REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE.

Ministère De l'Enseignement Supérieur et de la Recherche Scientifique.

Université Mouloud Mammeri de Tizi-Ouzou.

Faculté de Génie Electrique et Informatique.

Département D'Informatique.

Spécialité : Système Informatique.

Mémoire de fin d'étude master en informatique.

Mini-Excel :

-Evaluateur d'expression de types :arithmétiques, logiques et relationnelles.

Réalisé par :

MR: CHIKH Mohamed.

Promoteur :

MR : KHEMLICHE Salim

Le jury:

Président :MR OULARBI .

Promoteur :MR KHAMLICHE .

Examinateur :MR HABET.

Examinateur :MR DJEMAI.

Promotion 2010 /2011

Remerciement

Je remercie le bon DIEU de m'avoir accordé le pouvoir de finir ce projet.

Je remercie mon promoteur de m'avoir permis de soutenir.

Je remercie mes parent qui sont tout pour moi.

Je remercie mes frères et sœurs qui n'ont jamais négligé une aide dont j'avais besoin.

Je remercie tous ce qui m'ont aidé durant les moments difficiles (durant ma maladie).

Je remercie tous les amis qui m'ont aidé de prés ou de loin, de peu ou de trop…

Dédicaces

Je dédie ce travail à :

- mes parents.
- Mes frères et sœurs.
- Mes oncles, mes tentes et mes cousins.
- Mes amis de la cité universitaire de Oued Aissi.
- Mes amis de deuxième année master en informatique.
- Tous les étudiants et travailleur de l'Université Mouloud MAAMERI de Tizi Ouzou.
- Tous les gens qui me connaissent.

Sommaire.

Table des figures.

Résumé

Dans ce projet, on s'est intéressé à un des projet de la compilation qui est l'évaluation des expressions régulières ;donc on a essayé de faire un évaluateur d'expressions de types :Arithmétiques, logiques et relationnelles.

Mais cet évaluateur doit d'abord vérifier la validations de ces expressions régulières.

On a commencé par la présentation de logiciel Microsoft Excel car on va programmer une interface pour l'application qui a des caractéristiques semblables à celles de Microsoft Excel.

On a enchainé avec un chapitre de théories des langages et compilation où on va s'intéresser à tous ce qui concerne ce projet.

La conception de ce projet c'est la production de la grammaire qui génère les expressions régulières .La réalisation c'est la programmation de cette grammaire pour obtenir une application .

Après avoir effectué les quelques testes ; on terminera avec une conclusion générale.

Introduction

L'informatique est une science qui envahit tous les domaines (on entend souvent informatiser) et les expressions régulières font parti de cette science qui devient de plus en plus indispensable (l'informatique) ;elles (expressions régulières) sont des chaines de caractères décrites selon une syntaxe précise, utilisées dans l'édition et le contrôle de texte ainsi que la manipulation des langages de programmations **[A1]**.

Pour comprendre un peu plus cette notion d'expression rationnelle (régulière) on s'est intéressé à un type de ces expressions qui sont les expressions mathématiques.

Cependant Microsoft Excel est un logiciel qui fait des calculs sur les expressions mathématiques (arithmétiques, logiques et relationnelles),ce qui nous conduit à essayer de programmer une interface semblable à celle d'Excel où on testera la validité de ces expressions mathématiques suivi de leurs évaluations.

pour cela on commence par une brève présentation de Microsoft Excel suivie d'un chapitre de théorie des langages et compilation où on s'intéressera à quelques notions qui concerne ce projet, ensuite, on enchainera avec la conception où on construit la grammaire qui génère ces expressions ,après on réalisera l'application souhaitée et on finira par quelques testes.

Chapitre I : *Présentation de l'Excel.*

A. Microsoft Office : [A2]

On va présenter d'abord Microsoft Office car Excel en fait partie de l'Office.

a) Définition :

Microsoft Office est une suite bureautique éditée par Microsoft pour les plates-formes Windows et Macintosh.
Microsoft Office introduit désormais une version web utilisable sur les ordinateurs Linux par navigateur (Firefox, Safari ou Google Chrome). Une version pour téléphone est également disponible.

Figure1 :Logo de Microsoft Office 2010

Développeur : Microsoft
Dernière version : Office 2010 (12 mai 2010 pour les entreprises, 15 juin 2010 pour le grand public.)
Environnement : Windows, Mac OS X
Type : Suite bureautique
Licence : Propriétaire, EULA (License du logiciel)
Site Web : office.microsoft.com

b) Les Logiciels de Base :

Il existe 8 éditions différentes de Microsoft Office System 2007 et 6 éditions pour Microsoft Office System 2010. 3 de ces éditions sont destinées à un usage principalement personnel et incluent un certain nombre des logiciels considérés aujourd'hui comme les logiciels de base de Microsoft Office System.

1. Microsoft Office Word :

Microsoft Office Word est un logiciel de traitement de texte. Il est considéré comme le programme central de Microsoft Office. Il domine le marché du logiciel de traitement de texte.
L'extension de ses fichiers par défaut est le « .doc ».

2. Microsoft Office Excel :

Microsoft Office Excel est un tableur qui, comme presque tous les tableurs, sait également faire des graphiques, d'où son nom de tableur-grapheur. Comme Microsoft Word, il domine le marché. Il a été dès le début le concurrent de Lotus 1-2-3 mais il est devenu rapidement le logiciel dominant du marché. Il est disponible pour Windows et Macintosh.

3. Microsoft Office Outlook :

Microsoft Office Outlook, à ne pas confondre avec Microsoft Outlook Express est un Gestionnaire d'informations personnelles (également connu sous le nom de PIM, acronyme anglais de Personal Information Manager) et un client de communication par courriel. Désigné comme remplaçant de Windows Mail, il fait son arrivée dans la version d'Office 97. Il inclut un calendrier, un client de courriel, un gestionnaire de tâches et un carnet d'adresses.

4. Microsoft Office PowerPoint :

Microsoft Office PowerPoint est un créateur de présentations (succession de diapositives) pour Windows et Mac. Il est utilisé pour créer des présentations avec du texte, avec des

images, sons, vidéos et autres objets, qui peuvent être visualisées sur un écran ou projetées grâce à un projecteur appelé également beamer. Windows Mobile 2005 contient une version réduite.
Une Visionneuse gratuite téléchargeable sur Internet permet de montrer des présentations. Microsoft a repris l'idée de visionneuse gratuite pour Word et Excel.

5. Microsoft Office OneNote :

Microsoft Office OneNote est un logiciel destiné à la prise de notes pouvant être utilisé sur les Tablets PC ou sur les PC de bureau.

c) Logiciels spécialisés :

Ces logiciels sont inclus uniquement dans certaines éditions de la suite Office ou doivent être achetés à part.

1. Microsoft Office Access :

Microsoft Office Access est un système de gestion de base de données. Il nécessite quelques connaissances pour être utilisé efficacement.

2. Microsoft Office Publisher :

Microsoft Office Publisher est un logiciel de publication assistée par ordinateur, qui crée des lettres d'information, cartes de visite, papiers volants, cartes de vœux ou encore des cartes postales.
Il a été conçu pour aider des « non-professionnels » à créer et mettre en forme des publications. Publisher est aussi un support pour créer de grands travaux d'impression.

3. Microsoft Office InfoPath :

Microsoft Office InfoPath est une application qui permet à des utilisateurs de concevoir des formulaires basés sur le XML.

4. Microsoft Office Sharepoint Designer :

Microsoft Office SharePoint Designer est un éditeur HTML, un logiciel permettant de créer des pages web. Il a remplacé Microsoft Office Frontpage à partir de la version 2007 d'Office System. Ce logiciel est principalement dédié à la conception de pages pour Microsoft Office Sharepoint.

5. Microsoft Office Communicator :

Microsoft Office Communicator est un client de messagerie instantanée servant à se connecter sur les composants Microsoft Office communication Server.

6. Microsoft Office Groove :

Microsoft Office Groove est un outil de collaboration simplifié pour les petites entreprises.

7. Microsoft Office Visio :

Microsoft Office Visio est un outil de création de diagrammes.

8. Microsoft Office Project :

Microsoft Office Project est un logiciel de gestion de projets.

9. Microsoft Office MapPoint :

Microsoft Office Mappoint est un logiciel de cartographie dédié aux professionnels. Sa version grand public est Microsoft Autoroute Express.

d) Outils Microsoft Office :

Microsoft Office contient quelques outils intégrés au reste de la suite.

1. Microsoft Office Picture Manager :

Microsoft Office Picture Manager est un logiciel secondaire présent dans toutes les éditions de Microsoft Office System et fait non seulement office de classeur de photos simplifié, mais il sait également retoucher légèrement les photos en modifiant les tons de couleurs, de luminosité, de saturation, etc. Il a succédé à Microsoft Photo Editor.

2. Microsoft Office Producer :

Cet outil est en fait un complément gratuitement pour Office qui permet de faire des présentations multimédias en fusionnant PowerPoint et Windows Movie Maker 2. Ce logiciel est disponible sur le site web de Microsoft, mais il nécessite Microsoft Office PowerPoint 2002 ou 2003 pour fonctionner. Bien que son interface ressemble à celle de Office XP, ce logiciel est bien intégré à Office 2003.

e)Autres programmes intégrés dans les éditions pour Mac :

Microsoft Entourage – logiciel de gestion d'informations et client de messagerie (n'existe plus depuis la sortie de Microsoft Office 2011).

B. Microsoft Office Excel : [A3]

a) Définition :

Microsoft Excel, dont le nom officiel actuel est Microsoft Office Excel, est un tableur de la suite bureautique Microsoft Office qui est écrite et distribuée par l'éditeur Microsoft ; il est destiné à être utilisé sur des plateformes Microsoft Windows ou Macintosh Mac OS X.
Chaque feuille de Microsoft Excel est composée :

- de 16 384 lignes sur 256 colonnes jusqu'à la version 95 (7.0)
- de 65 536 lignes sur 256 colonnes jusqu'à la version 2003 (11.0)
- de 1 048 576 lignes sur 16 384 colonnes depuis la version 2007

Depuis la version 2002 le nombre maximum de feuilles par classeur est limité par la quantité de mémoire disponible mais, auparavant (*depuis Excel 5.0*), la limite du nombre de feuilles était de 256. Développé au départ par Microsoft pour le Macintosh et porté ensuite sur Microsoft Windows, Excel fait partie de la suite Microsoft Office et il est depuis sa version 5.0 sortie en 1993 le tableur dominant sur ces plates-formes.
Les fichiers produits à l'aide du logiciel portent l'extension xls (xlsx à partir de la version 2007).

b) Historique :

A l'origine, Microsoft a commercialisé un tableur appelé Multiplan en 1982 qui fut très populaire sur les systèmes CP/M (plus populaire que Lotus 1-2-3 en France, mais pas dans le reste du monde). La première version d'Excel est sortie en 1985 (*Multiplan ne fut pas réutilisé*) pour les ordinateurs Apple Macintosh alors que la première version pour Windows, la 2.05 (*en ligne avec celle pour Macintosh*), est sortie en 1987. Elle incluait un Moteur d'exécution (*Runtime*) de la version 2.1 de Microsoft Windows.
Avec l'introduction de *Windows 3.0* en 1990, les ventes d'Excel dépassèrent celles de Lotus 1-2-3 et il atteignit la première place dans la liste des ventes (*en nombre de licences*). Cet accomplissement permit à Microsoft de détrôner le leader mondial du logiciel en se positionnant comme un véritable concurrent et démontra ainsi ses capacités à développer des logiciels avec interface graphique utilisateur (GUI).
Les tableurs ont été les premières applications importantes pour les ordinateurs personnels et lorsque Excel fut empaqueté avec Microsoft Word et Microsoft PowerPoint dans la suite

Microsoft Office (4.x), l'interface de ces deux derniers programmes a dû être revue pour être conforme avec celle d'Excel.
La version 14 actuelle pour systèmes Windows (XP SP2 et ultérieur) se nomme aussi Microsoft Office Excel 2010 et la version actuelle pour Macintosh se nomme quant à elle, Microsoft Excel 2011.

Figure2 :Logo de Microsoft Office Excel 2010

Développeur : Microsoft
Dernière version : 2010 sous Windows et 2011 sous Mac OS X (15 juin 2010 sous Windows et 26 octobre 2010 sous Mac OS X)
Environnement : Microsoft Windows et Mac OS X
Type : Tableur grapheur
Licence : Propriétaire EULA End-User License Agreement :Contrat de License Utilisateur Final
Site Web : http //office.microsoft.com/en-us/FX010858001033.aspx/

c) Fonctionnement du tableur :

1. Principes simple des formules :

Il est possible de faire calculer beaucoup de choses à Excel en saisissant une « formule » dans une cellule.

- Par exemple commencez par saisir = dans la cellule **A10** ce qui indique à Excel que vous attendez de lui un calcul dont le résultat s'affichera dans **A10**.
- Ensuite entrez les *Arguments* après le = donc dans **A10** tapez **=A1+A2**
- Rendez la main à Excel en tapant la touche **retour à la ligne** ce qui signifie *à toi de jouer*.
- Le résultat de l'addition s'affiche dans **A10** (*A condition qu'il y ait des nombres dans A1 et A2 évidemment*)
- Similaire : dans **A10** si on saisit : **=A1*A2** le résultat sera la *multiplication* des nombres contenus dans A1 multipliés par A2.
- Similaire : dans **A10** si on saisit : **=A1/A2** le résultat sera la division des nombres contenus dans A1 divisé par A2.
- Similaire : dans **A10** si on saisit : **=A1+A2+A4** le résultat sera l'addition des nombres contenus dans A1 , A2 et A4.
- Similaire : dans **A10** si on saisit : **=A1+(A2*A4)** le résultat sera l'addition du contenu de A1 avec le résultat de la multiplication A2 par A4.

Note : la parenthèse permet de décomposer clairement à Excel le calcul qu'il doit effectuer

2. Principes simple des fonctions :

Excel contient aussi des Fonctions, paramétrables ou non, avec des catégories diverses répertoriées par les dernières utilisations (les plus courantes) ou en simple liste de choix...

Somme :

Par exemple la Fonction : '**SOMME**' effectue l'addition d'une zone désignée *plage*.
Sélectionnez par exemple la cellule **A21** et commencez par saisir = (Le = signifie à Excel que vous attendez de lui un calcul). Rentrer le nom de la Fonction : **SOMME**. Écrivez entre

parenthèses les références des cellules ce qui signifie à Excel quelle plage de cellules il doit prendre en compte pour calculer la somme. Vous obtenez ainsi dans la cellule A21 **=SOMME(A2:A20)** En appuyant sur **retour à la ligne**, Excel effectue la somme du contenu des cellules A2 jusqu'à A20 et l'affiche dans **A21**.
Nota : les ' **:** ' entre A2 et A20 signifient **jusqu'à.**

Moyenne :

Si vous tapez par exemple **=MOYENNE(A1:A20)** En appuyant sur **retour à la ligne** Excel effectue la moyenne du contenu des cellules A1 jusqu'à A20.

Maintenant :

Si vous tapez par exemple **=MAINTENANT()** En appuyant sur **retour à la ligne** Excel affiche la date du jour et l'heure (de façon dynamique) Nota : les parenthèses sont indispensables bien qu'il n'y ait pas d'arguments à décrire.

Autres fonctions :

Dans le **Menu Insertion**, la commande **Fonction…**, Excel propose un gros catalogue de fonction prêtes à l'emploi avec une description de leur usage.

Les Fonctions Complémentaires :

Certaines formules reposent sur des packs externes de fonctions complémentaires (par exemple, **Utilitaire d'analyses**). Les formules sont grossièrement les mêmes que pour OpenOffice.org Calc.

d) Caractéristiques de Microsoft Excel :

Cette application est un tableur ; autrement dit, elle se présente sous forme de tableaux structurés en lignes et colonnes dans des onglets séparés avec, pour chaque cellule qui compose chaque feuille, des caractéristiques particulières pour les calculs, des outils de génération de graphiques, des outils d'analyse croisée dynamique et un module de programmation par macro ou en développement direct avec le langage Visual Basic pour Application (VBA).

1. Graphiques :

Excel permet de dessiner automatiquement des graphiques de visualisation des données chiffrées.

2. Tableaux croisés dynamiques :[A4]

Un **tableau croisé dynamique** (en anglais ***pivot table***) est une fonctionnalité de certains tableurs qui permet de générer une synthèse d'une table de données brutes. Cela permet de créer des mises en forme de tableaux en choisissant les différents champs voulus, en abscisses ou en ordonnées.
La base de départ est une plage de cellules où chaque ligne correspond à un enregistrement (tableau du haut dans l'image). Dans le cas de l'exemple donné dans l'image c'est la plage "A1:E16". La première ligne de la plage est constituée des titres des champs (en-tête des colonnes en gras).
C'est avec ces titres de champs, que l'on sélectionne ou pas, qu'on arrange le tableau croisé dynamique (tableau du bas). On fait glisser les champs soit dans une colonne soit dans une ligne du modèle. Il existe un ordre des champs dans les lignes et les colonnes.
Un champ à faire glisser constituera les données du tableau croisé.
Dans l'exemple sur l'image, les champs « fournisseur » et « prix » ont été mis en ligne, les champs « Genre » et « taille » en colonne, dans cet ordre respectivement et le champ « Quantité » a été mis pour constituer les données.
Nota : Les sous-totaux des lignes et des colonnes ont été désactivés pour plus de lisibilité du tableau servant d'exemple.

Le tableau est dit croisé parce que l'on met des données à l'intersection (le « croisement ») entre des champs disposés en ligne et en colonne.
Il est dynamique, parce que l'on peut apporter des changements à la table des données brutes, il suffit de demander de « rafraîchir » pour que le tableau croisé soit actualisé. On peut aussi vouloir rééditer la position des différents champs, ce qui re-générera le tableau.

Intégration des tableaux croisés dynamiques dans les principaux tableurs

Sous MS Excel

Sous Excel, le tableau croisé dynamique est généré grâce à un assistant appelé à partir du menu Données/Rapport de tableau croisé dynamique.

Sous OpenOffice.org

OpenOffice préfère appeler le tableau croisé dynamique **Pilote de données**. On accède à l'assistant par le menu *Données* / *Pilote de données*.

Logiciels permettant de manipuler les tableaux croisés dynamiques :

Suites bureautiques :

- Microsoft Office (dans le module Excel)
- Open Office (dans le module OOo Calc)
- WordPerfect Office (dans le module Quattro Pro)
- statistica
- Google Docs (dans le module Spreadsheets)

Grâce à un langage de programmation :

Les tableaux croisés dynamiques peuvent être mis en œuvre directement via un langage de programmation, comme dans le cas du langage Oberon ou via un langage de macro utilisant les objets d'un tableur.

Macro en VBA :

Il est possible d'enregistrer une macro en Visual Basic for Applications dans Excel lors de la génération d'un tableau croisé dynamique. Le code généré peut être édité et modifié pour contrôler la formation du tableau.
Il faut commencer par sélectionner une plage de cellules :
Range(« A1:E16 »).Select
Puis il faut créer le tableau. L'objet associé s'appelle PivotTables :
ActiveWorkbook.PivotCaches.Add(SourceType:=xlDatabase, SourceData:= "Feuil1!R1C1:R16C5").CreatePivotTable TableDestination:="", TableName:= "Tableau croisé dynamique2", DefaultVersion:=xlPivotTableVersion10
On donne alors une destination à ce tableau croisé dynamique :
ActiveSheet.PivotTableWizard TableDestination:=ActiveSheet.Cells(3, 1)
ActiveSheet.Cells(3, 1).Select
Cet objet PivotTables est un conteneur des objets PivotFields, représentant les champs ajoutés, dont on peut choisir les propriétés :

- .orientation (arguments : =xlRowField xlColumnField ou xlHidden), pour choisir si le champ est mis en ligne, en colonne ou non utilisé,
- .position (=1, 2,3 ...), pour choisir la position du champ dans la ligne ou la colonne,
- .Subtotals = Array(False, False, False, False, False, ...), pour masquer les différents sous-totaux,

Il existe enfin une méthode .AddDataField permettant d'ajouter un certain champ (PivotFields) en tant que données à l'objet PivotTables. On donne deux arguments séparés par une virgule : le « Nom du champ » et xlSum,

3. Macros :

On peut considérer les produits Microsoft Office sous deux angles de vue. L'angle de l'utilisateur qui se servira du logiciel pour accomplir un certain nombre de tâches afin d'arriver à un certain résultat. Et l'angle de l'informaticien, qui pour certains, voient en MS Office une véritable plateforme de développement. En effet, les macros sous Excel (mais aussi Word, Access, PowerPoint, ...) possèdent une partie sous-jacente qui permet de contrôler entièrement l'application grâce au langage **Visual Basic for Applications**.
Dans un premier niveau, ces Macro-commandes permettent de mettre, sous la forme de lignes de commande, des tâches longues et/ou répétitives. Mais dans un deuxième, la puissance du langage permet, à l'aide de certaines bibliothèques de composants (Windows ou .NET par exemple), de créer de véritables applications dont Excel (ou Word ou Access) ne seraient que les interfaces graphiques.

4. Sécurité :

La protection par mot de passe pour verrouiller l'accès au contenu d'une feuille Excel permet de se mettre à l'abri de modifications faites par inadvertance par des utilisateurs (*étourdis*) ou tout simplement par pure confidentialité et de réserver la lecture aux titulaires des mots de passe.
Mais contrairement à une croyance répandue, l'utilisation d'un mot de passe ne procure pas une protection totalement efficace contre un utilisateur expérimenté et déterminé. Toutefois, le contournement d'un mot de passe n'est pas à la portée du premier venu :
En effet, à cause d'un niveau chiffrement faible, de nombreux logiciels permettent de découvrir instantanément le mot de passe utilisé par Excel, quelle que soit sa longueur ou le jeu de caractères utilisé. La parade consiste à choisir au moment de la création/sauvegarde du fichier, l'option qui propose un chiffrement plus fort que celui assigné par défaut.

e) Formats de fichier :

Jusqu'à la version 2007, Microsoft Excel utilisait un format propriétaire binaire caractérisé par l'acronyme BIFF (Binary Interchange File Format).
Excel 2007 utilise cette fois le format ouvert de fichier Open XML identique à la structure du format XML comme format principal : il porte le nom de **XML Spreadsheet** dont l'acronyme s'écrit XMLSS. Ce format avait été tout d'abord introduit dans la version 2002 mais n'était pas capable d'encoder les macros écrites avec du langage VBA.
Excel 2007 reste pour autant totalement compatible avec les formats des versions précédentes bien que Microsoft encourage l'usage de ce nouveau format XML. En parallèle, la plupart des versions de Microsoft Excel sont capables de lire les formats externes issus de fichiers CSV, DBF, SYLK ou encore DIF ; il dispose en plus d'un module d'importation intelligent de fichiers texte.

1. Formats de fichier binaire Microsoft Excel :

La spécification de *Format binaire* a été disponible de Microsoft depuis février 2008 et peut être téléchargée librement.

2. Formats de fichier Open XML pour Microsoft Office Excel 2007 :

Microsoft Excel 2007 autant que les autres applications de la suite Microsoft Office 2007, intègre le nouveau format de fichier faisant partie des spécifications du format XML ouvert (OOXML).
Les nouveaux formats Excel 2007 sont :

Format	Extension	Commentaires
Classeurs Excel	.xlsx	Le format Excel par défaut. En réalité, il s'agit d'un format compressé ZIP avec une structure XML sous forme de texte. Ce format est dédié à remplacer le format initial .xls mais il ne sait pas incorporer de macros pour des raisons de sécurité.
Format des classeur avec Macros-incoporées	.xlsm	Identique au format des classeurs Excel mais avec macros incorporées.
Format binaire des classeurs	.xlsb	Tout comme le format xlsm, ce format stocke les informations du classeur en binaire plutôt qu'en XML de manière à pouvoir ouvrir et enregistrer de manière plus efficace considérant que format est plutôt dédié aux classeurs très volumineux, pourvus de milliers de lignes sur plusieurs centaines de colonnes.
Format des modèles avec Macros-incoporées	.xltm	Tout comme le format des modèles XLT des versions précédentes qu'il remplace, ce format permet de créer des modèles avec macros incorporées.
Le format des macros complémentaires Excel	.xlam	Les macros complémentaires Excel sont considérées comme des bibliothèques de fonctions utilisées en référence, offrant des extra-fonctionnalités. Du fait de leur constitution, ce type de fichier supporte bien évidemment les macros.

Figure3 :Nouveaux formats Excel 2007

Type	Extension < 2007	Extension > 2007
Classeur sans macros	.xls	.xlsx
Classeur avec macros	.xls	.xlsm
Modèle sans macros	.xlt	.xltx
Modèle avec macros	.xlt	.xltm

Figure4 :Exemple de comparaison des extensions Excel avant et après la version 2007

f) Exportation des données issues de feuilles de calcul :

Microsoft Excel fournit un jeu de fonctions d'applications liées directement à l'interface pour pouvoir exploiter des données issues de feuilles de calcul Excel dans des environnements et/ou des applications hétérogènes. Parmi eux se trouvent des composants permettant notamment d'ouvrir des classeurs Excel directement depuis Internet à l'aide de contrôles ActiveX ou encore de composants (*PlugIns*) comme ceux disponibles dans Adobe Flash Player. D'autres fonctionnalités sont en mesure de copier des données issues de feuilles de calcul Excel directement au sein d'applications Web en utilisant le format CSV (*format texte séparé par des virgules*).

g) Programmation :

Une caractéristique très intéressante de Microsoft Excel est celle de pouvoir écrire du code à travers le langage Visual Basic pour application. La rédaction du code (*procédures, fonctions et macros*) s'effectue dans un éditeur indépendant du classeur : **Visual Basic Editor** (VBE). L'exploitation du classeur et de son contenu peut alors s'effectuer en utilisant des objets. Le code issu de procédures ou de fonctions peut être exécuté directement (F5) ou à partir de la feuille de calcul ciblée ou depuis un contrôle posé sur une feuille ou sur une fenêtre sous forme de formulaire (*UserForm*). On peut donc écrire, modifier, déplacer, copier ou agir avec une multitude de possibilités sur le contenu des feuilles de calcul (*données* et *graphiques*) et ce, de façon instantanée. Chaque feuille de calcul peut devenir une véritable interface en parfaite homogénéité avec le code qui la supervise ou qui gère ses calculs

h) Critiques et défauts :

Excel a été critiqué sous différentes formes et en particulier pour les problèmes de précision sur des calculs à virgule flottante face à d'autres outils dédiés notamment aux calculs statistiques. Les adeptes d'Excel répondaient que ces erreurs de précision ne touchaient qu'une

minorité de personnes qui connaissaient ce problème et que ces mêmes personnes, le plus souvent, avaient des solutions de contournement pour y parer.
Par ailleurs, Excel suppose que l'année de base de départ de l'environnement Excel est **1900** et que celle-ci est *bissextile*. L'objectif était d'être en mesure de rester compatible avec le bug rencontré dans le tableur Lotus 1-2-3.
Il continue d'être exploité ainsi encore aujourd'hui, même au sein du format de fichier XML ouvert.
Pour contrer certaines failles dans des calculs de date, Excel intègre et gère aussi la base de ses calculs à partir de l'année 1904.

i) Versions de Microsoft Excel :

Les versions de Microsoft Excel pour Windows :

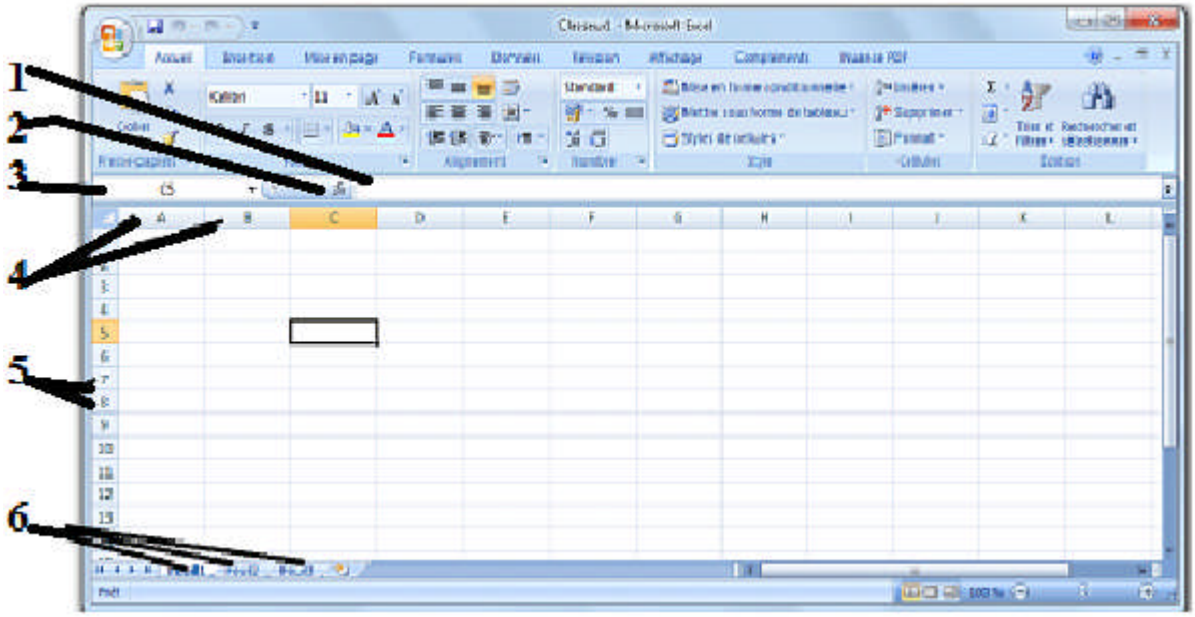
1987 - Excel 2.0 pour Windows
1990 - Excel 3.0
1992 - Excel 4.0
1993 - Excel 5.0 (Microsoft Office 4.2 et 4.3 et également une version 32-bit pour Windows NT tournant seulement sur PowerPC, DEC Alpha et MIPS OS)
1995 - Excel pour Windows 95 (version 7.0) – inclus aussi dans Microsoft Office 95
1997 - Excel 97 (version 8.0) - inclus aussi dans Microsoft Office 97 (x86 et aussi DEC Alpha version)
1999 - Excel 2000 (version 9.0) inclus aussi dans Office 2000
2001 - Excel 2002 (version 10) inclus aussi dans Office XP
2003 - Excel 2003 (version 11) inclus aussi dans Office 2003
2007 - Excel 2007 (version 12) inclus aussi dans Office 2007
2010 - Excel 2010 (version 14) inclus aussi dans Office 2010

Les versions de Microsoft Excel pour Apple Macintosh :

1985 - Excel 1.0
1988 - Excel 1.5
1989 - Excel 2.2
1990 - Excel 3.0
1992 - Excel 4.0
1993 - Excel 5.0 (Office 4.X – version Motorola 68000 et première version pour PowerPC)
1998 - Excel 8.0 (Office '98)
2000 - Excel 9.0 (Office 2001)
2001 - Excel 10.0 (Office v. X)
2004 - Excel 11.0 (Office 2004 pour Mac)
2008 - Excel 12.0 (Office 2008 pour Mac)
2010 - Excel 14.0 (Office 2011 pour Mac)

Les versions de Microsoft Excel pour OS/2 :

1989 - Excel 2.2
1990 - Excel 2.3
1991 - Excel 3.0



1 Barre des formules
2 Indication d'insertion de fonction
3 Zone nom(indique n° ligne et colonne; ici C5)
4 N° de colonne (avec des lettres)
5 N° de ligne (avec des chiffres)
6 N° de feuille

Figure5 : Image indiquant sur l'apparence de l'Excel

Toutes ces caractéristiques de Microsoft Excel, on doit les introduire dans l'application ou l'interface de l' application de ce projet qui est un évaluateur d'expressions régulières.

Chapitre II : *Théorie des langages et compilation.*

A. Théorie des langage :

a) Définition :[A5]

En informatique, la **théorie des langages** a pour objectif de décrire les langages formels, vus comme moyen de communication, d'un point de vue mathématique.
Un langage formel est un ensemble de mots. Un mot est une suite finie de symboles. L'ensemble de ces symboles est appelé alphabet, les symboles eux-mêmes sont des lettres.
Un langage (formel) doit pouvoir être décrit et analysé par une méthode formelle. Plusieurs sortes de mécanismes peuvent exister:

- Une description au moyen d'une grammaire formelle, permettant d'engendrer les mots du langage,
- Un dispositif d'analyse reconnaissant les mots du langage. Ce sont des automates. Les automates finis reconnaissent les langages rationnels, les automates à pile reconnaissent les langages algébriques, des automates plus sophistiqués existent pour d'autres classes de langages.
- Une description au moyen d'une formule logique
  La théorie des langages s'applique en particulier dans la réalisation des compilateurs.

b) Mots :[A5]

On se donne un ensemble A, appelé **alphabet** dont les éléments sont appelés des **lettres**.

- Un **mot** de longueur k est une suite $u = (a_1,a_2,...,a_k)$ de k lettres.
- L'ensemble des mots sur l'alphabet A est noté A^*.
  Le mot vide, de longueur 0, est noté : ε.

- On définit sur A^*, une loi de composition interne appelée concaténation. Elle associe à deux mots $(a_1,...,a_n)$ et $(b_1,...,b_m)$ le mot $(a_1,...,a_n,b_1,...,b_m)$ (de longueur $n + m$). Cette loi de composition interne est associative et admet le mot vide pour élément neutre, par conséquent A^*, muni de cette loi, est un monoïde. C'est un monoïde libre au sens de l'algèbre.

c) Langages :[A6]

Un langage est un ensemble de *mots*, qui sont simplement des séquences de symboles choisis dans un ensemble (en général fini) appelé *alphabet*. Formellement, si A est un ensemble, on note A^* le monoïde libre sur A, c'est-à-dire l'ensemble des suites finies d'éléments de A, muni de l'opération de concaténation de deux mots. Un langage sur l'alphabet A est par définition un sous-ensemble de A^*.
Souvent, les « symboles » que l'on considère lorsqu'on définit un langage par une grammaire formelle sont constitués de plusieurs caractères, de sorte qu'ils correspondent plutôt à ce que l'on appelle des mots dans la langue courante. De même, les « mots » du langage correspondent plutôt à des phrases ou à des textes. Lorsqu'il y a ambiguïté, on parle de *lettres* ou de *caractères* pour les symboles de l'alphabet utilisé pour coder les informations ; et on réserve le mot symbole pour ceux de l'alphabet abstrait, qui sont les éléments de base du langage.
Par exemple :

- A1 = { a, b, c, d, e } est un alphabet contenant 5 symboles, traditionnellement appelés *lettres* dans ce cas précis ;
- A2 = { 2, 5, @, $, & } est un autre alphabet contenant 5 symboles ;
- A3 = { Dét, Adj, Verbe, Nom, Coord, Prép } est un alphabet de 6 symboles pouvant décrire, par exemple, la structure syntaxique d'une phrase dans une langue naturelle.

d) Grammaires :[A6]

Une grammaire formelle (ou, simplement, grammaire) est constituée des quatre objets suivants:

- Un ensemble fini de symboles, appelés *symboles terminaux* (qui sont les « lettres » du langage), notés conventionnellement par des minuscules,
- Un ensemble fini de symboles, appelés *non-terminaux*, notés conventionnellement par des majuscules,
- Un élément de l'ensemble des non-terminaux, appelé *axiome*, noté conventionnellement S,
- Un ensemble de *règles de production*, qui sont des paires formées d'un non-terminal et d'une suite de terminaux et de non-terminaux ; par exemple, A → ABa.

Appliquer une règle de production consiste à remplacer dans un mot une occurrence du membre de gauche de cette règle par son membre de droite ; l'application successive de règles de productions s'appelle une dérivation. Le langage défini par une grammaire est l'ensemble des mots formés uniquement de symboles terminaux qui peuvent être atteints par dérivation à partir de l'axiome.

Ainsi, la grammaire définie par les terminaux {a, b}, le non-terminal S, l'axiome S et les deux règles de production suivantes :

S → aSb

S → ε (où ε représente le mot vide)

représente le langage des mots de la forme $a^n b^n$ (un certain nombre de *a* – éventuellement 0, en vertu de la deuxième règle –, suivis du même nombre de *b*) : {ε, ab, aabb, aaabbb…}.

e)Hiérarchie de Chomsky :[A6]

Lorsque le linguiste Noam Chomsky a dégagé la notion de grammaire formelle, il en a proposé une classification appelée de nos jours hiérarchie de Chomsky. Elle est formée des quatre niveaux suivants, du plus restrictif au plus large.

- Les langages de type 3, ou langages rationnels : ce sont les langages définis par une grammaire linéaire à gauche (c'est-à-dire une grammaire dont chaque membre droit de règle commence par un non-terminal), une grammaire linéaire à droite (c'est-à-dire une grammaire dont chaque membre droit de règle finit par un non-terminal) ou une expression rationnelle ; ou bien encore les langages reconnus par un automate fini.
- Les langages de type 2, ou langages algébriques : ce sont les langages définis par une grammaire formelle hors-contexte, ou bien encore les langages reconnaissables par un automate à pile non déterministe. La plupart des langages de programmation, sans être à proprement parler des langages algébriques, en sont assez proches pour que les techniques d'analyse des langages algébriques s'y adaptent.
- Les langages de type 1, ou langages contextuels : ce sont les langages définis par une grammaire contextuelle, ou encore les langages reconnaissables par une machine de Turing non-déterministe à ruban de longueur bornée par un multiple fixé de la longueur du mot d'entrée.
- Les langages de type 0, ou langages récursivement énumérables. Cet ensemble inclut tous les langages définis par une grammaire formelle. C'est aussi l'ensemble des langages acceptables par une machine de Turing (que l'on autorise à boucler sur un mot qui n'est pas du langage).

Outre les quatre types de la hiérarchie de Chomsky, il existe des classes intermédiaires remarquables :

- entre 3 et 2 : les langages hors-contextes déterministes, reconnaissables par automate à pile déterministe ;
- entre 1 et 0 : les langages récursifs, c'est-à-dire reconnaissables par une machine de Turing (celle-ci doit refuser les mots qui ne sont pas du langage).

Les six types ci-dessus sont strictement inclus les uns dans les autres. Si dans le type 1, on transforme « non déterministe » en « déterministe », on obtient un type plus petit, mais on ne sait pas montrer s'il est strictement inclus dans le type 1 ou s'il est égal à celui-ci.

f)Notion d'équivalence :[A6]

Équivalence forte :

Deux grammaires sont dites fortement équivalentes si et seulement si

1. Elles reconnaissent exactement le même langage.
2. Elles utilisent exactement le même arbre syntaxique pour analyser une même phrase.

Équivalence faible :

Deux grammaires sont dites faiblement équivalentes si et seulement si

1. Elles reconnaissent exactement le même langage.

Les grammaires fortement équivalentes sont donc toujours aussi faiblement équivalentes.

g)Grammaire régulière :[A7]

Une **grammaire régulière**, **rationnelle** ou **à états finie** permet de décrire un langage régulier. Bien que les expressions rationnelles soient le modèle de définition le plus courant d'un langage régulier, en particulier dans le domaine des langages de programmation, il ne s'en trouve pas moins que, techniquement, la reconnaissance d'un mot d'un langage régulier s'effectue par la traduction d'expressions rationnelles en grammaire régulière.

Notation : Une grammaire régulière se note de la même manière qu'une grammaire hors-contexte:

$G = \{N,\Sigma,P,S\}$

où N est le vocabulaire non terminal, Σ est le vocabulaire terminal, P est l'ensemble des règles de production et $S \in N$ est le non terminal initial.

Définition : L'ensemble des grammaires régulières est inclus dans l'ensemble des grammaires hors-contexte. Aussi les règles de production ne doivent comporter aucun symbole terminal dans leur partie gauche.

De plus, les règles de production doivent être de l'une des 2 formes suivantes:

- **grammaire régulière à droite :**

$$X \rightarrow aY$$
$$X \rightarrow a$$
$$X \rightarrow \epsilon$$

- **grammaire régulière à gauche**

$$X \rightarrow Ya$$
$$X \rightarrow a$$
$$X \rightarrow \epsilon$$

Où $X, Y \in N$, $\alpha \in \sum$ et ε est le mot vide.

A la vue des règles de production, on peut en déduire que le principe d'une grammaire régulière est de reconnaitre un mot en partant d'une de ses extrémités et de progresser vers l'autre bout du mot en reconnaissant un seul caractère à la fois.

h)Grammaire ambigüe :[A8]

Lorsqu'un même mot possède plusieurs arbres d'analyse, on parle **d'ambiguïté .**

i)Méthodes d'analyse d'une grammaire :[A8]

il y a deux méthodes qui sont assez connues qui sont :

LL : Left to right scanning usin Left most derivation.

LR : Left to right scanning usig Right most dérivation.

B. Compilation :

a) Définition : [A9]

Compiler signifie une description d'information et synthétiser une autre forme de celle-ci, (mieux adaptée à ce qu'on veut en faire) tout en maintenant la sémantique invariante.
Un compilateur peut être définie d'une façon générale comme un traducteur automatique d'une forme source en une forme cible. Dans le cas de la compilation d'un langage de programmation écrit dans langage (source) et la forme objet est un code (objet) exécutable une machine réelle ou virtuelle. Dans le cas d'une présentation d'erreurs, le compilateur doit en donner un diagnostique l'utilisateur.

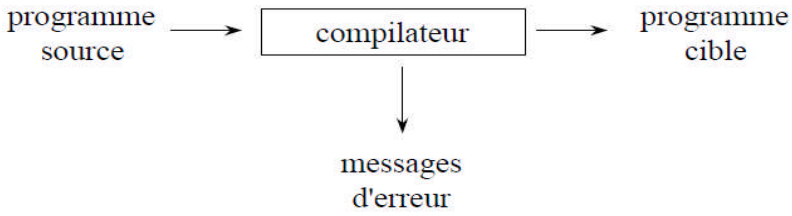


Figure 6 *: Schéma général d'un compilateur.*

b) Les phases d'un compilateur: [A10]

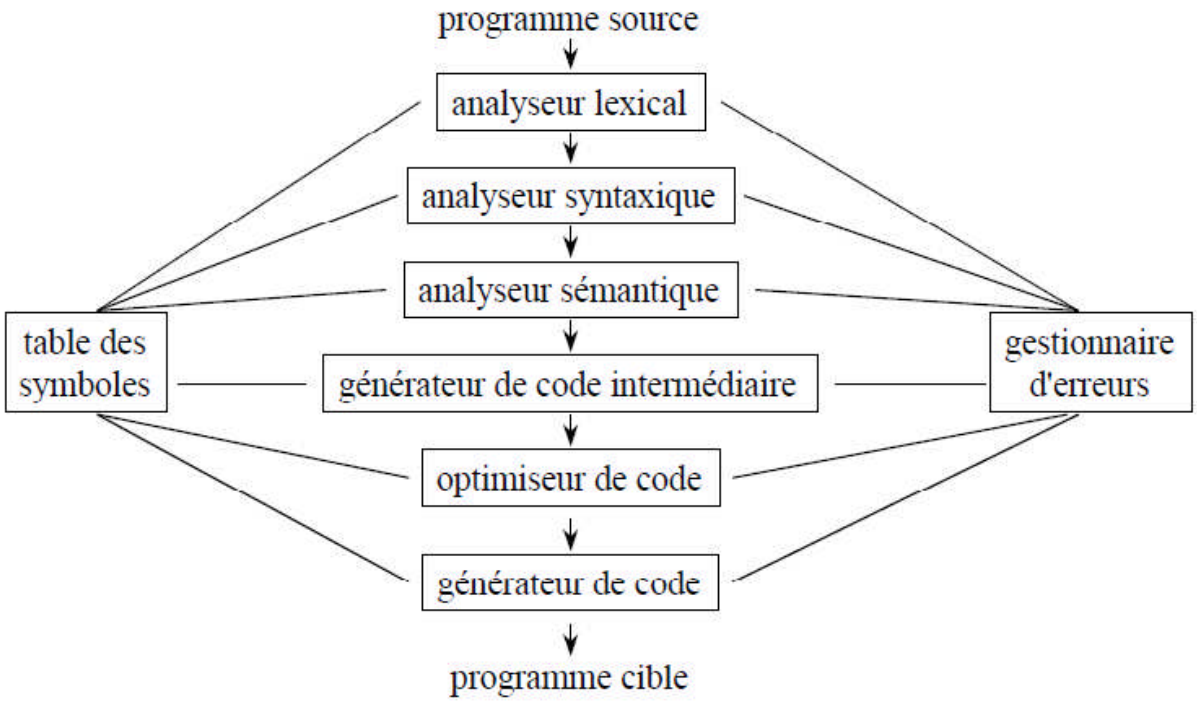


Figure 7 *: phases d'un compilateur*

c) Le rôles des différentes phases : [A10]

1. **L'analyse lexicale :** Consiste à :

- Lire le programme source caractère par caractère.
- Grouper les caractères en une séquence d'unités lexicales (les symboles du langage).
- Sauter les blancs et les commentaires.
- Eventuellement construire et initialiser la table des symboles.

2. **L'analyse syntaxique :**Consiste à vérifier que la séquence de terminaux retournée par la phase précédente est conforme à la syntaxe du langage considéré, décrite par une grammaire non contextuelle (ou grammaire de type 2).
3. **L'analyse sémantique:** L'**analyse sémantique** d'un message est la phase de son analyse qui en établit la signification en utilisant le sens des éléments (mots) du texte, par opposition aux analyses lexicales ou grammaticales qui décomposent le message à l'aide d'un lexique ou d'une grammaire.
4. **Génération de code intermédiaire:** Transformation du programme source « du haut niveau » en un programme class dans un langage intermédiaire.
5. **Code intermédiaire:** machine indépendante (non exécutable).cette étape améliore la quantité du code.
6. **Optimisation de code : l'optimisation** est la pratique qui consiste généralement à réduire le temps d'exécution d'une fonction, l'espace occupé par les données et le programme, ou la consommation d'énergie.

d) Les outils logiciels de compilation : **[A10]** Outils efficaces d'aide à la réalisation de compilateurs :

• Constructeurs d'analyseurs lexicaux (LEX) ;
• Constructeurs d'analyseurs syntaxiques ;
• Traducteurs dirigés par la syntaxe (YACC) ;
• Générateurs de code automatiques ;
•Moteurs d'analyse de flots de données.

- **JavaCC.**

e) Types d'analyses : **[A11]**

- **L'analyse LL [A11]** est une analyse descendante pour un sous-ensemble de grammaire non contextuelle. Il analyse l'entrée de gauche à droite (**L**eft to right) et en construit une dérivation à gauche (**L**eftmost derivation). Les grammaires analysables de cette façon sont nommées *grammaires LL*.

Une analyse LL est appelée analyse LL(k) lorsqu'elle utilise *k* lexèmes d'avance.

Générateurs d'analyseur LL(*k*) :

- ANTLR
- Coco/R
- JavaCC
- PCCTS : ancien nom de ANTLR,
- Ocaml Genlex Module

- **L' analyseur LR [A12]** (pour *Left to right, Rightmost derivation*) est un analyseur pour les grammaires non contextuelles qui lit l'entrée de gauche à droite et produit une dérivation droite. On parle aussi d'**analyseur LR(*k*)** où *k* représente le nombre de symboles "anticipés" et non consommés qui sont utilisés pour prendre des décisions d'analyse syntaxique.

D'habitude, *k* vaut 1 et est souvent omis. Une grammaire non contextuelle est appelée LR(*k*)

s'il existe un analyseur syntaxique LR(*k*) pour elle.

On dit qu'un analyseur syntaxique LR réalise une analyse ascendante car il essaye de déduire les productions du niveau du haut de la grammaire en les construisant à partir des **feuilles**.

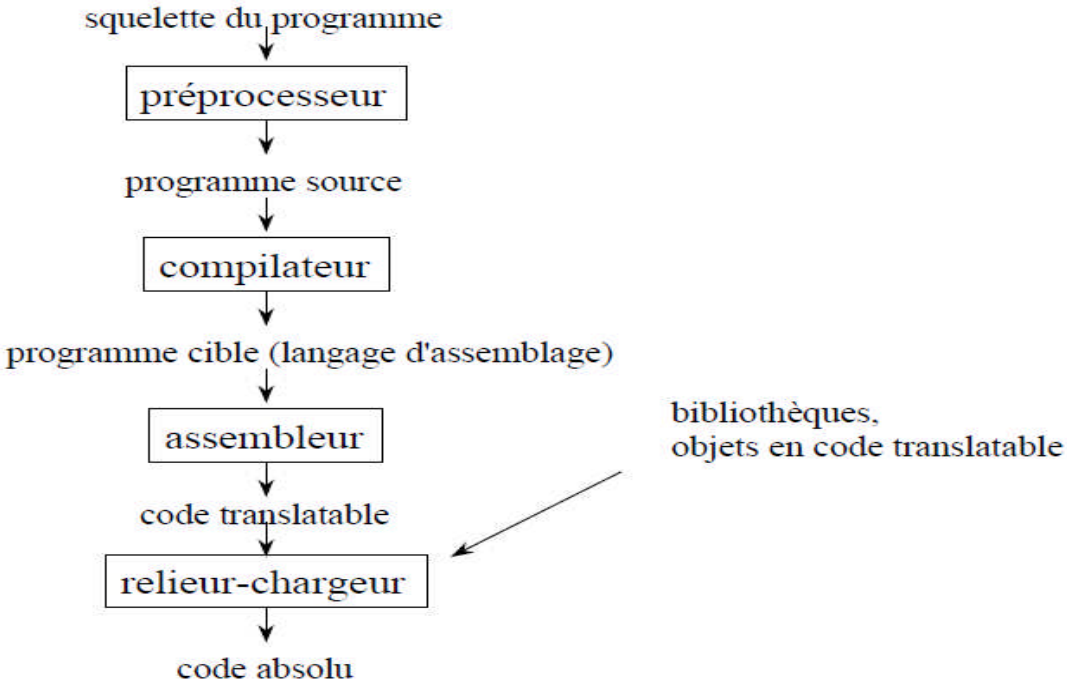


Figure 8 *: Environnement d'un compilateur.*

f) Quelques exemples d'évaluateurs d'expressions sur internet:

1. Evaluateur d'expressions régulières : [A13]

Figure 9 : *Interface de regexp.htm.*

Exemple :

- Masque : [a-z]+
- Chaine : BonjourATous
- Interrupteur : i

2. Evaluateur d'expressions régulières en JavaScript : [A13]

(OnFaitDuWeb.com)

(Programmation en web 2.0)
Il utilise simplement du javascript afin d'évaluer les expressions en cours de frappe.

3. Regular Expression Tool : [A14]

Figure 10 : ***Interface de Regular Expression Tool .***

Chapitre III : *conception et Réalisation.*

A. Conception :

a) Introduction :

Pour la réalisation de ce projet, on a besoin de quelques grammaire quand on va présenter dans de ce chapitre.

b) Grammaire générant les expressions arithmétiques :

E → E + E
| E - E
| T
T → T * T
| T / T
| F
F → Nombre
| - F

c) Grammaire générant les expressions logiques :

E → E OU E
| T
|not E
T → T ET
| T
| not T

d) Grammaire générant les expressions relationnelles :

T → T
| T <> T
| T > T
|T >= T
| T < T
| T <= T
|T=T

Ce projet ne contient pas seulement des opérations, il a aussi des fonctions comme la somme, moyenne, max…

En basant sur le tableau des priorités qui est le suivant, on obtient une grammaire ou qui génère toutes les expressions et fonctions possibles.

Priorité décroissante	**Associativité**
Ou logique	à gauche
Et logique	à gauche
Non logique	à droite
Opérateurs relationnels (= ;<> ;> ;>= ;< ;<=)	à gauche
(+) ;(-)	à gauche
(*) ;(/)	à gauche
(-) unaire	à droite
Fonction	à gauche

Figure 11 :*Tableau des priorités*.

La grammaire et donc :

<E_L> ➔ <E_L> OU <E_L>
| <T_L>
<T_L> ➔ <T_L> ET
| <T_L>
<T_L_R> ➔ <T_L_R> = <T_L_R>
| <T_L_R> <> <T_L_R>
| <T_L_R> > <T_L_R>
| <T_L_R> >= <T_L_R>
| <T_L_R> < <T_L_R>
| <T_L_R> <= <T_L_R>
| <F_L>
<F_L> ➔ E
| not <F_L>
E ➔ E + E
| E - E
| T
T ➔ T * T
| T / T
| F
F ➔ Nombre
| - F
| (<E_L>)
| Fonction

Fonction ➔ moyenne (<L_A>)
| variance (<L_A>)
| ecartype (<L_A>)
| max (<L_A>)
| min (<L_A>)
| somme (<L_A>)
| SommeCarres (<L_A>)
| SommeCarreEcarts (<L_A>)
| sin (Argument)
| cos (Argument)
| tan (Argument)
| sinh (Argument)
| cosh (Argument)
| tanh (Argument)
| puissance (Argument ; Argument)

<L_A> ➔ <L_A>< ;> Argument
| Argument

Argument ➔ <E_L>

Pour utiliser le langage Java et bien évidement le logiciel Java CC on doit transformer cette grammaire ambigüe en une grammaire de type LL(1).

e) Le langage Java : [A16]

Le langage Java est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au *SunWorld*.

Figure 12 *:Logo de Java*

Apparu en :23 mai 1995 .
Auteur :Sun Microsystems.
Développeurs :Oracle Corporation .
Paradigme :Programmation orientée objet, structurée et impérative.
Typage :Statique, fort, sûr, nominatif .
Influencé par :Objective-C, C++, Smalltalk, Eiffel .
A influencé :C#, D, J#, Ada 2005, Gambas .
Implémentations :Liste de JVM Système d'exploitation Multiplateformes Licence GNU GPL.

Site Web :www.java.com.
Dernière version : Java SE 7.

Figure 13 *:Duke, la mascotte de Java.*

f) Java CC : [A17]

Java Compiler Compiler (**JavaCC**) est un logiciel destiné à faciliter la réalisation de programmes informatiques en langage Java. En plus d'être un générateur de parser (un outil qui lit les spécifications d'une grammaire et qui la convertit en programme Java), JavaCC fournit d'autres possibilités relatives à la génération de parser comme la construction d'arbre et le débogage. C'est le plus utilisé des générateur de parser pour Java.
JavaCC prend comme entrée un fichier MaGrammaire.jj qui contient entre autres les descriptions des règles de la grammaire et produit un parser descendant (dans le fichier MaGrammaire.java). Une classe MaGrammaire est définie dans le fichier java. Elle implémente l'interface MaGrammaireConstants, définie dans MaGrammaireConstants.java et qui contient les définitions des mots clés de la grammaire.
C'est un logiciel libre distribué selon les termes de la licence BSD (Berkeley Software Distribution).

g) Eclipse : [A18]

Eclipse est un environnement de développement intégré libre extensible, universel et polyvalent, permettant de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions.
La spécificité d'Eclipse IDE vient du fait de son architecture totalement développée autour de la notion de plugin (en conformité avec la norme OSGi) : toutes les fonctionnalités de cet atelier logiciel sont développées en tant que plug-in.
Plusieurs logiciels commerciaux sont basés sur ce logiciel libre, comme par exemple *IBM Lotus Notes 8*, *IBM Symphony* ou *WebSphere Studio Application Developer*.

Figure 14 *:Logo Eclipse.*

Développeur : Eclipse Foundation
Dernière version : 3.6 Helios (23 juin 2010)
Environnements : Plateforme Java Langues Multilingue

Type : IDE
Licence : EPL
Site Web : www.eclipse.org.

h) Langages de programmation gérés par Eclipse : De nombreux langages sont supportés en 2008 (la plupart grâce à l'ajout de plugins), parmi lesquels : *Java, GAP pour system i, C#, Ada, C++, C, Cobol via LegacyJ, Objective Caml , Python, Perl (EPIC), Ruby, COBOL, Pascal, PHP (PDT), Javascript , XML, HTML, XUL, SQL, Action script, Coldfusion, Magik* .

i) NetBeans : [A19]

NetBeans est un environnement de développement intégré (EDI), placé en *open source* par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).
Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.
NetBeans constitue par ailleurs une plate forme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plate forme.

Figure 15 : *net beans*

Développeur :Sun Microsystems
Première version :1996, sous le nom de Xelfi
Dernière version :7.0 (20 avril 2011)
Environnements Plateforme Java
Langues Multilingue (français : partiellement)
Type IDE pour Java, PhP, C/C++, Fortran, JavaScript, Python, Ruby
Licence CDDL/ GPL
Site Web www.netbeans.org

B. Réalisation :

Apres Configuration de java cc avec Eclipse,on peut créer un programme en lançant :
File > New >JavaProject. Puis ,on clique sur New > Other > Java CC > Fichier d'exemple Java CC. Et ce qui nous donne un programme qui reconnue les expressions arithmitiques avec un point virgule a la fin. Ce programme contient un fichier de type jj (Java CC) et sept (7) classes de types Java.

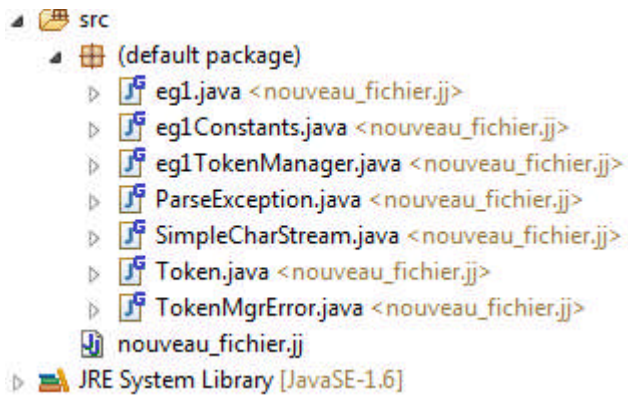


Figure 16 :*un programme initiale JavaCC sous Eclipse.*

Pour modifier le programme on doit modifier la classe le java cc (nouveau_fichier.jj) et exécuter le programme.

Après modification, on aura :

<E_L> ➔ <E_L> OU <E_L>

Devient :

<E_L> ➔ <T_L>X

X ➔ OU <T_L>X

| ε

<T_L> ➔ <T_L> ET <T_L>

Devient :

<T_L> ➔ <T_L_R> Y

Y ➔ ET <T_L_R>Y

| ε

<T_L_R> ➔ <T_L_R> '=' <T_L_R>

| <T_L_R> '<>' <T_L_R>

| <T_L_R> '>' <T_L_R>

|<T_L_R> '>=' <T_L_R>

| <T_L_R> '<' <T_L_R>

| <T_L_R> '<=' <T_L_R>

| <F_L>

Devient :

<T_L_R> ➔ <F_L>Z

Z ➔ '=' <T_L_R>Z

| '<>' <T_L_R>

| '>' <T_L_R>

| '>=' <T_L_R>

| '<' <T_L_R>

| '<=' <T_L_R>

| ε

En JavacCC, ça donne :

```
double  Expression_Logique()

{}

{Terme_Logique(<OU>Terme_Logique)*}
```

En JavacCC, ça donne :

```
double  Term_Logique()

{}

{Terme_Logique_Relationnel()( <OU>

                        Terme_Logique_ Rlationnel())*}
```

En JavacCC, ça donne :

```
double  Term_Logique_Relationnel()

{}

{Facteur_Logique()(<EGAL>Facteur_Logique()

                | <DIF>Facteur_Logique ()

                | <SUP>Facteur_Logique ()

                | <SUPEGAL>Facteur_Logique ()

                | <INF>Facteur_Logique ()

                | <INFEGAL>Facteur_Logique()  )*  }
```

E → E + E
| E - E

Devient :

E → TW
W → + T W
| - TW
| ε

T → T * T
| T / T

Devient :

E → FU
U → * FU
| / FU
| ε

<L_A> → <L_A>< ;> Argument
| Argument

Devient :

<L_A> → Argument<L_A>'
<L_A>' → ; Argument<L_A>'
| ε

En JavacCC, ça donne :

```
double  Expression ()

{}

{ Terme () (<PLUS>Terme()

            |<MOINS>Terme () )* }
```

En JavacCC, ça donne :

```
double  Terme ()

{}

{ Facteur () (<MUL>Facteur ()

            |<DIV>Facteur ()  )* }
```

En JavacCC, ça donne :

```
double  List_Argument ()

{}

{ Argument () (<point virgule> Argument ()  )* }
```

1) Ajout de routines et leurs emplacements :

<E_L> → <T_L> { X.h = <T_L>.s ; }
X { <E_L>.s=X.s ; }
X → OU
<T_L> { X.h = <T_L>.s OU X.h }
X { X.s = X.s ; }
| ε { X.s = X.h ; }

<T_L> → <T_L_R> { Y.h = <T_L_R>.s ; }
Y { <T_L>.s=Y.s ; }
Y → ET
<T_L_R> { Y.h = <T_L_R>.s ET Y.h }
Y { Y.s = Y.s ; }
| ε { Y.s = Y.h ; }

<T_L_R> → <F_L> { Z.h = <F_L>.s ; }
Z { <F_L>.s=Z.s ; }

```
Z          ➔ '= '
             <F_L>  {  Z.h = <F_L>.s ==  Z.h ;}
             Z      {   Z.s = Z.s ;   }
            | '<> '
             <F_L>  {  Z.h = <F_L>.s  <>  Z.h ; }
             Z      {   Z.s = Z.s ;   }
            | '> '
             <F_L>  {  Z.h = <F_L>.s  >  Z.h ; }
             Z      {   Z.s = Z.s ;   }
            | '>= '
             <F_L>  {  Z.h = <F_L>.s  >=  Z.h ; }
             Z      {   Z.s = Z.s ;   }
           | '< '
             <F_L>  {  Z.h = <F_L>.s  <  Z.h ; }
             Z      {   Z.s = Z.s ;   }
           | '<= '
             <F_L>  {  Z.h = <F_L>.s  <=  Z.h ; }
             Z      {   Z.s = Z.s ;   }
          | ε    {   Z.s = Z.h  ;  }


<F_L>      ➔ E       {  <F_L>.s = E. ;  }
           | not  <F_L>   { <F_L>.s =  not<F_L>.s ;  }
E          ➔  T    {   W.h = E.s ; }
              W   {   E.s=W.s ;  }
W          ➔  +
             T     { W.h = T.s  +  W.h ; }
             W   {   W.s=W.s ;  }
            | -
               T       { W.h = T.s  -  W.h ; }
               W       {   W.s=W.s ;  }   }
            | ε     {   W.s=W.h ;  }   }
T          ➔  F    {   T'.h = T.s ; }
              T'   {   T.s=T'.s ;  }
T'         ➔  *
             F     { T'.h = F.s  +  T'.h ; }
             T'   {   T'.s=T'.s ;  }
            | /
               F       { T'.h = F.s  -  T'.h ; }
               T'      {   T'.s=T'.s ;  }   }
            | ε     {   T'.s=T'.h ;  }   }
```

```
F            ➔ Nombre        { F.s=nombre.s ;    }
               | - F       {   F.s= - F.s ;   }
               | (<E_L>)   {   F.s=<E_L>.s ;     }
               | Fonction   {   F.s=Fonction.s ;   }
Fonction     ➔ moyenne
               (                    {cpt=0 ;  somme=0 ;  }
               Argument       {  somme= somme + Argument.s ;   cpt++ ; }
               X1      {  Fonction.s = Somme / cpt ;     }
               )
X1           ➔ ' ;'
               Argument     {   somme =somme + Argument.s ;   cpt++ ;    }
               X1
             | Argument   {   somme = somme = Argument.s ;   cpt++ ;  }

Fonction     ➔ variance
               (                {cpt=0 ;  somme=0 ; somme_carres=0 ; }
               Argument    {  somme= somme + Argument.s ;
                              Somme_carre=somme_carres+Argument.s* Argument.s;
cpt++ ; }
               X2      {  Fonction.s = (somme_carres –somme*somme / cpt)cpt ;     }
               )
X2           ➔ ' ;'
               Argument     {   somme =somme + Argument.s ;   cpt++ ;
                               Somme_carres=somme_carres+ Argument.s* Argument.s
}
               X2
             | Argument   {   somme = somme =+Argument.s ;   cpt++ ;
                               Somme_carres=somme_carres+ Argument.s* Argument.s
}
Fonction     ➔ Ecartype
               (                {cpt=0 ;  somme=0 ; somme_carres=0 ; }
               Argument    {  somme= somme + Argument.s ;
                              Somme_carre=somme_carres+Argument.s* Argument.s;
cpt++ ; }
               X3      {  Fonction.s = racine_carrée((somme_carres –somme*somme /
cpt)cpt)) ;    }
               )
X3           ➔ ' ;'
               Argument     {   somme =somme + Argument.s ;   cpt++ ;
                               Somme_carres=somme_carres+ Argument.s* Argument.s
}
               X3
             | Argument   {   somme = somme + Argument.s ;   cpt++ ;
```

Somme_carres=somme_carres+ Argument.s* Argument.s
}
Fonction ➔ Max
(Argument { max = Argument**.s** ;}
X4 { Fonction.s = max ; }
)
X4 ➔ ‘ ;’
Argument { si (max<Argument.s) max = Argument ; }
X4
| Argument { si (max<Argument.s) max = Argument ; }
Fonction ➔ min
(Argument { man = Argument**.s** ;}
X5 { Fonction.s = man ; }
)
X5 ➔ ‘ ;’
Argument { si (man<Argument.s) man = Argument ; }
X5
| Argument { si (man<Argument.s) man = Argument ; }
Fonction ➔ somme
({ somme=0 ; }
Argument { somme= somme + Argument.s ;}
X6 { Fonction.s = Somme ; }
)
X6 ➔ ‘ ;’
Argument { somme =somme + Argument.s ; }
X6
| Argument { somme = somme = Argument.s ; }
Fonction ➔ somme_carres
({ somme_carres=0 ; }
Argument { Somme_carres=somme_carres+Argument.s* Argument.s; }
X7 { Fonction.s = somme_carres ; }
)
X7 ➔ ‘ ;’
Argument { Somme_carres=somme_carres+ Argument.s* Argument.s
}
X7
| Argument { Somme_carres=somme_carres+ Argument.s* Argument.s
}
Fonction ➔ Somme_carre_ecarts
({cpt=0 ; somme=0 ; somme_carres=0 ; }
Argument { somme= somme + Argument.s ;

```
                                    Somme_carre=somme_carres+Argument.s* Argument.s;
cpt++ ; }
                    X8      {  Fonction.s = somme_carres –somme*somme / cpt ;     }
                    )
X8              ➔ ‘ ;’
                    Argument     {   somme =somme + Argument.s ;   cpt++ ;
                                     Somme_carres=somme_carres+ Argument.s* Argument.s
}
                    X8
                  | Argument   {   somme = somme + Argument.s ;   cpt++ ;
                                     Somme_carres=somme_carres+ Argument.s* Argument.s
}
Fonction      ➔ sin
                    (
                    Argument   { fonction.s=sin(Argument.s) ;  }
                    )
Fonction      ➔ cos
                    (
                    Argument   { fonction.s=cos(Argument.s) ;  }
                    )
Fonction      ➔ tan
                    (
                    Argument   { fonction.s=tan(Argument.s) ;  }
                    )
Fonction      ➔ sinh
                    (
                    Argument   { fonction.s=sinh(Argument.s) ;  }
                    )
Fonction      ➔ cosh
                    (
                    Argument   { fonction.s=cosh(Argument.s) ;  }
                    )
Fonction      ➔ tan
                    (
                    Argument   { fonction.s=tanh(Argument.s) ;  }
                    )
Fonction      ➔ puissance
                    (
                    Argument  { t=Argument. s; }
                    ‘;’
                     Argument  { fonction.s=power(t , Argument.s) ;  }
                    )
```

2) Evaluateur avec descente récursive :

```
Z()
{  Résultat=expression_Logique() ;
   Si (tc==' # ')   Afficher(Résultat) ;     }
expression_Logique()
{      Return   =X(Terme_Logique() )  }

X(h)
{  si (tc== 'ou')   Return   X(h or Terme_Logique() ) ;
  Sinon  return h ;   }
Terme_Logique()
{  return   Y(Terme-Logique_Relationnel()) ;   }

Y(h)
{ si (tc=='et')   return  Y(h  and Terme_Logique_Relationnel()) ;
  Sinon return h ; }

Terme_logique_relationnel()
{     return Z(Facteur_Logique() ;}
Z(h)
{   si (tc=='=') return (h==Facteur_Logique()) ;
    Sinon si (tc=='<>') return (h <>  Facteur_Logique()) ;
    Sinon si (tc=='>') return (h  >  Facteur_Logique()) ;
    Sinon si (tc=='>=') return (h  >=  Facteur_Logique()) ;
    Sinon si (tc=='<') return (h  <  Facteur_Logique()) ;
    Sinon si (tc=='<=') return (h  <=  Facteur_Logique()) ;
    Sinon return   h ;        }

Facteur_Logique()
{si (tc=='not')  return  not(Facteur_Logique()) ;
Sinon  return  E() ;       }

E()
{   return (W(T())) ;   }
```

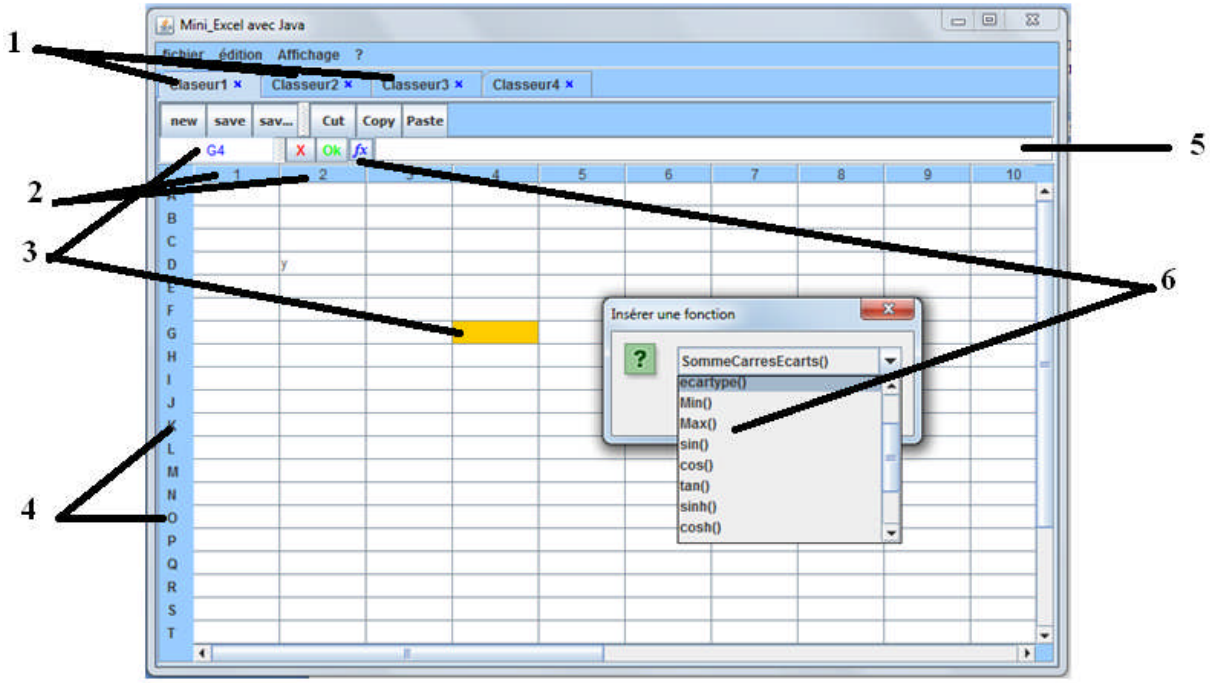
```
W(h)
{  Si (tc=='+')  return W(h+T()) ;
   Sinon si  (tc=='-')  return (h-T()) ;
  Inon return h ;        }

T()
{  return (T'(F())) ;         }

T'(h)
{  Si (tc=='*')
              Tc=ts ;
               return(h*F()) ;
  Sinon si
                Tc=ts ;
                (tc=='/')  return  (h/F()) ;
  Sinon  return h ;          }

F()
{ si tc=='('
 Begin
        Tc=ts ;
        V=E() ;
        Si tc==')'
        Begin
             Tc=ts ;
             Return  V ;
        End
        Sinon  « ERREUR »
  End
  Sinon
        Si  (tc== chiffre)
             Tc=ts ;
             Return chiffr.val ;
        Sinon return fonction() ;
}

Fonction
{
     Si (tc=='moyenne')  et ts=='('
                                  Tc=ts ; tc=ts ;
                              Somme = Argument() ;    cpt =1 ;
                                  X1(somme,cpt) ;
                              Si  tc==')'
                                    Tc=ts ;
```

```
                                    Return  somme/cpt ;
                                Sinon « «ERREUR »
     Sinon si (tc==’Variance’)  et ts==’(‘
                                    Tc=ts ; tc=ts ;
                                Somme = Argument() ;    somme_carre=somme*somme ;  cpt
=1 ;
                                    X2(somme,somme_carres,cpt) ;
                                Si  tc==’)’
                                    Tc=ts ;
                                    Return  (somme_carres-somme*somme/cpt)/cpt ;
                                Sinon « «ERREUR »
     Sinon  Si (tc==’ecartype’)  et ts==’(‘
                                    Tc=ts ; tc=ts ;
                                Somme = Argument() ;    somme_carre=somme*somme ;  cpt
=1 ;
                                    X2(somme,somme_carres,cpt) ;
                                Si  tc==’)’
                                    Tc=ts ;
                                    Return   racine (somme_carres-somme*somme/cpt)/cpt ;
                                Sinon « «ERREUR »
     Sinon  Si (tc==’max’)  et ts==’(‘
                                    Tc=t ;   tc=ts ;
                                Max=Argument() ;
                                Max = X4(max) ;
                                Si tc==’)’
                                        Tc=ts ;
                                    Return max ;
                            Sinon « ERREUR »
     Sinon  Si (tc==’min’)  et ts==’(‘
                                    Tc=t ;   tc=ts ;
                                Min=Argument() ;
                                Min = X5(min) ;
                                Si tc==’)’
                                        Tc=ts ;
                                    Return min ;
                            Sinon « ERREUR »
Si (tc==’Somme’)  et ts==’(‘
                                    Tc=ts ; tc=ts ;
                                Somme = Argument() ;
                                    X6(somme) ;
                                Si  tc==’)’
                                    Tc=ts ;
                                    Return  somme;
                                Sinon « «ERREUR »
```

```
Sinon si (tc==’Somme_carres’)  et ts==’(‘
                                    Tc=ts ; tc=ts ;
                                somme_carre = Argument ();
                                    somme_carre= somme_carre* somme_carre;
                                    somme_carres =  X7(somme_carres) ;
                                Si  tc==’)’
                                        Tc=ts ;
                                        Return  somme_carres ;
                                Sinon « «ERREUR »
Sinon si (tc==’ somme_carres_ecarts’)  et ts==’(‘
                                    Tc=ts ; tc=ts ;
                                Somme = Argument() ;    somme_carre=somme*somme ;  cpt
=1 ;
                                     X2(somme,somme_carres,cpt) ;
                                Si  tc==’)’
                                        Tc=ts ;
                                        Return  (somme_carres-somme*somme/cpt);
                                Sinon « «ERREUR »
Sinon si (tc==’ sin’)  et ts==’(‘
                                    Tc=ts ; tc=ts ;
                                     V=sin(Argument()) ;
                                Si tc==’)’
                                      Tc=ts ;
                                              Return V ;
                                Sinon « ERREUR »
Sinon si (tc==’ cos’)  et ts==’(‘
                                    Tc=ts ; tc=ts ;
                                     V=cos(Argument()) ;
                                Si tc==’)’
                                      Tc=ts ;
                                              Return V ;
                                Sinon « ERREUR »
Sinon si (tc==’tan’)  et ts==’(‘
                                    Tc=ts ; tc=ts ;
                                     V=tan(Argument()) ;
                                Si tc==’)’
                                      Tc=ts ;
                                              Return V ;
                                Sinon « ERREUR »
Sinon si (tc==’ sinh’)  et ts==’(‘
                                    Tc=ts ; tc=ts ;
                                     V=sinh(Argument()) ;
                                Si tc==’)’
                                      Tc=ts ;
```

```
                                        Return V ;
                                    Sinon « ERREUR »
Sinon si (tc==’ cosh’)  et ts==’(‘
                                    Tc=ts ; tc=ts ;
                                     V=cosh(Argument()) ;
                                    Si tc==’)’
                                       Tc=ts ;
                                            Return V ;
                                    Sinon « ERREUR »
Sinon si (tc==’ tanh’)  et ts==’(‘
                                    Tc=ts ; tc=ts ;
                                     V=tanh(Argument()) ;
                                    Si tc==’)’
                                       Tc=ts ;
                                            Return V ;
                                    Sinon « ERREUR »
Sinon si (tc==’ puissance’)  et ts==’(‘
                                    Tc=ts ; tc=ts ;
                                     V1=Argument() ;
                                   Si   tc==’ ;’
                                        Tc=ts ;
                                        Resultat=pow(v1,Argument) ;
                                   Sinon  « ERREUR »
                                          Si tc==’)’
                                                Tc=ts ;
                                                      Return V ;
                                          Sinon « ERREUR »
   Sinon « ERREUR »
}
```

- Pour le lancement de testes et la récupération de résultat on a besoin d’une table qui ressemble à celle de l’Excel, on va la faire avec la plate netbeans et on introduit le source comme une classe dans le programme de Eclipse. On obtient l’interface suivante:



1 : Classeurs (ressemble à feuilles en Excel)
2 :N° de colonne
3 : Nom de la zone de travail (n° ligne et colonne) (ici c'est G4)
4 : N° de ligne
5 : Zone d'insertion (on insert les formules, expressions, fonctions)
6 : Fonctions et insertion de fonctions

Figure 17 :*Plate-forme de l'application réalisée.*

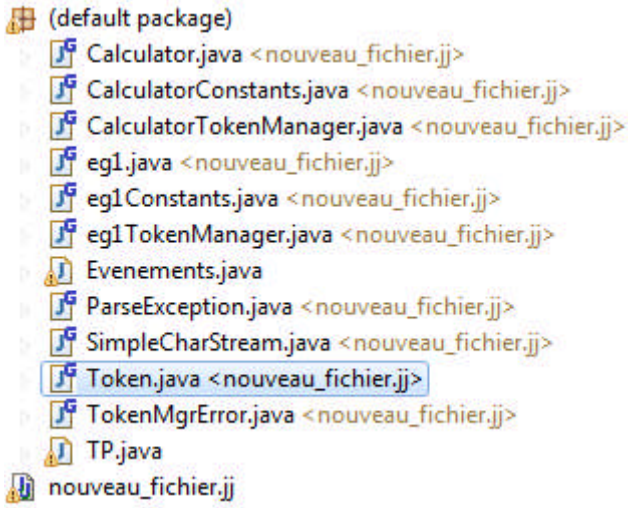


Figure 18 : *Programme finale de l'application projet fait.*

Chapitre IV : *Testes*

Dans ce chapitre, on va présenter quelque testes effectuer sur l'application de ce projet :

1) **Fichier > nouveau** ==> nouveau tableau (feuille).
2) **Fichier > ouvrir** ==> permet de créer un fichier ou de nommer la feuille de travail.
3) **Fichier >enregistrer** ou **enregistrer Sous** ==> permet d'enregistrer un fichier
4) **Fichier > quitter** ==> étendre l'application.
5) **OK** ==> permet de valider le teste ou tester ce qu'on a écrit.
6) **Copy, cut, save et paste** ==> chacun nous permet son fonctionnement .
7) **F(x)** ==> permet de choisir une fonction parmi les fonctions programmer sous l'application ; la somme, le sinus, le maximum,…
8) **Somme(2,4,6)** ==> donne un résultat qui est :12 .
   Le résultat sera mis dans une case qu'on a choisi.
9) **Somme(A1,2)** ==> donne ou annonce que la case A1 est vide.
   Par ce que la case A1 n'est pas rempli.
10) **Max(A1,B5,C7)** ==> permit de donner le maximum entre le contenu de A1(ici c'est -48),le contenu de B5 (ici c'est 500) et le contenu de C7 (ici c'est 87) ; le résultat est rendu dans G1qui est 500 (contenu de B5)
11) **Tan (0)** ==> Donne le résultat qui est 0 dans une case sélectionnée (ici c'est la case A8)
12) **La sélection d'une fonction** ==> se fera d'une façon suivante :
    Cliquer sur f(x) > sélection de la fonction (ici c'est Max() (maximum)) > OK.

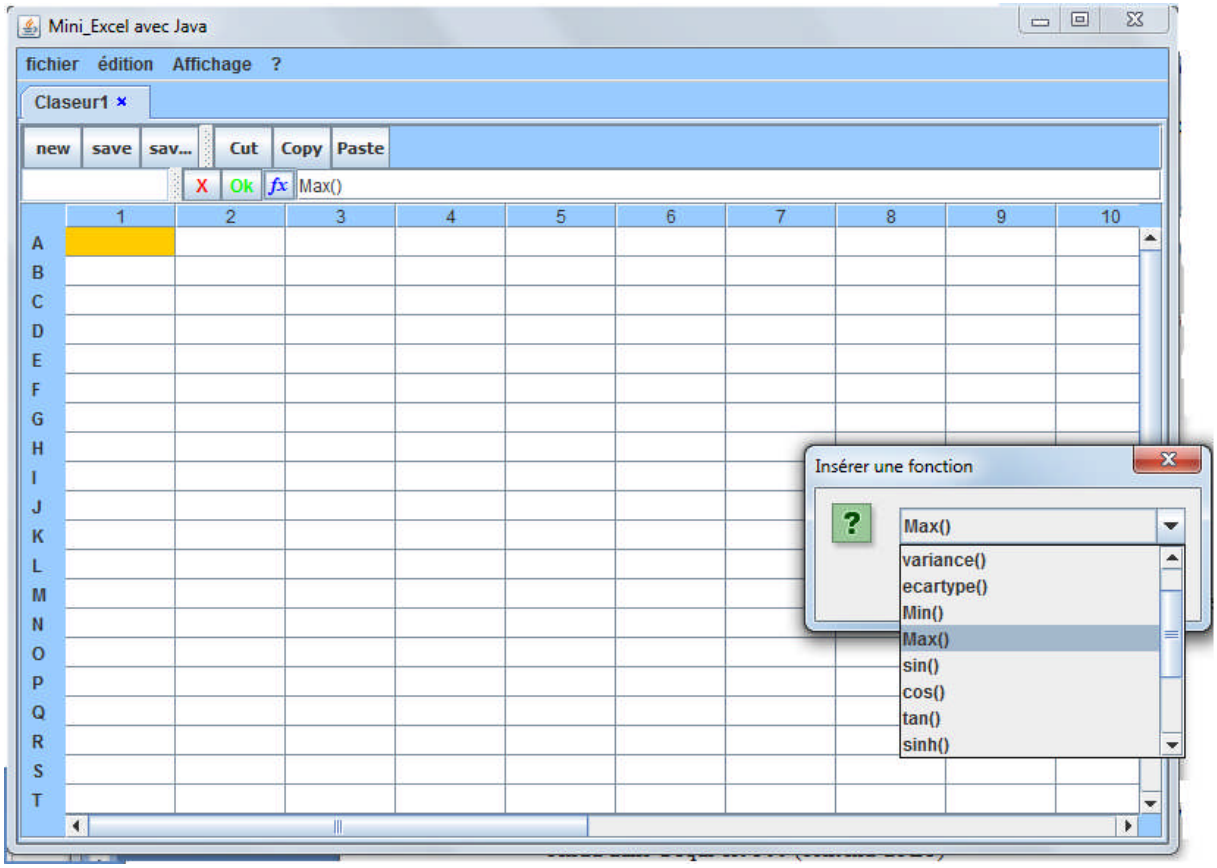


Figure n°19 : *Sélection d'une fonction*

13) **Après la sélection de la fonction** SommeCarres(), on lui insére 3 et 8 ; on insere aussi 1 et &.

Donc la formule sera :1 & SommeCarres (3,8).

Le résultat c'est : 1

Explication : SommeCarres (3,8) = $3^2 + 8^2 = 9 + 64 + 73$.

1 & 73 = 1 . Car on a supposé que 0= FAUX et tous les autres chiffres et nombres sont considérés égale à 1 =VRAI dans les opérations logiques.

Conclusion

A la fin de ce projet, on a eu la chance d'enrichir nos connaissances sur la compilation en générale, en assimilant les analyseurs LL et LR, et les évaluateurs d'expressions régulières en particulier, grâce à la présence sur internet des travaux qui sont entrain de se faire sur ce type de recherche et les sites spécialisés dans ce domaine.

L'application qu'on a programmé effectue la vérification lexicale et syntaxique des expressions arithmétiques, logiques et relationnelles et les évalue en donnant un résultat.

On s'est familiarisé un peut plus avec le langage Java, avec les IDEs Eclipse et NetBeans ; on a aussi apprit qu'est ce que c'est que le logiciel Java CC, ses caractéristiques, ses objectifs et ses fonctionnalités.

On a appris un plus sur le logiciel Microsoft Excel comme les dessins et ses fonctions ;
on a aussi appris beaucoup sur les logiciels de Microsoft Office et d'autres logiciels comme ceux de Macintosh et OS /2.

Bibliographie

[A1]http://fr.wikipedia.or/wiki/Expression_rationnelle.htm,29 mai 2011 à 00:31.
[A2] http://fr.wikipedia.org/wiki/Microsoft Office (Microsoft Office-Wikipédia.htm), 18 avril 2011 à 19:09.
[A3] http://fr.wikipedia.org/wiki/Excel (Microsoft Excel - Wikipédia.htm), 21 avril 2011 à 07:30.
[A4] http://fr.wikipedia.org/wiki/Tableau_crois%C3%A9_dynamique
[A5] http://fr.wikipedia.org/wiki/Théorie_des_langages (Théorie_des_langage.htm), le 14 mars 2011 à 16:14.
[A6] http://fr.wikipedia.org/wiki/Wikip%C3%A9dia:%C3%80_propos
[A7] http://fr.wikipedia.org/wiki/Grammaire_régulière(Grammaire_régulière.htm) ,23 septembre 2010 à 12:50
http://fr.wikipedia.org/wiki/Wikip%C3%A9dia:Avertissements_g%C3%A9n%C3%A9raux
[A8] cours compilation quatrième année ingénieur,2009-2010.
[A9]http://fr.wikipedia.org/wiki/Compilation_(informatique)(Compilation_(informatique).htm), 14 mai 2011 à 07:42.
[A10] cours de compilation 3eme année LMD, UMMTO, 2010-2011.
[A11] http://encyclo.voila.fr/wiki/Analyse_LR (Anlyse_LR.htm), Dimanche 29 Mai 2011.
[A12] http://encyclo.voila.fr/wiki/Analyse_LR (Anlyse_LR.htm), Dimanche 29 Mai 2011.
[A13] http://www.wikistuce.info/doku.php/regexp /regexp.htm, 18/12/2006 18:20
[A14] http://www.**Évaluateur d'expressions régulières en JavaScript « OnFaitDuWeb.com.htm**
[A15] http://www./regexp.html
[A16] http://fr.wikipedia.org/ Java_(langage).htm
[A17] http://fr.wikipedia.org/ JavaCC.htm
[A18] http://fr.wikipedia.org/ Eclipse_(logiciel).htm
[A19] http://fr.wikipedia.org/ NetBeans.htm