Projet compilation

Rapport intermédiaire de Projet : Projet compilation 2021

Auteurs :
Coline Chataing
Ahmed Ziani
Laury Thiebaux

Encadrants :
Suzanne Collin
Alexandre Bourbeillon
Sebastien Da Silva

12 janvier 2022

Table des matières

Chapitre 1

Introduction

Dans le cadre du projet de compilation, nous effectuons les premières étapes de création d'un compilateur pour le langage CircC, qui est un fragement du langage C. Pour réaliser ce compilateur, nous avons dû d'abord écrire une grammaire reconnaissant ce langage pour ensuite créer les méthodes de construction de l'Arbre Abstrait ainsi que la table des symboles et les contrôles sémantiques nécessaires. Ce compilateur devra également signaler par un message explicite les erreurs syntaxiques, sémantiques et lexicales rencontrées.

Chapitre 2

Grammaire du langage

2.1 Aspects syntaxiques

Le langage CirC créé permet de reconnaitre les constructions suivantes :

Instruction	**Langage Circ**
Les déclarations de variables, les affectations	int a ; int ident = 1 ; int a,b ;
Les structures	struct type var * (int a) ; struct var int a = 1 ; ;
Les conditions	if (condition) {} [else {}]
L'itération	while (condition) {}
Les comparateurs reconnus Les tests d'égalité	+,-, etc ==,>= etc
Le retour	return (expression) ;
Les déclarations de fonction	int a(int b, int c){ }

2.2 Langage CirC

La grammaire du langage CirC est en annexe.

Nous nous sommes tout d'abord basés sur la grammaire donnée dans le sujet, mais nous avons rajouté certaines règles. Nous avons aussi dérécursivé. Des labels ont été ajoutés.

Chapitre 3

Gestion de projet

3.1 Choix de la méthode de fonctionnement

Nous avons décidé de nous inspirer d'une méthode de projet connue : **la méthode Agile**. En effet, nous savons qu'il y aura plusieurs rendus intermédiaires, et la structure du projet se prête bien à une progression se basant sur les différents échelons.

3.1.1 Réunions

L'équipe s'est réunie en distanciel sur la plateforme Discord, mais surtout à l'école en présentiel. Nous avons aussi fait des séances de travail, où nous avancions chacun de notre côté avec la possibilité de se poser des questions.

3.1.2 Langage

Le langage utilisé lors du projet est JAVA, avec utilisation du .jar ANTLR dans sa version 4. Pour rédiger la grammaire de Cir-C, nous avons dû utiliser nos compétences en C notamment pour la création de tests.

3.2 Outils de gestion de projet

3.2.1 La Matrice SWOT

Nous avons construit une matrice SWOT au début de notre projet. Cela nous a permis de voir nos compétences dans certains domaines et ainsi d'être plus efficaces dans la répartition des tâches.

FIGURE 3.1 – Matrice SWOT du projet

3.2.2 Liste des tâches et temps

Ce tableau résume le temps passé par les différents membres du projet sur les différentes tâches. Ne sont pas incluses les réunions, et les séances de TP.

	Ahmed	Laury	Coline
Rédaction de la grammaire initiale	3	3	2
Corrections de la grammaire	10	6	6
Création de test	2	2	2
Création AST	1	4	1
Rédaction du rapport LaTeXet gestion de projet	1	2	6
Total (en heures)	17	17	17

TABLE 3.1 – Matrice RACI

3.2.3 Diagramme de Gantt du projet

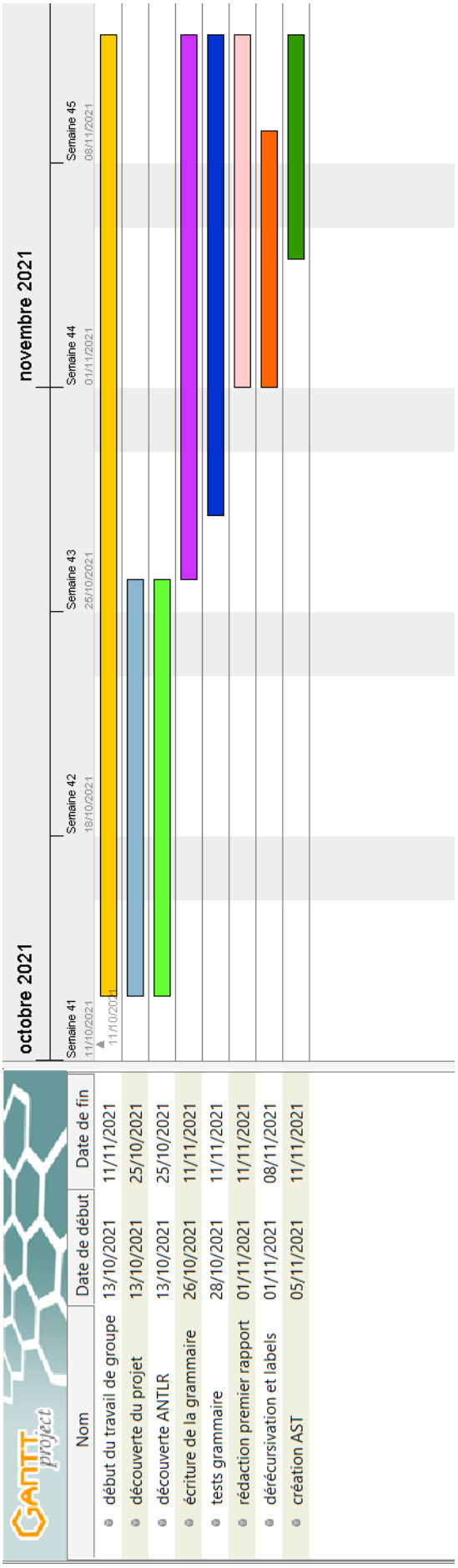

FIGURE 3.2 – Diagramme de Gantt du projet

3.3 Annexe

3.3.1 Grammaire

```
grammar circ;

@header{
package parser;
}

program :
    decl* EOF;

decl :
    decl_typ | decl_fct | decl_vars ;

decl_vars :
      'int' (IDENT',')* IDENT ';'                       #Decla
    | 'struct' IDENT IDENT* ('(' '*' IDENT ')'',')* ';' #Struct
    | 'int' IDENT '=' ENTIER';'                         #DeclaAffect ;

decl_typ :
    'struct' IDENT '{' decl_vars* '}' ';';
decl_fct :
    'int' IDENT '(' (param ',')* param?')' bloc
    | 'struct' IDENT '*' IDENT '(' param ','* ')' bloc;

param :
    'int' IDENT
    | 'struct' IDENT  '*' IDENT;

expr :  ENTIER expr1?                           #EntierExpr
        |IDENT expr1                            #IdentExpr
        | IDENT ('(' (expr',')* ')' expr1? )?   #IdentExprPointeur
        | '!' expr expr1?                       #ExclaExpr
        | '-' expr expr1?                       #TiretExpr
        | 'sizeof' '(' 'struct' IDENT ')' expr1? #SizeofExpr
        | '(' expr ')' expr1?                   #ExprExpr  ;

expr1 : '->' IDENT expr1?           #Fleche
        | OPERATEUR expr expr1?     #OpExpr;

instruction :
     expr ';' #ExprSeule
    | 'if' '(' expr ')' instruction #IfThen
    | 'if' '(' expr ')' instruction 'else' instruction #IfThenElse
    | 'while' '(' expr ')' instruction #While
    | bloc  #BlocInstruct
    | affectation #AffectInstruct
    | 'return' expr ';'    #Return ;

affectation : IDENT '=' expr2 ';' ;

bloc :
    '{' decl_vars* instruction* '}';

OPERATEUR : '=' | '==' | '!=' | '<' | '<=' | '>' | '>=' | '+' | '-' | '*' | '/' | '&&' | '||';


ENTIER : '0' | ('1'..'9') ('0'..'9')* | CARACTERE;
IDENT : ('a'..'z'|'A'..'Z')('a'..'z'|'A'..'Z'|'0'..'9'|'_')*;
CARACTERE : '\''('!'|'#'|'$'|'%'|'('|')'|';'|'+'|','|'-'|'.'|'&'
           | ('0'..'9')|':'|';'|'<'|'='|'>'|'?'|'@'|'['|']'|'^'
           | '_'|'`'|'{'|'}'|'~'|'a'..'z'|'A'..'Z'|'/'|'|')'\''
           ;
WS : ('\n'|' '|'\t'|'\r'|'/*' .*? '*/' | '//' ~[\r\n]*)+ -> skip;
```

FIGURE 3.3 – Grammaire

3.3.2 Tests et arbres

3.3.2.1 Test d'un petit main

```c
int a = 2;
int b = 0;
int c = 15;

/* test de commentaire */

//test de commentaire

int main (){
if (a) {
    b=1+2+3*5+9;
}
else {
    b=2;
}
}
```

FIGURE 3.4 – code du test "good.c"

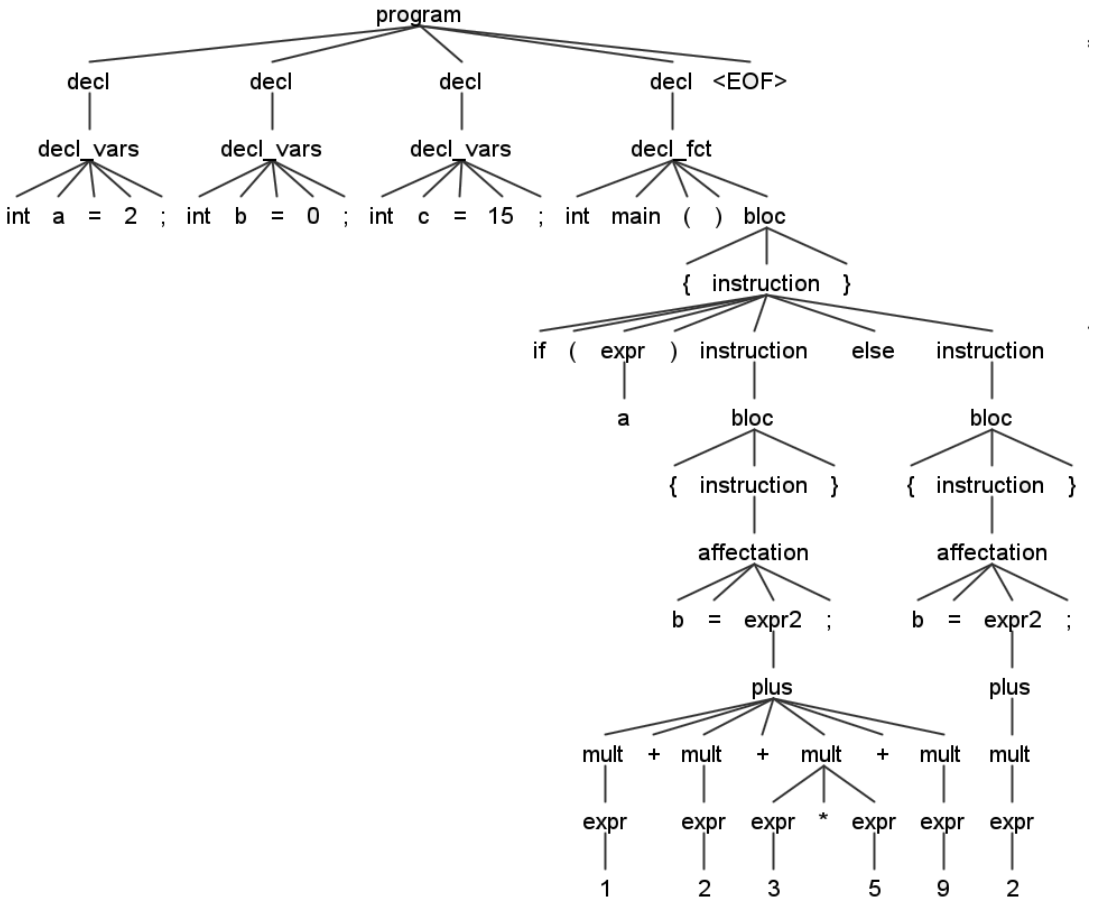

FIGURE 3.5 – arbre obtenu du test "good.c"

3.3.2.2 Test de struct

```c
struct person_t
{
    int nom;
    int tel;
};

struct person_t* my_function (int id) {
    struct person_t p1;
    return sizeof(struct person_t);
}

int a = 2;
int b = 0;


int main (){
if (a) {
    b=1+2;
}
else {
    b=2;
}
}
```

FIGURE 3.6 – code du test "struct.c"

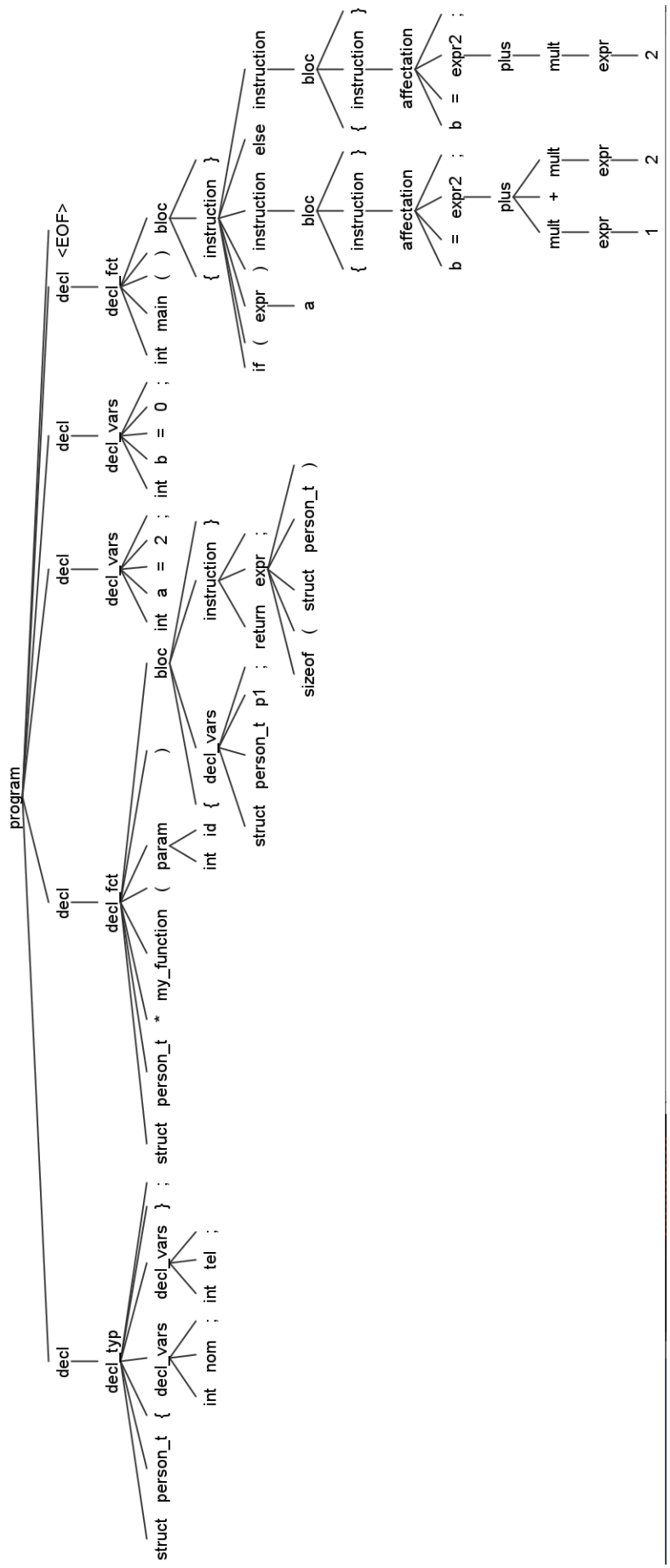

FIGURE 3.7 – arbre obtenu du test struct.c"

3.3.2.3 Test d'une boucle

```
int main(int lignes){
    int compteur = 0;
    while (lignes != compteur){
        compteur = compteur +1;
    }
    return compteur;
}
```

FIGURE 3.8 – code du test "boucle.c"

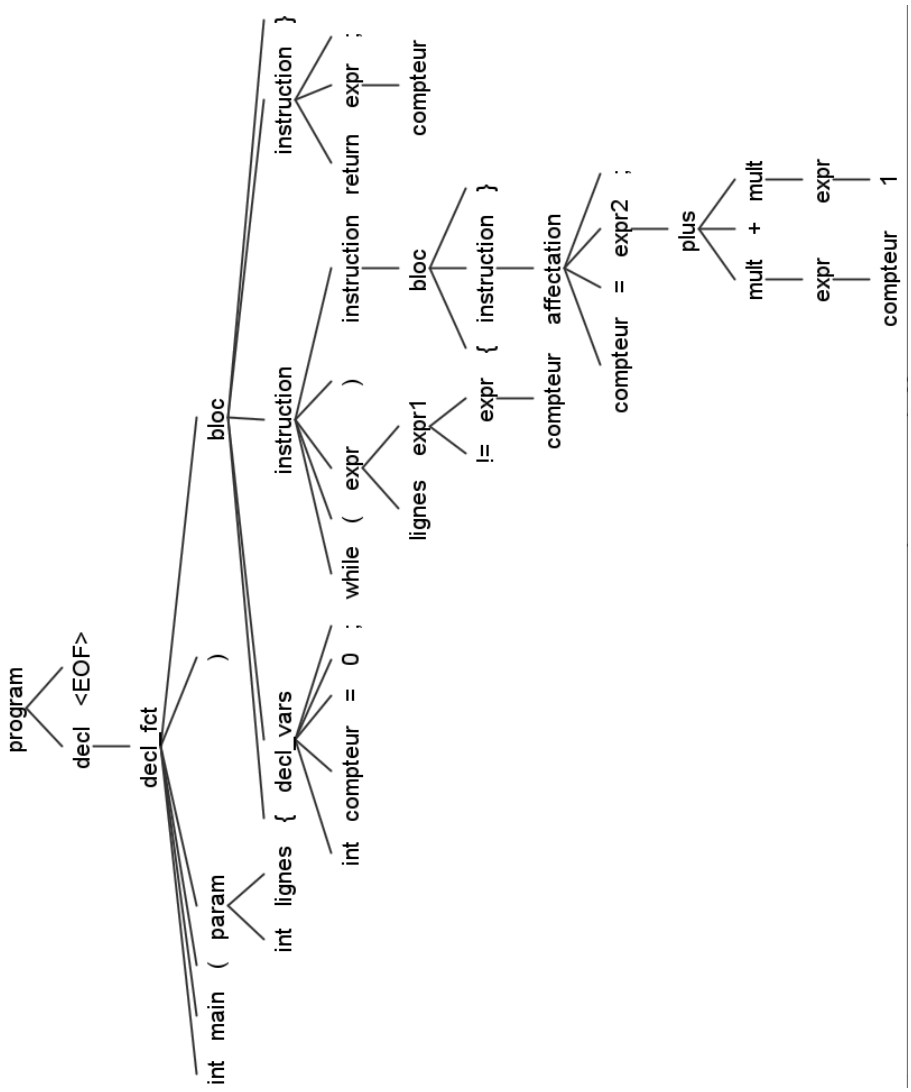

FIGURE 3.9 – arbre obtenu du test boucle.c"

3.3.2.4 Un test volontairement erroné

Ce test est un code auquel il manque une accolade. Il n'est pas reconu par la grammaire, comme attendu.

```
int a = 2;
int b = 0;
int c = 15;

int main (){
if (a) {
    b=1+2+3*5+9;
}
else {
    b=2;

}
```

FIGURE 3.10 – code du test "boucle.c"

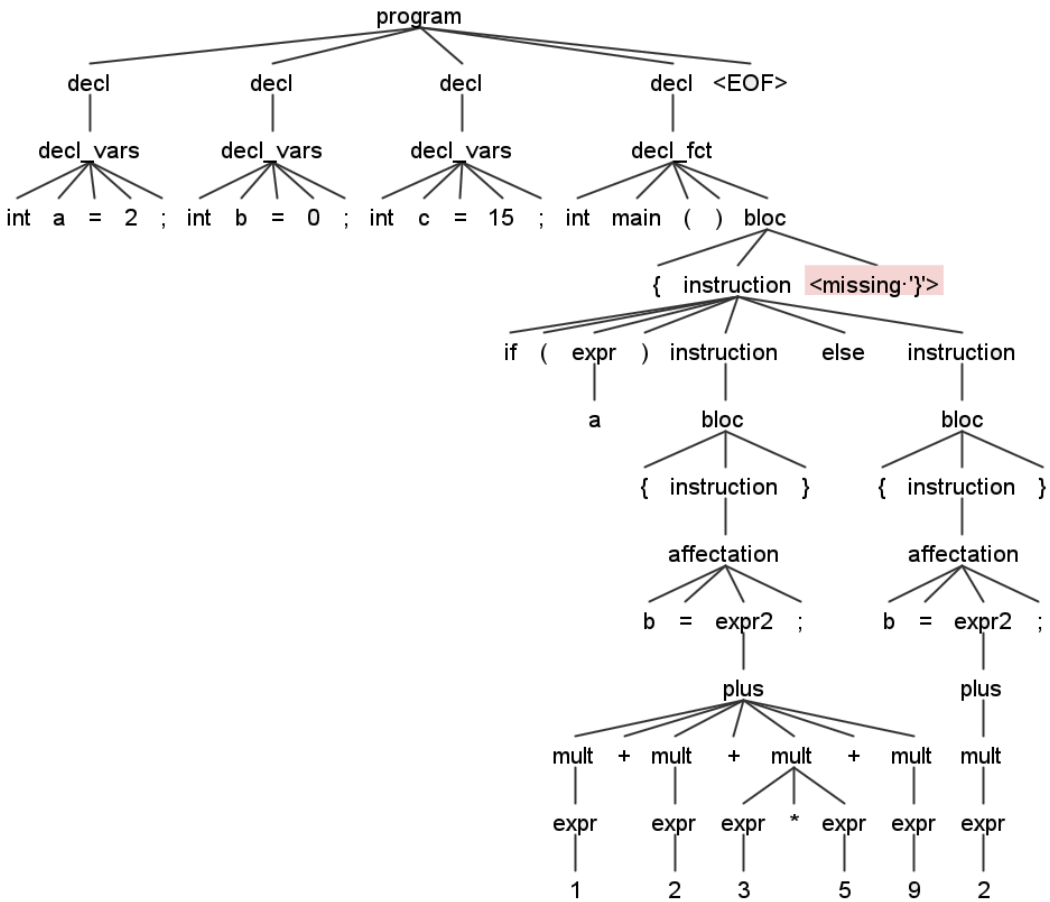

FIGURE 3.11 – arbre obtenu du test boucle.c"