Projet de réalisation d'un mini compilateur pour un mini Langage avec les Outils FLEX et BISON

I- Introduction

Ce projet a pour but de concevoir un mini-compilateur pour un langage nommé «**MiniSoft**», en intégrant les différentes étapes de la compilation : l'analyse lexicale, syntaxique, sémantique, ainsi que la génération de code intermédiaire.
De plus, la gestion parallèle de la table des symboles et le traitement des différentes erreurs devront être pris en charge lors des phases d'analyse du processus de compilation.

II- Description du mini langage « MiniSoft »

II.1. Structure générale

La structure générale du programme est la suivante :

```
MainPrgm  L3_software ;
Var
        < !- Partie déclaration - !>
BeginPg
{
          {-- Partie Instruction --}
}
EndPg ;
```

II.2. Partie déclaration

Cette section est dédiée à la déclaration de toutes les variables (simples ou tableaux) et constantes nécessaires au bon fonctionnement du programme.

1. Déclaration de variables

La syntaxe de déclaration d'une variable est la suivante :

- **Variable simple :**

let <nom_var>: <type> ;

- **Tableau :**

let <nom_var>: **[**<type>**;**<taille>**]**;

<nom_var> est un identificateur (**voir section II.2.3**).

<type> est le nom du type de la variable (**voir section II.2.4**).

Projet de réalisation d'un mini compilateur pour un mini Langage avec les Outils FLEX et BISON

NB : <taille> d'un tableau est un entier positif.

⇨ La **déclaration** d'un **ensemble de variables** s'effectue en séparant les noms de variables par une virgule.

Exemple :

< ! - Déclaration de 3 variables - !>

let x,y,z : Int ;

< !- Déclaration de 2 tableaux de même taille et type !->

let A,B :[Int ; 10] ;

2. Déclaration d'une constante

La syntaxe de déclaration d'une constante est la suivante :

@define Const <nom_const> : <type>=Valeur ;

Exemple :

@define Const Software : Int = 91 ;

⇨ Valeur est une constante.

3. Identificateur

Un identificateur est une suite de lettres minuscules, de chiffres et/ou de tirets bas (« _ »), qui commence par une lettre alphabétique, ne dépasse pas 14 caractères, ne se termine pas par un tiret bas et ne contient pas de tirets bas consécutifs.

⇨ Les noms de programme, ainsi que les noms de variables et de constantes, sont des identificateurs.

Exemple: Mon_program, Abbb, achmb, ….

4. Types

Le type peut être : Int, Float.

Type	Description	Exemple
Int	Une constante entière est une suite de chiffres. Elle peut être signée ou non signée tel que sa valeur est entre **-32768** et **32767**. Si la constante entière est signée, elle doit être mise entre parenthèses.	**5, (-5), (+5)**
Float	Une constante réelle est une suite de chiffres contenant le point décimal. Elle peut être signée ou non signée. Si la constante réelle est signée, elle doit être mise entre parenthèses.	**5.5, (-5.76), (+ 568.77)**

Projet de réalisation d'un mini compilateur pour un mini Langage avec les Outils FLEX et BISON

II.3. Partie Instruction

Les instructions autorisées dans notre langage sont :

Instruction	Description	Exemple
Affectation	Idf **:=** Expression ;	A : = 23 ; B [5] : = (H+ 2) / (5 – b +(-3)) ;

Condition	**if** (condition) **then**
	{
	< ! - Instructions -!>
	} **else** {
	<! - Instructions - !>
	}
		if (Aa +2 > Bb) **then**
		{
		H : = J – 5.5;
		} **else** {
		H : = H-1 ;
		}

Boucle	**do** {
	< ! - Instruction - !>
	} **while** (condition) **;**
		do {
		x: =x+1;
		} **while** (x<100);

	for <idf> **from** <initialisation> **to** <condition_arrêt> **step** <pas> {
	< ! - Instruction - !>
	}
		for i **from** 1 **to** 10 **step** 2 {
		Ka := Hj * i;
		}

Lecture et écriture (input, output)	L'**output** permet d'afficher des informations à l'écran, tandis que l'**input** permet à l'utilisateur de saisir des données.	**input**(UserName) ; **output**("Vous avez saisi: ", UserName) ;

Important :

- Une condition est une expression qui renvoie '1' ou '0'. Elle peut prendre la forme d'une expression logique ou de comparaison.
- On peut avoir des **boucles** imbriquées.
- Toute instruction doit se terminer par un **point-virgule**.

Opérateurs

1. **Opérateurs arithmétiques**

   + , - , * , /

2. **Opérateurs logiques**
   - **(**expression1 **AND** expression2**) :** le **et** logique**.**
   - **(**expression1 **OR** expression2**) :** le **ou** logique**.**
   - **!(expression) :** la négation**.**
3. **Opérateurs de comparaison**
   - expression1> expression2

Projet de réalisation d'un mini compilateur pour un mini Langage avec les Outils FLEX et BISON

- expression 1< expression2
- expression1 >= expression2
- expression1 <= expression2
- expression1 == expression2
- expression1 != expression2

⇨ **Associativité et priorité des opérateurs**

⇨ **Associativité :** gauche pour tous les opérateurs.

⇨ **Priorité :** les priorités sont données par la table suivante par ordre croissant :

Opérateurs Logiques	OR
	AND
	!
Opérateurs de comparaison	< > >= <= == !=
Opérateurs arithmétiques	+ -
	* /

Priorité ↓

II.4. Commentaires

Un commentaire peut être écrit sur une seule ligne en mettant le texte entre < !- et - !>

ou sur plusieurs lignes en le mettant entre **{--** et **--}** .

Exemple :

< ! - Ceci est un commentaire sur une seule ligne ! ->

{--

Ceci est un commentaire sur

Plusieurs lignes

--}

III- Travail à réaliser

Ci-dessous les différentes briques à implémenter afin de réaliser le compilateur demandé :

1. **Analyse lexicale avec l'outil FLEX**
2. **Analyse syntaxique avec l'outil Bison**
3. **Analyse sémantique pour les erreurs suivantes :**
   - Idf non déclaré
   - Idf double déclarée
   - Non-compatibilité de type
   - Division sur 0 dans le cas d'une constante.

Projet de réalisation d'un mini compilateur pour un mini Langage avec les Outils FLEX et BISON

- Modification de la valeur d'une constante.
- Dépassement de la taille d'un tableau.

- **Gestion de la table de symboles :** La table de symboles, créée lors de l'analyse lexicale, regroupe les informations nécessaires sur les variables, constantes et tableaux, et est mise à jour au fil de la compilation. Elle doit avoir au minimum les champs suivants :
  - ⇨ **NomEntité, CodeEntité, Valeur …etc**

- Génération des quadruplets.

4. Traitement des erreurs :

Il est demandé d'afficher les messages d'erreur appropriés à chaque étape du processus de compilation, en précisant le type d'erreur, l'entité concernée, ainsi que le numéro de ligne et la colonne où l'erreur se produit.

BON COURAGE

Par : SAYOUD Lynda