

## CAHIER DES CHARGES

Réf. : PSN-EXP/BEPAM/2025-00057

**SENSIBLE:** Non

**Objet : Nouveau langage pour le compilateur du logiciel probabiliste KANT**

**Documents associés :**

### HISTORIQUE DES MODIFICATIONS DU DOCUMENT

Indice	Date	Nature de la modification
1	24/06/2026	Version initiale du document

Nom et visa du rédacteur :  Gaëtan BAYLE-RUAULT DES COURCHAMPS  Date :	Nom et visa du vérificateur :  Nadia RAHNI  Date :	Nom et visa de l'approbateur :  Emmanuel RAIMOND  Date :
--	---	---

## Table des matières

1	CONTEXTE DE LA PRESTATION.....	4
2	LES ACTIVITES DU SCEPS ET DU BEPAM.....	4
3	OBJET DE LA PRESTATION .....	4
3.1	Le code KANT.....	4
3.2	Motivations de la prestation.....	5
4	DESCRIPTION DE LA PRESTATION .....	5
4.1	Volet 1 : Prise en main de KANT et du prototype .....	6
4.1.1	<i>Prise en main de KANT et de son architecture</i> .....	6
4.1.2	<i>Prise en main du prototype</i> .....	6
4.2	Volet 2 : Extension du prototype et adaptation du Compilateur .....	6
4.2.1	<i>Extension du prototype</i> .....	6
4.2.2	<i>Adaptation du compilateur pour produire le code Lua du prototype</i> .....	7
4.3	Volet 3 : Extension à tout le langage KANT (hors fonctions externes) .....	7
4.4	Volet 4 : Extension aux fonctions externes .....	8
4.5	Volet 5 : Débogage et cas-tests sémantiques.....	8
5	DOCUMENTS ET DONNEES D'ENTREE .....	8
6	MODALITES D'EXECUTION.....	9
6.1	Durée .....	9
6.2	Délais de réalisation .....	9
6.3	Lieu d'exécution .....	9
6.4	Organisation et suivi de la prestation .....	9
7	LIVRABLES.....	10
7.1	Modalités de réception.....	10
7.2	Résultats attendus.....	11
8	SAVOIR-FAIRE REQUIS.....	11
9	INTERLOCUTEURS ASNR .....	12
10	ANNEXE : SYNTAXE RESUMEE DU LANGAGE KANT .....	13
10.1.1	<i>Caractères et mots réservés</i> .....	13
10.1.2	<i>Lignes et commentaires</i> .....	13
10.1.3	<i>Identificateurs</i> .....	13

10.1.4	Constantes .....	13
10.1.5	Types de données .....	13
10.1.6	Déclaration .....	13
10.1.7	Opérateurs.....	14
10.1.8	Expressions.....	14
10.1.9	Bloc d'instructions .....	14
10.1.10	Questions.....	14
10.1.11	Fonction interne.....	14
10.1.12	- Opérateurs du langage.....	14
10.1.13	Fonctions du langage .....	14
10.1.14	Variables spéciales .....	15

## **1 CONTEXTE DE LA PRESTATION**

Le présent cahier des charges porte sur l'utilisation du langage Lua pour le compilateur du logiciel probabiliste KANT.

Cette prestation sera effectuée pour le compte du Bureau d'Évaluation Probabiliste des Accidents Majeurs (BEPAM), au sein du Service de la Conduite des réacteurs et des Etudes Probabilistes de Sûreté (SCEPS) de la Direction de l'Expertise en Sûreté de l'ASNR.

Le présent cahier des charges définit les objectifs et les conditions techniques de réalisation de la prestation.

## **2 LES ACTIVITES DU SCEPS ET DU BEPAM**

Le SCEPS a pour mission de mener des expertises de sûreté et des études dans le domaine de la conduite des réacteurs et de la fiabilité humaine, ainsi que des Études Probabilistes de Sûreté (EPS). Ces missions concernent les réacteurs électrogènes en fonctionnement (REP, EPR), les réacteurs EPR2, et les installations nucléaires LUDD (Laboratoires, Usines, Déchets et Démantèlement).

Le BEPAM est, notamment, chargé de :

- Développer les méthodes, les outils probabilistes et les outils de calcul nécessaires aux EPS de niveau 2 (EPS2) ;
- Réaliser les quantifications probabilistes des EPS2 et de leur prolongement éventuel ;
- Contribuer à la définition et au suivi des études nécessaires à la réalisation des EPS2 et réaliser certaines de ces études, notamment pour l'évaluation des différents modes de défaillance du confinement et des rejets radioactifs induits ;
- Contribuer à l'utilisation des EPS2 dans l'expertise de sûreté, coordonner et participer à l'analyse des EPS2 réalisées par l'exploitant ;
- Evaluer, en liaison avec les autres unités de l'ASNR, la cohérence des dispositions prévues dans les plans d'urgence interne et les plans particuliers d'intervention vis-à-vis des rejets calculés dans les EPS2 ;
- Contribuer à l'utilisation des EPS2 pour la gestion de crise et à la validation des outils de diagnostic et de pronostic du Centre Technique de Crise de l'ASNR.

## **3 OBJET DE LA PRESTATION**

### **3.1 Le code KANT**

Développé à partir de 1996 par le BEPAM, le code probabiliste KANT permet le développement d'une EPS2 grâce aux concepts suivants :

- L'entrée de l'étude est un ensemble d'EDI (État Dégradé de l'Installation) décrivant l'état de l'installation nucléaire en cas de fusion du cœur. Chaque EDI est une combinaison de valeurs prises par certaines variables, que l'on désigne comme « variables EDI » ;
- Un ADAG (Arbre de Déroulement des Accidents Graves) est un enchaînement d'événements, modélisés par des « questions » que l'on représente comme les feuilles d'un arbre. Les nœuds de l'arbre définissent les variables et paramètres auxquels accèdent leurs descendants ;

- Chaque question a un nombre fini de réponses possibles. Son comportement est décrit dans un langage simple ;
- Le langage KANT permet également de définir des fonctions, que toute question d'un ADAG peut utiliser (cf. § 10 pour une description du langage KANT) ;
- Un regroupement est une suite ordonnée de variables. À chaque succession de réponses qui mène à la sortie de la dernière question, KANT prend les valeurs des variables du regroupement pour former une FPA (Famille de Progression d'Accident). Les variables du regroupement sont dites « variables FPA » ;
- La suite des réponses menant d'un EDI au regroupement est appelée « séquence ». Il arrive souvent que plusieurs séquences aboutissent à la même FPA.

Le code KANT est constitué de plusieurs modules, dont les principaux sont donnés ci-dessous :

- Développeur : cette interface graphique sert à développer et représenter l'ADAG. Elle permet notamment de constituer et de gérer les EDI, et décrire précisément le déroulement de l'accident via les questions et les fonctions écrites en langage KANT ;
- Compilateur : ce programme, en ligne de commande uniquement, compile le langage KANT en un code intermédiaire (p-code) exécuté par une machine virtuelle simple dans le module Moteur ;
- Quantificateur : cette interface graphique aide à définir les études, il s'agit en particulier des valeurs des paramètres à considérer ou des lois de probabilité à leur attribuer, et des EDI à sélectionner ; elle peut également lancer des quantifications de manière plus conviviale qu'en « ligne de commande » (cf. ci-dessous) ;
- Moteur : ce programme en ligne de commande réalise les quantifications, à partir d'un ADAG défini par le Développeur, et d'une étude définie par le Quantificateur.

En plus de ces modules, KANT utilise, via son module moteur, deux bibliothèques qui ne sont pas concernées par la prestation : MER pour le calcul des rejets, PATSY pour l'appel à des fonctions métier depuis l'ADAG.

### **3.2 Motivations de la prestation**

Les modules de KANT sont développés sous Windows. Le module Moteur existe aussi dans une version pour Linux.

Des développements effectués en 2023 et 2024 ont doté le moteur et le compilateur d'un certain nombre de tests unitaires, et ont rendu plus lisible le format des fichiers de travail.

D'autres travaux ont porté sur le compilateur de KANT. Actuellement ce dernier traduit le langage KANT en un code intermédiaire (p-code), qui est exécuté par le module Moteur. Or il est apparu que le p-code pouvait être avantageusement remplacé par le langage Lua, dont les interpréteurs sont fiables et rapides. Ce choix repose sur les performances constatées avec le prototype réalisé pendant ces travaux.

## **4 DESCRIPTION DE LA PRESTATION**

La prestation concerne les modules Moteur et Compilateur du logiciel probabiliste KANT.

Elle consiste essentiellement à permettre au compilateur de KANT de produire sa sortie en Lua au lieu du p-code, et au moteur d'exécuter ce code Lua. Il n'est pas demandé de modifier le langage KANT lui-même.

La prestation s'articule autour des volets décrits ci-après.

- Volet 1 : Prise en main de KANT et du prototype
- Volet 2 : Extension du prototype et adaptation du Compilateur
- Volet 3 : Extension à tout le langage KANT (hors fonctions externes)
- Volet 4 : Extension aux fonctions externes
- Volet 5 : Support du débogage et cas-tests sémantiques

#### **4.1 Volet 1 : Prise en main de KANT et du prototype**

##### Prise en main de KANT et de son architecture :

Le titulaire se familiarisera suffisamment avec l'utilisation de KANT pour être en mesure de développer, compiler et exécuter un ADAG élémentaire.

L'intervenant se familiarisera également avec le code source C++ de KANT, pour être en mesure de :

- Recompiler les modules Moteur et Compilateur de KANT ;
- Comprendre la production du p-code par le compilateur ;
- Comprendre l'exécution du p-code par le Moteur ;
- Rejouer les cas-tests disponibles et en comprendre la structure.

##### Prise en main du prototype :

Le prototype consiste en un ajout au module Moteur (300 lignes de code C++). Il traite les aspects suivants de l'exécution d'une question de l'ADAG à partir de code LUA :

- Passage des valeurs des variables du code C++ du moteur vers le code LUA ;
- Déclaration d'une variable locale de type entier, réel, ou tableau ;
- Boucles et exécution conditionnelle dans le code Lua ;
- Passage des valeurs des variables du code Lua vers le code C++ du moteur.

Le titulaire se familiarisera au besoin avec le langage Lua et étudiera et exécutera le prototype. Il portera ensuite ce prototype vers la branche courante de développement de KANT, lequel est hébergé sur un serveur Gitlab de l'ASNR.

Nota : La branche de développement courante de KANT a peu de différences avec celle du prototype dans les zones de code concernées.

#### **4.2 Volet 2 : Extension du prototype et adaptation du Compilateur**

##### Extension du prototype

Le langage KANT décompose une question de l'ADAG en quatre zones : déclaration, zone commune, zone spécifique, zone contrôle. Actuellement, le prototype contient le code Lua qui correspond à la zone commune.

Le titulaire étendra ce code Lua aux autres zones et adaptera le code du moteur pour qu'il exécute ces zones. Un cas-test correspondant à ces développements devra être produit.

## Adaptation du compilateur pour produire le code Lua du prototype

Le titulaire adaptera le compilateur pour produire un code Lua équivalent à celui utilisé pour l'extension du prototype.

Le compilateur KANT devra être capable de produire du code Lua en plus du p-code. Pour cela, le générateur du code Lua devra être implémenté comme une fonction membre supplémentaire de la classe C++ chargée de la compilation<sup>1</sup>.

Les développements pourront se limiter au sous-ensemble du langage KANT utilisé dans le prototype. Ce sous ensemble est constitué des éléments suivants :

- Déclaration de variables locales ;
- Fonctions spéciales `ecrire()` (noms et valeurs de variables) et `reponse()` (index de la réponse à une question précédente) ;
- Boucle `pour()` ;
- Contrôle si ... `sinon_si` ... `sinon` ... `fin_si`.

Un cas-test correspondant à ces développements devra être produit.

### **4.3 Volet 3 : Extension à tout le langage KANT (hors fonctions externes)**

Le volet 3 prolonge le volet 2. Le titulaire adaptera le compilateur pour produire le code Lua couvrant toutes les instructions du langage KANT, excepté celles qui concernent l'appel à des fonctions externes.

Ces instructions sont listées ci-après :

- Structures de contrôle :
  - o `tant_que ... fin_tant_que`
  - o `selon ... cas ... default ... fin_selon`
  - o `aller_a`
- Fonctions intégrées au langage :
  - o `MIN()`, `MAX()`, `puissance()`
- Fonctions écrivant dans la console :
  - o `ecrire()` : cette fonction (traitée au lot 2) rappelle le nom de la question courante, puis le nom et la valeur de chaque variable en argument
  - o `ecrire_var_par()` : donne le nom et la valeur de chaque variable et de chaque paramètre utilisé dans la question courante
  - o `ecrire_sequence()` : écrit le chemin actuel, i.e. toutes les questions de la séquence courante, avec le numéro de la réponse suivie
- Fonctions écrivant dans un fichier :
  - o `fouvrir()`
  - o `fecrire()`
  - o `fecrire_var_par()`

---

<sup>1</sup> La classe devra recevoir une fonction membre `produire_lua()`, en plus de la fonction membre actuelle `produire_p_code()`.

- `fecrire_sequence()`
- Déclaration de fonctions internes en langage KANT :
  - `fonction`
  - `ZONE_FONCTION`
  - `retour()`

Le titulaire élaborera un cas-test utilisant toutes ces instructions.

#### 4.4 Volet 4 : Extension aux fonctions externes

Le langage KANT permet d'appeler des fonctions de bibliothèques externes respectant une API C particulière<sup>2</sup>.

Lors des travaux antérieurs, une étude de faisabilité a permis d'identifier des solutions techniques pour appeler des fonctions externes depuis le code Lua.

Le titulaire exploitera cette étude de faisabilité pour modifier le compilateur et lui faire produire le code Lua adéquat, et adaptera le code du module Moteur en conséquence.

La validation se fera par comparaison avec le comportement de la version actuelle de KANT.

#### 4.5 Volet 5 : Débogage et cas-tests sémantiques

En mode débogage, le moteur de KANT s'arrête et attend une instruction de l'utilisateur sur son entrée standard.

Le titulaire devra reproduire ce comportement, en réalisant par exemple un « hook » Lua qui appelle la fonction du moteur chargée d'attendre les instructions de l'utilisateur.

Par ailleurs, le langage Lua étant plus faiblement typé que le langage KANT, certaines vérifications de type doivent rester assurées au niveau de la compilation. Le titulaire proposera des cas-test pour vérifier cet aspect<sup>3</sup>.

La validation se fera par comparaison avec le comportement de la version actuelle de KANT.

Les prestations des volets 1 à 5 sont forfaitaires et fermes.

### 5 DOCUMENTS ET DONNEES D'ENTREE

Les documents suivants, nécessaires au déroulement de la prestation, seront mis à disposition du titulaire :

- Manuel utilisateur de KANT ;
- Code source C++ de KANT ;
- Documentation de développement de KANT ;
- Code source du prototype et rapport de l'étude de faisabilité.

---

<sup>2</sup> Les fonctions ont un prototype C double `f(...)`, et les arguments passés sont des pointeurs sur des entiers ou des « double ».

<sup>3</sup> Par exemple, une affectation de type entier = reel doit être refusée.



## **6 MODALITES D'EXECUTION**

### **6.1 Durée**

Le présent marché est prévu pour une durée ferme de 5 mois à compter de la notification.

### **6.2 Delais de réalisation**

Le soumissionnaire devra proposer dans son offre technique et commerciale les meilleurs délais de réalisation pour chaque volet (1 à 5).

### **6.3 Lieu d'exécution**

La prestation pourra être réalisée, en fonction du contexte :

- Dans les locaux de l'ASNR situés 31, avenue de la division Leclerc - 92260 Fontenay aux roses :
  - o Au titre de sa présence dans les locaux de l'ASNR, le personnel du Titulaire affecté aux prestations, objet du présent marché, sera tenu de respecter :
    - L'ensemble des dispositions législatives et réglementaires selon le code du travail fixant les prescriptions particulières d'hygiène et de sécurité applicables aux travaux effectués dans un établissement par une entreprise extérieure.
    - Les règlements intérieurs en vigueur pour l'installation, et leur annexe, la « charte relative au bon usage des systèmes d'information de l'ASNR », qui en fait partie intégrante.
- Dans les locaux du titulaire étant entendu que le Titulaire se rend, en tant que de besoin, dans les locaux de l'ASNR à Fontenay-aux-Roses.
  - o Les réunions de suivi de la prestation (lancement, clôture, point intermédiaire d'avancement dans chaque volet) pourront à ce titre être réalisées en visio ou audioconférence. Les réunions qui seront réalisées à distance par audioconférence ou visioconférence se feront avec utilisation exclusive de Microsoft Teams.

### **6.4 Organisation et suivi de la prestation**

Lors de la réunion de démarrage, le titulaire présentera son Plan Particulier d'Assurance Qualité (PPAQ) décrivant en particulier les principes de fonctionnement de la prestation (planning, présentation de l'organisation du Titulaire, identification et processus de réception, des livrables, organisation du suivi de la réalisation, proposition d'indicateurs de satisfaction, ...).

Le titulaire devra être en mesure d'assurer la continuité de la prestation en cas de départ ou d'indisponibilité de la personne chargée de sa réalisation. En cas de changement de chargé d'affaires au cours de la prestation, une période de recouvrement de 10 jours ouvrés a minima devra être garantie.

La prestation débutera par une réunion de lancement avec le responsable technique de PSN-EXP/BEPAM désigné. Elle donnera lieu à l'établissement d'un compte rendu de la part du titulaire et à la transmission du plan d'assurance qualité pour la prestation (PAQP).

Des réunions d'avancement seront organisées périodiquement pour assurer le suivi de la prestation. Elles porteront sur :

- La revue des objectifs attendus fixés à la réunion d'avancement précédente (les objectifs de la 1ère réunion d'avancement seront fixés au cours de la réunion d'enclenchement de la prestation) ;
- Le point d'avancement de chaque tâche et la conformité avec les objectifs attendus ;
- La validation de la recevabilité de chaque livrable ;
- Les éventuelles difficultés rencontrées.

D'autres réunions seront éventuellement organisées à la demande de l'ASNR ou de la société retenue, afin d'orienter au mieux le titulaire dans la réalisation des tâches. En cas de difficulté rencontrée, la réunion fera l'objet d'un compte-rendu à l'initiative du titulaire. Une réunion de clôture aura lieu en fin de prestation. Elle aura pour objectif de constater la bonne fin d'exécution des travaux qui ont été confiés au titulaire. Ce dernier remettra à cette occasion les derniers livrables de la prestation.

## **7 LIVRABLES**

### **7.1 Modalités de réception**

Tous les documents et logiciels qui font l'objet de la prestation sont la propriété de l'ASNR. La diffusion de ces documents et logiciels est intégralement sous la responsabilité de PSN-EXP/SCEPS.

Les règles et le formalisme documentaire à respecter sont ceux définis dans le programme d'assurance de la qualité de l'ASNR.

Pour ce qui concerne les livrables de la prestation, une copie dûment signée sera livrée par courrier électronique.

La réception des livrables sera réalisée suivant le principe défini ci-dessous :

- Remise d'un exemplaire indicé, validé et signé par le titulaire ;
- Analyse par l'ASNR du livrable pour vérification ;
- Version finale du livrable :
  - o Si le livrable est accepté sans remarque ou avec des remarques mineures, le fournisseur prendra en compte les éventuelles remarques et représentera le livrable en incrémentant l'indice (exemple indice 1, un seul aller-retour pour des corrections mineures donne l'indice 1.1) ;
  - o Si le livrable est refusé avec remarques, le fournisseur traitera ces remarques et soumettra, dans un délai à définir d'un commun accord entre les parties, une nouvelle version du livrable à l'indice supérieur pour approbation. Le délai ouvert au titulaire pour présenter à nouveau le livrable après ajournement ne constitue pas une prolongation du délai contractuel d'exécution des prestations.
  - o L'acceptation sans réserve restante de l'ensemble des livrables constituera leur réception.

## 7.2 Résultats attendus

Les livrables attendus de la prestation sont listés ci-après.

- Volet 1 : Prise en main de KANT et du prototype
  - o Code source du prototype porté pour la branche de développement principale ;
  - o Rapport résumant l'architecture du prototype et les principes de transcription du langage KANT en Lua.
- Volet 2 : Extension du prototype et adaptation du Compilateur
  - o Code source du moteur et du compilateur ;
  - o Cas-test de validation associé (code et fichiers entrée / sortie) ;
  - o Rapport décrivant la solution technique adoptée.
- Volet 3 : Extension à tout le langage (hors fonctions externes)
  - o Code source du compilateur ;
  - o Cas-test de validation (code et fichiers entrée / sortie) ;
  - o Rapport décrivant la solution technique adoptée.
- Volet 4 : Extension aux fonctions externes
  - o Code source du compilateur et du moteur ;
  - o Cas-tests de validation (code et fichiers entrée / sortie) ;
  - o Rapport décrivant la solution technique adoptée.
- Volet 5 : Débogage et cas-tests sémantiques
  - o Code source du compilateur et du moteur ;
  - o Cas-tests de validation (code et fichiers entrée / sortie) ;
  - o Rapport décrivant la solution technique adoptée.

## 8 **SAVOIR-FAIRE REQUIS**

Compétences techniques requises :

- Maîtrise du développement en C et C++
- Connaissance des langages de scripting (perl, bash, batch DOS, Lua est un plus)
- Expérience en théorie des langages et compilation
- Utilisation de Git pour la gestion de configuration
- Travail avec Visual Studio 2022

Expérience :

- Projets similaires réalisés avec ces technologies
- Durée d'expérience dans ces domaines

## **9 INTERLOCUTEURS ASNR**

L'interlocuteur technique et pour la présente consultation est :

- Gaëtan Bayle-Ruault des Courchamps, Tel : 01.58.35.97.53, Email : [gaetan.bayleruaultdescourchamps@asnr.fr](mailto:gaetan.bayleruaultdescourchamps@asnr.fr)
- L'interlocuteur commercial pour la présente consultation est :
- Farid AZZOUG, Tel: 07.86.91.90.78, Email: [farid.azzoug@asnr.fr](mailto:farid.azzoug@asnr.fr)

## 10 ANNEXE : SYNTAXE RESUMEE DU LANGAGE KANT

La grammaire ci-dessous est un abrégé de la grammaire formelle originale de KANT.

Le langage est sensible à la casse.

Les notations reprennent un certain nombre de conventions courantes dans les expressions régulières du style de PERL :

{ ... } : élément obligatoire  
[ ... ] : élément facultatif (noté avec espaces dans les délimiteurs [ ])  
litteral : valeur d'un élément terminal  
<element> : élément non-littéral produit par règle lexicale ou syntaxique  
<element> | <element> : choix de l'un ou l'autre.  
<element>\* : 0 ou 1 occurrence, ou occurrences successives.  
<element>+ : 1 occurrence, ou occurrences successives  
<element>? : \* ou 1 occurrence  
<element> ::= ... : définit une règle de production possible pour <element>  
[0-9][a-z][A-Z] : chiffre, minuscule, majuscule  
[^xyz] : un caractère qui n'est pas dans xyz  
[xyz] : choix entre caractères xyz

### 10.1.1 Caractères et mots réservés

# [ ] ( ) \$ + - " \ < > ,  
ZONE\_DECLARATION ZONE\_COMMUNE ZONE\_SPECIFIQUE ZONE\_CONTROLE  
FONCTION ZONE\_FONCTION retour FIN\_ZONE  
entier reel booleen faux vrai  
et ou non  
si alors sinon fin\_si  
aller\_a aller\_a\_fin\_adag stop break  
tant\_que fin\_tant\_que pour fin\_pour  
selon cas default fin\_selon  
probabilite reponse min max puissance  
lire\_fichier ecrire\_var\_par

### 10.1.2 Lignes et commentaires

- Commentaire : \$ en début de ligne, s'arrête en fin de ligne ;
- Prolongation de ligne : \
- Espaces : espaces et tabulations (en-dehors d'une chaîne) sont ignorés et comptent juste pour un séparateur.

### 10.1.3 Identificateurs

<nomsimple> ::= [a-zA-Z][a-zA-Z0-9]\*  
<nomvar> ::= <nomsimple>  
<nomparam> ::= #<nomsimple>

### 10.1.4 Constantes

<constante> ::= <entier> | <reel> | vrai | faux  
<entier> ::= [0-9]+  
<reel> ::= [0-9][.][0-9]\* |[eE][+-]?[0-9]+  
<chaîne> ::= "[^"]\|\""]"

### 10.1.5 Types de données

<type> ::= entier | reel | booleen | chaîne  
<type> ::= <type> \ [<constante>]

### 10.1.6 Déclaration

<declaration> ::= <type> <dlist>  
<dlist> ::= <dvar> [ , <dvar> ]\*  
<dvar> ::= <nom> | <nom> \ [<constante>]

### 10.1.7 Opérateurs

<opunaire> ::= + | - | non  
<opbin> ::= + | - | \* | /  
<opbin> ::= et | ou | < | > | => | =< | =

### 10.1.8 Expressions

<expresssion> ::= <constante> | <nomparam> | <nomvar>  
<expression> ::= <expression> <opbin> <expression>  
<expression> ::= <opunaire> <expression>  
<expression> ::= ( <expression> )

### 10.1.9 Bloc d'instructions

<bloc> ::= <instruction> \*  
<instruction> ::= <controle> | <procedure> | <affectation>

#### 10.1.9.1 Affectation

<affectation> ::= <variable>[<index>] = <expression>

#### 10.1.9.2 Contrôles

<controle> ::= tant\_que(<condition>) <bloc> fin\_tant\_que  
<controle> ::= pour(<nomvar>, <expression>, <expression>[, <constante>])  
fin\_pour  
<controle> ::= si (<condition>) alors <bloc>  
[sinon\_si (<condition>) alors <bloc>]\* sinon <bloc> fin\_si  
<controle> ::= selon(<expression>)  
[cas(<constante>) <bloc>]\* [default <bloc>] fin\_selon  
<controle> ::= aller\_a <nom> | aller\_a\_fin\_adag | stop | break

#### 10.1.9.3 Entrée/sortie:

\$ Valeurs lues vers une variable tableau d'entiers ou de réels.  
lire\_fichier(<variable>, <nom\_fichier>, <taille>)  
ecrire\_sequence()  
fecrire\_sequence(<nom>)  
ecrire\_var\_par()  
fecrire\_var\_par(<nom>)

### 10.1.10 Questions

<question> ::=  
[ZONE\_DECLARATION <declaration>\* FIN\_ZONE]  
[ZONE\_COMMUNE <instruction>\* FIN\_ZONE]  
[ZONE\_SPECIFIQUE <instruction>\* FIN\_ZONE]  
[ZONE\_CONTROLE <instruction>\* FIN\_ZONE]

### 10.1.11 Fonction interne

<fonction\_i> ::=  
FONCTION <type> <nom>(<argument>[, <argument>]\*)  
ZONE\_DECLARATION <declaration>\* FIN\_ZONE  
ZONE\_FONCTION <bloc> retour (<variable>) FIN\_ZONE

### 10.1.12 - Opérateurs du langage

- Opérateurs multiplicatifs : \* /
- Opérateurs additifs : + -
- Opérateurs relationnels = < > <= >=
- Opérateurs logiques : et ou non

### 10.1.13 Fonctions du langage

- min(x,y)
- max(x,y)
- puissance(x,y) : renvoie  $x^y$

- reponse(<nomsimple>) : numéro de réponse actuellement exploré pour la question <nomsimple>

#### 10.1.14 Variables spéciales

- probabillite[] : tableau de réels, avec une entrée par réponse possible à la question
- reponse : numéro de réponse actuellement exploré pour la question actuelle.