

# Algorithme #1

Michael  
X  NATIS

# Blocs Boucles Conditions





Compétence demandée :  
Maîtriser les 5 concepts de la  
construction algorithmique

1. Variables
2. Instructions de base
3. Blocs
4. Conditions
5. Boucles

1. Variables
2. Instructions de base
3. Blocs
4. Conditions
5. Boucles





# 1. Variables



Les **variables** sont **typées** !

Les **variables** sont **TYPÉES** !

Les **variables** sont **TYPÉES** !



Type = Structure de données

Type = Structure de données





Les **types** permettent à  
l'ordinateur **d'identifier les**  
**actions possibles**

Les **types** permettent à  
l'ordinateur **d'identifier les**  
**actions possibles**

Les **types** prennent un  
**espace** différent en RAM



Les **types** permettent à  
l'ordinateur d'identifier les  
actions possibles

Les **types** prennent un  
**espace** différent en RAM



## 2. Instructions de base

# Affectation



```
taille <- 34
```

```
taille <- 34
```

```
toto <- [23, 34, 32, 3]
```

```
taille <- 34
```

```
toto <- [23, 34, 32, 3]
```

```
resultat <- 'Petit'
```



```
taille <- 34
```

```
toto <- [23, 34, 32, 3]
```

```
resultat <- 'Petit'
```

```
yop <- Vrai
```

Structure de données	Actions possibles
Nombre	Addition Soustraction Division Multiplication
Chaîne de caractères	Concaténation
Boolean	Et Ou Non
Tableau	Adressage (position) Ajout Suppression
Dictionnaire	Adressage (clé) Ajout Suppression



Structure de données	Actions possibles
Nombre	Addition Soustraction Division Multiplication
Chaîne de caractères	Concaténation
Boolean	Et Ou Non
Tableau	Adressage (position)
Dictionnaire	Adressage (clé) Ajout Suppression





## 3. Blocs

```
@DebutBloc  
resultat <- 'Grand'  
taille <- 34  
@FinBloc
```

Un **bloc** permet de  
**rassembler** des  
instructions



Les variables définies  
dans un bloc meurent à  
la fin du bloc

Les variables définies  
dans un bloc meurent à  
la fin du bloc

Portée (scope)



Un bloc est un ensemble  
d'instructions qui  
peuvent être  
conditionnés ou répétés

Un bloc est un ensemble  
d'instructions qui  
peuvent être  
conditionnés ou répétés





## 4. Conditions

Une condition permet de  
conditionner l'exécution  
d'un bloc

```
taille <- 34

resultat <- 'Petit'
@Si taille >= 50
  @DebutBloc
  resultat <- 'Grand'
  @FinBloc
```



```
taille <- 34

resultat <- 'Petit'
@Si taille >= 50
  @DebutBloc
  resultat <- 'Grand'
  @FinBloc
```

```
taille <- 34

resultat <- 'Petit'
@Si @Non (taille < 50)
  @DebutBloc
  resultat <- 'Grand'
  @FinBloc
```

Une condition se base  
sur 1 ou plusieurs  
prédicats

Une condition se base  
sur 1 ou plusieurs  
prédicats



La valeur logique d'un  
prédictat est toujours  
« Vrai » ou « Faux »

```
taille <- 34  
forme <- 'Rectangle'  
  
resultat <- 'Petit'  
@Si taille >= 50 @Et forme = 'Rectangle'  
  @DebutBloc  
  resultat <- 'Grand'  
  @FinBloc
```

```
taille <- 34  
forme <- 'Rectangle'  
  
resultat <- 'Petit'  
@Si taille >= 50 @Ou forme = 'Rectangle'  
  @DebutBloc  
  resultat <- 'Grand'  
  @FinBloc
```

```
taille <- 34
forme <- 'Rectangle'

@Si taille >= 50 @Ou forme = 'Rectangle'
  @DebutBloc
  resultat <- 'Grand'
  @FinBloc
@Sinon
  @DebutBloc
  resultat <- 'Petit'
  @FinBloc
```

## Opérateurs binaires sur les prédicats

**ET** = « et en même temps ... »

**OU** = « ou soit ... »



# Opérateurs binaires

a ET b

c OU d

## Opérateurs unaires sur les prédicats

**NON** = « ne pas ... » ou  
« contraire »

# Opérateurs unaires sur les prédicats

NON (a)

# Table de vérité

Les tables de vérité présentent tous les résultats possibles d'une opération logique



a	b	a ET b
Faux	Faux	Faux
Faux	Vrai	Faux
Vrai	Faux	Faux
Vrai	Vrai	Vrai

a	b	a OU b
Faux	Faux	Faux
Faux	Vrai	Vrai
Vrai	Faux	Vrai
Vrai	Vrai	Vrai



# Loi De Morgan

La loi De Morgan permet de « **casser** » un **NON**  
englobant un **ET** ou un **OU**

# Loi De Morgan

$$\begin{aligned}\text{NON (a ET b)} &= \text{NON (a) OU NON (b)} \\ \text{NON (a OU b)} &= \text{NON (a) ET NON (b)}\end{aligned}$$











# 5. Boucles

# Les boucles

Les boucles permettent de **répéter un bloc d'instructions**

Il y a 3 types de boucles pour  
répéter un bloc

1. @PourChaque
2. @Pour @De @A
3. @TantQue ou Boucle + @Stop

# 1. Il faut s'arrêter à la fin du tableau

```
tab <- [23, 43, 32, 4, 3]  
  
@PourChaque element @Dans tab  
  @DebutBloc  
  Afficher element  
  @FinBloc
```

## 2. Il faut s'arrêter avec un nombre maximal

```
tab <- [23, 43, 32, 4, 3]
```

```
@Pour i @De 0 @A 4  
  @DebutBloc  
  Afficher tab[i]  
  @FinBloc
```

## 2. Il faut s'arrêter avec un nombre maximal



```
tab <- [23, 43, 32, 4, 3]
```

```
@Pour i @De 0 @A 4  
  @DebutBloc  
  Afficher tab[i]  
  @FinBloc
```



### 3. Il faut s'arrêter avec une condition


```
tab <- [23, 43, 32, 4, 3]

position <- 0
@TantQue tab[position] < 30
  @DebutBloc
    position <- position + 1
  @FinBloc
Afficher position
```

```
tab <- [23, 43, 32, 4, 3]

position <- 0
@PourChaque element @Dans tab
  @DebutBloc
    @Si element >= 30
      @DebutBloc
        Afficher position
      @Stop
    @FinBloc
  position <- position + 1
@FinBloc
```

### 3. Il faut s'arrêter avec une condition



```
tab <- [23, 43, 32, 4, 3]

position <- 0
@TantQue tab[position] < 30
  @DebutBloc
    position <- position + 1
  @FinBloc
Afficher position
```

```
tab <- [23, 43, 32, 4, 3]

position <- 0
@PourChaque element @Dans tab
  @DebutBloc
    @Si element >= 30
      @DebutBloc
        Afficher position
      @Stop
    @FinBloc
  position <- position + 1
@FinBloc
```







