

```
var NomDeVariable: Type;
```

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

**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=ExpressionDuTypeDeLaVariable;*

**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  $+$ ,  $-$ ,  $*$ ,  $/$ .

## Rééls

Une variable de type **réel** 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

début	instruction1	instruction2	fin
0	1	2	3
1	2	3	4
2	3	4	5
3	4	5	6
4	5	6	7
5	6	7	8
6	7	8	9
7	8	9	10
8	9	10	11
9	10	11	12
10	11	12	13
11	12	13	14
12	13	14	15
13	14	15	16
14	15	16	17
15	16	17	18
16	17	18	19
17	18	19	20
18	19	20	21
19	20	21	22
20	21	22	23
21	22	23	24
22	23	24	25
23	24	25	26
24	25	26	27
25	26	27	28
26	27	28	29
27	28	29	30
28	29	30	31
29	30	31	32
30	31	32	33
31	32	33	34
32	33	34	35
33	34	35	36
34	35	36	37
35	36	37	38
36	37	38	39
37	38	39	40
38	39	40	41
39	40	41	42
40	41	42	43
41	42	43	44
42	43	44	45
43	44	45	46
44	45	46	47
45	46	47	48
46	47	48	49
47	48	49	50
48	49	50	51
49	50	51	52
50	51	52	53
51	52	53	54
52	53	54	55
53	54	55	56
54	55	56	57
55	56	57	58
56	57	58	59
57	58	59	60
58	59	60	61
59	60	61	62
60	61	62	63
61	62	63	64
62	63	64	65
63	64	65	66
64	65	66	67
65	66	67	68
66	67	68	69
67	68	69	70
68	69	70	71
69	70	71	72
70	71	72	73
71	72	73	74
72	73	74	75
73	74	75	76
74	75	76	77
75	76	77	78
76	77	78	79
77	78	79	80
78	79	80	81
79	80	81	82
80	81	82	83
81	82	83	84
82	83	84	85
83	84	85	86
84	85	86	87
85	86	87	88
86	87	88	89
87	88	89	90
88	89	90	91
89	90	91	92
90	91	92	93
91	92	93	94
92	93	94	95
93	94	95	96
94	95	96	97
95	96	97	98
96	97	98	99
97	98	99	100
98	99	100	101
99	100	101	102
100	101	102	103
101	102	103	104
102	103	104	105
103	104	105	106

Alternative

Alternative simple (traduction Python):

```
si ExpressionBooléenne alors   BlocInstruction1   sinon
    BlocInstruction2   fin;
```

Alternative multiple (traduction Python):

selon que      cas cas1 : BlocInstruction1      cas cas2 : BlocInstruction2

autrement : *BlocInstruction*      fin**selonque** Répétition

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

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

**tant que***Expression Booléenne* **faire** *BlocInstruction* **fin tant que;**

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 Expression Booléenne fin répéter:**

## 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*).