# Introduction

Ce présent projet représente notre travaille sur la réalisation d’un compilateur pour le langage propre de l’ENSIAS. Et nous avons travaillé avec le langage C et sans l’utilisation des générateurs de l’analyseur lexicale comme FLEX ou les générateurs de l’analyseur syntaxique comme BISON.

Les trois éléments essentiels d’un compilateur sont l’analyseur lexicale, l’analyseur syntaxique et celui de sémantique. Nous nous sommes concentrés premièrement dans la définition des mots-clés de types de données, des opérateurs arithmétique et logique, des séparateurs ainsi que du commentaire pour nous faciliter la tâche d’élaboration de l’analyseur Lexicale.

Le plan de ce rapport sera le suivant :

1. **Grammaire**
2. **Documentation :**
   1. **Analyseur Lexicale**
   2. **Analyseur Syntaxique**
   3. **Analyseur Sémantique**
3. **Fonctionnement du compilateur**
4. **Limites du compilateur**
5. **Conclusion**

## Identifier

Case sensitive.

## Keywords

**Data types :**

Int – float – string – double – char – long – list<,,,,> - File

*Remarques :*

Pour accéder aux éléments de la liste : list[index] or list[name\_of\_element]

**Opérateur :**

***Opérateur arithmétique****:* + , - , / , % , \* , ^ ,

***Opérateur de comparaison****:* == , != , <= , < , >= , > ,

**Opérateur d’affectation** : =

**Opérateur logique** : & , |

## Commentary

Com : ;

## White Spaces

\n, \t, espace,

## Conditions

--- **If elif** ---

If(condition){

}elif(condition){

}else{

}

--- **While** ---

While ( condition ){

}

--- **Do while**---

Do{

}while( condition ) ;

--- **For in** ---

For i in (1,n){

}

For i in variable(){

}

## Structure of Program

Libraries :

Constants :

Variables :

Functions :

Main() :

**Librairies :**

Lib : name\_of\_library1, name\_of\_library2, ….;

H : name\_of\_header1, name\_of\_header2, ….;

**Constants:**

**Variables:**

**Functions:**

Name\_of\_function(argument1, argument2, ….){

Return something;

}

**Main():**

I. Grammaire

Prog : Lib | Const | Var | Func | Main

Lib   : Libraries : [ lib: ID{,ID}; ] | [ h: ID{,ID}; ] | e

Const : Constants : { ID = Value; } | e

Var : Variables : { ID [= Value] | e ; } | e

Func : Functions : [ { ID ( { ID | ID,} ) { Insts | Return } } ] | e

Main : Main ( {ID | ID,} ) { Insts | Return }

Name\_of\_function (){

Constants : kjqsdfkj=qsdfq;

Variables : kjnqjnjqsd=qsdf ;

}

Value  : int | long | float | double | string | char | | List | File

Int : Num { Num }

Long : Num { Num } L

Float : Num { Num } [, Num { Num }] | e F

Double : Num { Num } [, Num { Num }] | e D

String : “ { Alpha | Num | Special } ”

Char : ‘Alpha’

List : List < { ID | ID, } >

File : FILE ( string , ‘AccessType’);

AccessType : r | w

Insts : Inst { Inst }

Inst : CallFunction | Decision | Loop | Expr

CallFunction : ID ( { ID | ID, } ) ;

Condition : ( ID | CallFunction | Value ) Comp\_Op ( ID | CallFunction | Value )

Decision :   [ if ( condition ) { Insts } ] | [ if ( condition ) { Insts } elif { Insts } ]

Loop : for ID in Value { Insts } | for ID in (Num, Num) { Insts } | while (condition ){ Insts } |   do{ Ints }while (condition);

Expr :

Return : return ID | Value ;

Operator : Comp\_Op | Arith\_Op | Assign\_Op | Logic\_Op

Comp\_Op : == | < | <= | > | >= | !=

Arith\_Op : + | - | / | \* | ^ | %

Assign\_Op : =

Logic\_Op : and | or

ID  : \_ | lettre { \_ | lettre | chiffre}

Alpha : a | .. | z | A | .. | Z

Num : 0 | .. | 9

Special : tous les caractères spéciaux