الجمهوريــة الجزائريــة الديمقراطيــة الشـعبية

République Algérienne Démocratique et Populaire

وزارة التعليـم العـالي والبحـث العلـمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

ÉCOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE

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

Projet de compilation

2ème année Cycle Supérieur (2CS)

Groupe : 2CS SIL 2

Thème :

L'analyse sémantique du langage "X" et génération du code intermédiaire

Réalisé par :

Beldjoudi Wassim

Guerraiche Ahmed Amine

Menassel Rayane Ibrahim

Tirouche Mohamed Mahdi

Encadrés par :

ABDMEZIEM Mohamed Riyadh

Promotion : 2024 - 2025

Table des matières

1-Introduction générale:

L'analyse sémantique est une étape essentielle du processus de compilation, car elle garantit que le programme respecte les règles contextuelles du langage, telles que la cohérence des types, la portée des variables et la validité des structures de contrôle.

2-Langage X

Le langage « X » a été conçu pour être simple et structuré. Ses principales caractéristiques incluent :

- **Structure du programme** : en-tête (« program »), définitions (« struct »), variables (« var ») et instructions principales entre « start » et « finish ».
- **Types** : primitifs (« int », « flt », « bool », « str ») et structures complexes.
- **Instructions supportées** : affectations, conditions, boucles, entrée/sortie.
- **Opérateurs** : logiques, arithmétiques et de comparaison.

3-Grammaire du langage "X"

La grammaire de X a été implémentée pour couvrir les principaux aspects du langage, y compris :

- **Structures de contrôle** : conditions (« check », « else »), boucles (« cycle », « while »).
- **Déclarations** : variables simples,structurées et tableaux.
- **Opérations** : arithmétiques, logiques et de comparaison.

4-Table des symboles :

La table des symboles est une structure essentielle dans le processus de compilation, utilisée pour stocker et gérer les informations contextuelles des éléments du programme, comme les variables, tableaux et structures.

Chaque entrée est définie par la structure `SymbolTableEntry`, qui contient les informations suivantes :

- **Nom** : Identifiant du symbole.
- **Type** : Type de données (ex. : `int`, `flt`, etc.).
- **Valeur** : Valeur initiale (si applicable).
- **Ligne** : Numéro de ligne de déclaration.
- **Portée** : Niveau de portée (global ou local).
- **Tableau** : Indique si le symbole est un tableau et sa taille.

- **Structure** : Indique si le symbole fait partie d'une structure et le nom de celle-ci.

symbol_table.h

```h
#define MAX_NAME_LENGTH 50
#define MAX_TYPE_LENGTH 20
#define MAX_VALUE_LENGTH 100

typedef struct {
    char name[MAX_NAME_LENGTH];
    char type[MAX_TYPE_LENGTH];
    char value[MAX_VALUE_LENGTH];
    int line;
    int scope;
    int isArray;
    int arraySize;
    int isStruct;
    char parentStruct[MAX_NAME_LENGTH];
} SymbolTableEntry;

SymbolTableEntry* allocSymbolTable();
SymbolTableEntry* insertEntry(SymbolTableEntry* symbolTable,int* tableSize,char* name,
    char* type,char* value,int line,int scope,int isArray,int arraySize,int isStruct,
    char* parentStruct
);
SymbolTableEntry* searchSymbol(SymbolTableEntry* symbolTable, int tableSize, char* name);
void updateEntry(SymbolTableEntry* symbolTable, int tableSize, char* name, int scope,
     char* value, char* parentStruct);
void printSymbolTable(SymbolTableEntry* symbolTable, int tableSize);
void freeSymbolTable(SymbolTableEntry* symbolTable);

#endif // SYMBOL_TABLE_H
```

5-Langage intermédiaire et routines sémantiques:

5.1.Quadruplets :

Pour les quadruplets , voici les fonctions utilisées :

quadruplets.h > QUAD_H

```h
#ifndef QUAD_H
#define QUAD_H

typedef struct quadruplet {
    char op[20];    // Opérateur
    char opr1[20];  // Premier opérande
    char opr2[20];  // Deuxième opérande
    char res[20];   // Résultat
} quadruplet;

typedef struct quad_list {
    quadruplet* quads;
    int size;
    int capacity;
} quad_list;

// Global quad list
extern quad_list* global_quad_list;
extern int temp_var_counter;

// Function declarations
quad_list* init_quad_list();
void add_quad(quad_list* list, char* op, char* opr1, char* opr2, char* res);
void update_quad(quad_list* list, int index, char* op, char* opr1, char* opr2, char* res);
char* new_temp();
void print_quads(quad_list* list);
void free_quad_list(quad_list* list);

#endif
```

5.2.Exemple de traduction: Instructions conditionnelles

Pour cette instruction , on a introduit 3 routines permettant à la fois de :

- **Générer les quadruplets .**
- **Vérification des types.**
- **Mettre à jour les adresses de branchement.**

```
376  if_else_stmt:
377      DEBUT_INST_IF_ELSE ELSE block
378      {
379          char next_saut[20];
380          sprintf(next_saut, "%d", global_quad_list->size);
381          strcpy(global_quad_list->quads[sauv_fin_else].opr1 , next_saut);
382      };
383
384  DEBUT_INST_IF_ELSE:
385      DEBUT_IF_ELSE block
386      {
387          add_quad(global_quad_list,"BR","","","");
388          char temp[20];
389          sprintf(temp, "%d", global_quad_list->size);
390          strcpy(global_quad_list->quads[sauv_else].opr1 , temp);
391          sauv_fin_else = global_quad_list->size-1;
392      }
393  ;
394
395  DEBUT_IF_ELSE:
396      CHECK LPAREN expression RPAREN
397      {
398          sauv_else = global_quad_list->size-1;
399          sauv_fin_if = global_quad_list->size-1;
400      };
```

6-Tests et validations:

- **Les commandes à exécuter :**

$ bison -d x_parser.y

$ flex x_lexer.l

$ gcc -o x_compiler x_parser.tab.c lex.yy.c symbol_table.c quadruplets.c -lfl -lm

$ ./x_compiler programme_test.x

- **Le programme de test**

Un programme de test a été écrit pour valider l'analyse sémantique. Le fichier d'entrée suit la syntaxe du langage X et comprend diverses structures comme des boucles, des conditions, et des déclarations.

```
program langageX.

struct {
    etudiant = {
        int id,
        str nom
        }.
}
var{
    int A.
    etudiant mahdi.
    bool boolean.
}

start
    A = 5.

    check (A >= 7) {
        boolean = true.
    } else {
        mahdi->id = 2025.
        mahdi->nom = "Mahdi".
    }
    cycle (A from 1 to 5 by 1) {
        mahdi->id = mahdi->id + 5.
    }

finish
```

- **Résultats :**

```
Programme reconnu : langageX

-----------------------------------------Symbol Table-----------------------------------------------------------------
Name            | Type         | Value      | Line | Scope | isArray | ArraySize | isStruct | ParentStruct         |
-----------------------------------------------------------------------------------------------------------------------
id              | int          |            | 5    | 1     | 0       | 0         | 0        | etudiant             |
nom             | str          |            | 6    | 1     | 0       | 0         | 0        | etudiant             |
etudiant        | struct       |            | 7    | 0     | 0       | 0         | 1        |                      |
A               | int          |            | 10   | 0     | 0       | 0         | 0        |                      |
mahdi           | etudiant     |            | 11   | 0     | 0       | 0         | 1        |                      |
boolean         | bool         |            | 12   | 0     | 0       | 0         | 0        |                      |

Analyse syntaxique terminée avec le code : 0

=== Quadruplets Generated ===
0: (=, 5, , A)
1: (BL, 4, A, 7)
2: (=, true, , boolean)
3: (BR, 6, , )
4: (=, 2025, , id)
5: (=, "Mahdi", , nom)
6: (=, 1, , A)
7: (+, id, 5, t1)
8: (=, t1, , id)
9: (+, A, 1, t2)
10: (=, t2, , A)
11: (BLE, 7, A, 5)
```