Université des Sciences et de la Technologie Houari Boumediene
Faculté d'Electronique et d'Informatique
Département d'Informatique

Licence 3 ACAD — Année Universitaire : 2022/2023
Module Compilation

Projet de compilation du langage minipy
Avec les Outils FLEX et BISON

1. Introduction :

Le but de ce projet est de réaliser un mini-compilateur en langage C effectuant une analyse lexicale en utilisant l'outil FLEX et une analyse syntaxico-sémantique en utilisant l'outil BISON ainsi que la génération du code intermédiaire.

La génération de la table des symboles ainsi que le traitement des différentes erreurs doivent être également réalisés.

2. Description du Langage minipy:

Un programme en minipy est composé d'une suite de déclarations et d'instructions. Chaque instruction doit être sur une seule ligne (et elle ne se termine pas par un ' ;').

Les blocs sont identifiés par **l'indentation**, (quatre espaces) au lieu d'accolades comme en C. Une augmentation de l'indentation marque le début d'un bloc, et une réduction de l'indentation marque la fin du bloc courant.

Exemple de programme en minipy

```
#Le programme calcule la somme des x premiers termes d'une suite
# arithmétique
int x=50
resultat =0
while (x!=0) :
   resultat = resultat + x
   x = x-1
```

2.1. Commentaire :

Un commentaire est précédé par un '#'. Il doit être ignoré par le compilateur.

Exemple de commentaire

```
# ceci est un commentaire
```

2.2. Déclarations :

Une déclaration peut être de variables simples (entiers, réels, caractères, booléen) ou bien de variables structurées (tableaux).

Le type peut ne pas être spécifié mais dans ce cas, la variable doit être initialisée avec une valeur (avec laquelle on devinera le type) lors de la déclaration.

2.2.1. Les types

Le type peut être : int , float, char, bool.

Type	Description
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
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.
char	Une variable de type char représente un caractère
bool	Une variable de type bool peut prendre deux valeurs : **true**, **false**

2.2.2. Identificateur

Un identificateur est une suite alpha-numérique qui commence par une lettre majuscule suivie d'une suite de chiffres et lettres minuscules. Un IDF ne doit pas contenir plus de 8 caractères.

2.2.3. Déclaration des variables de type simple

La déclaration de variables a les deux formes.

Forme	Description	Exemple
1	**TYPE** *liste_variables*	int x int y, z
2	*nom_variable1* = **val1**	x=100 y='C'

2.2.4. Déclaration des tableaux

La déclaration d'un tableau a la forme suivante:

Description	Exemple
type nom [taille]	bool Tab[10]

2.2.5. Operateurs

Il y a trois types d'opérateurs : **arithmétiques, logiques et de comparaisons.**

Type	Operateurs
Arithmétiques	**+, - , *, /**
logiques	**and, or , not**
Comparaison	>, <, >=, <=, ==, !=

- ***Associativité et priorité des opérateurs :***

Les associativités et les priorités des opérateurs sont données par la table suivante par ordre croissant :

Opérateur		Associativité
Opérateurs de comparaison Opérateurs Arithmétiques	> >= == != <= <	Gauche
Opérateurs de comparaison Opérateurs Arithmétiques Opérateurs de comparaison	+ -	Gauche
	* /	Gauche

2.2.6. Les conditions

Une condition est une expression qui renvoie '1' ou '0'. Elle peut prendre la forme d'une expression de comparaison ou une expression logique.

2.3. Instructions

Affectation	
Description	**Exemple**
Idf=expression	A=(X+7+B)/ (5,3-(-2)) A= 'c'

IF ELSE	
Description	**Exemple**
if (condition) : instruction 1 instruction2 **else :** instruction 3 instruction4 **Note :** Le premier bloc est exécuté ssi la condition est vérifiée. Sinon le bloc « else » sera exécuté s'il existe.	if (Aa > Bb): Cc=E+2.6 else : Cc=0

Boucle For version 1	
Description	**Exemple**
for nom-variable **in range (**borne-inf **,** borne-sup**) :** liste instructions **Note :** la variable prend successivement les valeurs entre borne-inf et borne-sup. On peut avoir des boucles imbriquées.	y=0 for I in range (0,5) : y=y+i

Boucle For version 2	
Description	**Exemple**
for nom-variable **in** nom-tableau **:** liste instructions **Note :** la variable prend successivement la valeur d'éléments du tableau de l'index 0 jusqu'au dernier élément. On peut avoir des boucles imbriquées.	y=0 for I in tab: y=y+i

Boucle while	
Description	**Exemple**
while (Condition) **:** instruction 1 instruction 2 **Note :** le bloc d'instructions est exécuté ssi la condition est vérifiée. On peut avoir des boucles imbriquées.	while (i <n) : i=i+2

3. Analyse Lexicale avec l'outil FLEX :

Son but est d'associer à chaque mot du programme source la catégorie lexicale à laquelle il appartient. Pour cela, il est demandé de définir les différentes entités lexicales à l'aide d'expressions régulières et de générer le programme FLEX correspondant.

4. Analyse syntaxico-sémantique avec l'outil BISON :

Pour implémenter l'analyseur syntaxico-sémantique, il va falloir écrire la grammaire qui génère le langage défini au-dessus. La grammaire associée doit être LALR. En effet l'outil BISON est un analyseur ascendant qui opère sur des grammaires LALR. Il faudra spécifier dans le fichier BISON les différentes règles de la grammaire ainsi que les règles de priorités pour les opérateurs afin de résoudre les conflits. Les routines sémantiques doivent être associées aux règles dans le fichier BISON.

5. Gestion de la table de symboles :

La table de symboles doit regrouper l'ensemble des variables et constantes définies par le programmeur avec toutes les informations nécessaires pour le processus de compilation. Il est demandé de prévoir des procédures pour permettre de **rechercher** et d'**insérer** des éléments dans la table des symboles.

6. Génération du code intermédiaire

Le code intermédiaire doit être généré sous forme de quadruplets.

7. Traitement des erreurs :

Il est demandé d'afficher les messages d'erreurs adéquats à chaque étape du processus de compilation. Ainsi, lorsqu'une erreur lexicale ou syntaxique est détectée par votre compilateur, elle doit être signalée le plus précisément possible, par sa nature et sa localisation dans le fichier source. On adoptera le format suivant pour cette signalisation :

Type_ de_ l'erreur, line 4, colonne 56, entité qui a générée l'erreur.