الجمهورية الجزَائِرية الديمقرَاطية الشعبية
RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
وزَارة التعليم العَالي و البحث العلمي
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جَامعة عمَار ثليجي بِالأَغوَاط
UNIVERSITE AMAR TELIDJI LAGHOUAT

FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

MÉMOIRE DE LICENCE

Domaine : Mathématiques et Informatique

Filière : Informatique

Option : Systèmes et Réseaux Informatiques

Par :

Ladjedel ahlem **Benchaoui amel**

Thème :

IMPLÉMENTATION D'UN COMPILATEUR SOURCE À SOURCE D'UN ALGORITHME

Proposé par : M^{lle} **Fatima Zohra BOUSBAA**

Année universitaire 2019/2020

Dédicaces

Je voudrais remercier dieu pour toute l'énergie qu'il m'a donné durant ces trois années et me donné la capacité de terminer ce que j'ai commencé jusqu'au bout.

Je voudrais dédier mon travail à mes chers parents, mes chers frères mes grands-pères mes tantes et mes oncles ainsi que toute ma famille sans qui je n'aurais jamais pu faire ce travail.

Ladjedel ahlem

Dédicaces

Je voudrais remercier dieu pour toute l'énergie qu'il m'a donné durant ces trois années et me donné la capacité de terminer ce que j'ai commencé jusqu'au bout.

Je voudrais dédier mon travail à mes chers parents, mes chers frères mes grands-pères mes tantes et mes oncles ainsi que toute ma famille sans qui je n'aurais jamais pu faire ce travail.

Benchaoui amel

Remerciement

Nous tenons d'abord à remercier tout particulièrement M^{lle} Fatima Zohra BOUSBAA qui nous a permis de bénéficier de son encadrement. Les conseils qu'elle nous a prodigué, la patience, la confiance qu'il nous a témoignés ont été déterminants dans la réalisation de notre travail.

Nous tenons à remercier également tous nos enseignants pour la qualité de l'enseignement et le personnel de département d'Informatique de nous avoir appris à aimer le monde numérique et digital.

Table des matières

Table des figures

Liste des tableaux

CHAPITRE 1

Introduction générale

1.1 Context et objectifs

Au cours de ces dernières années, la programmation est devenue indispensable dans tous les domaines, grâce à l'utilisation de l'ordinateur qui rond automatique, rapide, facile la réalisation des différentes tâches. On trouve que l'utilisateur ne peut exprimer ses idées qu'en langage naturel, alors que l'ordinateur ne comprend que le langage machine.

Il est évident que pour pouvoir communiquer avec un ordinateur, il faut utiliser un langage compréhensible par les deux parties. Donc les langages de programmations ont été nés comme un moyen d'interaction entre la machine et les utilisateurs.

Les langages de programmation permettent aux programmeurs d'écrire leurs programmes avec une notation plus naturelle pour l'humain que le code machine exécuté par les processeurs. Les premiers langages de programmation développés n'offrent qu'une simple couche d'abstraction sur le code machine sous-jacent, ils permettent d'exprimer les instructions avec des codes symboliques et les adresses de mémoire avec des étiquettes. Les besoins des utilisateurs varient, évoluent, et les langages de programmation aussi, presque chaque jour des nouveaux langages sont crées et des nouvelles fonctionnalités sont ajoutés aux langages de programmation anciens.

L'exécution d'un programme écrit avec un langage de programmation ne peut pas être directement effectuée par le processeur. Une première solution à ce problème consiste à préalablement traduire le programme source en code machine à l'aide d'un compilateur [1]. La seconde solution consiste à décoder le programme source et à le traduire vers un autre langage de programmation afin de le compiler, càd., la compilataion source à source.

Dans ce mémoire, nous avons consacré à réaliser un compilateur source à source qui traduit un algorithme vers un programme équivalent dans le langage de programmation Pascal. L'objectif principal de notre projet est de présenter et d'implémenter les différentes étapes de la compilation, le choix de traduire un algorithme à pour but de faciliter la tâche aux étudiants à comprendre les principes et les notions d'algorithmique et d'apprendre la programmation en Pascal.

1.2 Organisation du mémoire

Pour arriver au but fini de notre projet, notre mémoire est composé de trois chapitres qui sont structurés comme suit :

- Le premier chapitre consiste à la présentation des langages de programmation et les concepts de base de la compilation.
- Le deuxième chapitre s'intéresse à la conception de notre travail, nous utilisons les différentes méthodes et techniques de réalisation d'un compilateur.
- Le dernier chapitre comprend la description de l'application développée. Un exemple d'exécution est donné pour illustrer les différentes caractéristiques de l'application.
- Finalement, on termine le mémoire par une conclusion qui résume notre travail et donne quelques améliorations futures.

CHAPITRE 2

Les langages de programmation et la compilation : vue générale

Sommaire

2.1 Introduction

Afin de réduire la complexité de la conception et de la construction d'ordinateurs, presque tous ça est fait pour exécuter des commandes relativement simples (mais le fait très rapide). Un programme pour un ordinateur doit être construit en combinant ces commandes simples dans un programme ce qu'on appelle le langage machine. Puisqu'il s'agit d'un processus ennuyeux et peuvent commettre des erreurs, la programmation est effectuée à l'aide d'un langage de haut niveau. Ce langage peut être très différent du langage machine que l'ordinateur peut exécuter, donc certains moyens de combler le fossé sont nécessaires. C'est là que le compilateur entre en jeu [1].

Dans ce chapitre, nous présentons une description des concepts des langages de programmations, des définitions plus précises du compilateur et leurs structures internes.

2.2 Les langages de programmation

Il existe différents types de langages informatiques et même différentes familles de langages informatiques. Ces langages mettent parfois en œuvre des concepts fort différents. On peut déjà définir des catégories en fonction de la « distance ». Un langage proche du langage machine sera dit de « bas niveau », un langage proche du langage humain sera dit de « haut niveau ». On peut distinguer les trois niveaux suivants [2] :

- Le langage machine.
- Le langage assembleur.
- Les langages évolués.

Et il y a aussi plusieurs catégories de langages évolués basées sur des approches conceptuelles différentes [2] :

2.2.1 La programmation impérative

Correspond aux programmes traditionnels séquentiels (ou procéduraux) qui sont des suites d'instructions manipulant des variables.

2.2.2 Les langages orientés objets

Ces langages sont devenus extrêmement populaires. Ce sont largement les plus utilisés. Le programme n'est plus vu comme l'exécution purement séquentielle d'une suite d'instruction se traduisant assez rapidement en langage proche de la machine mais comme la manipulation d'objets évolués.

2.2.3 Les langages fonctionnels

Il sont basés sur la manipulation d'expressions mathématiques uniquement. Il sont mis en œuvre par emboîtement d'appels de fonctions.

2.3 La compilation

La compilation est une transformation préservant l'équivalence d'un code écrit dans un Langage facilement compréhensible par l'humain (Langage de haut niveau) vers un langage machine (Langage de plus bas niveau) [3].
De façon très générale, la compilation d'un programme et son exécution peuvent être vues graphiquement sur le schéma suivant [4] [5].

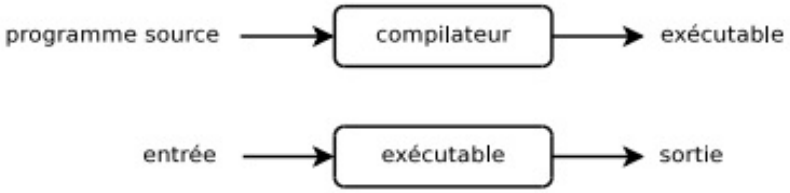


FIGURE 2.1 – Schéma du compilation

2.3.1 Type de compilateur

– Compilateur traditionnel : Le plus connu des types de compilateur est le compilateur de langage de programmation traditionnel. Celui-ci accepte un ou plusieurs fichiers de code source constituant un programme en entrée et produit, en sortie, un ou plusieurs fichiers binaires formés d'instructions pour un processeur. Ces fichiers binaires peuvent ensuite être exécutés sur l'ordinateur [6]. Ceci est illustré à la figure 2.2.

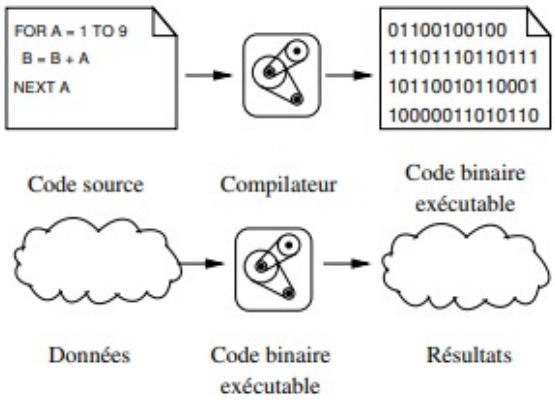


FIGURE 2.2 – Un compilateur traditionnel

– Compilateur source à source : Un compilateur qui prend le code source d'un langage de programmation et le traduit dans le code source d'un autre langage de programmation est appelé un compilateur source-à-source [7].

  Un exemple type est un compilateur qui lit une description de schéma de données en XML et qui produit un ensemble de classes (fichiers sources) en langage JAVA pour encapsuler l'accès à une base de données avec un modèle objet [8].

2.3.2 Les étapes de la compilation

La compilation se décompose en deux partie chaque partie comprends 3 phases [9] :

– Partie d'analyse : Cette partie permet de :
  – Partitionner le programme en ses constituants.

  - Elaborer la structure syntaxique du programme.
  - Ajouter quelques propriétés sémantiques.
- Partie de synthèse : Permet de créer un programme cible à partir de la représentation intermédiaire du programme source.

2.3.3 Gestion de la table de symboles

La table des symboles est la structure de données utilisée servant à stocker les informations qui concernent les entités lexicales du programme source [9]. Cette table est utilisée tout au long de la compilation, depuis son remplissage par l'analyseur lexical jusqu'à la génération du code [10].

2.3.4 Gestion des erreurs

La gestion des erreurs est une partie importante d'un compilateur. Une bonne gestion des erreurs permet de détecter un maximum d'erreurs et de déterminer le plus précisément possible la cause de l'erreur. Chaque phase détecte son propre type d'erreurs [11] :

- L'analyseur lexical détecte les caractères interdits et les mots non reconnus.
- L'analyseur syntaxique détecte les erreurs de construction du programme. Par exemple « ; » manquant à la fin d'une instruction Pascal.
- L'analyseur sémantique détecte les erreurs de type, les variables non déclarées,...etc.

Remarque : La plupart des compilateurs sont réalisés (écrits) dans un langage de haut niveau et non en assembleur Les avantages sont multiples :

- Facilité la manipulation de concepts avancés.
- Maintenabilité accrue du compilateur.
- Portage sur d'autres machines plus aisé.

Par exemple le compilateur C++ de Bjorne St roustrup est écrit en C il est même possible d'écrire un compilateur pour un langage L dans ce langage L (gcc est écrit en C)(bootst rap) [12].

2.4 Conclusion

Dans ce chapitre, nous avons présenté les langages de programmation comme un outil d'interaction entre l'utilisateur et l'ordinateur et nous avons présenté aussi les notions de base de réalisation d'un compilateur. Dans le chapitre suivant on va présenter la conception de notre compilateur avec les différentes techniques utilisées.

CHAPITRE 3

La conception de notre compilateur $ALgo\_Comp$

Sommaire

3.1 Introduction

La compilation est la traduction automatique d'une description écrite dans un langage vers un autre langage. Très souvent, ce terme est restreint à la traduction d'un langage de programmation de haut niveau tel que le C vers le langage machine du processeur qui devra exécuter le programme [13]. Dans ce chapitre, nous présentons notre compilateur qui traite les instructions écrites dans un algorithme donné pour les traduire en Pascal. Le but de ce chapitre est de présenter les étapes de notre compilation de notre mini-langage ALgorithme_Compiler ($ALgo\_Comp$) : (i) analyseur lexical, (ii)analyseur syntaxique et (iii) analyseur sémantique.

3.2 L'analyse lexicale de $ALgo\_Comp$

Le flot de caractères formant le programme source est lu de gauche à droite et groupé en unités lexicales, qui sont des suites de caractères ayant une signification collective. Les identificateurs, les mots-clés, les constantes et les opérateurs sont des unités lexicales, les commentaires et les blancs séparant les caractères formant les unités lexicales sont éliminés au cours de cette phase.

3.2.1 Les unités lexicales

Les identificateurs : Un identificateur est un mot composé de lettres, de chiffres et du caractère "_" et commençant obligatoirement par une lettre. la longueur max d'un identificateur est 32 caractères.
Les mots réservés : algorithme , variable, var, de, à, cas, parmi, constante, entier, réel, booléenne, et, ou, non, vrai, faux, début , lire, écrire, si, sinon, alors, tq, pour, jusquà, répéter, faire, fin, lireln, écrireln, enregistrement, chaine, caractère, type, tableau, fonction, procédure.
Les opérateurs arithmétiques : "+" "-" "$\times$" "$\div$"
Les opérateurs relationnelles : " $<$" " $<=$" " $>$" " $>=$" "=" " $\neq$"
Chaine de caractère : une chaine de caractère est une séquence de caractères quelconques entre deux apostrophes, si on veut que l'apostrophe apparaisse dans la chaine, il faut la doubler, par exemple : 'ceci est une chaine 1200AGF sd3 !'.
L'opérateur d'affection : $\leftarrow$
Les séparateurs : "," "." " :" "(" ")" " ;" "[" "]" "^"

Une fonction importante de cette phase est d'enregistrer tous les entités lexicales dans la table des symboles. toutes les autres phases de la compilation sont en relation permanente avec cette table.

3.2.2 L'automate union déterministe de $ALgo\_Comp$

La Figure 3.1 illustre la représentation graphique d'automate union déterministe de notre compilateur et la Table 3.1 donne les transitions du même automate.
Remarque :
Ch est l'ensemble de chiffres {0,1,...,9}.
L est l'ensemble de lettres {a,...,z,A,...Z}.
Char : est l'ensemble de tous les caractères.

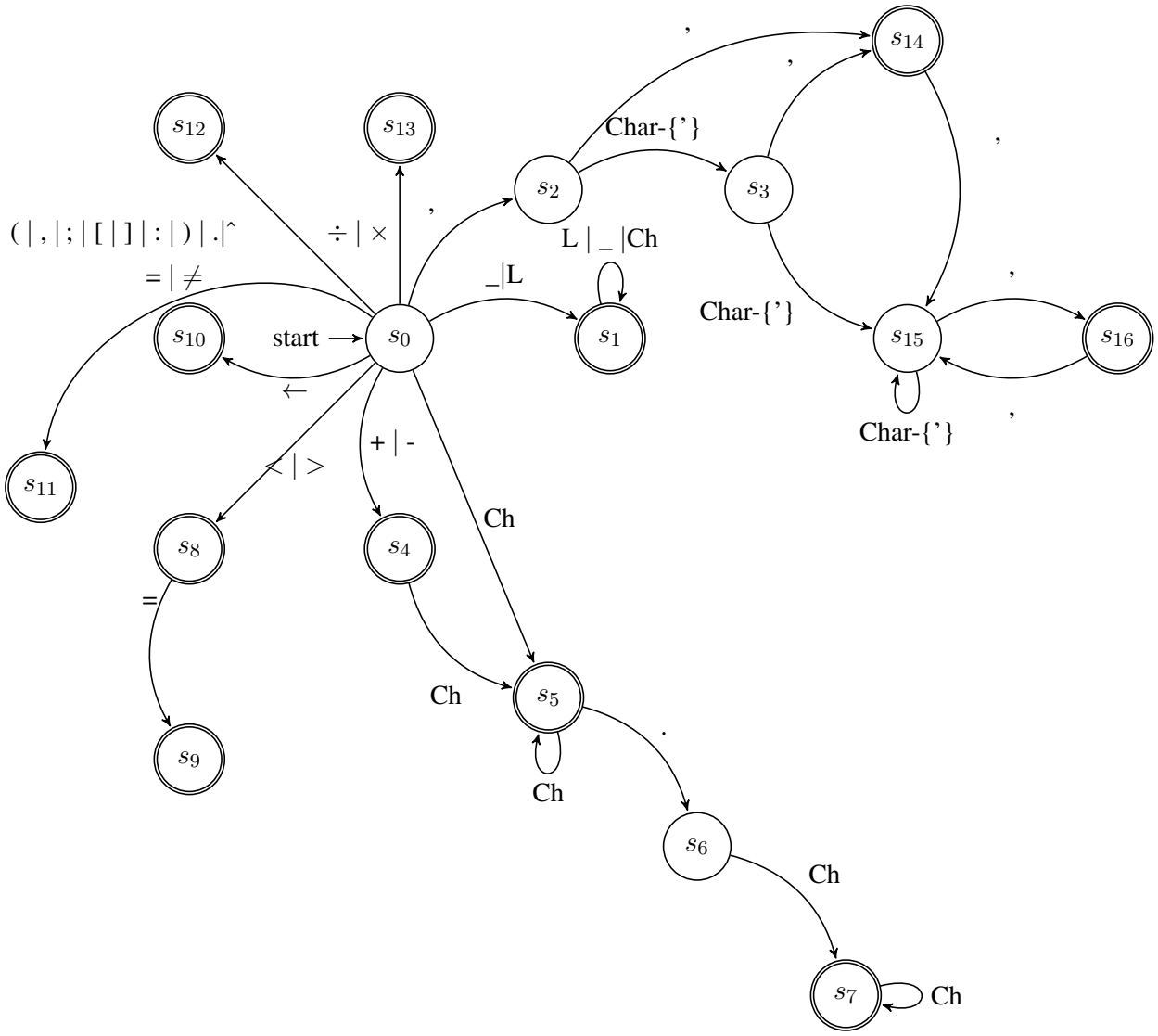


FIGURE 3.1 – L'Automate Union Déterministe de $ALgo\_Comp$

S \ C	‘	×	÷	.	–	=	L	ch	(	)	:	;	,	+	-	Char-{’}	>	<	←	[	]	≠	^
s_0	s_2	s_{12}	s_{13}	s_{12}	s_1	s_{11}	s_1	s_5	s_{12}	s_{12}	s_{12}	s_{12}	s_{12}	s_4	s_4		s_8	s_8	s_8	s_{12}	s_{12}	s_{11}	s_{12}
s_1					s_1		s_1	s_1															
s_2	s_{14}	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_3
s_3	s_{14}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}
s_4								s_5															
s_5				s_6				s_5															
s_6								s_7															
s_7								s_7															
s_8						s_9																	
s_9																							
s_{10}																							
s_{11}																							
s_{12}																							
s_{13}																							
s_{14}	s_{15}																						
s_{15}	s_{16}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}	s_{15}
s_{16}	s_{15}																						

TABLE 3.1 – Table de transition de $ALgo\_Comp$

Remarque : Toute case vide dans la table contient une erreur.

3.3 L'analyse syntaxique de $ALgo\_Comp$

Cette phase consiste à regrouper les unités lexicales du programme source en structures grammaticales (i.e. vérifie si un programme est correctement écrit selon la grammaire qui spécifie la structure syntaxique du langage). En général, cette analyse est représentée par un arbre.

Il existe deux catégories d'analyse qui sont l'analyse descendante et l'analyse ascendante, le principe générale de l'analyse descendante est de construire l'arbre de dérivation à partir de l'axiome vers les unités lexicales. Par contre avec une analyse ascendante on essaye de lire les entités du texte source et par une série de réductions remonter à l'axiome de la grammaire du langage. Il existe plus qu'une méthode pour cette analyse comme : SLR, LR(0), LR (1) et la méthode LALR.

L'analyse descendante est une méthode simple à implémenter et efficace mais il existe un ensemble minimal de grammaire qui accepte cette dernière [9], c'est pourquoi nous avons choisi la méthode la plus fiable parmi les méthodes ascendantes, nous avons utilisé l'analyse LR(1).

3.3.1 La grammaire de $ALgo\_Comp$

<ALGO1>	$\longrightarrow$	<ALGO>
<ALGO>	$\longrightarrow$	algorithme id ; <NVTYPE> <CONST> <VAR> <ACTION> debut <CODE> fin
<NVTYPE>	$\longrightarrow$	type id = <T> | ϵ
<T>	$\longrightarrow$	<T1> <NVTYPE>
<T1>	$\longrightarrow$	<TABLEAU> | enregistement <CHAMPS>
<CHAMPS>	$\longrightarrow$	id : <TYPE> <CH> fin ;
<CH>	$\longrightarrow$	<CHAMPS> | ϵ
<TABLEAU>	$\longrightarrow$	tableau [int .. int <DIM>] de <TYPETAB> ;
<DIM>	$\longrightarrow$	,int ..int | ϵ
<TYPETAB>	$\longrightarrow$	entier | réel | booléenne | chaine | caractère | id
<TYPE>	$\longrightarrow$	<TYPETAB> ; | ^id ; |<TABLEAU>
<CONST>	$\longrightarrow$	constante <C> | ϵ
<C>	$\longrightarrow$	id opaff <VALEUR> ; C1
<C1>	$\longrightarrow$	<C> | ϵ
<VALEUR>	$\longrightarrow$	int | real | char | string | vrai | faux
<VAR>	$\longrightarrow$	variable <V> | ϵ
<V>	$\longrightarrow$	<ID> : <TYPEVAR> ; <V1>
<V1>	$\longrightarrow$	<V> | ϵ
<TYPEVAR>	$\longrightarrow$	<TABLEAU> | <TYPETAB>
<ID>	$\longrightarrow$	id <D>
<D>	$\longrightarrow$	,<ID> | ϵ
<ACTION>	$\longrightarrow$	<FONCTION> | <PROCEDURE> | ϵ
<FONCTION>	$\longrightarrow$	fonction id (<ENTRER>) : <TYPETAB> ; <CONST> <VAR> debut <CODE> fin ; <ACTION>
<PROCEDURE>	$\longrightarrow$	procédure id (<ENTRER> <SORTIE> <SORTIE>) ; <CONST> <VAR> debut <CODE> fin ; <ACTION>
<ENTRER>	$\longrightarrow$	<ID> : <TYPETAB> ; <ENTRER> | ϵ
<SORTIE>	$\longrightarrow$	var <ENTRER> | ϵ
<CODE>	$\longrightarrow$	<INSTRUCTION> <CODE1>
<CODE1>	$\longrightarrow$	<CODE> | ϵ
<INSTRUCTION>	$\longrightarrow$	<AFFECTATION> | <BOUCLE> | <CONDITION>|CASE> | <LIRE> | <ECRIRE> | <APPEL_P>

<AFFECTATION>	$\longrightarrow$	id <AFF> opaff <EXPRE> ;
<EXPRE>	$\longrightarrow$	<EXPRESSION> | vrai | faux | string | char
<AFF>	$\longrightarrow$	<TAB> | .id <TAB> | ^ id
<TAB>	$\longrightarrow$	[<EXPRESSION> <EXP>] | ϵ
<EXP>	$\longrightarrow$	,<EXPRESSION> | ϵ
<APPEL_P>	$\longrightarrow$	id (<E>) ;
<E>	$\longrightarrow$	<EXPRE> <E1> | ϵ
<E1>	$\longrightarrow$	,<EXPRE> <E1> | ϵ
<BOUCLE>	$\longrightarrow$	pour id opaff <EXPRESSION> à <EXPRESSION> faire debut <CODE1> fin ; | tq (<COND>) faire debut <CODE1> fin ; | répéter debut <CODE1> fin ; jusquà (<COND>) ;
<CONDITION>	$\longrightarrow$	si (<COND>) alors debut <CODE1> fin ; <SINON>
<SINON>	$\longrightarrow$	sinon debut <CODE1> fin ; | ϵ
<COND>	$\longrightarrow$	<COND> ou <COND1> | <COND1>
<COND1>	$\longrightarrow$	<COND1> et <COND2> | <COND2>
<COND2>	$\longrightarrow$	non <COND2> | (<COND>) | <EX>
<EX>	$\longrightarrow$	<EXPRESSION> oprel <EXPRESSION> | vrai | faux | id <K>
<CASE>	$\longrightarrow$	cas (id) parmi <CAS> sinon debut <CODE> fin ;
<CAS>	$\longrightarrow$	<X> : debut <CODE> fin ; <CAS1>
<X>	$\longrightarrow$	<VALEUR><VAL>
<VAL>	$\longrightarrow$	, <X> | ϵ
<CAS1>	$\longrightarrow$	<CAS> | ϵ
<LIRE>	$\longrightarrow$	lire (id <AFF> <L>) ; | lireln (id <AFF> <L>) ;
<L>	$\longrightarrow$	,id <AFF> <L> | ϵ
<ECRIRE>	$\longrightarrow$	écrire (<EXPRE> <S>) ; | écrireln (<EXPRE> <S>) ;
<S>	$\longrightarrow$	,<EXPRE> <S> | ϵ
<K>	$\longrightarrow$	(<E>) | <AFF>
<EXPRESSION>	$\longrightarrow$	<EXPRESSION> + <A> | <EXPRESSION> - <A> | <A>
<A>	$\longrightarrow$	<A> $\div$ <B> | <A> $\times$ <B> | <B>
<B>	$\longrightarrow$	int | real | id<K> | (<EXPRESSION>)

3.3.2 L'ensemble des items LR(1) de $ALgo\_Comp$

$N°$	GOTO	Ensemble d'item
$\mathbb{I}_0$	Ferm(ALGO1 ⟶.ALGO,#)	{[<ALGO1> ⟶.<ALGO>,#], [<ALGO> ⟶.algorithme id ; <NVTYPE> <CONST> <VAR> <ACTION> debut <CODE> fin., #]}
$\mathbb{I}_1$	GOTO($\mathbb{I}_0$, ALGO)	{[<ALGO1> ⟶<ALGO>.,#]}
$\mathbb{I}_2$	GOTO ($\mathbb{I}_0$, algorithme)	{[<ALGO>⟶algorithme.id ;<NVTYPE><CONST> <VAR><ACTION>debut<CODE>fin., #]}
$\mathbb{I}_3$	GOTO ($\mathbb{I}_2$, id)	{[<ALGO> ⟶ algorithme id . ; <NVTYPE> <CONST> <VAR> <ACTION> debut <CODE> fin., #]}
$\mathbb{I}_4$	GOTO ($\mathbb{I}_3$, ;)	{[<ALGO> ⟶ algorithme id ;. <NVTYPE> <CONST> <VAR> <ACTION> debut <CODE> fin., #], [<NVTYPE>⟶.type id=<T>,constante,variable,fonction, procédure, debut], [<NVTYPE>⟶.,constante,variable,fonction,procédure, debut] }
$\mathbb{I}_5$	GOTO ($\mathbb{I}_4$, NVTYPE)	{[<ALGO> ⟶ algorithme id ; <NVTYPE>. <CONST> <VAR> <ACTION> debut <CODE> fin., #] , [<CONST>⟶.constante <C>,variable,fonction,procédure, debut], [<CONST>⟶.,variable,fonction,procédure, debut]}
$\mathbb{I}_6$	GOTO ($\mathbb{I}_4$, type)	{ [<NVTYPE>⟶type .id=<T>,constante, variable, fonction, procédure, debut]}
$\mathbb{I}_7$	GOTO ($\mathbb{I}_5$, CONST)	{ [<ALGO> ⟶ algorithme id ; <NVTYPE> <CONST>. <VAR> <ACTION> debut <CODE> fin., #], [<VAR> ⟶.variable <V>,fonction,procédure, debut], [<VAR>⟶.,fonction,procédure,debut] }
$\mathbb{I}_8$	GOTO ($\mathbb{I}_5$, constante)	{[<CONST>⟶constante.<C>,variable,fonction, procédure,debut], [<C>⟶.id opaff<VALEUR> ;<C1>,variable,fonction, procédure,debut] }
$\mathbb{I}_9$	GOTO ($\mathbb{I}_6$, id)	{[<NVTYPE>⟶type id.=<T>,constante, variable, fonction,procédure,debut]}
$\mathbb{I}_{10}$	GOTO ($\mathbb{I}_7$, VAR)	{[<ALGO>⟶algorithme id ;<NVTYPE><CONST> <VAR>.<ACTION>debut<CODE>fin.,#], [<ACTION>⟶ .<FONCTION>,debut], [<ACTION>⟶ .<PROCEDURE>,debut], [<ACTION>⟶ .,debut], [<FONCTION>⟶.fonction id(<ENTRER>) : <TYPETAB> ;<CONST><VAR>debut<CODE>fin ; <ACTION>,debut], {[<PROCEDURE>⟶.procédure id(<ENTRER> <SORTIE> <SORTIE>) ;<CONST><VAR>debut <CODE>fin ;<ACTION>,debut]
$\mathbb{I}_{11}$	GOTO ($\mathbb{I}_7$, variable)	{[<VAR>⟶variable.<V>,fonction,procédure, debut], [<V>⟶.<ID> :<TYPEVAR> ;<V1>,fonction,procédure, debut], [<ID>⟶.id<D>, :] }
$\mathbb{I}_{12}$	GOTO ($\mathbb{I}_8$, C)	{[<CONST>⟶constante<C>.,variable,fonction, procédure,debut] }
$\mathbb{I}_{13}$	GOTO ($\mathbb{I}_8$, id)	{[<C>⟶id.opaff<VALEUR> ;<C1>,variable,fonction, procédure,debut] }

$N°$	GOTO	Ensemble d'item
$\mathbb{I}_{14}$	GOTO ($\mathbb{I}_9$, =)	{[<NVTYPE>⟶type id=.<T>,constante, variable, fonction,procédure,debut], [<T>⟶.<T1><NVTYPE>,constante,variable,fonction, procédure,debut], [<T1>⟶.<TABLEAU>,type,constante,variable,fonction, procédure,debut], [<T1>⟶.enregistrement<CHAMPS>,type,constante, variable,fonction,procédure,debut], [<TABLEAU>⟶.tableau[int..int<DIM>]de<TYPETAB> ; ,type,constante,variable,fonction,procédure,debut] }
$\mathbb{I}_{15}$	GOTO ($\mathbb{I}_{10}$, ACTION)	{[<ALGO>⟶algorithme id ;<NVTYPE><CONST> <VAR><ACTION>.debut<CODE>fin.,#] }
$\mathbb{I}_{16}$	GOTO ($\mathbb{I}_{10}$, FONCTION)	{[<ACTION>⟶<FONCTION>.,debut] }
$\mathbb{I}_{17}$	GOTO ($\mathbb{I}_{10}$, PROCEDURE)	{[<ACTION>⟶<PROCEDURE>.,debut] }
$\mathbb{I}_{18}$	GOTO ($\mathbb{I}_{10}$, fonction)	{[<FONCTION>⟶fonction.id(<ENTRER>) : <TYPETAB> ;<CONST><VAR>debut<CODE>fin ; <ACTION>,debut] }
$\mathbb{I}_{19}$	GOTO ($\mathbb{I}_{10}$, procedure)	{[<PROCEDURE>⟶procédure.id(<ENTRER> <SORTIE><SORTIE>) ;<CONST><VAR>debut <CODE>fin ;<ACTION>,debut] }
$\mathbb{I}_{20}$	GOTO ($\mathbb{I}_{11}$, V)	{[<VAR>⟶variable<V>.,fonction,procédure, debut] }
$\mathbb{I}_{21}$	GOTO ($\mathbb{I}_{11}$, ID)	{[<V>⟶<ID>. :<TYPEVAR> ;<V1>,fonction,procédure, debut] }
$\mathbb{I}_{22}$	GOTO ($\mathbb{I}_{11}$, id)	{[<ID>⟶id.<D>, :], [<D>⟶.,<ID>, :], [<D>⟶., :] }
$\mathbb{I}_{23}$	GOTO ($\mathbb{I}_{13}$, opaff)	{[<C>⟶id opaff.<VALEUR> ;<C1>,variable,fonction, procédure,debut], [<VALEUR>⟶.int, ;], [<VALEUR>⟶.real, ;], [<VALEUR>⟶.string, ;], [<VALEUR>⟶.char, ;], [<VALEUR>⟶.vrai, ;], [<VALEUR>⟶.faux, ;] }
$\mathbb{I}_{24}$	GOTO ($\mathbb{I}_{14}$, T)	{[<NVTYPE>⟶type id=<T>.,constante, variable, fonction,procédure,debut] }
$\mathbb{I}_{25}$	GOTO ($\mathbb{I}_{14}$, T1)	{[<T>⟶<T1>.<NVTYPE>,constante,variable,fonction, procédure,debut], [<NVTYPE>⟶.type id=<T>,constante,variable,fonction, procédure,debut], [<NVTYPE>⟶.,constante,variable,fonction,procédure, debut]}
$\mathbb{I}_{26}$	GOTO ($\mathbb{I}_{14}$, TABLEAU)	{[<T1>⟶<TABLEAU>.,type,constante,variable,fonction, procédure,debut] }
$\mathbb{I}_{27}$	GOTO ($\mathbb{I}_{14}$, tableau)	{[<TABLEAU>⟶tableau.[int..int<DIM>]de<TYPETAB> ; ,type,constante,variable,fonction,procédure,debut] }
$\mathbb{I}_{28}$	GOTO($\mathbb{I}_{14}$,enregistrement)	{[<T1>⟶enregistrement.<CHAMPS>,type,constante, variable,fonction,procédure,debut], [<CHAMPS>⟶.id :<TYPE> ;<CH>fin ;,type,constante, variable,fonction,procédure,debut] }

$N°$	GOTO	Ensemble d'item
$\mathbb{I}_{29}$	GOTO ($\mathbb{I}_{15}$, debut)	{[<ALGO>⟶algorithme id ;<NVTYPE><CONST> <VAR><ACTION>debut.<CODE>fin.,#], [<CODE>⟶.<INSTRUCTION><CODE1>,fin] , [<INSTRUCTION>⟶.<AFFECTATION>,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<INSTRUCTION>⟶.<BOUCLE>,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<INSTRUCTION>⟶.<CONDITION>,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<INSTRUCTION>⟶.<CASE>,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<INSTRUCTION>⟶.<LIRE>,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<INSTRUCTION>⟶.<ECRIRE>,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<INSTRUCTION>⟶.<APPEL_P>,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<AFFECTATION>⟶.id<AFF>opaff<EXPRE> ;,fin,id, pour,tq,répéter,si,cas,lire,lireln,écrire,écrireln], [<BOUCLE>⟶.pour id opaff<EXPRESSION>à <EXPRESSION>faire debut<CODE1>fin ;,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<BOUCLE>⟶.tq(<COND>)faire debut<CODE1>fin ; ,fin, id,pour,tq,répéter,si, cas,lire,lireln,écrire,écrireln], [<BOUCLE>⟶.répéter debut <CODE1> fin ;jusquà (<COND>) ;,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire, écrireln] [<CONDITION>⟶.si(<COND>)alors debut <CODE1> fin ; <SINON>, fin,id,pour, tq,répéter,si,cas,lire,lireln, écrire, écrireln] [<CASE>⟶.cas(id)parmi<CAS>sinon debut<CODE> fin ;,fin,id,pour,tq,répéter,si,cas,lire,lireln,écrire, écrireln] [<LIRE>⟶.lire (id<AFF><L>) ; ,fin,id,pour,tq,répéter, si,cas,lire,lireln,écrire,écrireln] [<LIRE>⟶.lireln(id<AFF><L>) ;,fin,id,pour,tq,répéter, si,cas,lire,lireln,écrire,écrireln] [<ECRIRE>⟶.écrire(<EXPRE><S>) ;,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] [<ECRIRE>⟶.écrireln(<EXPRE><S>) ;,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] [<APPEL_P>⟶.id(<E>) ;,fin,id,pour,tq,répéter,si, cas,lire,lireln,écrire,écrireln] }
$\mathbb{I}_{30}$	GOTO ($\mathbb{I}_{18}$, id)	{[<FONCTION>⟶fonction id.(<ENTRER>) : <TYPETAB> ;<CONST><VAR>debut<CODE>fin ; <ACTION>,debut] }
$\mathbb{I}_{31}$	GOTO ($\mathbb{I}_{19}$, id)	{[<PROCEDURE>⟶procédure id.(<ENTRER> <SORTIE><SORTIE>) ;<CONST><VAR>debut <CODE>fin ;<ACTION>,debut] }

$N°$	GOTO	Ensemble d'item
$\mathbb{I}_{32}$	GOTO ($\mathbb{I}_{21}$, :)	{[<V>⟶<ID> :.<TYPEVAR> ;<V1>,fonction,procédure, debut], [<TYPEVAR>⟶.<TABLEAU>, ;], [<TYPEVAR>⟶.<TYPETAB>, ;],[<TYPETAB>⟶.id, ;] [<TABLEAU>⟶.tableau[int..int<DIM>]de <TYPETAB> ;, ;], [<TYPETAB>⟶.entier, ;], [<TYPETAB>⟶.réel, ;], [<TYPETAB>⟶.booléenne, ;], [<TYPETAB>⟶.chaine, ;],[<TYPETAB>⟶.caractère, ;]}
$\mathbb{I}_{33}$	GOTO ($\mathbb{I}_{22}$, D)	{[<ID>⟶id<D>., :] }
$\mathbb{I}_{34}$	GOTO ($\mathbb{I}_{22}$, ,)	{[<D>⟶,.<ID>, :],[<ID>⟶.id<D>, :] }
$\mathbb{I}_{35}$	GOTO ($\mathbb{I}_{23}$, VALEUR)	{[<C>⟶id opaff<VALEUR>. ;<C1>,variable,fonction, procédure,debut] }
$\mathbb{I}_{36}$	GOTO ($\mathbb{I}_{23}$, int)	{[<VALEUR>⟶int., ;] }
$\mathbb{I}_{37}$	GOTO ($\mathbb{I}_{23}$, real)	{[<VALEUR>⟶real., ;] }
$\mathbb{I}_{38}$	GOTO ($\mathbb{I}_{23}$, string)	{[<VALEUR>⟶string., ;] }
$\mathbb{I}_{39}$	GOTO ($\mathbb{I}_{23}$, char)	{[<VALEUR>⟶char., ;] }
$\mathbb{I}_{40}$	GOTO ($\mathbb{I}_{23}$, vrai)	{[<VALEUR>⟶vrai., ;] }
$\mathbb{I}_{41}$	GOTO ($\mathbb{I}_{23}$, faux)	{[<VALEUR>⟶faux., ;] }
$\mathbb{I}_{42}$	GOTO ($\mathbb{I}_{25}$, NVTYPE)	{[<T>⟶<T1><NVTYPE>.,constante,variable,fonction, procédure,debut] }
$\mathbb{I}_{6}$	GOTO ($\mathbb{I}_{25}$, type)	
$\mathbb{I}_{43}$	GOTO ($\mathbb{I}_{27}$, [ )	{[<TABLEAU>⟶tableau[.int..int<DIM>]de<TYPETAB> ; ,type,constante,variable,fonction,procédure,debut] }
$\mathbb{I}_{44}$	GOTO ($\mathbb{I}_{28}$, CHAMPS)	{[<T1>⟶enregistrement<CHAMPS>.,type,constante, variable,fonction,procédure,debut] }
$\mathbb{I}_{45}$	GOTO ($\mathbb{I}_{28}$, id)	{[<CHAMPS>⟶id. :<TYPE><CH>fin ;,type,constante, variable,fonction,procédure,debut] }
$\mathbb{I}_{46}$	GOTO ($\mathbb{I}_{29}$, CODE)	{[<ALGO>⟶algorithme id ;<NVTYPE><CONST> <VAR><ACTION>debut<CODE>.fin.,#] }

$N°$	GOTO	Ensemble d'item
$\mathbb{I}_{47}$	GOTO ($\mathbb{I}_{29}$, INSTRUCTION)	{[<CODE>⟶<INSTRUCTION>.<CODE1>,fin] , [<CODE1>⟶.<CODE>,fin], [<CODE1>⟶.,fin], [<CODE>⟶.<INSTRUCTION><CODE1>,fin] , [<INSTRUCTION>⟶.<AFFECTATION>,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<INSTRUCTION>⟶.<BOUCLE>,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<INSTRUCTION>⟶.<CONDITION>,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<INSTRUCTION>⟶.<CASE>,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<INSTRUCTION>⟶.<LIRE>,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<INSTRUCTION>⟶.<ECRIRE>,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<INSTRUCTION>⟶.<APPEL_P>,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<AFFECTATION>⟶.id<AFF>opaff<EXPRE> ;,fin,id, pour,tq,répéter,si,cas,lire,lireln,écrire,écrireln], [<BOUCLE>⟶.pour id opaff<EXPRESSION>à <EXPRESSION>faire debut<CODE1>fin ;,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<BOUCLE>⟶.tq(<COND>)faire debut<CODE1>fin ;,fin,id, pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln], [<BOUCLE>⟶.répéter debut<CODE1>fin ;jusquà(<COND>) ;,fin, id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] [<CONDITION>⟶.si(<COND>)alors debut<CODE1>fin ;<SINON> ,fin, id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] [<CASE>⟶.cas(id)parmi<CAS>sinon debut<CODE>fin ;,fin ,id,pour,tq,répéter,si,cas,lire,lireln,écrire,écrireln] [<LIRE>⟶.lire (id<AFF><L>) ; ,fin,id,pour,tq,répéter, si,cas,lire,lireln,écrire,écrireln] [<LIRE>⟶.lireln(id<AFF><L>) ;,fin,id,pour,tq,répéter, si,cas,lire,lireln,écrire,écrireln] [<ECRIRE>⟶.écrire(<EXPRE><S>) ;,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] [<ECRIRE>⟶.écrireln(<EXPRE><S>) ;,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] [<APPEL_P>⟶.id(<E>) ;,fin,id,pour,tq,répéter,si, cas,lire,lireln,écrire,écrireln] }
$\mathbb{I}_{48}$	GOTO ($\mathbb{I}_{29}$, AFFECTATION)	{[<INSTRUCTION>⟶<AFFECTATION>.,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] }
$\mathbb{I}_{49}$	GOTO ($\mathbb{I}_{29}$, BOUCLE)	{[<INSTRUCTION>⟶<BOUCLE>.,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] }
$\mathbb{I}_{50}$	GOTO ($\mathbb{I}_{29}$, CONDITION)	{[<INSTRUCTION>⟶<CONDITION>.,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] }
$\mathbb{I}_{51}$	GOTO ($\mathbb{I}_{29}$, CASE)	{[<INSTRUCTION>⟶<CASE>.,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] }
$\mathbb{I}_{52}$	GOTO ($\mathbb{I}_{29}$, LIRE)	{[<INSTRUCTION>⟶<LIRE>.,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] }
$\mathbb{I}_{53}$	GOTO ($\mathbb{I}_{29}$, ECRIRE)	{[<INSTRUCTION>⟶<ECRIRE>.,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] }
$\mathbb{I}_{54}$	GOTO ($\mathbb{I}_{29}$, APPEL_P)	{[<INSTRUCTION>⟶<APPEL_P>.,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] }

$N°$	GOTO	Ensemble d'item
$\mathbb{I}_{55}$	GOTO ($\mathbb{I}_{29}$, id)	{[<AFFECTATION>⟶id.<AFF>opaff<EXPRE> ;,fin,id, pour,tq,répéter,si,cas,lire,lireln,écrire,écrireln], [<APPEL_P>⟶id.(<E>) ;,fin,id,pour,tq,répéter,si, cas,lire,lireln,écrire,écrireln] [<AFF>⟶.<TAB>,opaff] [<AFF>⟶..id<TAB>,opaff] [<AFF>⟶.^id,opaff] [<TAB>⟶.[<EXPRESSION><EXP>],opaff] [<TAB>⟶.,opaff] }
$\mathbb{I}_{56}$	GOTO ($\mathbb{I}_{29}$, pour)	{ [<BOUCLE>⟶pour.id opaff<EXPRESSION>à <EXPRESSION>faire debut<CODE1>fin ;,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] }
$\mathbb{I}_{57}$	GOTO ($\mathbb{I}_{29}$, tq)	{[<BOUCLE>⟶tq.(<COND>)faire debut<CODE1>fin ;,fin,id, pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] }
$\mathbb{I}_{58}$	GOTO ($\mathbb{I}_{29}$, répéter)	{[<BOUCLE>⟶répéter.debut <CODE1>fin ;jusquà(<COND>) ;,fin, id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln]
$\mathbb{I}_{59}$	GOTO ($\mathbb{I}_{29}$, si)	{[<CONDITION>⟶si.(<COND>)alors debut<CODE1>fin ;<SINON> ,fin, id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] }
$\mathbb{I}_{60}$	GOTO ($\mathbb{I}_{29}$, cas)	{[<CASE>⟶cas.(id)parmi<CAS>sinon debut<CODE>fin ;,fin ,id,pour,tq,répéter,si,cas,lire,lireln,écrire,écrireln] }
$\mathbb{I}_{61}$	GOTO ($\mathbb{I}_{29}$, lire)	{[<LIRE>⟶lire.(id<AFF><L>) ;,fin,id,pour,tq,répéter, si,cas,lire,lireln,écrire,écrireln] }
$\mathbb{I}_{62}$	GOTO ($\mathbb{I}_{29}$, lireln)	{[<LIRE>⟶lireln.(id<AFF><L>) ;,fin,id,pour,tq,répéter, si,cas,lire,lireln,écrire,écrireln] }
$\mathbb{I}_{63}$	GOTO ($\mathbb{I}_{29}$, écrire)	{[<ECRIRE>⟶écrire.(<EXPRE><S>) ;,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] }
$\mathbb{I}_{64}$	GOTO ($\mathbb{I}_{29}$, écrireln)	{[<ECRIRE>⟶écrireln.(<EXPRE><S>) ;,fin,id,pour,tq, répéter,si,cas,lire,lireln,écrire,écrireln] }
$\mathbb{I}_{65}$	GOTO ($\mathbb{I}_{30}$, ()	{[<FONCTION>⟶fonction id(.<ENTRER>) : <TYPETAB> ;<CONST><VAR>debut<CODE>fin ; <ACTION>,debut], [<ENTRER>⟶.<ID> :<TYPETAB> ;<ENTRER>,)], [<ENTRER>⟶.,)], [<ID>⟶.id<D>, :] }
$\mathbb{I}_{66}$	GOTO ($\mathbb{I}_{31}$, ()	{[<PROCEDURE>⟶procédure id(.<ENTRER> <SORTIE><SORTIE>) ;<CONST><VAR>debut <CODE>fin ;<ACTION>,debut], [<ENTRER>⟶.<ID> :<TYPETAB> ;<ENTRER>,),var], [<ENTRER>⟶.,),var], [<ID>⟶.id<D>, :] }

$N°$	GOTO	Ensemble d'item
$\mathbb{I}_{67}$	GOTO ($\mathbb{I}_{32}$, TYPEVAR)	{[<V>⟶<ID> :<TYPEVAR>. ;<V1>,fonction,procédure, debut] }
$\mathbb{I}_{68}$	GOTO ($\mathbb{I}_{32}$, TABLEAU)	{[<TYPEVAR>⟶<TABLEAU>., ;] }
$\mathbb{I}_{69}$	GOTO ($\mathbb{I}_{32}$, TYPETAB)	{[<TYPEVAR>⟶<TYPETAB>., ;] }
$\mathbb{I}_{70}$	GOTO ($\mathbb{I}_{32}$, tableau)	{[<TABLEAU>⟶tableau.[int..int<DIM>]de <TYPETAB> ;, ;] }
$\mathbb{I}_{71}$	GOTO ($\mathbb{I}_{32}$, entier)	{ [<TYPETAB>⟶entier., ;] }
$\mathbb{I}_{72}$	GOTO ($\mathbb{I}_{32}$, réel)	{ [<TYPETAB>⟶réel., ;] }
$\mathbb{I}_{73}$	GOTO ($\mathbb{I}_{32}$, chaine)	{ [<TYPETAB>⟶chaine., ;] }
$\mathbb{I}_{74}$	GOTO ($\mathbb{I}_{32}$, booléenne)	{ [<TYPETAB>⟶booléenne., ;] }
$\mathbb{I}_{75}$	GOTO ($\mathbb{I}_{32}$, caractère)	{ [<TYPETAB>⟶caractère., ;] }
$\mathbb{I}_{76}$	GOTO ($\mathbb{I}_{32}$, id)	{ [<TYPETAB>⟶id., ;] }
$\mathbb{I}_{77}$	GOTO ($\mathbb{I}_{34}$, ID)	{ [<D>⟶,<ID>., :] }
$\mathbb{I}_{22}$	GOTO ($\mathbb{I}_{34}$, id)	
$\mathbb{I}_{78}$	GOTO ($\mathbb{I}_{35}$, ;)	{ [<C>⟶id opaff<VALEUR> ;.<C1>,variable,fonction, procédure,debut],[<C1>⟶.<C>,variable,fonction, debut, procédure],[<C1>⟶.,variable,fonction,procédure,debut], [<C>⟶.id opaff<VALEUR> ;<C1>,variable,fonction, procédure,debut]}
$\mathbb{I}_{79}$	GOTO ($\mathbb{I}_{43}$, int)	{[<TABLEAU>⟶tableau[int..int<DIM>]de<TYPETAB> ; ,type,constante,variable,fonction,procédure,debut] }
$\mathbb{I}_{80}$	GOTO ($\mathbb{I}_{45}$, :)	{ [<CHAMPS>⟶id :.<TYPE><CH>fin ;,type,constante, variable,fonction,procédure,debut], [<TYPE>⟶.^id ;, ;], [<TYPE>⟶.<TYPETAB> ;, ;], [<TYPE>⟶.<TABLEAU> ;, ;], [<TABLEAU>⟶.tableau[int..int<DIM>]de<TYPETAB> ;, ;],[<TYPETAB>⟶.id, ;], [<TYPETAB>⟶.entier, ;], [<TYPETAB>⟶.réel, ;], [<TYPETAB>⟶.booléenne, ;], [<TYPETAB>⟶.chaine, ;],[<TYPETAB>⟶.caractère, ;]}
$\mathbb{I}_{81}$	GOTO ($\mathbb{I}_{46}$, fin)	{[<ALGO>⟶algorithme id ;<NVTYPE><CONST> <VAR><ACTION>debut<CODE>fin.,#] }
$\mathbb{I}_{82}$	GOTO ($\mathbb{I}_{47}$, CODE1)	{ [<CODE>⟶<INSTRUCTION><CODE1>.,fin] }
$\mathbb{I}_{83}$	GOTO ($\mathbb{I}_{47}$, CODE)	{ [<CODE1>⟶<CODE>.,fin] }
$\mathbb{I}_{47}$	GOTO ($\mathbb{I}_{47}$, INSTRUCTION)	
$\mathbb{I}_{48}$	GOTO ($\mathbb{I}_{47}$, AFFECTATION)	
$\mathbb{I}_{49}$	GOTO ($\mathbb{I}_{47}$, BOUCLE)	

$N°$	GOTO	Ensemble d'item
$\mathbb{I}_{50}$	GOTO ($\mathbb{I}_{47}$, CONDITION)	
$\mathbb{I}_{51}$	GOTO ($\mathbb{I}_{47}$, CASE)	
$\mathbb{I}_{52}$	GOTO ($\mathbb{I}_{47}$, LIRE)	
$\mathbb{I}_{53}$	GOTO ($\mathbb{I}_{47}$, ECRIRE)	
$\mathbb{I}_{54}$	GOTO ($\mathbb{I}_{47}$, APPEL_P)	
$\mathbb{I}_{55}$	GOTO ($\mathbb{I}_{47}$, id)	
$\mathbb{I}_{56}$	GOTO ($\mathbb{I}_{47}$, pour)	
$\mathbb{I}_{57}$	GOTO ($\mathbb{I}_{47}$, tq)	
$\mathbb{I}_{58}$	GOTO ($\mathbb{I}_{47}$, répéter)	
$\mathbb{I}_{59}$	GOTO ($\mathbb{I}_{47}$, si)	
$\mathbb{I}_{60}$	GOTO ($\mathbb{I}_{47}$, cas)	
$\mathbb{I}_{61}$	GOTO ($\mathbb{I}_{47}$, lire)	
$\mathbb{I}_{62}$	GOTO ($\mathbb{I}_{47}$, lireln)	
$\mathbb{I}_{63}$	GOTO ($\mathbb{I}_{47}$, écrire)	
$\mathbb{I}_{64}$	GOTO ($\mathbb{I}_{47}$, écrireln)	
...	...	...
$\mathbb{I}_{670}$	GOTO ($\mathbb{I}_{665}$, ;)	{[<SINON> ⟶sinon debut <CODE1>fin ; ,fin,id,pour,tq,répéter,si,cas,lire,lireln,écrire,écrireln] }

Remarque : La même procédure est répété jusqu'a ce qu'il n'y ait plus d'items à rajouter dans $\mathbb{I}$, avec notre grammaire, nous avons construit $\mathbb{I}_{670}$ ensembles d'items.
$\mathbb{I} = \{\mathbb{I}_0, ..., \mathbb{I}_{670}\}$

3.3.3 La table d'analyse de $ALgo\_Comp$

La Table 3.2 donne un dictionnaire des symboles de notre grammaire (S_G) afin de simplifier l'affichage de la table d'analyse.

TABLE 3.2 – Index de la table d'analyse de $ALgo\_Comp$

S_G	S	S_G	S	S_G	S	S_G	S	S_G	S
.	S_1	algorithme	S_2	id	S_3	;	S_4	debut	S_5
fin	S_6	type	S_7	=	S_8	enregistrement	S_9	:	S_{10}
tableau	S_{11}	[	S_{12}	int	S_{13}	]	S_{14}	de	S_{15}
,	S_{16}	entier	S_{17}	réel	S_{18}	booléene	S_{19}	chaine	S_{20}
caractère	S_{21}	^	S_{22}	constante	S_{23}	opaff	S_{24}	real	S_{25}
char	S_{26}	string	S_{27}	vrai	S_{28}	faux	S_{29}	variable	S_{30}
fonction	S_{31}	(	S_{32}	)	S_{33}	procédure	S_{34}	var	S_{35}
pour	S_{36}	à	S_{37}	faire	S_{38}	tq	S_{39}	répéter	S_{40}
jusquà	S_{41}	si	S_{42}	alors	S_{43}	sinon	S_{44}	ou	S_{45}
et	S_{46}	non	S_{47}	oprel	S_{48}	cas	S_{49}	parmi	S_{50}
lire	S_{51}	lireln	S_{52}	écrire	S_{53}	écrireln	S_{54}	+	S_{55}
-	S_{56}	*	S_{57}	/	S_{58}	ALGO	S_{59}	NVTYPE	S_{60}
T	S_{61}	T1	S_{62}	CHAMPS	S_{63}	CH	S_{64}	TABLEAU	S_{65}
DIM	S_{66}	TYPETAB	S_{67}	TYPE	S_{68}	CONST	S_{69}	C	S_{70}
C1	S_{71}	VALEUR	S_{72}	VAR	S_{73}	V	S_{74}	V1	S_{75}
TYPEVAR	S_{76}	ID	S_{77}	D	S_{78}	ACTION	S_{79}	FONCTION	S_{80}
PROCEDURE	S_{81}	ENTRER	S_{82}	SORTIE	S_{83}	CODE	S_{84}	CODE1	S_{85}
INSTRUCTION	S_{86}	AFFECTATION	S_{87}	EXPRE	S_{88}	AFF	S_{89}	TAB	S_{90}
EXP	S_{91}	APPEL_P	S_{92}	E	S_{93}	E1	S_{94}	BOUCLE	S_{95}
CONDITION	S_{96}	SINON	S_{97}	COND	S_{98}	COND1	S_{99}	COND2	S_{100}
EX	S_{101}	CASE	S_{102}	CAS	S_{103}	CAS1	S_{104}	VAL	S_{105}
LIRE	S_{106}	L	S_{107}	ECRIRE	S_{108}	S	S_{109}	K	S_{110}
EXPRESSION	S_{111}	A	S_{112}	B	S_{113}	X	S_{114}	#	S_{115}

TABLE 3.3 – Table d'analyse de $ALgo\_Comp$

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}	S_{17}	S_{18}	S_{19}	S_{20}	S_{21}	S_{22}	S_{23}	S_{24}	S_{25}	S_{26}	S_{27}	S_{28}	S_{29}	S_{30}
I_0		D_2																												
I_1																														
I_2			D_3																											
I_3				D_4																										
I_4					R_4		D_6																R_4							R_4
I_5					R_{24}																		D_8							R_{24}
I_6			D_9																											
I_7					R_{35}																									D_{11}
I_8			D_{13}							D_{14}																				
I_9																														
I_{10}					R_{46}																									
I_{11}			D_{22}																											
I_{12}					R_{23}																									R_{23}
I_{13}																								D_{23}						
I_{14}									D_{28}		D_{27}																			
I_{15}					D_{29}																									
I_{16}					R_{44}																									
I_{17}					R_{45}																									
I_{18}			D_{30}																											
I_{19}			D_{31}																											
I_{20}					R_{34}																									
I_{21}										D_{32}																				
I_{22}										R_{43}						D_{34}														
I_{23}													D_{36}												D_{37}	D_{39}	D_{38}	D_{40}	D_{41}	
I_{24}					R_3																		R_3							R_3
I_{25}					R_4		D_6																R_4							R_4
I_{26}					R_6		R_6																R_6							R_6
I_{27}												D_{43}																		
I_{28}			D_{45}																											
I_{29}			D_{55}																											
I_{30}																														
I_{31}																														
I_{32}			D_{76}								D_{70}						D_{71}	D_{72}	D_{74}	D_{73}	D_{75}									
I_{33}										R_{41}																				
I_{34}			D_{22}																											
I_{35}				D_{78}																										
I_{36}				R_{28}																										
I_{37}				R_{29}																										
I_{38}				R_{31}																										
I_{39}				R_{30}																										
I_{40}				R_{32}																										
I_{41}				R_{33}																										
I_{42}					R_5																		R_5							R_5
I_{43}													D_{79}																	
I_{44}					R_7		R_7																R_7							R_7
I_{45}										D_{80}																				
I_{46}						D_{81}																								
I_{47}			D_{55}			R_{55}																								
I_{48}			R_{56}			R_{56}																								
I_{49}			R_{57}			R_{57}																								
I_{50}			R_{58}			R_{58}																								
I_{51}			R_{59}			R_{59}																								
I_{52}			R_{60}			R_{60}																								
I_{53}			R_{61}			R_{61}																								
I_{54}			R_{62}			R_{62}																								
I_{55}	D_{87}											D_{89}										D_{88}		R_{73}						
I_{56}			D_{90}																											
I_{57}																														
I_{58}					D_{92}																									
I_{59}																														
I_{60}																														

	S_{31}	S_{32}	S_{33}	S_{34}	S_{35}	S_{36}	S_{37}	S_{38}	S_{39}	S_{40}	S_{41}	S_{42}	S_{43}	S_{44}	S_{45}	S_{46}	S_{47}	S_{48}	S_{49}	S_{50}	S_{51}	S_{52}	S_{53}	S_{54}	S_{55}	S_{56}	S_{57}	S_{58}	S_{59}	S_{60}
I_0																													I_1	
I_1																														
I_2																														
I_3																														I_5
I_4	R_4			R_4																										
I_5	R_{24}			R_{24}																										
I_6																														
I_7	R_{35}			R_{35}																										
I_8																														
I_9																														
I_{10}	D_{18}			D_{19}																										
I_{11}																														
I_{12}	R_{23}			R_{23}																										
I_{13}																														
I_{14}																														
I_{15}																														
I_{16}																														
I_{17}																														
I_{18}																														
I_{19}																														
I_{20}	R_{34}			R_{34}																										
I_{21}																														
I_{22}																														
I_{23}																														
I_{24}	R_3			R_3																										
I_{25}	R_4			R_4																										I_{42}
I_{26}	R_6			R_6																										
I_{27}																														
I_{28}																														
I_{29}						D_{56}			D_{57}	D_{58}		D_{59}							D_{60}		D_{61}	D_{62}	D_{63}	D_{64}						
I_{30}		D_{65}																												
I_{31}		D_{66}																												
I_{32}																														
I_{33}																														
I_{34}																														
I_{35}																														
I_{36}																														
I_{37}																														
I_{38}																														
I_{39}																														
I_{40}																														
I_{41}																														
I_{42}	R_5			R_5																										
I_{43}																														
I_{44}	R_7			R_7																										
I_{45}																														
I_{46}																														
I_{47}						D_{56}			D_{57}	D_{58}		D_{59}							D_{60}		D_{61}	D_{62}	D_{63}	D_{64}						
I_{48}						R_{56}			R_{56}	R_{56}		R_{56}							R_{56}		R_{56}	R_{56}	R_{56}	R_{56}						
I_{49}						R_{57}			R_{57}	R_{57}		R_{57}							R_{57}		R_{57}	R_{57}	R_{57}	R_{57}						
I_{50}						R_{58}			R_{58}	R_{58}		R_{58}							R_{58}		R_{58}	R_{58}	R_{58}	R_{58}						
I_{51}						R_{59}			R_{59}	R_{59}		R_{59}							R_{59}		R_{59}	R_{59}	R_{59}	R_{59}						
I_{52}						R_{60}			R_{60}	R_{60}		R_{60}							R_{60}		R_{60}	R_{60}	R_{60}	R_{60}						
I_{53}						R_{61}			R_{61}	R_{61}		R_{61}							R_{61}		R_{61}	R_{61}	R_{61}	R_{61}						
I_{54}						R_{62}			R_{62}	R_{62}		R_{62}							R_{62}		R_{62}	R_{62}	R_{62}	R_{62}						
I_{55}		D_{84}																												
I_{56}																														
I_{57}		D_{91}																												
I_{58}																														
I_{59}		D_{93}																												
I_{60}		D_{94}																												

	S_{61}	S_{62}	S_{63}	S_{64}	S_{65}	S_{66}	S_{67}	S_{68}	S_{69}	S_{70}	S_{71}	S_{72}	S_{73}	S_{74}	S_{75}	S_{76}	S_{77}	S_{78}	S_{79}	S_{80}	S_{81}	S_{82}	S_{83}	S_{84}	S_{85}	S_{86}	S_{87}	S_{88}	S_{89}	S_{90}
I_0																														
I_1																														
I_2																														
I_3																														
I_4									I_7																				
I_5																														
I_6													I_{10}																	
I_7																														
I_8										I_{12}																				
I_9																														
I_{10}																			I_{15}	I_{16}	I_{17}								
I_{11}														I_{20}			I_{21}													
I_{12}																														
I_{13}																														
I_{14}	I_{24}	I_{25}			I_{26}																									
I_{15}																														
I_{16}																														
I_{17}																														
I_{18}																														
I_{19}																														
I_{20}																														
I_{21}																		I_{33}												
I_{22}																														
I_{23}												I_{35}																		
I_{24}																														
I_{25}																														
I_{26}																														
I_{27}																														
I_{28}			I_{44}																											
I_{29}																							I_{46}			I_{47}	I_{48}			
I_{30}																														
I_{31}																														
I_{32}					I_{68}		I_{69}									I_{67}														
I_{33}																														
I_{34}																	I_{77}													
I_{35}																														
I_{36}																														
I_{37}																														
I_{38}																														
I_{39}																														
I_{40}																														
I_{41}																														
I_{42}																														
I_{43}																														
I_{44}																														
I_{45}																														
I_{46}																														
I_{47}																							I_{83}	I_{82}	I_{47}	I_{48}				
I_{48}																														
I_{49}																														
I_{50}																														
I_{51}																														
I_{52}																														
I_{53}																														
I_{54}																														
I_{55}																												I_{85}	I_{86}	
I_{56}																														
I_{57}																														
I_{58}																														
I_{59}																														
I_{60}																														

	S_{91}	S_{92}	S_{93}	S_{94}	S_{95}	S_{96}	S_{97}	S_{98}	S_{99}	S_{100}	S_{101}	S_{102}	S_{103}	S_{104}	S_{105}	S_{106}	S_{107}	S_{108}	S_{109}	S_{100}	S_{111}	S_{112}	S_{113}	S_{114}	S_{115}
I_0																									
I_1																									accepté
I_2																									
I_3																									
I_4																									
I_5																									
I_6																									
I_7																									
I_8																									
I_9																									
I_{10}																									
I_{11}																									
I_{12}																									
I_{13}																									
I_{14}																									
I_{15}																									
I_{16}																									
I_{17}																									
I_{18}																									
I_{19}																									
I_{20}																									
I_{21}																									
I_{22}																									
I_{23}																									
I_{24}																									
I_{25}																									
I_{26}																									
I_{27}																									
I_{28}																									
I_{29}		I_{54}			I_{49}	I_{50}						I_{51}				I_{52}		I_{53}							
I_{30}																									
I_{31}																									
I_{32}																									
I_{33}																									
I_{34}																									
I_{35}																									
I_{36}																									
I_{37}																									
I_{38}																									
I_{39}																									
I_{40}																									
I_{41}																									
I_{42}																									
I_{43}																									
I_{44}																									
I_{45}																									
I_{46}																									
I_{47}		I_{54}			I_{49}	I_{50}						I_{51}				I_{52}		I_{53}							
I_{48}																									
I_{49}																									
I_{50}																									
I_{51}																									
I_{52}																									
I_{53}																									
I_{54}																									
I_{55}																									
I_{56}																									
I_{57}																									
I_{58}																									
I_{59}																									
I_{60}																									

3.4 L'analyse sémantique de $ALgo\_Comp$

Dans cette phase des traitements (portions de code) dits également actions sémantiques sont insérées à des endroits précis de la grammaire pour répondre à des spécificité du langage considéré. Par exemple, cette phase vérifie que les opérandes de chaque opérateur sont conformes aux spécifications du langage source. Des exemples de taches liées au contrôle de type sont :

- Mémoriser les types définis par l'utilisateur.
- Traiter les déclarations de variables et les types qui leur sont appliqués.
- Contrôler les types des opérandes des opérations arithmétiques et logique et déduire le type du résultat.

3.5 Réalisation de traducteur de $ALgo\_Comp$

Après la réalisation de trois premières étapes de compilation, nous utilisons un traducteur vers le langage de programmation Pascal pour continuer l'exécution des autres étapes.
Notre traducteur accepte en entrée une représentation textuelle d'un algorithme et que produit en sortie une représentation du même algorithme dans le langage Pascal.

3.6 Conclusion

Ce chapitre nous a permis de présenter et détailler la conception de notre compilateur avec les outils et les techniques utilisés pour développer notre application. Dans le chapitre suivant nous allons présenter l'application développé dans le cadre de ce projet.

CHAPITRE 4

Présentation de l'application et les outils de développement

Sommaire

4.1 Introduction

Dans ce chapitre, nous présentons quelques travaux connexes. Ensuite, nous présentons les outils de développement utilisés dans ce projet et l'application développée qui est le résultat du travail cité dans le chapitre précédent, nous expliquons en détail notre application à travers des exemples.

4.2 Travaux connexes

4.2.1 Algosim

Est un logiciel qui permet la conception des algorithmes grâce à une interface graphique sans saisir de code. les structures sont testé au fur à mesure. le code est exécuté dans une console graphique. Il est possible

d'exécuter le code pas à pas. l'algorithme créé peut généré un algorithme en pseudo-code au format PDF et le programme pascal.

4.2.2 Mobile compiler

Un compilateur d'un algorithme qui traduit un algorithme vers un programme écrit dans le langage de programmation Pascal [8], à pour but d'aider les étudiants de premmière année à corriger leurs algorithmes.

4.2.3 Comparaison, critiques et motivation

Algosim est simple à utiliser, mais ne contient pas un éditeur de texte comme les logiciels classiques de programmation. Par contre, Mobile compiler contient un éditeur de texte, mais ne prend pas en compte les structures dynamiques et les sous-programmes, qui est l'intérêt principal de notre projet. La nouveauté de notre application est :

- Permet de gérer les structures de données dynamiques.
- Aider l'utilisateur à écrire son algorithme à travers des instructions simples qui sont réalisées automatiquement par le programmeur.

4.3 Outils de développement

4.3.1 Le langage JAVA

Java est un langage de programmation orienté objet, développé par SunMicrosystems et destiné à fonctionner dans une machine virtuelle, il permet de créer des logiciels compatibles avec des nombreux systèmes d'exploitation. Java est considéré comme un langage adaptable aux plusieurs domaines puisque une application web implémentée par celle-ci peut avoir des extensions ou des modifications dans le future. De plus, java permet de réduire le temps de développement d'une application grâce à la réutilisation du code développé.

4.3.2 NetBeans IDE

NetBeans est un environnement de développement intégré (IDE), placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License) et GPLv2. En plus de Java. Q NetBeans permet également de supporter différents langages, comme C, C++, Java-script, XML, Groovy, PHP et HTML de façon native ainsi que bien d'autres (comme Python) par l'ajout de greffons. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web) .

4.4 Environnement de rédaction

4.4.1 Latex

LATEX est un langage et un système de composition de documents. Il s'agit d'une collection de macro-commandes destinées à faciliter l'utilisation du « processeur de texte ». il permet de rédiger des documents dont la mise en page est réalisée automatiquement en se conformant du mieux possible à des normes typographyques.

4.5 Présentation de notre logiciel Algo_Comp

Nous allons présenter dans cette section, le travail réalisé dans le cadre de ce projet par l'intermédiaire des captures d'écrans de quelques interfaces de notre application.
Cette première capture présente la fenêtre d'accueil de notre application (Figure 4.1). La fenêtre d'accueil de notre application s'affiche comme suit :

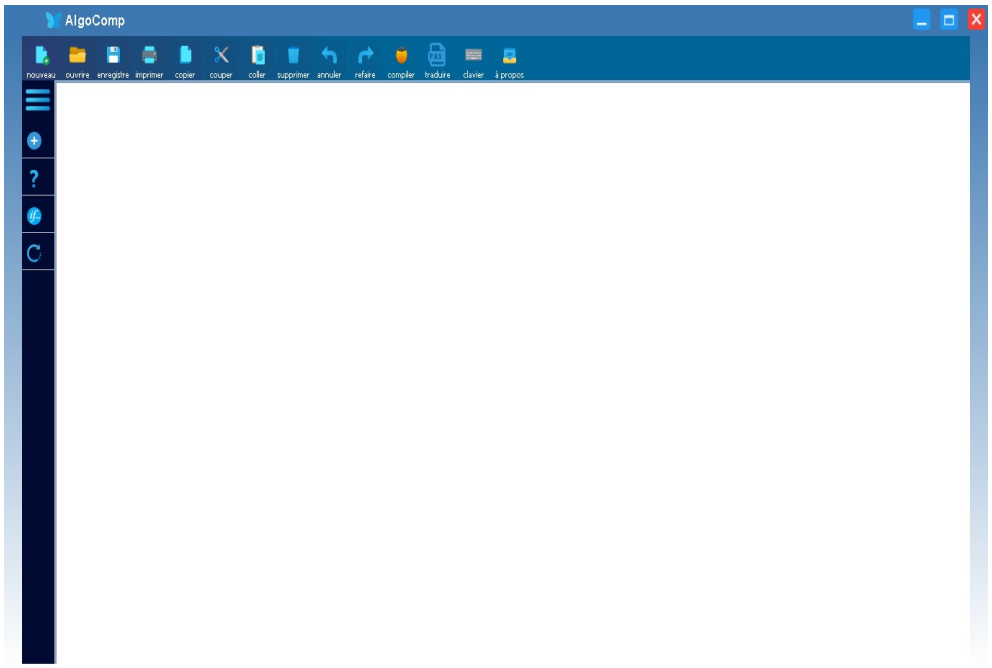


FIGURE 4.1 – Fenêtre principal de $ALgo\_Comp$

La fenêtre principal comme il est illustré dans la figure 4.1 contient un éditeur de text pour l'écriture du code et deux barres d'outils un barre verticale pour faciliter la rédaction d' un algorithme (Figure 4.2) et la deuxième est horizontale présente les différentes fonctionnalités du logiciel.

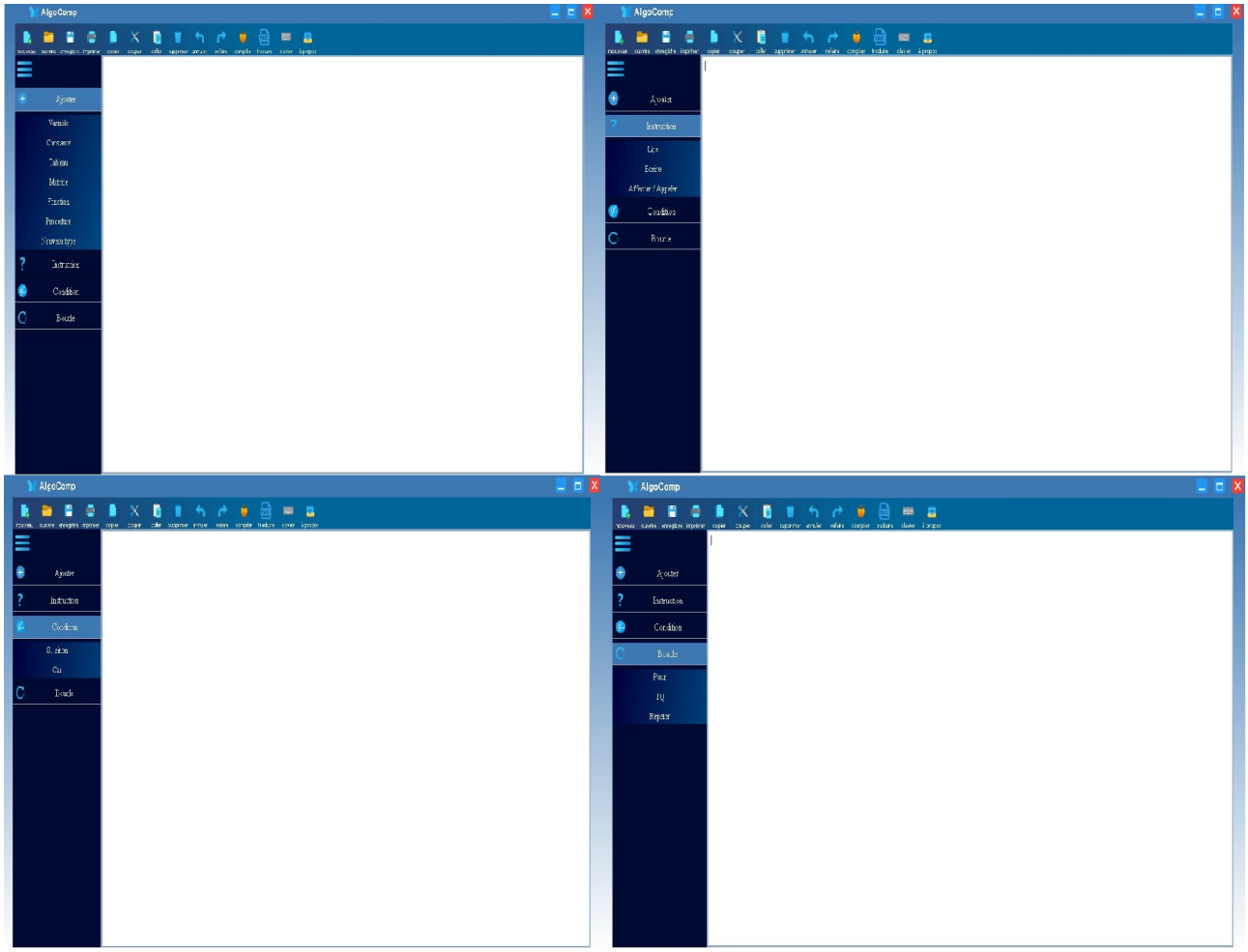


FIGURE 4.2 – Présentation des boutons du barre de rédaction

La figure 4.3 illustre un exemple pour déclarer une variable à l'aide du bouton ajouter. ce bouton permet d'ajouter la déclaration d'une variable son écriture de l'instruction complète.

1. On utilise le button AJOUTER et on choisi VARIABLE.
2. On choisi le nom de la variable et son type puis on utilise le button AJOUTER.
3. Le résultat obtenu est illustré dans la Figure 4.3.

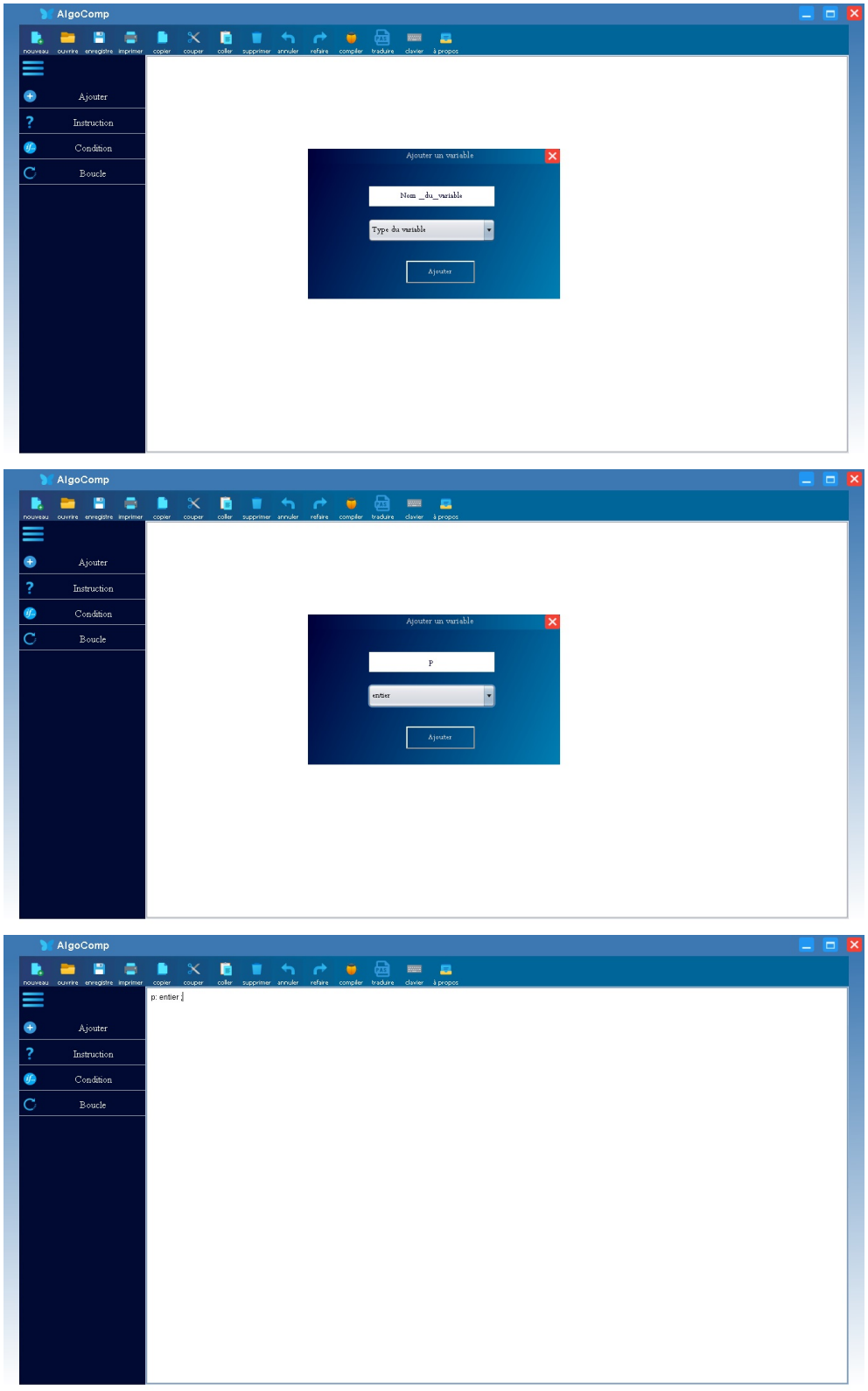


FIGURE 4.3 – Exemple de déclaration d'une variable

Remarque : Nous ne pouvons pas déclarer une variable sans précision de son type, comme il est illustré dans la Figure 4.4 .

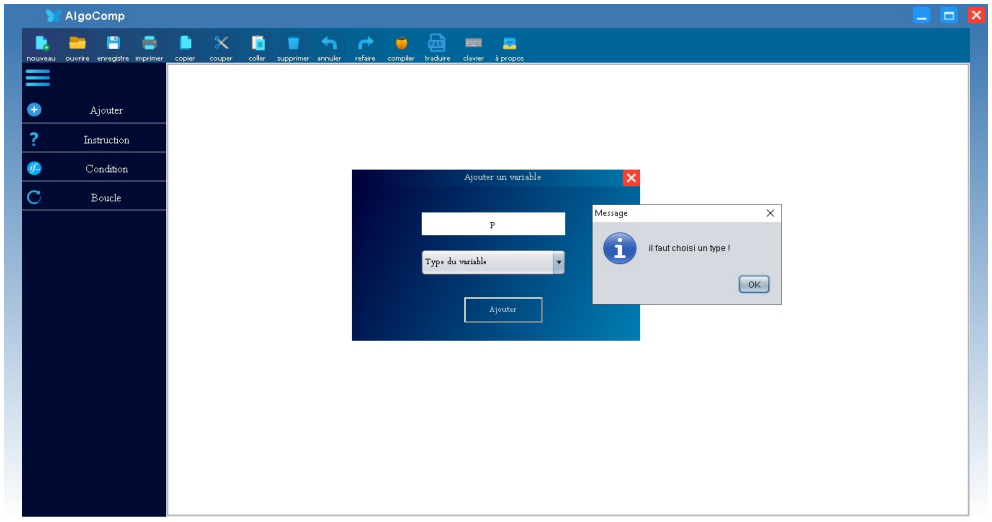
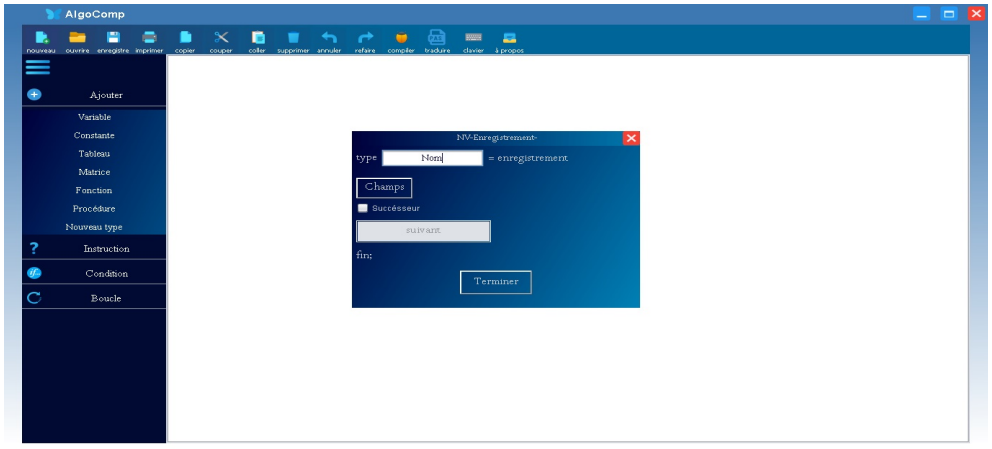


FIGURE 4.4 – Exemple de déclaration d'une variable sans précision de son type

Un autre exemple pour déclarer un nouveau type d'enregistrement ou liste chainée.

1. On click sur le button AJOUTER et on choisi NVTYPE, puis ENREGISTREMENT.
2. On précise le nom de l'enregistrement et on click sur CHAMPS pour on ajoute les champs, une autre fenêtre s'affiche.
3. On ajoute le nom et le type de chaque champ puis on click sur TERMINER.
4. Le résultat obtenu est s'affiche dans la Figure 4.5.



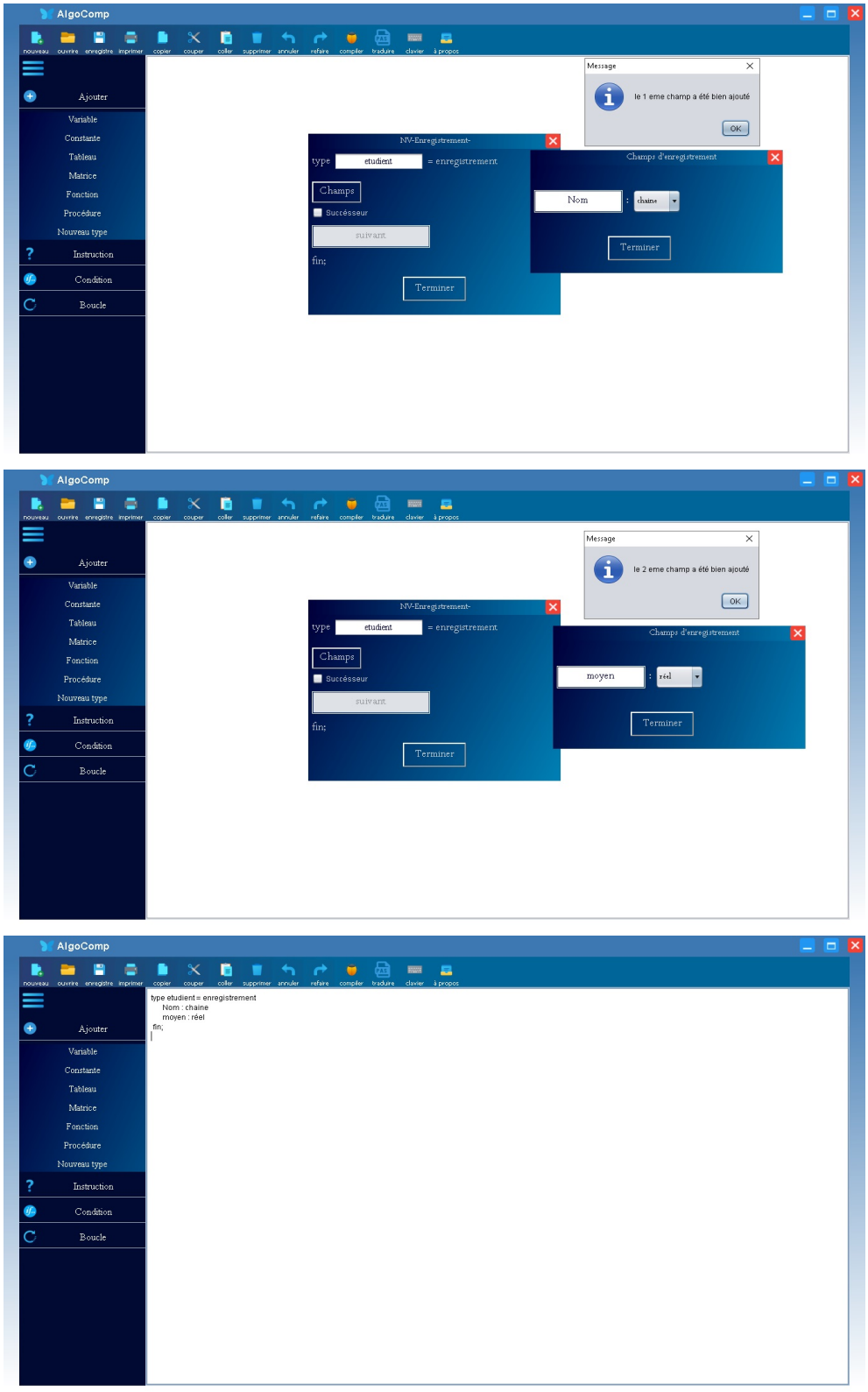


FIGURE 4.5 – Exemple de déclaration d'un enregistrement

5. Si nous voulons déclarer une liste chainée, on utilise SUCCESSEUR et on précise le nom du successeur puis TERMINER. 6. Le résultat obtenu est illustré dans la Figure 4.6.

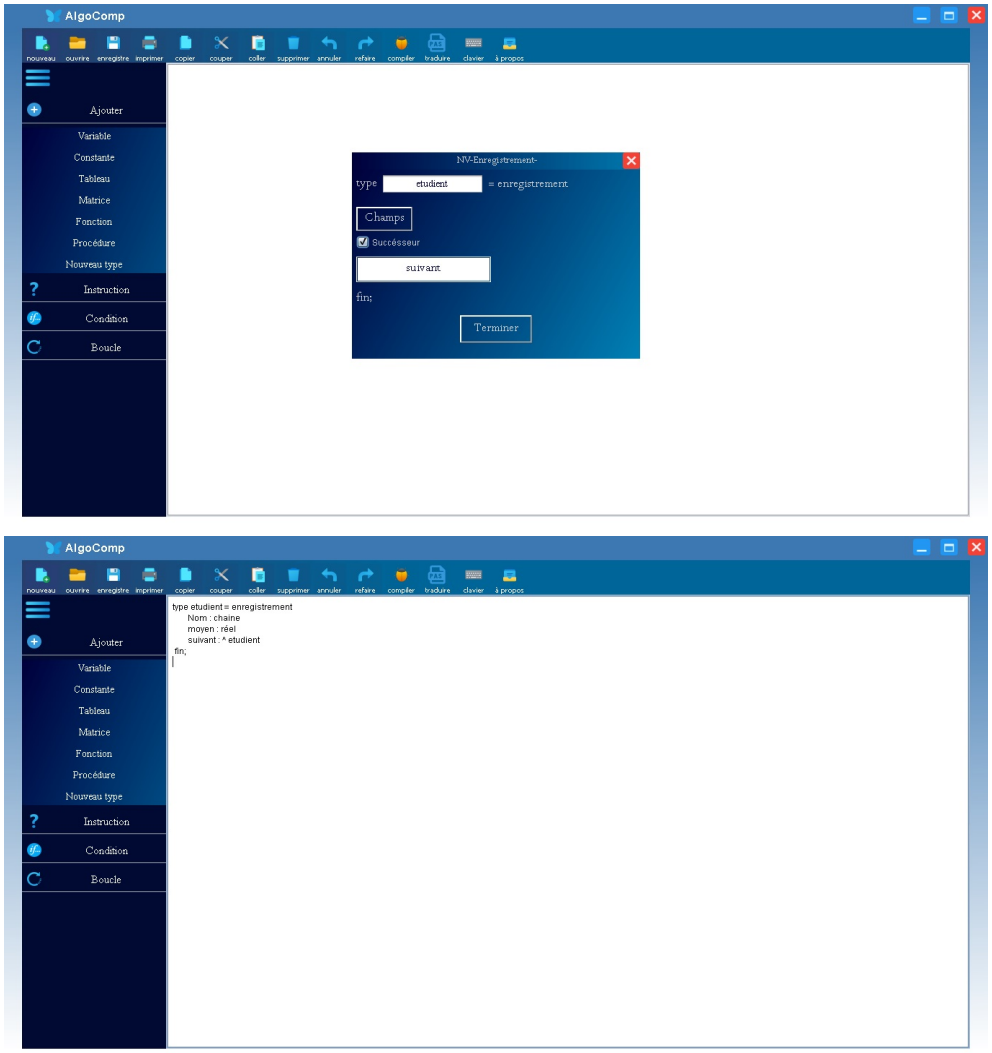


FIGURE 4.6 – Exemple de déclaration d'une liste chainée

La figure 4.7 montre un algorithme codé dans $ALgo\_Comp$.

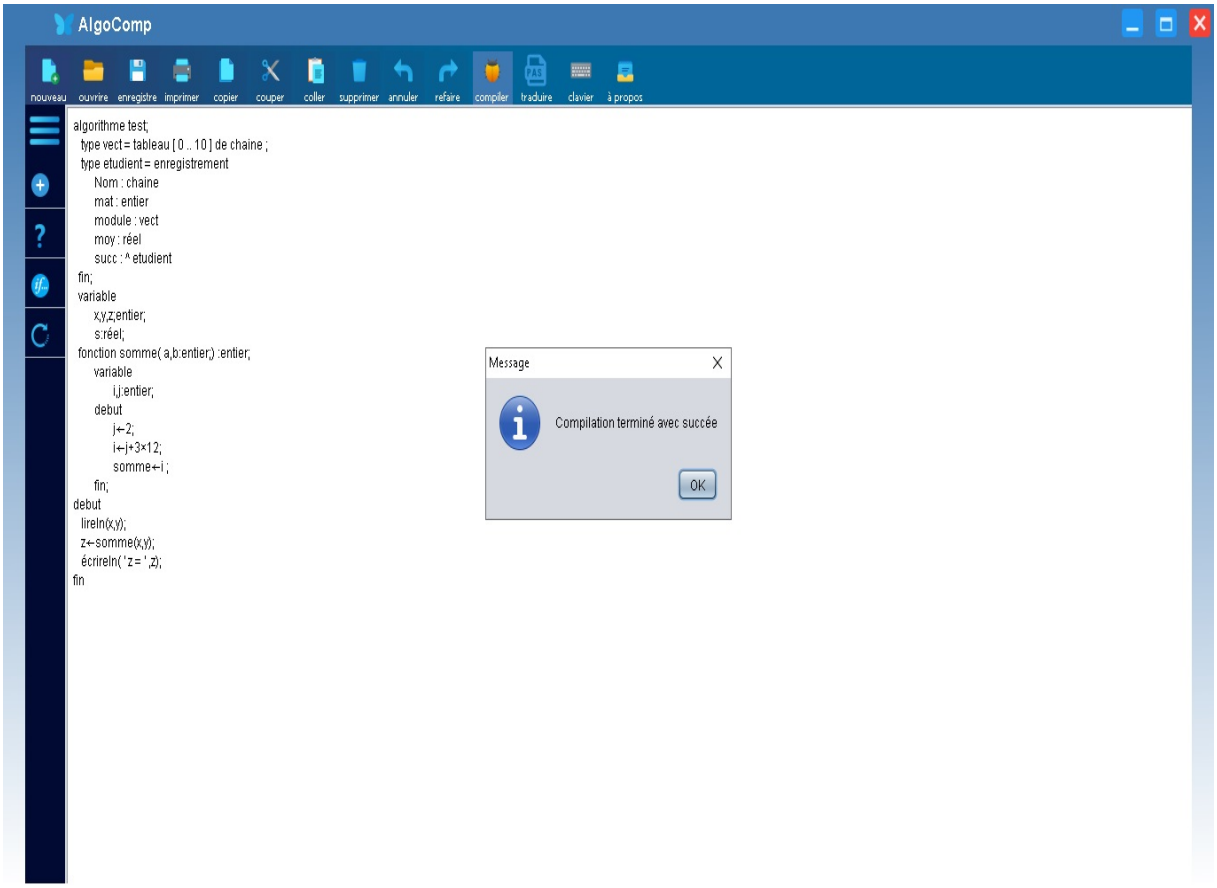


FIGURE 4.7 – Exemple de compilation d'un code correcte

Le résultat de compilation s'affiche dans la même fenêtre et indique que l'algorithme est correcte.

La traduction de ce code produit un fichier text contient la répresentation du même code en langage Pascal (Figure 4.8).

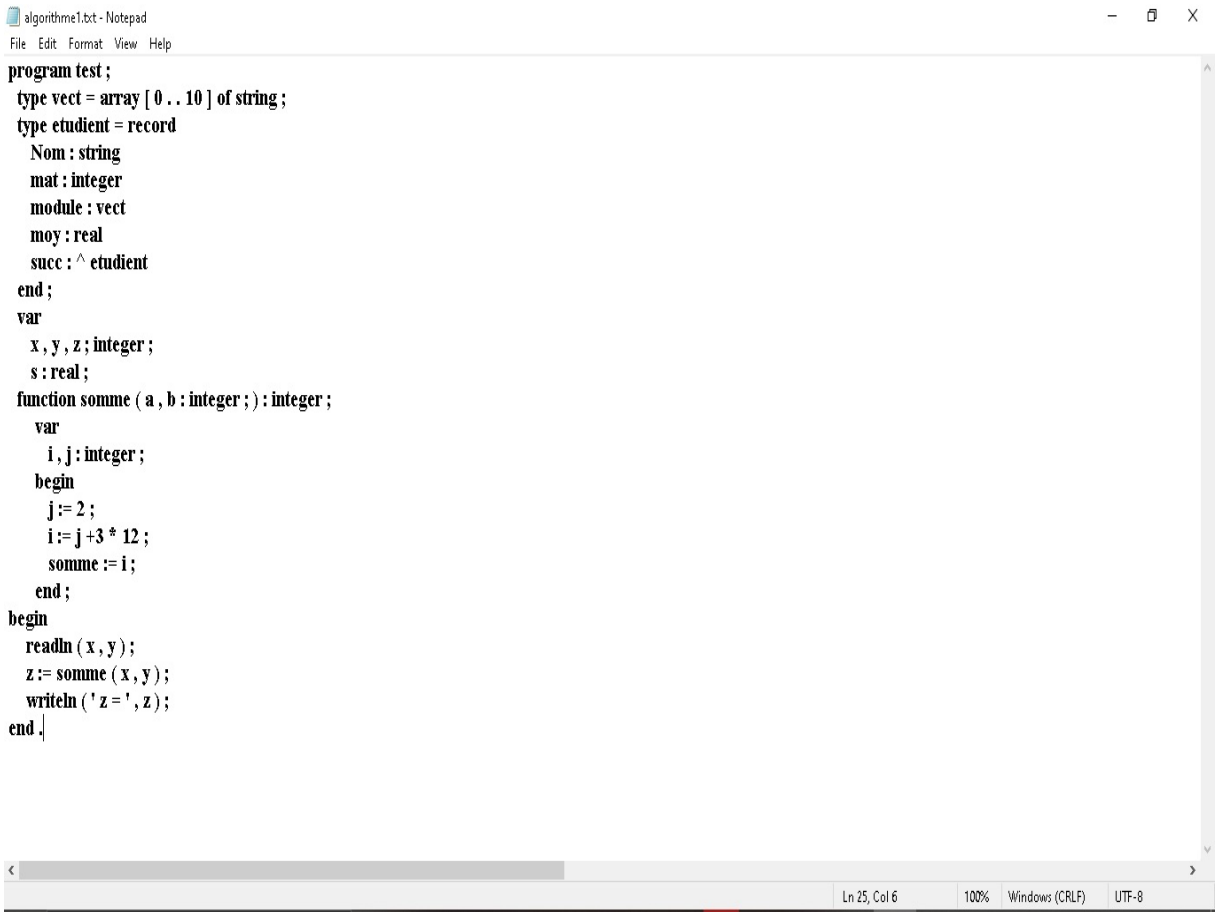


FIGURE 4.8 – La traduction d'un code correcte

Un autre exemple correcte :

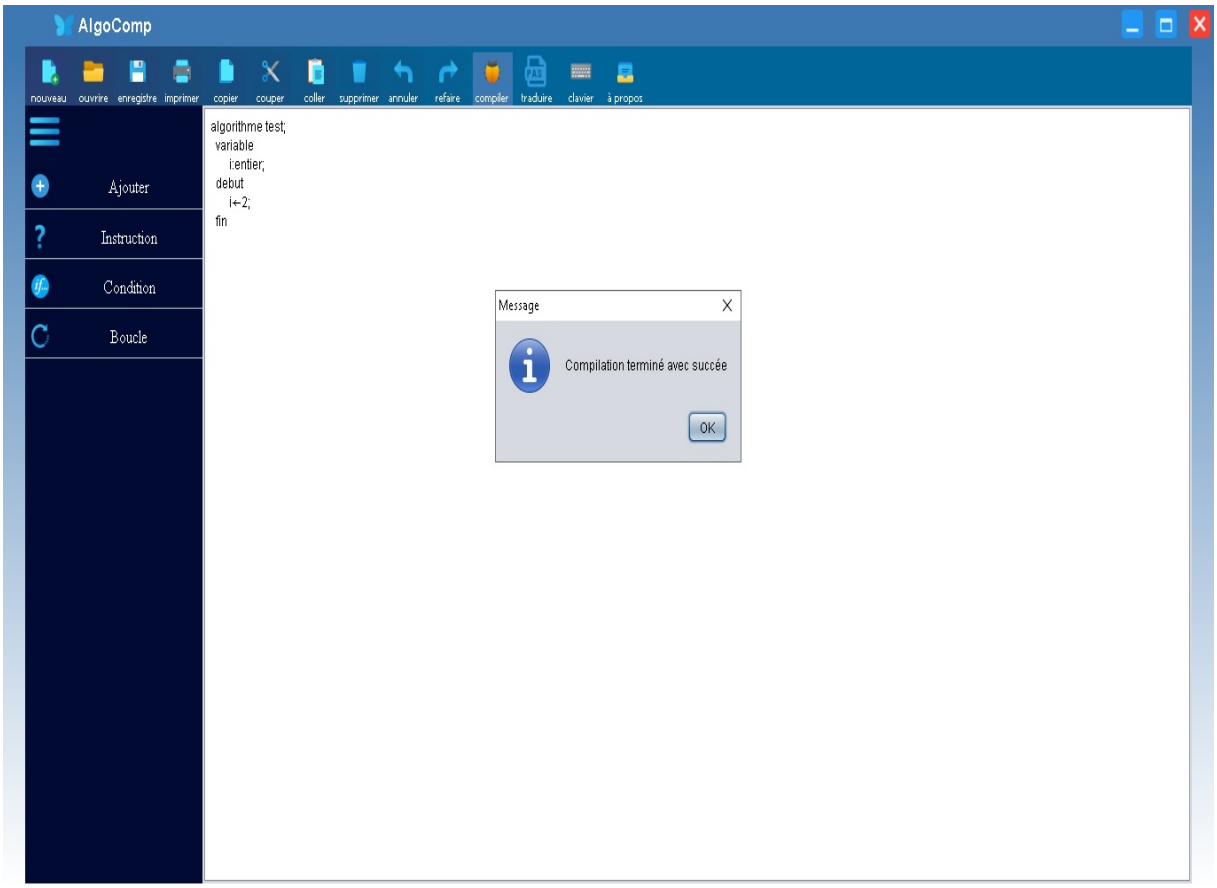


FIGURE 4.9 – Exemple de compilation d'un code correcte

Un autre exemple d'un algorithme incorrecte est illustré dans la figure 4.10. Le résultat de compilation s'affiche dans la même fenêtre et indique la ligne d'erreur (erreur lexicale) afin de mentionner que l'algorithme est incorrecte.

Un exemple contient une erreur syntaxique est illustré dans la Figure 4.11.

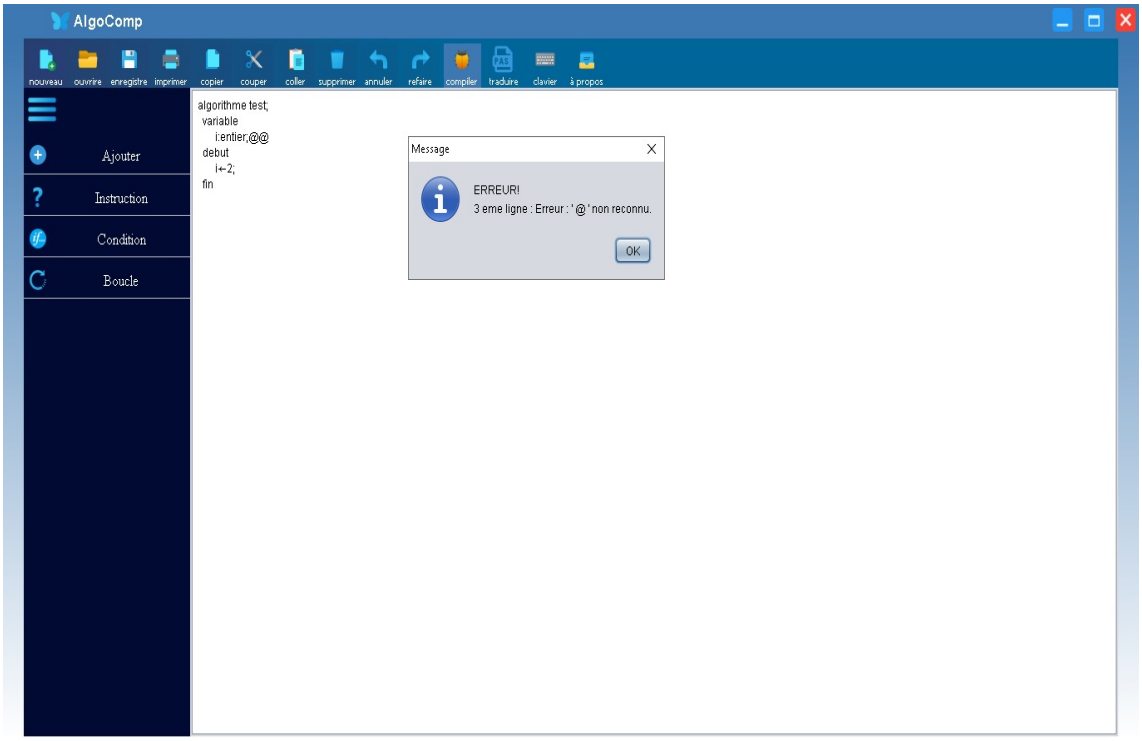


FIGURE 4.10 – Exemple de compilation d'un code incorrecte (contient une erreur lexicale)

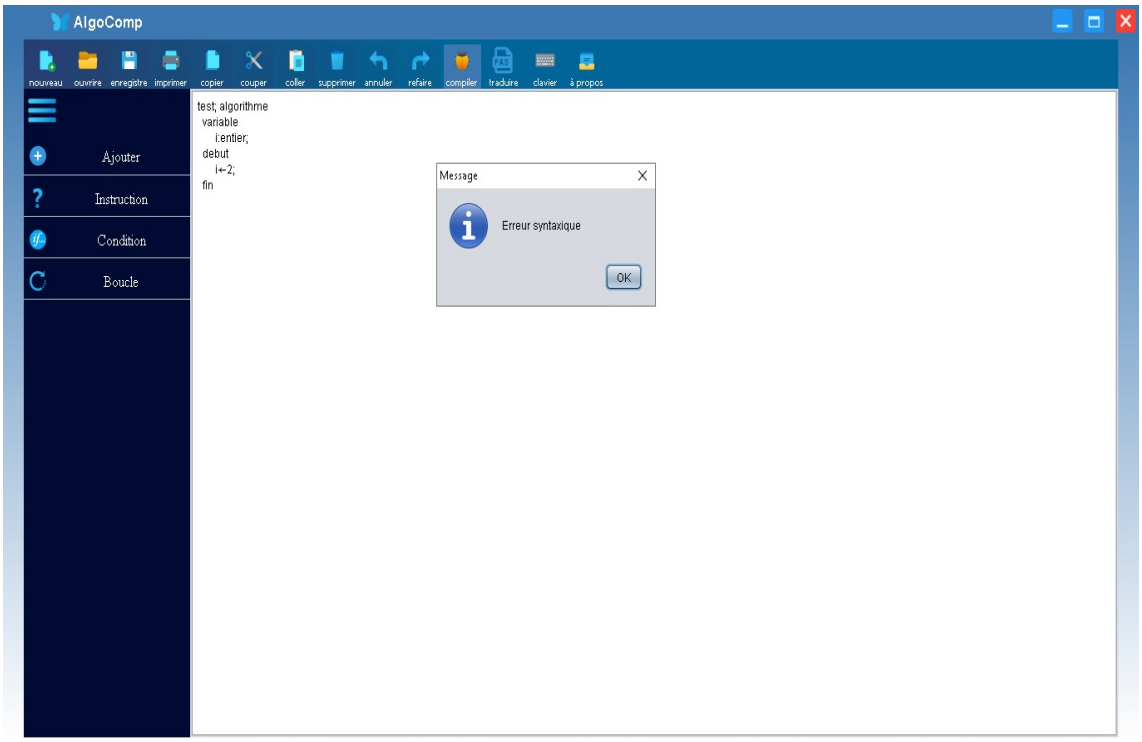


FIGURE 4.11 – Exemple de compilation d'un code incorrecte (contient une erreur syntaxique)

4.6 Conclusion

Dans ce chapitre, nous avons exposé le produit final de ce projet de fin d'étude qui est notre compilateur $ALgo\_Comp$. Au début, on a montré les outils de développement de l'application, puis on a présenté les différentes interfaces qui permettent aux utilisateurs d'utiliser notre application d'une manière plus efficace.

Conclusion générale

Le but principal de notre travail était de développer un compilateur d'un algorithme $ALgo\_Comp$, pour cela nous avons commencé par présenter les langages de programmation et la compilation. Ensuite, nous avons développé un compilateur en traitant les principes de bases de la compilation : l'analyse lexicale, l'analyse syntaxique, l'analyse sémantique et le processus de traduction avec les outils fondamentaux utilisés pour effectués ces analyses : fondement de base de la théorie des langages (automate, grammaire, ...etc.) et les méthodes algorithmique des différentes étapes de compilation.

Dans ce contexte, nous avons développé une application qui traduit un algorithme vers un programme écrit dans le langage de programmation Pascal.

Nous avons décomposé notre application en deux différentes parties : (i) la première pour les étudiants de la première année Licence Mathématiques et Informatique, à pour but d'aider les étudiants à coder et corriger leurs algorithmes et simplifier le passage vers la programmation en pascal. (ii) La deuxième partie pour les étudiants de la troisième année Licence Informatique, à pour but de présenter les différentes étapes et outils de la compilation pour faciliter d'apprendre le module de compilation dans le domaine de l'informatique.

Ce projet nous a permis :

- De bien comprendre et appliquer les différents techniques et méthodes d'analyse, de la compilation et de la programmation.
- De maitriser bien le langage de programmation java et l'outil de rédaction LaTeX.

Pour cela, il est intéressant d'améliorer l'application pour obtenir une nouvelle version contient les nouvelles fonctionnalités suivantes :

- L'ajout des autres options et structures comme l'utilisation des fichiers dans notre application.
- l'intégration de notre application dans un environnement mobile.

.

Bibliographie

[1] Torben Ægidius Mogensen, basics of compiler design, anniversary edition, 2010.

[2] Marc Jachym, Support de cours de langage C, ENS de Cachan, 2017.

[3] Maher Helaoui, Support de cours de compilation, 2016/2017.

[4] Ramzi BOULKROUNE, Support de cours de compilation, Université de Annaba, 2009.

[5] Sami KHALFOUI et Moez HAMMAMI, Support de cours de compilation, ENSI, 2006/2007.

[6] Etienne M.Gagnon et Jean privat, Introduction à la compilation et à l'interprétation, Support de cours, UQAM, 2009/2013.

[7] H. Drias, Compilation : Cours et exercices, OPU, 1993.

[8] Ahmed KACHNA, abdelhamid HAOUHAT, simulation et implementation de compilateur d'un algorithme, Mémoir de licence en informatique, Université de laghouat (UATL), 2016/2017.

[9] Fatima Zohra Bousbaa, Support de cours de compilation, Université Amar Telidji de Laghouat (UATL), 2019/2020.

[12] Mr. Meadi et M.Nadjib, Support de cours de compilation, Université de Biskra, 2011/2012.

[11] Compilation et théorie de langages, support de cours, Université de Bretagne Occidentale , 2003.

[10] Tahar ALLAOUI, Support de cours de compilation, Université Amar Telidji de Laghouat (UATL), 2006/2007.

[13] Alfred V.Aho, Monica S.Lam, Ravi Sethi, Jefrey D.Ulman, Compilers principles ,techniques,&Tools, Second edition, 2007.