var *NomDeVariable***:** *Type***;**

Type est à prendre pour l'instant dans l'ensemble {**entier**,**car**,**booléen**,**réél**}

Définition 2.3. Les **Expressions** sont constituées à l'aide de variables déjà déclarées, de valeurs, de parenthèses et d'opérateurs du (des)type(s) des variables concernées.

Définition 2.4. L'**affectation** est l'instruction qui permet de stocker une valeur dans une variable.

On écrit

*NomDeVariable***=***ExressionDuTypeDeLaVariable***;**

Toute variable doit être déclarée et recevoir une valeur initiale.

---

2. Types de base

Booléens

Une variable de type **booléen** prend comme valeur **VRAI** ou **FAUX**. Les opérations usuelles sont ET, OU et NON qui sont données dans les tables qui suivent.

Entiers

Une variable de type **entier** peut prendre comme valeur l'ensemble des nombres entiers signés. Les opérations associées sont les opérations usuelles +,-,*,/.

Réels

Une variable de type **réél** peut prendre comme valeur l'ensemble des nombres réels. Les opérations associées sont les opérations usuelles +,-,*,/.

Caractères

Une variable de type **car** peut prendre comme valeur l'ensemble des caractères imprimables. On notera les valeurs entre guillemets. On considère souvent que les caractères sont ordonnés dans l'ordre alphabétique.

Attention

Les valeurs

"1" qui est un caractère,

1 qui est un entier,

1. qui est un réel sont différentes et ne seront pas codés de la même manière dans la mémoire de la machine.

Comparaison

Les opérateurs **<, ?, ==, !=, >, ?** permettent de comparer les valeurs de type entier, réel et caractère. Le résultat de cette comparaison est une valeur booléenne.

---

3. Structures de contrôle

Il y a trois structures principale de contrôle qui permettent de construire des algorithmes Bloc d'instruction

début *instruction1* *instruction2* . **fin**

Alternative

Alternative simple (traduction Python):

si *ExpressionBooléenne* **alors** *BlocInstruction1* **sinon**

BlocInstruction2 **finsi;**

Alternative multiple (traduction Python):

selon que **cas** *cas1* **:** *BlocInstruction1* **cas** *cas2* **:** *BlocInstruction2*

.

autrement : *BlocInstruction* **finselonque** Répétition

L'instruction **exit** permet d'arrêter la répétition.

le bloc d'instruction peut ne pas être éxécuté (traduction Python):

tant que*ExpressionBooléenne* **faire** *BlocInstruction* **fintantque;**

le bloc d'instruction peut ne pas être exécuté et il y a une variable indicatrice (traduction Python):

pour *VariableIndicatrice* **allant de** *ValeurInitiale* **à** *ValeurFinale* **par pas de** *ValeurPas* **faire** *BlocInstruction* **finpour;**

le bloc d'instruction est exécuté au moins une fois (ne se traduit pas directement en Python)

répéter

BlocInstruction

jusqu'à *ExpressionBooléenne* **finrépéter;**

---

4. Fonctions

Une fonction est une section d'algorithme qui a un objectif bien défini et un nom. En général, elle communique avec l'extérieur par le biais de *paramètres* typés. Elle possède des variables locales qui ne sont pas visibles à l'extérieur de la fonction. Ces variables peuvent être des fonctions. Une fonction retourne une valeur par l'instruction simple **retourne(***Expression***)**.