

ED CHAINE DE PRODUCTION DE PROGRAMMES

Exercice 1 : Petit compilateur

On considère un petit langage simple défini comme suit:

```

<instruction> ::= <affectation> | <conditionnelle>
<conditionnelle> ::= ? <condition> : <affectation>
<condition> ::= <expression> {=|#} <expression>
<affectation> ::= <identificateur> = <expression>
<expression> ::= <facteur> {+|-} <expression> | <facteur>
<facteur> ::= <terme> {*|/} <facteur> | <terme>
<terme> ::= <identificateur> | <nombre> | ( <expression> )

```

Les identificateurs sont des unités lexicales constituées soit d'une lettre unique, soit d'une lettre suivie d'un chiffre. Les nombres sont également des unités lexicales constituées de chiffres; on ne prend en compte que des entiers positifs ou nuls. Ces unités lexicales peuvent donc être définies par les règles suivantes:

```

<identificateur> ::= <lettre> | <lettre><chiffre>
<nombre> ::= <chiffre><nombre> | <chiffre>

```

Les espaces ne peuvent se trouver à l'intérieur d'une unité lexicale, et n'ont pas de signification. Une instruction doit se trouver toute entière sur une seule ligne, et une ligne ne peut contenir qu'une seule instruction.

A- L'analyse lexicale consiste à reconnaître la suite des unités lexicales d'une ligne.

Donner le résultat fourni par l'analyseur lexical pour les lignes suivantes:

```

A1 = A * 100 + 2*B
? A1 = B : C = D
?A=3
AB=32
A = 2 ! 3

```

B- L'analyse syntaxique consiste à vérifier la concordance de la suite des codes syntaxiques telle qu'elle résulte de l'analyse lexicale avec les règles de syntaxe du langage. Elle permet aussi de retrouver la structure syntaxique du "programme".

Donner la structure syntaxique pour chacune des lignes suivantes:

```

A1 = A * 100 + 2*B
? A1 = B : C = D
?A=3
AB=32
A = 2 ! 3

```

Exercice 2 : Compilation

Soit le langage défini par les règles de Backus-Naur suivantes:

```

< programme > ::= PROGRAM < nom de programme > < corps de programme >
< corps de programme > ::= < suite de declarations > < suite d'instructions > FIN
< suite de declarations > ::= < declaration > | < declaration > < suite de declarations >
< declaration > ::= < identificateur > : REEL := < valeur réelle > ; | < identificateur > : ENTIER :=
< valeur entière > ;
< suite d'instructions > ::= < instruction > | < instruction > < suite d'instructions >
< instruction > ::= < addition > | < multiplication >
< identificateur > ::= < lettre >
< valeur entière > ::= < chiffre >
< valeur réelle > ::= < chiffre > , < chiffre >
< nom de programme > ::= < lettre > < chiffre >
< lettre > ::= A | B | C | D | E... | X | Y | Z
< chiffre > ::= 0 | 1 | 2 | 3 | 4... | 9

```

A. Une addition est de la forme $A := B + 3$; ou $A := B + 3,2$;

A et B sont des identificateurs, 3 est un entier et 3,2 est un réel.

Une multiplication est de la forme suivante : $A := B * C$; A, B et C sont des identificateurs.

Ecrivez la règle BNF donnant la syntaxe d'une addition et celle donnant la syntaxe d'une multiplication.

B. Soit à présent le programme suivant:

```

PROGRAM A3 /
  A : REEL := 3,2;
  B : ENTIER := 3,2;
  A := A + 5,1 ;
  B := B * 61,
  FIN

```

Entourez les unités lexicales reconnues. Signalez les éventuelles erreurs lexicales rencontrées à ce niveau.

C. Pour chaque phrase du programme donnée ci-dessous, donnez l'arbre syntaxique correspondant. Signalez les éventuelles erreurs syntaxiques rencontrées à ce niveau.

```

Phrase 1 A : REEL := 3,2;
Phrase 2 B : ENTIER := 3,2;
Phrase 3 A := A + 5,1;
Phrase 4 B := B * 61,

```

Exercice 3 : Étude du chargeur et de l'éditeur de liens

Soit une machine disposant d'une mémoire de 10000 emplacements. Chaque emplacement peut contenir un entier, qui peut représenter une donnée ou une instruction.

La machine dispose de 5 instructions qui sont codées comme suit:

```

LOAD <adresse>      10000 + adresse      met MEMOIRE[adresse] dans
l'accumulateur
ADD <constante>    20000 + constante    ajoute la constante à l'accumulateur
MUL <constante>    30000 + constante    multiplie l'accumulateur par la
constante
STORE <adresse>    40000 + adresse      met l'accumulateur dans
MEMOIRE[adresse]
JMP <adresse>      50000 + adresse      aller à l'instruction située à
adresse

```

On considère le module suivant:

```

      NOM      ESSAI  nom du module
      PUBLIC   INCR
      EXTERN   SUITE
A:    WORD     =25    réservation d'un emplacement pour A initialisé à 25
B:    WORD     1      réservation d'un emplacement pour B non initialisé
INCR: LOAD    A
      MUL      10
      ADD      1
      STORE    B
      JMP      SUITE
      FIN

```

Le résultat d'une compilation est un fichier dont les enregistrements décrivent non seulement la traduction du module source dans le langage de la machine, mais aussi les informations nécessaires pour relier ce module aux autres modules qui constitueront un programme. Ces enregistrements peuvent prendre diverses formes, suivant qu'ils contiennent le code ou les données du module, ou servent à l'établissement des liaisons intermodules.

Les enregistrements générés par le compilateur sont les suivants :

```

entête:          ( identificateur_module : t_identificateur;
                  taille : entier );
lien_à_satisfaire: ( identificateur_l_a_s : t_identificateur );
lien_utilisable:  ( identificateur_l_u : t_identificateur;
                  valeur_l_u : entier );
code_absolu, code_à_translater:
                  ( adresse_code : entier;
                    valeur_code : entier );
code_avec_l_a_s:  ( adr_code_l_a_s : entier;
                  numéro_l_a_s : entier;
                  valeur_code_l_a_s : entier );
adresse_lancement: ( adr_lanc : entier );
fin_module:      ( ) ;

```

1. Le champ `taille` de l'enregistrement `entête` précise le nombre d'emplacements occupés par le module.
2. Les liens à satisfaire sont implicitement numérotés, à partir de 1, suivant l'ordre d'apparition, dans le module, des enregistrements `lien_à_satisfaire` qui les contiennent.

A.1- Donner la suite d'enregistrements que doit produire l'assembleur pour ce module.

A.2- Lors d'une édition de liens d'un programme, le module `ESSAI` est placé à l'emplacement relatif 123 dans le programme exécutable. `SUITE` est un lien utilisable dans un autre module, à l'emplacement relatif 8. Cet autre module est placé immédiatement derrière le module `ESSAI` dans le programme exécutable. Donner la suite des enregistrements produits par l'éditeur de liens en provenance du module `ESSAI`.

A.3- Le chargeur place le programme à l'emplacement 1042 de la mémoire. Donner le contenu des emplacements de la mémoire qui contiennent ce module.

B- La structure de ces enregistrements est utilisable aussi bien dans les modules objets, entre les compilateurs et l'éditeur de liens, que dans les programmes exécutables, entre l'éditeur de liens et le chargeur. Expliquer pourquoi. Quels enregistrements ne doivent pas se trouver dans un programme exécutable?

Exercice 4 : Exemple d'édition de liens

On dispose d'un ensemble de modules définis comme suit:

```

module PROGRAMME  taille:      332
                   liens à satisfaire:  OUVRIR
                                           LIRE
                                           FERMER
                                           EDITER
                   adresse lancement:   133
module ETIQUETTE  taille:      128
                   liens utilisables:   NOM          0
                                           SOCIETE     32
                                           ADRESSE     64
                                           CODEPOST    96
                                           VILLE       101
module LECTURE    taille:      840
                   liens utilisables:   OUVRIR        0
                                           LIRE         340
                                           FERMER       732
                   liens à satisfaire:  NOM
                                           SOCIETE
                                           ADRESSE
                                           CODEPOST
                                           VILLE
module IMPRESSION taille:      212
                   liens utilisables:   IMPRIMER      0
module EDITION    taille:      642
                   liens utilisables:   EDITER         0
                   liens à satisfaire:  NOM
                                           SOCIETE
                                           ADRESSE
                                           CODEPOST
                                           VILLE
                                           IMPRIMER

```

On effectue l'édition de liens des modules `PROGRAMME`, `ETIQUETTE`, `LECTURE`, `IMPRESSION` et `EDITION`. Donner, en justifiant brièvement votre réponse:

- les adresses d'implantations de ces modules,

- la taille totale du programme résultant,
- la table des liens,
- l'adresse de lancement du programme résultant.