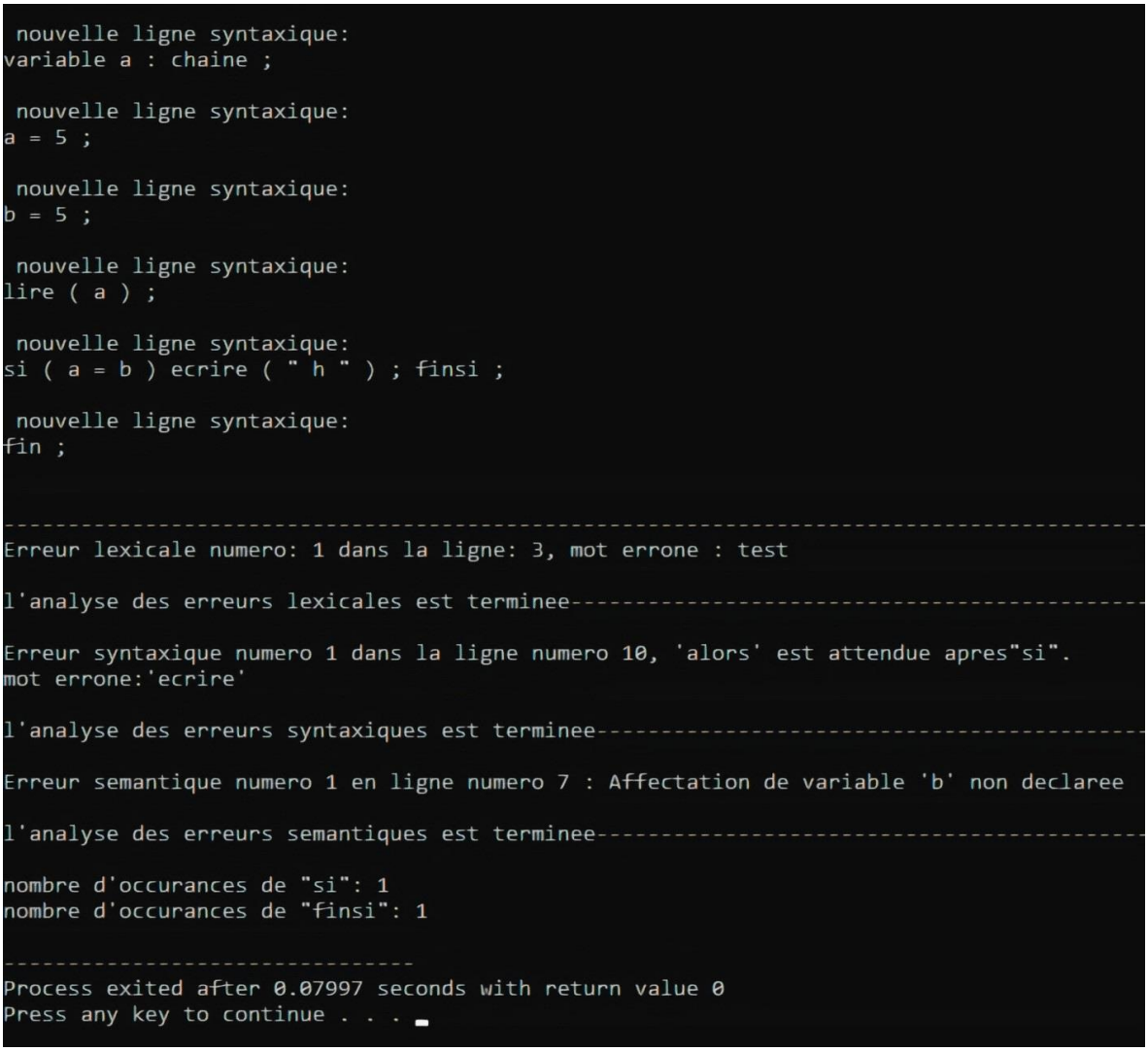
Etudiant : **AIT AHLAL Mouaad** | Complexité et algorithmique avancé | 30/01/ 2024

# Rapport de projet

REALISATION D'UN COMPILATEUR EN C

PROFESSEUR : **HNINI ABDELHALIM**




ANNEE UNIVERSITAIRE :

2023 / 2024

# Résumé :

Mon projet traite l'implémentation d'un analyseur lexical, syntaxique et sémantique pour un langage algorithmique. Le code source, écrit en langage C, vise à analyser un fichier source écrit dans le langage cible, détectant ainsi les erreurs lexicales, syntaxiques et sémantiques éventuelles.

**Les fonctions que j'ai implémenté :**

1. **Analyse Lexicale :**
    - Le programme parcourt le fichier source caractère par caractère, identifiant les mots du dictionnaire et signalant les erreurs lexicales.
2. **Analyse Syntaxique :**
    - Une liste chaînée est utilisée pour stocker les mots extraits. Des erreurs syntaxiques, telles que l'absence de mots clés essentiels, sont signalées.
3. **Analyse Sémantique :**
    - L'analyse sémantique vérifie la cohérence sémantique du programme, Les erreurs de l'utilisation de variables non déclarées, sont identifiées. Je note que j'ai traité les cas spéciales de déclarions de variables, car ils y restent beaucoup de cas à traiter dans le contexte sémantique de l'algorithme.

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int nlig = 0; // compteur de numero de ligne
int ncar = 0; // compteur de numero de caractere
int j=1; //compteur d'erreurs lexicales
int k=1; //compteur d'erreurs syntaxiques
int l=1; //compteur d'erreurs semantiques
int err_lex_existe =0;// verification s'il y a au moins une
erreur lexicale dans l'algorithme
char semicolon[2] =";\0"; // variable qui contient ';' car on va
l'utiliser plusieurs fois
char deux_pts[2] =":\0"; //variable qui contient ':' car on va
l'utiliser plusieurs fois
int existe_si =0;
int existe_affectation=0;
int compteur_si=0;
int compteur_finsi=0;
int verif_algo=0; // verifier l'existance du mot Algorithme au
debut
int verif_declaration=0; // verifier l'existance du mot
Declaration au debut
int verif_debut=0; // verifier l'existance du mot Debut au debut
int verif_fin=0; // verifier l'existance du mot Fin à la fin
int verif_apostrophe =0; //verifier " lors de l'écriture d'une
chaine
int error =0;
int variables_declarees[26] = {0}; // Tableau pour stocker les
variables déclarées (analyse sémantique) (par exemple, de A à Z)
int variables_dec_type[26] = {0}; // Tableau pour stocker les
variables déclarées (analyse sémantique) (par exemple, de A à Z)
int variables_initialisees[26] = {0}; // Tableau pour vérifier
les variables initialisées
int variables_ini_type[26] = {0}; // Tableau pour stocker les
types des variables initialisées
int pass_si = 0; //pass: si on analyse syntaxiquement des
instructions dedans si ou bien dedans sinon
int liste_finsi = 0; // marquant pour la liste chainée de si
typedef struct liste {
      char Tab[10];
      struct liste *next;
} mot;
mot* debut_copie = NULL;
void Analyser_lix(char Tab_Ligne[], FILE*e) {
      int verification_lex = 0; //variable qui indique s'il y a
des erreurs lexicales
      int i = 0; // compteur pour le remplissage du tableau
      char c;
      char Tab_Dic[10]; // Augmentation de la taille du tableau
pour stocker les mots du dictionnaire

      FILE *d = fopen("dictionnaire.txt", "r");
```

```
53          if (d == NULL) {
54                printf("Problème dans le fichier dictionnaire.\n");
55                return; // Sortie de la fonction en cas d'échec
56    d'ouverture du fichier
57          }
58
59          // Boucle pour parcourir le fichier dictionnaire et comparer
60    chaque mot avec Tab_Ligne
61          while ((c = fgetc(d)) != EOF) {
62                if (c == '\n') {
63                      Tab_Dic[i] = '\0'; // Ajouter le caractère de
64    fin de chaîne
65                      if (strcmp(Tab_Ligne, Tab_Dic) == 0) {
66                            verification_lex = 1;
67                            break; // Sortir de la boucle dès qu'un
68    mot correspond
69                      }
70                      i = 0; // Réinitialiser l'index pour le prochain
71    mot du dictionnaire
72                } else {
73                      Tab_Dic[i++] = c;
74                }
75          }
76
77          if (verification_lex == 0) {
78                err_lex_existe = 1;
79                if (e == NULL) {
80                      printf("Impossible d'ouvrir le fichier des
81    erreurs.\n");
82                      return;
83                }
84                fprintf(e, "Erreur lexicale numero: %d dans la ligne:
85    %d, mot errone : %s\n\n",j, nlig, Tab_Ligne);
86                j++;
87          }
88          fclose(d); // Fermeture du fichier dictionnaire
89    }
90
91    void InsererFin(mot **debut, const char Tab_Ligne[10]) {
92          mot* nouveau = (mot *)malloc(sizeof(mot));
93          strcpy(nouveau->Tab, Tab_Ligne);
94          nouveau->next = NULL;
95          if (*debut == NULL) {
96                *debut = nouveau;
97          } else {
98                mot *temp = *debut;
99                while (temp->next != NULL) {
100                     temp = temp->next;
101               }
102               temp->next = nouveau;
103         }
```

```
104 }
105
106 void Afficher(mot *debut) {
107       mot *temp = debut;
108       printf("\n nouvelle ligne syntaxique:\n");
109       while (temp != NULL) {
110             printf("%s ", temp->Tab);
111             temp = temp->next;
112       }
113       printf("\n");
114 }
115
116 void Analyse_syntaxique(mot* debut, FILE* s,int nlig) {
117
118       mot* temp = debut;
119
120       if (strcmp(temp->Tab, "Algorithme") != 0 && verif_algo ==0)
121 {
122             fprintf(s, "Erreur syntaxique numero %d dans la ligne
123 numero %d, il faut commencer par le mot \"Algorithme\" a la place
124 de le mot '%s'\n", k, nlig, temp->Tab);
125             k++;
126             verif_algo = 1;
127             error =1;
128       } else if(strcmp(temp->Tab, "Algorithme") == 0) {
129             verif_algo = 1;
130       }
131
132       else if (strcmp(temp->Tab, "debut") != 0 && verif_debut ==0
133 && error ==0 ) {
134             fprintf(s, "Erreur syntaxique numero %d dans la ligne
135 numero %d, il faut commencer par le mot \"debut\" a la place de
136 le mot '%s'\n", k, nlig, temp->Tab);
137             k++;
138             verif_debut = 1;
139             error =1;
140       } else if(strcmp(temp->Tab, "debut") == 0) {
141             verif_debut = 1;
142       }
143
144       else if (strcmp(temp->Tab, "Declaration") != 0 &&
145 verif_declaration ==0 && error ==0) {
146             fprintf(s, "Erreur syntaxique numero %d dans la ligne
147 numero %d, il faut commencer par le mot \"Declaration\" a la
148 place de le mot '%s'\n", k, nlig, temp->Tab);
149             k++;
150             verif_declaration = 1;
151             error =1;
152       } else if(strcmp(temp->Tab, "Declaration") == 0) {
153             verif_declaration = 1;
154       }
```

```
155
156
157   // verifier si fin. existe
158
159         temp=debut;
160         while(temp->next->next !=NULL) {
161               temp = temp->next;
162         }
163         if (strcmp(temp->Tab,"fin")==0 || strcmp(temp->Tab,
164   "fin.\0")==0) verif_fin =1;
165
166         temp=debut;
167         while(temp->next !=NULL) {
168               if(strcmp(temp->Tab,"=\0")==0) existe_affectation = 1;
169               temp = temp->next;
170         }
171
172         //verifier si les mots se terminent par un ';'
173         temp =debut;
174         while(temp->next!=NULL) {
175               temp = temp->next;
176         }
177         if(strcmp(temp->Tab,semicolon)!=0) {
178               fprintf(s,"Erreur syntaxique numero %d dans la ligne
179   numero %d, ';' est attendu apres le mot '%s'\n",k,nlig,temp-
180   >Tab);
181               k++;
182         }
183
184
185
186         temp=debut;
187         if(strcmp(temp->Tab,"si") !=0) {
188               if(pass_si ==0) {
189                     while(temp->next!=NULL) {
190                           temp=temp->next;
191                           if((strcmp(temp->Tab,"variable") ==
192   0)||(strcmp(temp->Tab,"var") == 0)) {
193                                 fprintf(s,"Erreur syntaxique numero
194   %d dans la ligne numero %d, la declaration de chaque variable
195   doit se faire sur une nouvelle ligne. erreur: le mot
196   '%s'\n",k,nlig,temp->Tab);
197                                 k++;
198                           }
199                           if(strcmp(temp->Tab,"lire") == 0) {
200                                 fprintf(s,"Erreur syntaxique numero
201   %d dans la ligne numero %d, chaque lecture doit se faire sur une
202   nouvelle ligne. erreur: le mot '%s'\n",k,nlig,temp->Tab);
203                                 k++;
204                           }
205                           if(strcmp(temp->Tab,"ecrire") == 0) {
```

```
                                    fprintf(s,"Erreur syntaxique numero
%d dans la ligne numero %d, chaque ecriture doit se faire sur une
nouvelle ligne. erreur: le mot '%s'\n",k,nlig,temp->Tab);
                                    k++;
                              }
                        }
                  }
            }
            temp = debut;
            if((strcmp(temp->Tab,"variable") == 0)||(strcmp(temp-
>Tab,"var") == 0)) {
                  temp=temp->next;
                  if(strlen(temp->Tab)!=1) {
                        fprintf(s,"Erreur syntaxique numero %d dans la
ligne numero %d, l'identificateur doit comporter un seul
caractere. mot errone:'%s'\n",k,nlig,temp->Tab);
                        k++;
                  }
                  temp=temp->next;
                  if((strcmp(temp->Tab,deux_pts) != 0)) {
                        fprintf(s,"Erreur syntaxique numero %d dans la
ligne numero %d, ':' est attendue. mot
errone:'%s'\n",k,nlig,temp->Tab);
                        k++;
                  }
                  temp=temp->next;
                  if((strcmp(temp->Tab,"entier") == 0) || (strcmp(temp-
>Tab,"reel") == 0) || (strcmp(temp->Tab,"chaine") == 0) ) {}
                  else {
                        fprintf(s,"Erreur syntaxique numero %d dans la
ligne numero %d, type de variable inconnu. mot
errone:'%s'\n",k,nlig,temp->Tab);
                        k++;
                  }
            }
            if(strcmp(temp->Tab,"lire") == 0) {
                  temp=debut;
                  temp=temp->next;
                  if((strcmp(temp->Tab,"(\0") != 0)) {
                        fprintf(s,"Erreur syntaxique numero %d dans la
ligne numero %d, '(' est attendue. mot
errone:'%s'\n",k,nlig,temp->Tab);
                        k++;
                  }
                  temp=temp->next;
                  temp=temp->next;
                  if((strcmp(temp->Tab,")\0") != 0)) {
                        fprintf(s,"Erreur syntaxique numero %d dans la
ligne numero %d, ')' est attendue. mot errone:'%s' (la lecture
doit se faire variable par variable)\n",k,nlig,temp->Tab);
                        k++;
```

```
257                 }
258           }
259           temp=debut;
260           if(strcmp(temp->Tab,"ecrire") == 0) {
261                 temp=temp->next;
262                 if((strcmp(temp->Tab,"(\0") != 0)) {
263                       fprintf(s,"Erreur syntaxique numero %d dans la
264 ligne numero %d, '(' est attendue. mot
265 errone:'%s'\n",k,nlig,temp->Tab);
266                       k++;
267                 } else if(strcmp(temp->next->Tab,"\"\0") == 0) {
268                       temp=temp->next;
269                       while(temp->next !=NULL) {
270                             temp=temp->next;
271                             if(strcmp(temp->Tab,"\"\0") == 0) {
272                                   verif_apostrophe =1;
273                                   temp=temp->next;
274                                   if(strcmp(temp->Tab,")\0") != 0) {
275                                         fprintf(s,"Erreur syntaxique
276 numero %d dans la ligne numero %d, ')' est attendue. mot
277 errone:'%s'\n",k,nlig,temp->Tab);
278                                         k++;
279                                   }
280                             }
281                       }
282                       if(verif_apostrophe ==0) {
283                             fprintf(s,"Erreur syntaxique numero %d
284 dans la ligne numero %d, \" est attendue. mot
285 errone:'%s'\n",k,nlig,temp->Tab);
286                             k++;
287                       }
288                 } else if(!(temp->next->Tab[0] >= 'a' && temp->next-
289 >Tab[0] <= 'z')) {
290                       fprintf(s,"Erreur syntaxique numero %d dans la
291 ligne numero %d, une variable (a - z) est attendue. mot
292 errone:'%s'\n",k,nlig,temp->Tab);
293                       k++;
294                 } else if(strcmp(temp->next->next->Tab,")\0")!=0) {
295                       temp=temp->next;
296                       fprintf(s,"Erreur syntaxique numero %d dans la
297 ligne numero %d, ')' est attendue. mot errone:'%s' (l'ecriture
298 doit se faire variable par variable)\n",k,nlig,temp->Tab);
299                       k++;
300                 }
301           }
302           //analyse de si sinon
303
304           temp =debut;
305           if(strcmp(temp->Tab,"si")==0) {
306                 mot* tempfinsi = debut;
307                 while(tempfinsi->next->next !=NULL) {
```

```
                        tempfinsi = tempfinsi->next;
                }
                if(strcmp(tempfinsi->Tab,"finsi")!=0) {
                        fprintf(s,"Erreur syntaxique numero %d dans la
ligne numero %d, \"finsi\" est attendue apres l'instruction
\"si\". mot errone:'%s'\n",k,nlig,tempfinsi->Tab);
                        k++;
                }
                mot* tempsi = NULL;
                compteur_si++;
                existe_si=1;
                temp=temp->next;
                if(strcmp(temp->Tab,"(\0")==0) {
                        int verifparenthese = 0;
                        temp=temp->next;
                        mot* analyse_si = temp;
                        mot* analyse_si_r = temp;
                        while(strcmp(temp->Tab,")\0")!=0) {
                                temp=temp->next;
                                if(strcmp(temp->Tab,")\0")==0) {
                                        verifparenthese =1;
                                        tempsi = temp;
                                }
                        }

                        if(verifparenthese == 1) {
                                while(analyse_si->next != tempsi) {
                                        analyse_si= analyse_si->next;
                                }
                                analyse_si->next = NULL;
                                analyse_si = analyse_si_r;
                                InsererFin(&analyse_si,";\0");
                                Analyse_syntaxique(analyse_si,s,nlig);
                                temp = tempsi;
                                temp=temp->next;
                                if(strcmp(temp->Tab,"alors")!=0 &&
strcmp(temp->Tab,"si")!=0) {
                                        fprintf(s,"Erreur syntaxique numero
%d dans la ligne numero %d, 'alors' est attendue
apres\"si\".\nmot errone:'%s'\n",k,nlig,temp->Tab);
                                        k++;
                                } else {
                                        while(temp->next->next !=NULL) {
                                                temp=temp->next;
                                                if(strcmp(temp->Tab,";\0")==0)
nlig++;
                                                pass_si = 1;

        Analyse_syntaxique(temp,s,nlig);
                                        pass_si = 0;
                                }
```

```
                        }
                    } else {
                        fprintf(s,"Erreur syntaxique numero %d
dans la ligne numero %d, ')' est attendue apres l'instruction
\"si\". mot errone:'%s'\n",k,nlig,temp->Tab);
                        k++;
                    }
                } else {
                    fprintf(s,"Erreur syntaxique numero %d dans la
ligne numero %d, '(' est attendue apres l'instruction \"si\". mot
errone:'%s'\n",k,nlig,temp->Tab);
                    k++;
                }
                if(strcmp(temp->Tab,"sinon")==0 && strcmp(temp->next-
>Tab,"si")!=0 ) {
                    temp=temp->next;
                    pass_si = 1;
                    Analyse_syntaxique(temp,s,nlig);
                    pass_si = 0;
                }

        }
        // treat sinon si without previous si
        temp=debut;
        if(strcmp(temp->Tab,"sinon")==0 && strcmp(temp->next-
>Tab,"si")==0) {
                if(existe_si ==0) {
                    fprintf(s,"Erreur syntaxique numero %d dans la
ligne numero %d, \"sinon si\" est utilisee sans l'instruction
\"si\". mots errones:'%s %s'\n",k,nlig,temp->Tab,temp->next-
>Tab);
                    k++;
                }
        }
        // traiter sinon doit etre après une si
        else if((strcmp(temp->Tab,"sinon")==0 && strcmp(temp->next-
>Tab,"si")!=0)) {
                if(existe_si ==0) {
                    fprintf(s,"Erreur syntaxique numero %d dans la
ligne numero %d, \"sinon\" est utilisee sans l'instruction
\"si\". mots errones:'%s %s'\n",k,nlig,temp->Tab,temp->next-
>Tab);
                    k++;
                } else {
                    //temp=temp->next;
                    //Analyse_syntaxique(temp,s,nlig);
                }
        }
        //analyse de l'affectation;
        mot* tempdebut;
        temp=debut->next;
```

```
410         tempdebut= debut;
411         if (existe_affectation == 1) {
412               existe_affectation = 0;
413               char variable = tempdebut->Tab[0];
414               if (!(variable >= 'a' && variable <= 'z')) {
415                     fprintf(s, "Erreur syntaxique numero %d en ligne
416   numero %d : Affectation non validee, membre gauche de
417   l'operateur'\n",k,nlig, variable);
418                     k++;
419               } else if(strcmp(tempdebut->next->Tab,"=\0")!=0) {
420                     fprintf(s, "Erreur syntaxique numero %d en ligne
421   numero %d : Affectation non validee, \"=\" est attendue a la
422   place de %s\n",k,nlig, tempdebut->Tab);
423                     k++;
424               } else {
425                     int err;
426                     err=0;
427                     temp = temp->next;
428                     if (!((temp->Tab[0] >= 'a' && temp->Tab[0] <=
429   'z') || (temp->Tab[0] >= '1' && temp->Tab[0] <= '9' ))) {
430                           err=1;
431                     }
432
433                     if(err == 1) {
434                           fprintf(s, "Erreur syntaxique numero %d en
435   ligne numero %d : Affectation non validee, membre droit de
436   l'operateur'\n",k,nlig, variable);
437                           k++;
438                     }
439               }
440
441         }
442   }
443   //fin de l'analyse syntaxique
444
445
446   void Analyse_semantique(mot* debut, FILE* sem, int nlig) {
447         mot* temp = debut;
448         mot* tempdebut = debut;
449
450         //1 = entier 2=reel 3= chaine
451         /*void stocker(mot *debut) {
452         if (debut == NULL) {
453             printf("\n");
454             return;
455         }
456
457         stocker(debut->next);
458         printf("%s ", debut->Tab);
459         }*/
460         // Vérification des déclarations de variables
```

```
461         if (strcmp(temp->Tab, "variable") == 0 || strcmp(temp->Tab,
462 "var") == 0) {
463                 temp=temp->next;
464                 char variable = temp->Tab[0];
465                 if (variable >= 'a' && variable <= 'z') {
466                       variables_declarees[variable - 'a'] = 1; //
467 Marquer la variable comme déclarée
468                       temp = temp->next;
469                       temp = temp->next;
470                       if(strcmp(temp->Tab,"entier")==0) {
471                             variables_dec_type[variable - 'a'] = 1; //
472 Marquer la variable comme déclarée entier
473                       }
474                       if(strcmp(temp->Tab,"reel")==0) {
475                             variables_dec_type[variable - 'a'] = 2; //
476 Marquer la variable comme déclarée reel
477                       }
478                       if(strcmp(temp->Tab,"chaine")==0) {
479                             variables_dec_type[variable - 'a'] = 3; //
480 Marquer la variable comme déclarée chaine
481                       }
482                 }
483         }
484
485         temp =debut->next;
486         // Vérification de l'initialisation des variables
487         if (strcmp(temp->Tab, "=\0") == 0) {
488                 char variable = tempdebut->Tab[0];
489                 if (variable >= 'a' && variable <= 'z') {
490                       variables_initialisees[variable - 'a'] = 1; //
491 Marquer la variable comme initialisée
492                 }
493         }
494
495         // Vérification d'utilisation de variables non déclarées
496         temp=debut->next;
497
498         if (strcmp(temp->Tab, "=\0") == 0) {
499                 char variable = tempdebut->Tab[0];
500                 if (variable >= 'a' && variable <= 'z' &&
501 !variables_declarees[variable - 'a']) {
502                       fprintf(sem, "Erreur semantique numero %d en
503 ligne numero %d : Affectation de variable '%c' non
504 declaree\n",l,nlig, variable);
505                       l++;
506                 }
507         }
508         temp=debut->next;
509
510         if (strcmp(temp->Tab, "(\0") == 0) {
511                 char variable = temp->next->Tab[0];
```

```
512             if ((variable >= 'a' && variable <= 'z') &&
513 !variables_declarees[variable - 'a']) {
514                     fprintf(sem, "Erreur semantique numero %d en
515 ligne numero %d : utilisation de variable '%c' non
516 declaree\n",l,nlig, variable);
517                     l++;
518             }
519       }
520       //Fin de Vérification d'utilisation de variables non
521 déclarées
522       // Vérification des variables non initialisées
523       int c;
524       if((strcmp(tempdebut->Tab,"ecrire")==0 && strcmp(tempdebut-
525 >next->next->Tab,"\"\0")!=0) || strcmp(tempdebut->Tab,"si")==0 ||
526 strcmp(tempdebut->next->Tab,"si")==0 ) {
527             if(strcmp(tempdebut->next->Tab,"si")==0) c =
528 tempdebut->next->next->next->Tab[0];
529             else c = tempdebut->next->next->Tab[0];
530             if (variables_declarees[c-'a'] &&
531 !variables_initialisees[c-'a']) {
532                     fprintf(sem, "Erreur semantique numero %d en
533 ligne numero %d : Variable '%c' declaree mais non
534 initialisee\n",l,nlig, c);
535                     l++;
536             }
537       }
538 //ici nouveau cas
539 }
540
541
542 int main() {
543       FILE *e = fopen("erreurs_lexicales.txt", "w"); // ouverture
544 du fichier erreurs lexicales
545       FILE *s = fopen("erreurs_syntaxiques.txt", "w"); //
546 ouverture du fichier erreurs syntaxiques
547       FILE *sem = fopen("erreurs_semantiques.txt", "w"); //
548 ouverture du fichier erreurs semantiques
549       mot *debut = NULL;
550       // Ouvrir le fichier algorithme
551       FILE *f = fopen("algorithme.txt", "r");
552       //vérifier si le fichier existe
553       if (f == NULL) {
554             printf("Impossible d'ouvrir le fichier.\n");
555             return 1;
556       }
557       char c; // lecture de chaque caractere
558       char Tab_Ligne[10];
559       int i=0; //tableau des lignes avec son compteur i
560       int espace=0; //variable pour résoudre le cas des espaces
561 consecutifs multiples
562       //parcourir les caractères
```

```
563         //mot* resultat= (mot*)malloc(sizeof(mot));
564         while ((c = fgetc(f)) != EOF) {
565               if(c == '\n') {
566                     nlig++;
567                     i = 0;
568                     ncar=0;
569               }
570               if (c == ' ' || c=='\n') {
571                     if (espace == 0) {
572                           c = '\n';
573                           i = 0;
574                           Analyser_lix(Tab_Ligne,e);
575                           if(strcmp(Tab_Ligne,semicolon)!=0) {
576                                 InsererFin(&debut,Tab_Ligne);
577                                 if(strcmp(Tab_Ligne,"finsi")==0) {
578                                       liste_finsi = 1;
579                                       compteur_finsi++;
580                                       nlig--;
581                                 }
582                           } else {
583                                 //traiter la liste chainée spéciale
584   de si
585                                 if (strcmp(debut->Tab,"si")==0 &&
586   liste_finsi == 0) {
587                                       InsererFin(&debut,Tab_Ligne);
588                                 } else {
589                                       InsererFin(&debut,Tab_Ligne);
590                                       Afficher(debut);
591                                       //ici on doit commencer
592   l'analyse syntaxique
593
594         Analyse_syntaxique(debut,s,nlig);
595                                       //ici on doit commencer
596   l'analyse semantique
597
598         Analyse_semantique(debut,sem,nlig);
599                                       debut_copie =debut;
600                                       debut =NULL;
601                                 }
602                           }
603                     }
604                     espace++;
605               } else {
606                     Tab_Ligne[i++] = c;
607                     ncar++;
608                     Tab_Ligne[i] = '\0';
609                     espace = 0;
610               }
611         }
612
```

```
613         //pour traiter le cas si une ou plusieurs parties de
614   l'algorithme n'est pas terminée par un ';'
615         if(debut!=NULL) {
616               Analyse_syntaxique(debut,s,nlig);
617               Afficher(debut);
618         }
619
620         //traiter si le fichier finis par fin.
621         if(verif_fin ==0) {
622               mot* temp =debut_copie;
623               while(temp->next!=NULL) {
624                     temp = temp->next;
625               }
626               nlig++;
627               fprintf(s,"Erreur syntaxique numero %d dans la ligne
628   numero %d, il faut finir par le mot fin.  mot errone:
629   %s\n",k,nlig,temp->Tab);
630               k++;
631         }
632         //traiter le cas special de nbr de si et finsi après la fin
633   de fichier
634         if(compteur_si != compteur_finsi) {
635               mot* temp =debut_copie;
636               while(temp->next!=NULL) {
637                     temp = temp->next;
638               }
639               nlig++;
640               fprintf(s,"Erreur syntaxique numero %d dans la ligne
641   numero %d, le nombre des \"si\" doit etre egal au nombre des
642   \"finsi\" dans l'algorithme'. mot errone:'%s'\n",k,nlig,temp-
643   >Tab);
644               k++;
645         }
646
647         printf("\n");
648
649         printf("\n");
650         fclose(f);
651         fclose(e); // Fermeture du fichier erreurs lexicales
652         fclose(s); // Fermeture du fichier erreurs syntaxiques
653         fclose(sem); // Fermeture du fichier erreurs semantiques
654         printf("-----------------------------------------------------
655   ---------------------------------------------------------------
656   ----\n");
657
658
659         //maintenant on affiche les erreurs lexicales (s'il y a
660   aucune erreur rien ne va être affiché)
661
662         if (err_lex_existe ==1) {
663               FILE *err_lix = fopen("erreurs_lexicales.txt", "r");
```

```
664             if (err_lix == NULL) {
665                   printf("Impossible d'ouvrir le fichier des
666 erreurs lexicales.\n");
667                   return 1;
668             }
669             char c_err;
670             while ((c_err = fgetc(err_lix)) != EOF) {
671                   printf("%c", c_err);
672             }
673             fclose(err_lix); // Fermeture du fichier erreurs.txt
674 
675       }
676       //else{
677       printf("l'analyse des erreurs lexicales est terminee--------
678 ------------------------------------------------------\n\n");
679       //ici affichage des erreurs syntaxiques
680       FILE *err_syn = fopen("erreurs_syntaxiques.txt", "r");
681       if (err_syn == NULL) {
682             printf("Impossible d'ouvrir le fichier des erreurs
683 syntaxiques.\n");
684             return 1;
685       }
686       char c_err;
687       while ((c_err = fgetc(err_syn)) != EOF) {
688             printf("%c", c_err);
689       }
690       fclose(err_syn); // Fermeture du fichier erreurs.txt
691       printf("\nl'analyse des erreurs syntaxiques est terminee----
692 ------------------------------------------------------\n\n");
693       fclose(err_syn);//fermeture du fichier
694 
695       FILE *err_sem = fopen("erreurs_semantiques.txt", "r");
696       if (err_sem == NULL) {
697             printf("Impossible d'ouvrir le fichier des erreurs
698 semantiques.\n");
699             return 1;
700       }
701       char c_errsem;
702       while ((c_errsem = fgetc(err_sem)) != EOF) {
703             printf("%c", c_errsem);
704       }
705       fclose(err_sem); // Fermeture du fichier erreurs.txt
706       printf("\nl'analyse des erreurs semantiques est terminee----
707 ------------------------------------------------------\n\n");
708 
709       //}
710       free(debut);
711       printf("nombre d'occurances de \"si\": %d \nnombre
712 d'occurances de \"finsi\": %d \n",compteur_si,compteur_finsi);
713       return 0;
714 }
```