ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE

2ème Année Cycle Supérieur (2CS)

2018/2019

PROJET COMPIL

Encadré par : Mr CHEBIEB ABDELKRIM

Réalisé par : Bouzidi Halima

Groupe : SIQ3

I. Forme de Backus-Naur (BNF) :

La forme de Backus-Naur (souvent abrégée en BNF, de l'anglais Backus-Naur Form) est une notation permettant de décrire les règles syntaxiques (en) des langages de programmation. C'est donc un métalangage. Elle est utilisée dans certains livres pour décrire le langage étudié, mais également par de nombreux logiciels d'analyse syntaxique pour travailler sur des fichiers sources de plusieurs langages différents. Elle est une notation pour des grammaires formelles de type hors-contexte (car on définit les termes hors de leur contexte, pour replacer ensuite la définition desdits termes dans ce contexte).

Cette syntaxe a été conçue par John Backus et Peter Naur lors de la création de la grammaire du langage Algol 60.

II. La grammaire du langage de spécifications :

i. Les déclarations des symboles terminaux de la grammaire :

<POINTS> ::= :

< COMMA > ::= ,

< LBR > ::= (

< RBR > ::=)

< SEMIC > ::= ;

< ASSIGN > ::= :=

< Relation > ::= ::

< OPENACCOLADE > ::= {

< CLOSEDACCOLADE > ::= }

< Equal > ::= =

< Different > ::= <>

< Comp > ::= < | > | <= | >=

< APOSTROPHE > ::= ‘

< PLUS_SIGN > ::= +

< MINUS_SIGN > ::= -

< MULT_SIGN > ::= *

< DIV_SIGN > ::= /

< CHOIX > ::= ~

< AND > ::= AND | And | and

< THEN > ::= THEN | Then | then

< DEBUT > ::= DEBUT | Debut | debut

< IF > ::= IF | If | if

< INTERFACE > ::= INTERFACE | Interface | interface

< INIT > ::= INITIALISATIONS | Initialisations | initialisations

< ACTIONS > ::= ACTIONS | Actions | actions

< COMPOSANT > ::= COMP

< PROP > ::= PROP

< EVT > ::= EVT | EVET

< INV > ::= INV

< FIN > ::= FIN

< Int > ::= int

< Bool > ::= Boolean | Bool | bool

< String > ::= STRING | String | string

< Char > ::= CHAR | Char | char

// Ici on déclare les types standards des composants qu'on peut en avoir dans le programme de spécifications.

< TypeCompo > ::= Fenetre | Boite Dialogue| Panel| Bouton | Champs Texte | Combo Box | Liste Deroulante | Case A cochet | Bouton Radio| Icone|Menu

// Ici on déclare les actions standards sur les composants qu'on peut en avoir dans le programme de spécifications.

< Type1> ::= DbClick | Click |Drag | Drop | ChrKeyPress | NumKeyPress | EntrerKeyPress | EnterPress

< Type2Fen > ::= ClickBtFerm | ClickBtRed | ClickBtArg | ClickTitre

< LOGIC_CONSTANT > ::= True | False | TRUE | FALSE | true | false

< NUMBER > ::= (<DIGIT>)+

< IDLIB > ::= <LETTER>(<LETTER>|<DIGIT>)*

< LETTER > ::= [a - z , A - Z]

< DIGIT > ::= [0 - 9]

ii. Les déclarations des symboles non terminaux de la grammaire :

< Prog > ::= < S > <EOF>

< S > ::= <DEBUT> <Interface> <Init> <Actions> <FIN>

<Interface> ::= <INTERFACE> <ListComposant>

// Entre une déclaration d'un composant et un autre on doit mettre obligatoirement un ' ;'

<ListComposant> ::= <Composant> <SEMIC> (<Composant><SEMIC>)*

<Composant> ::= <COMPOSANT> <IDLIB> <Composant> <POINTS> <TypeCompo> <choice1> <Propi> <choice>

<choice1> ::= (<LDB> <IDLIB> <RDB> <SEMIC>) | <SEMIC>

//Le symbole non terminal "choice" nous permet de générer soit des Evènements seulement ou bien des déclarations de de type « INV » et des évènements.

<choice> ::= (<Invs> <Events>) | <Events>

<Invs> ::= <INV> <ListInv>

// Chaque declaration de type INV doit se terminer obligatoirement par ';'

<ListInv> ::= <Inv> (<SEMIC> <Inv>)*

<Inv> ::= <IDLIB> <ASSIGN> (<IDLIB> | <NUMBER> | <LOGIC_CONSTANT>)

// Generation des propriétés.

<Propi> ::= <PROP> <ListProp>

// Chaque déclaration d'une propriété doit se terminer obligatoirement par ';'

<ListProp> ::= <Prop> <SEMIC> (<Prop> <SEMIC>)*

//La déclaration de la propriété doit être sous la forme suivante :

« Nom de la propriété » : « Type de la propriété »

<Prop> ::= <ListIDLIB> <POINTS> <IDLIBs>

<ListIDLIB> ::= <IDLIB> (<COMMA> <IDLIB>)*

// Le type d'une propriété est soit : énumération, chaine de caractères, entier ou bien booléen.

<IDLIBs> ::= (<OPENACCOLADE> <ListeIDLIB> <CLOSEDACCOLADE>)| <Bool> | <Int> | <String> | <Char>

<ListeIDLIB> ::= <IDLIB> (<COMMA> <IDLIB>)*

<Events> ::= <EVT> <ListEvent>

// La déclaration de chaque évènement doit se terminer obligatoirement par un ' ;'

<ListEvent> ::= <Event> <SEMIC> (<Event> <SEMIC>)*

<Event> ::= <TypeEvent> <choice3> <ListTypeEvent> <CLOSEDACCOLADE>

<choice3> ::= (<LDB> <IDLIB> <RDB> <OPENACCOLADE>) | <OPENACCOLADE>

<TypeEvent> ::= <Type1> | <Type2Fen>

<ListTypeEvent> ::= (<IFCond> | <Aff>) (<SEMIC> (<IFCond> | <Aff>))*

<IFCond> ::= <IF> <Condition> <THEN> (<Affectation> | <IFCond>)

<Condition> ::= <IDLIB> <SensCond> (<IDLIB> | <NUMBER> | <LOGIC_CONSTANT>)

<SensCond> ::= <Equal> | <Diffrent> | <Comp>

<Affectation> ::= <Aff> (<AND> <Aff>)*

// L'affectation d'une valeur à une propriété dans une action doit se faire à travers le seul symbole de l'affectation ' :=' et non pas

<Aff> ::= <E> (<ASSIGN> <E>)?

<E> ::= <T> ((<PLUS_SIGN>|<MINUS_SIGN>) <T>)*

<T> ::= <F> ((<MULT_SIGN>|<DIV_SIGN>) <F>)*

<F> ::= <LBR> <E> <RBR> | <IDLIB> | <NUMBER> | <LOGIC_CONSTANT> | (< APOSTROPHE > <IDLIB> < APOSTROPHE >)

<Init> ::= <INIT> <ListInit>

<ListInit> ::= <InitObj> <SEMIC> (<InitObj> <SEMIC>)*

<InitObj> ::= <IDLIB> <Relation> <IDLIB> <ASSIGN> (<IDLIB> | <NUMBER> | <LOGIC_CONSTANT>)

<Actions> ::= <ACTIONS> <ListActions>

<ListActions> ::= ((<OPENACCOLADE> <ListActions> <transit> <ListActions> <repeat> <CLOSEDACCOLADE>) <repeat>) | <singleAction>

<repeat> ::= (<transit> <ListActions>)*

<transit> ::= <SEMIC>|<PLUS_SIGN>|<CHOIX>

<singleAction> ::= <Type1><LBR><IDLIB><RBR>