École Nationale d'Ingénieurs de Tunis
Département TIC
Sections : 2Info

A.U. 2023/2024

Compilation
Série 1

1. A quelle phase de la compilation peut-on détecter les erreurs suivantes ?
   - Identificateur invalide : `$x`.
   - Parenthèse manquante.
   - La fonction ne retourne pas de valeur.
   - Indice de tableau non valide.
   - Paramètre d'appel de fonction manquant.
   - Mot clé du langage utilisé comme identificateur.
   - Tentative de modification d'une constante.
   - Variable non déclarée.
   - Opérateur manquant.
   - Instruction non atteignable.
   - Commentaire non fermé.
   - `sqrt("a*a+b*b")`.

2. On désire concevoir un analyseur lexical pour vérifier le format des notes que l'enseignant attribue à ses étudiants. Ces notes sont supposées vérifier les spécifications suivantes :

   - Les notes sont codées par les chiffres de `0` à `9` et la virgule (`,`).
   - Les notes sont comprises entre 0 et 20.
   - La partie entière des notes doit toujours contenir deux chiffres. Si la note est inférieure à 10 alors le chiffre le plus à gauche est obligatoirement 0 et si la note est inférieure à 1 alors la partie entière est composée de deux 0.
   - L'enseignant est supposé noter au demi-point près, c'est-à-dire que la partie décimale, si elle n'est pas nulle, est codée par le seul chiffre 5. Si la partie décimale est nulle alors ni la virgule ni la partie décimale ne doivent figurer dans la note.

   Proposez une définition régulière dénotant ces notes ainsi qu'un automate fini déterministe qui les reconnaît.

3. Un langage, représentant un petit sous-ensemble du langage Pascal, est constitué par les unités lexicales suivantes :
   - Des identificateurs (`IDENT`) commençant par une lettre suivie d'une combinaison quelconque de lettres ou de chiffres.
   - Des constantes numériques entières non signées (`CONST`) sans limitation de longueur.
   - L'affectation `:=`

   Ces unités sont combinées pour former des instructions (`INSTR`) de la forme :

   ```
   IDENT := CONST;
   ```

Construire l'analyseur lexical correspondant à ce langage en utilisant l'approche basée sur les définitions régulières et les automates à états finis.

4. Il s'agit de concevoir un automate fini déterministe reconnaissant le langage des entiers sans signes et sans exposant codés en bases 2, 8, 10 ou 16. Ce langage est dénoté par la définition régulière suivante :

```
Bin = 0-1
Oct = 0-7
Dec = 0-9
Decnn = 1-9
Hex = Dec | a-f | A-F
EntBin = 0(b | B)(_? Bin)+
EntOct = 0(o | O)(_? Oct)+
EntHex = 0(x | X)(_? Hex)+
EntDec = Decnn(_? Dec)* | 0+(_? 0)*
Ent = EntDec | EntHex | EntOct | EntBin
```

   (a) Proposez un automate fini déterministe reconnaissant le langage dénoté par `EntBin`. Faite de même pour les langages dénotés par `EntOct`, `EntHex` et `EntDec`.

   (b) En combinant les automates répondant à la question précédente, déduisez un automate fini (non déterministe) reconnaissant le langage dénoté par `Ent`.

   (c) Déduisez un automate fini déterministe reconnaissant le langage dénoté par `Ent`.

5. On se propose de reconnaître le langage des mots codant des sommes en dollar. Un mot de ce langage est représenté par un nombre décimal positif arrondi au centième le plus proche. Chaque nombre est précédé (à gauche) par le caractère \$, avec une virgule séparant chaque groupe de trois chiffres se trouvant à gauche du point décimal, plus deux chiffres à droite du point décimal, par exemple \$8,937.43 et \$7,777,777.77.

   Proposez un automate fini déterministe qui reconnaît ce langage.

6. La reconnaissance des commentaires compris entre deux suites de symboles R_1 et R_2 peut se faire par un automate reconnaissant le langage dénoté par $R_1(\overline{\Sigma^* R_2 \Sigma^*})R_2$ qui s'interprète comme suit : lire R_1, lire une expression ne comportant pas R_2 puis lire R_2. Pour construire un tel automate, on se propose de suivre les étapes suivantes :

   (a) Construire un automate reconnaissant $R_2 = */$.

   (b) Construire un automate reconnaissant $\Sigma^* R_2 \Sigma^*$.

   (c) Rendre ce dernier automate déterministe et complet puis l'inverser.

   (d) Construire un nouvel automate en ajoutant $R_1 = /*$ au début et R_2 à la fin.

   (e) Rendre l'automate obtenu déterministe

7. Les langages C^{++} and Java offrent au programmeur la possibilité d'écrire des commentaires se limitant à une seule ligne. De tels commentaires sont délimités par la suite de symboles `//`, en début de commentaire, et par le retour à la ligne (`\n`) en fin de commentaire. Le langage de tous ces commentaires est alors dénoté par l'expression régulière

`//(^\n)*\n`, où l'expression $\hat{}\sigma$ désigne n'importe quel symbole de l'alphabet sauf le symbole σ.

Proposez un automate fini déterministe reconnaissant le langage des commentaires décrit ci-dessus.