République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي

ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE

المدرسة الوطنية العليا للإعلام الآلي
(المعهد الوطني للتكوين في الإعلام الآلي سابقا)
École nationale Supérieure d'Informatique
ex. INI (Institut National de formation en Informatique)

implémentation d'un analyseur lexico-syntaxique du langage Mini-C,

et d'un producteur du code intermédiaire sous forme de quadruplets

Projet de compile 2CS SIQ3.

Nom Prénom

- KORBAA Hamza
- BOUCHEFIRAT Loukmane

Promoteur : M.AbdelKrim CHEBIEB

Table des matières

1 Introduction

Le but de ce projet est de développer un compilateur pour le langage Mini-c, Compilateur devra générer une forme intermédiaire sous forme de quadruplets. Votre Compilateur aura pour entrées des programmes écrits dans le langage Mini-c.
Si un programme est correct (après les vérifications lexicale, syntaxique,..) un fichier de sortie sera génère contenant le code intermédiaire sous format quadruplets correspondant au code source donnée en entrée.
Si des erreurs se produisent, une stratégie consiste à s'arrêter à la première erreur et donner à l'utilisateur un message significatif de cette erreur pour faciliter la correction

- Si un programme est correct (après les vérifications lexicale, syntaxique,..) un fichier de sortie sera génère contenant le code intermédiaire sous format quadruplets correspondant au code source donnée en entrée ;
- Si des erreurs se produisent, une stratégie consiste à s'arrêter à la première erreur et donner à l'utilisateur un message significatif de cette erreur pour faciliter la correction ;

2 Analyse syntaxique

2.1 La grammaire du langage Mini-C

La grammaire du langage Mini-C est la suivante :

- **Function** → identifier (ArgList) CompoundStmt
- ArgList → Arg | ArgList , Arg
- Arg → Type identifier
- Declaration → Type IdentList ;
- Type → int | float
- IdentList → identifier , IdentList | identifier
- Stmt → ForStmt
  | WhileStmt | Expr ;
  | IfStmt
  | CompoundStmt
  | Declaration
  | ;
- ForStmt → for (Expr ; OptExpr ; OptExpr) Stmt
- OptExpr → Expr | ε
- WhileStmt → while (Expr) Stmt
- IfStmt → if (Expr) Stmt ElsePart

- ElsePart → else Stmt | ε
- CompoundStmt → StmtList
- StmtList → StmtList Stmt | ε
- Expr → identifier = Expr | Rvalue
- Rvalue → Rvalue Compare Mag | Mag
- Compare → == | < | > | <= | >= |!=
- Mag → Mag + Term | Mag Term | Term
- Term → Term * Factor | Term / Factor | Factor
- Factor → (Expr) | - Factor | + Factor | identifier | number

2.2 Transformation de la grammaire

La grammaire présentée précédemment n'est pas LL(1) ; elle contient des productions récursives à gauche. Elle n'est, de ce fait, pas directement exploitable pour la réalisation d'un traducteur par descente récursive. Il faut la rendre LL(1). Le résultat obtenu après transformation est le suivant :

- Function → identifier (ArgList) CompoundStmt
- ArgList → Type identifier NewArgList
- NewArgList → , Arg NewArgList | ε
- Arg → Type identifier
- Declaration → Type IdentList ;
- Type → int | float
- IdentList → identifier IdentListEnd
- IdentListEnd → , IdentList | ε
- Stmt → ForStmt
  | WhileStmt
  | Expr ;
  | IfStmt
  | CompoundStmt
  | Declaration
  | ;
- ForStmt → for (Expr ; ForStmtEnd
- ForStmtEnd → Expr ; ForStmtEnd1 | ; ForStmtEnd1
- ForStmtEnd1 → Expr) Stmt |) Stmt
- WhileStmt → while (Expr) Stmt
- IfStmt → if (Expr) Stmt IfStmtEnd

- IfStmtEnd → else Stmt | ε
- CompoundStmt → StmtList
- StmtList → StmtList Stmt | ε
- Expr → identifier ExprFol
  | (Expr) NewTerm NewMag NewRvalue
  | - Factor NewTerm NewMag NewRvalue
  | + Factor NewTerm NewMag NewRvalue
  | number NewTerm NewMag NewRvalue
- ExprFol → = Expr | NewTerm NewMag NewRvalue
- NewRvalue → Compare Mag NewRvalue | ε
- Compare → == | < | > | <= | >= |!=
- Mag → Term NewMag
- NewMag → + Term NewMag | - Term NewMag | ε
- Term → Factor NewTerm
- NewTerm → * Factor NewTerm | / Factor NewTerm | ε
- Factor → (Expr) | - Factor | + Factor | identifier | number

Ainsi transformée, la grammaire remplie les conditions nécessaires pour être LL(1), encore faut-il que la table d'analyse LL(1) soit mono-définie. On va donc la construire pour voir si tel est le cas.

Symbole	**DEBUTs**	**SUIVANTs**
Function	INT FLOAT	#
ArgList	INT FLOAT	)
NewArgList	, e	)
Arg	INT FLOAT	) ,
Declaration	INT FLOAT	{ } ; (- + FOR WHILE IF INT FLOAT IDENTIFIER NUMBER
Type	INT FLOAT	IDENTIFIER
IdentList	IDENTIFIER	;
IdentListEnd	, e	;
Stmt	{ ; (- + FOR WHILE IF INT FLOAT IDENTIFIER NUMBER	{ } ; (- + FOR WHILE IF INT FLOAT IDENTIFIER NUMBER
ForStmt	FOR	{ } ; (- + FOR WHILE IF INT FLOAT IDENTIFIER NUMBER
ForStmtEnd	(- + ; IDENTIFIER NUMBER	{ } ; (- + FOR WHILE IF INT FLOAT IDENTIFIER NUMBER
ForStmtEnd1	( ) - + IDENTIFIER NUMBER	{ } ; (- + FOR WHILE IF INT FLOAT IDENTIFIER NUMBER
WhileStmt	WHILE	{ } ; (- + FOR WHILE IF INT FLOAT IDENTIFIER NUMBER
IfStmt	IF	{ } ; (- + FOR WHILE IF INT FLOAT IDENTIFIER NUMBER
IfStmtEnd	ELSE e	{ } ; (- + FOR WHILE IF INT FLOAT IDENTIFIER NUMBER

Table 2.1 continued from previous page

CompoundStmt	{	{ } ; (- + FOR WHILE IF INT FLOAT IDENTIFIER NUMBER
StmtList	{; (- + FOR WHILE IF INT FLOAT IDENTIFIER NUMBER e	}
Expr	(- + IDENTIFIER NUMBER	;)
ExprFol	= * / e	;)
NewRvalue	EG NG IN IE SU SE e	;)
Compare	EG NG IN IE SU SE	(- + IDENTIFIER NUMBER
Mag	(- + IDENTIFIER NUMBER	;) EG NG IN IE SU SE
NewMag	+ - e	;) EG NG IN IE SU SE
Term	(- + IDENTIFIER NUMBER	+ - ;) EG NG IN IE SU SE
NewTerm	* / e	+ - ;) EG NG IN IE SU SE
Factor	(- + IDENTIFIER NUMBER	* / + - ;) EG NG IN IE SU SE

TABLE 2.1: Table DEBUT et SUIVANTs)

3 Analyse semantique

3.1 Routines semantiques

3.1.1 La boucle for

"for" "(" Expr **R1** ";" (Expr() **R2**)? ";" (Expr() **R3**)? ")" Stmt() **R4**

R1 : `adr3=adr2=adr1= getCourant();`

R2 : `adr2 =getCourant(); genereQuad("JZ","","","");` `adr3 =getCourant();`
`genereQuad("JUMP","","","");`

R3 : `genereQuad("JUMP","","",adr1); modifyQuad(adr3 , 4 ,getCourant());`

R4 : `genereQuad("JUMP","","",adr2+2); modifyQuad(adr2 , 4 , getCourant());`

3.1.2 La boucle while

"while" **R1** "(" Expr() ")" **R2** Stmt() **R3**

R1 : `adr1 = getCourant();`

R2 : `adr2 =getCourant(); genereQuad("JZ","","","");`

R3 : `genereQuad("JUMP","","",adr1); modifyQuad(adr2 , 4 ,getCourant());`

3.1.3 Si Sinon

"if" "(" Expr() ")" **R1** "{" Stmt() "}" **R2** (**R3** "else" Stmt() **R4**)?

R1 : `adr1= getCourant(); genereQuad("JZ","","","");`

R2 : `modifyQuad( adr1 , 4,getCourant());`

R3 : `adr2 = getCourant(); genereQuad("JUMP","","","");`

`modifyQuad(adr1 , 4 ,getCourant());`

R4 : `modifyQuad(adr2 , 4 , getCourant());`

3.1.4 L'expression arithmétique binaire(+ -)

f1 = Term() ((s = "+" f2 = Term() | s = "-" f2 = Term() **R1**) **R2**) * **R3**

R1 : t = newTemp() ; genereQuad("NEG" , f2 , "" , t) ; f2 = t;

R2 : t = newTemp() ; genereQuad("ADD" , f1 , f2 , t) ; f1 = t;

R3 : Mag() = f1

3.1.5 L'expression arithmétique binaire (* /)

f1 = Factor() ((s= "*" f2= Factor() | s= "/" f2 = Factor())**R1**)* **R2**

R1 : t = newTemp() ; if (s.kind == MULT) inst = "MUL" ; else inst = "DIV" ;

genereQuad(inst , f1 , f2 , t) ; f1 = t;

R2 Term() = f1 ;

3.1.6 L'expression arithmétique unaire(+ -)

"(" Expr() ")" | (t = <NUMBER> | t = <IDENTIFIER>) **R1**

| (s = "-" f = Factor() **R2** | s = "+" f = Factor()) **R3**

R1 : Factor() = t.image;

R2 : t = newTemp() ; genereQuad("NEG" , f , "" , t) ; f = t;

R3 : Factor() = f;

3.1.7 La comparaison

(t = "==" | t= "!=" | t= "<" | t= "<=" | t= ">" | t= ">=") **R1**

R1 : if (t.kind==EG) inst = "JE";
else if (t.kind ==NE) inst = "JNE";
else if (t.kind==IN) inst = "JL";
else if (t.kind ==IE) inst = "JLE";
else if (t.kind == SU) inst = "JG";
else if (t.kind == SE) inst = "JGE";
else inst="";
Compare()= inst;

4 Les tests

4.1 Test N°=1

```
1   int f (float x)
2   {
3       float y ;
4       x = y + 5 * 5;
5       for (i = 1 ; i<10;i=i+1 )
6       {
7           x=x+5;
8       }
9       }
```

FIGURE 4.1: Le code de test 1

```
java MiniC ../Tests/test1

Mini-C compiler wirtten by Korbaa Hamza , Bouchfirat Loukmane
The compiler created a file ../Tests/test1_code.txt containing the assembl
y code

========================= ASSEMBLY CODE ========================

  NUM        | OP         | S1         | S2         | RES
-------------------------------------------------------------------
  0          | MUL        | 5          | 5          | tem0
  1          | ADD        | y          | tem0       | tem1
  2          | MOV        | tem1       |            | x
  3          | MOV        | 1          |            | i
  4          | cmp        | i          | 10         |
  5          | JL         |            |            | 7
  6          | MOV        | 0          |            | tem2
  7          | JUMP       |            |            | 9
  8          | MOV        | 1          |            | tem2
  9          | JZ         |            |            | 17
  10         | JUMP       |            |            | 14
  11         | ADD        | i          | 1          | tem3
  12         | MOV        | tem3       |            | i
  13         | JUMP       |            |            | 4
  14         | ADD        | x          | 5          | tem4
  15         | MOV        | tem4       |            | x
  16         | JUMP       |            |            | 11
  17         | halt       |            |            | f
```

FIGURE 4.2: Le code produit par le compilateur

4.2 Test N°=2

```
1   int fact (int n)
2   {
3       int i ;
4       i=1;
5       if (n != 0 ){
6           while (i < n )
7           {
8               i = i * ( i + 1) ;
9               i = i +1 ;
10          }
11      }
12  }
13
```

FIGURE 4.3: Le code de test 2

```
java MiniC ../Tests/test2

Mini-C compiler wirtten by Korbaa Hamza , Bouchfirat Loukmane
The compiler created a file ../Tests/test2_code.txt containing the assembl
y code

========================== ASSEMBLY CODE ==========================

  NUM        | OP         | S1         | S2         | RES
-------------------------------------------------------------------
  0          | MOV        | 1          |            | i
  1          | cmp        | n          | 0          |
  2          | JNE        |            |            | 4
  3          | MOV        | 0          |            | tem0
  4          | JUMP       |            |            | 6
  5          | MOV        | 1          |            | tem0
  6          | JZ         |            |            | 19
  7          | cmp        | i          | n          |
  8          | JL         |            |            | 10
  9          | MOV        | 0          |            | tem1
  10         | JUMP       |            |            | 12
  11         | MOV        | 1          |            | tem1
  12         | JZ         |            |            | 19
  13         | ADD        | i          | 1          | tem2
  14         | MUL        | i          | tem2       | tem3
  15         | MOV        | tem3       |            | i
  16         | ADD        | i          | 1          | tem4
  17         | MOV        | tem4       |            | i
  18         | JUMP       |            |            | 7
  19         | halt       |            |            | fact
```

FIGURE 4.4: Le code produit par le compilateur

4.3 Test N°=3

```
1   int  m(int n, float x)
2   {
3       float k ;
4       k= 1;
5       for ( i = 0 ; i< n ;i= i +2 )
6       {
7           k = i * x + 5 * i - (k - x) ;
8       }
9   }
10
```

FIGURE 4.5: Le code de test 3

```
java MiniC ../Tests/test3

Mini-C compiler wirtten by Korbaa Hamza , Bouchfirat Loukmane
The compiler created a file ../Tests/test3_code.txt containing the assembl
y code

========================== ASSEMBLY CODE ==========================

  NUM        | OP         | S1         | S2         | RES
-------------------------------------------------------------------
  0          | MOV        | 1          |            | k
  1          | MOV        | 0          |            | i
  2          | cmp        | i          | n          |
  3          | JL         |            |            | 5
  4          | MOV        | 0          |            | tem0
  5          | JUMP       |            |            | 7
  6          | MOV        | 1          |            | tem0
  7          | JZ         |            |            | 21
  8          | JUMP       |            |            | 12
  9          | ADD        | i          | 2          | tem1
  10         | MOV        | tem1       |            | i
  11         | JUMP       |            |            | 2
  12         | MUL        | i          | x          | tem2
  13         | MUL        | 5          | i          | tem3
  14         | ADD        | tem2       | tem3       | tem4
  15         | NEG        | x          |            | tem5
  16         | ADD        | k          | tem5       | tem6
  17         | NEG        | tem6       |            | tem7
  18         | ADD        | tem4       | tem7       | tem8
  19         | MOV        | tem8       |            | k
  20         | JUMP       |            |            | 9
  21         | halt       |            |            | m
```

FIGURE 4.6: Le code produit par le compilateur

4.4 Test N°=4

```
1   int  function(int a, int b)
2   {
3   |    while (a != b )
4   |    {
5   |    |    a = a + b //  ";" manquante
6   |    }
7   }
8
```

FIGURE 4.7: Le code de test 4

```
java MiniC ../Tests/test4
Encountered " "}" "} "" at line 6, column 5.
Was expecting one of:
    "==" ...
    "!=" ...
    "<" ...
    "<=" ...
    ">" ...
    ">=" ...
    ";" ...
    "+" ...
    "-" ...
    "*" ...
    "/" ...
```

FIGURE 4.8: Le code produit par le compilateur

Bibliographie

[1] BERGMANN, S. D. (2017). Compiler Design : Theory, Tools, and Examples. *Rowan University, bergmann@rowan.edu*, 320.

[2] REIS, A. J. D. (s. d.). Compiler Construction Using Java, JavaCC, and Yace. *EEE ©computer society*, 600-660.