

# PHP de base

## Table des matières

<b>Définition.....</b>	<b>4</b>
<b>Le pseudo-code .....</b>	<b>6</b>
<b>Chapitre 1 : L’algorithmique : les bases .....</b>	<b>7</b>
Les variables .....	7
Le typage des variables .....	7
Déclarer une variable .....	7
Les constantes .....	8
Bonnes pratiques.....	8
Affectation.....	8
Exercice 1.1.....	8
<i>Consigne</i> .....	8
Opérations entrées/sorties .....	9
<i>Echo</i> .....	9
\$_POST['name'].....	9
Exercice 1.2.....	9
<i>Consigne</i> .....	9
<i>Aperçu</i> .....	10
<b>Chapitre 2 : Les opérateurs .....</b>	<b>11</b>
Les opérations arithmétiques.....	11
Les opérateurs arithmétiques en PHP.....	11
Exercice 2.1.....	11
Exercice 2.2.....	11
<i>Consigne</i> .....	11
<i>Correctif</i> .....	11
Exercice 2.3.....	11
<i>Consigne</i> .....	11
Exercice 2.4.....	11
<i>Consigne</i> .....	11
<b>Chapitre 3 : Les opérateurs conditionnels.....</b>	<b>12</b>
Les booléens .....	12
Comparer des variables.....	12
<i>PHP</i> .....	12

Combiner des comparaisons .....	13
Les tables de vérité : ET – OU – NON .....	13
Exercice 3.1.....	13
<i>Consigne</i> .....	13
<b>Chapitre 4 : Les structures conditionnelles .....</b>	<b>14</b>
SI... SINON.....	14
Beaucoup de conditions.....	15
L'indentation .....	15
SI... SINON SI... SINON .....	16
SELON QUE .....	17
Exercice 4.1.....	18
<i>Consigne</i> .....	18
Exercice 4.2.....	18
<i>Consigne</i> .....	18
Exercice 4.3.....	18
<i>Consigne</i> .....	18
Exercice 4.4.....	18
<i>Consigne</i> .....	18
Exercice 4.5.....	18
<i>Consigne</i> .....	18
<b>Chapitre 5 : Les structures itératives .....</b>	<b>19</b>
Les boucles .....	19
Les types de boucles.....	19
TANT QUE ... FAIRE .....	20
Exercice 5.1.....	20
<i>Consigne</i> .....	20
Exercice 5.2.....	20
<i>Consigne</i> .....	20
Exercice 5.3.....	20
<i>Consigne</i> .....	20
Exercice 5.4.....	20
<i>Consigne</i> .....	20
FAIRE... TANT QUE .....	21
POUR.....	21
<b>Chapitre 6 : Les tableaux.....</b>	<b>22</b>
Utilité.....	22

<i>Déclarer un tableau</i> .....	22
<i>Affectation dans un tableau</i> .....	22
<i>Trucs et astuces</i> .....	23
Exercice 6.1.....	23
<i>Consigne</i> .....	23
Exercice 6.2.....	23
<i>Consigne</i> .....	23
Exercice 6.3.....	24
Consigne .....	24
Exercice 6.4.....	24
<i>Consigne</i> .....	24
Exercice 6.5.....	24
<i>Consigne</i> .....	24
Exercice 6.6.....	24
<i>Consigne</i> .....	24
Exercice 6.7.....	24
<i>Consigne</i> .....	24
Exercice 6.8.....	24
<i>Consigne</i> .....	24
Exercice 6.9.....	24
<i>Consigne</i> .....	24
Exercice 6.10.....	24
<i>Consigne</i> .....	24
Exercice 6.11.....	25
<i>Consigne</i> .....	25
Exercice 6.12.....	25
<i>Consigne</i> .....	25
Projet 1 : Le tableau .....	25
Projet 2 : Le calendrier ordonné.....	25
Projet 3 : Le calendrier jour .....	26
Projet 4 : Les températures .....	26

## Définition d'un algorithme

Ensemble de règles opératoires dont l'application permet de **résoudre un problème** énoncé au moyen d'un **nombre fini d'opérations**. Un algorithme peut être traduit, grâce à un langage de programmation, en un programme exécutable par un ordinateur.

Source : Dictionnaire Larousse.

### Exemple :

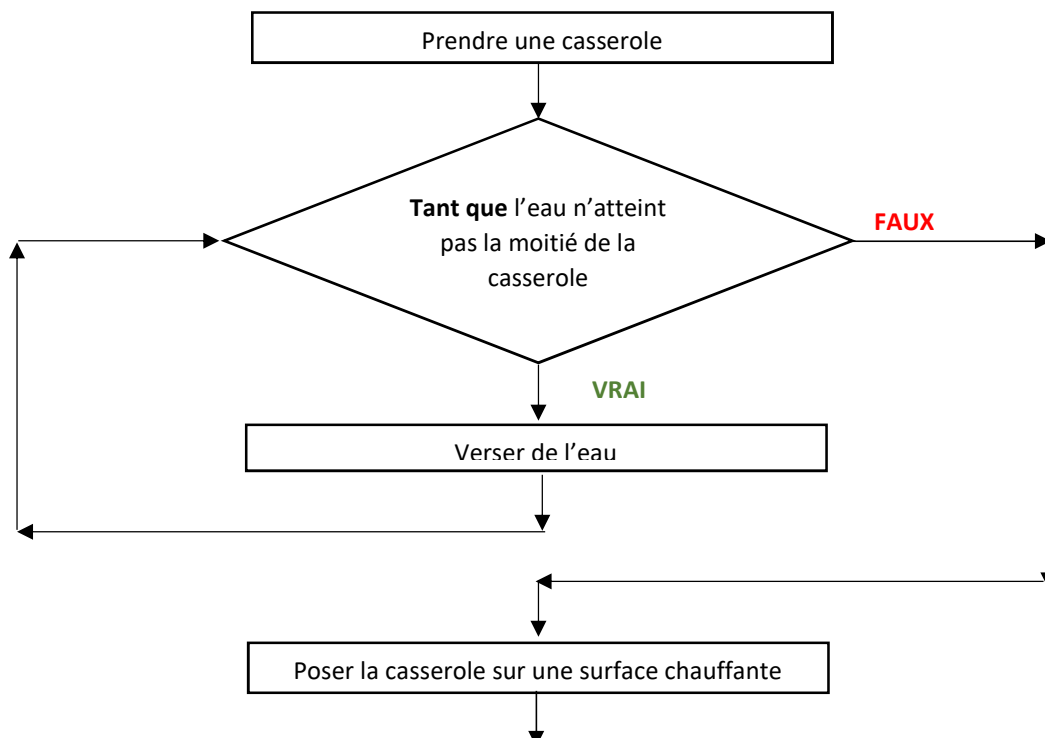
Pour faire cuire du riz :

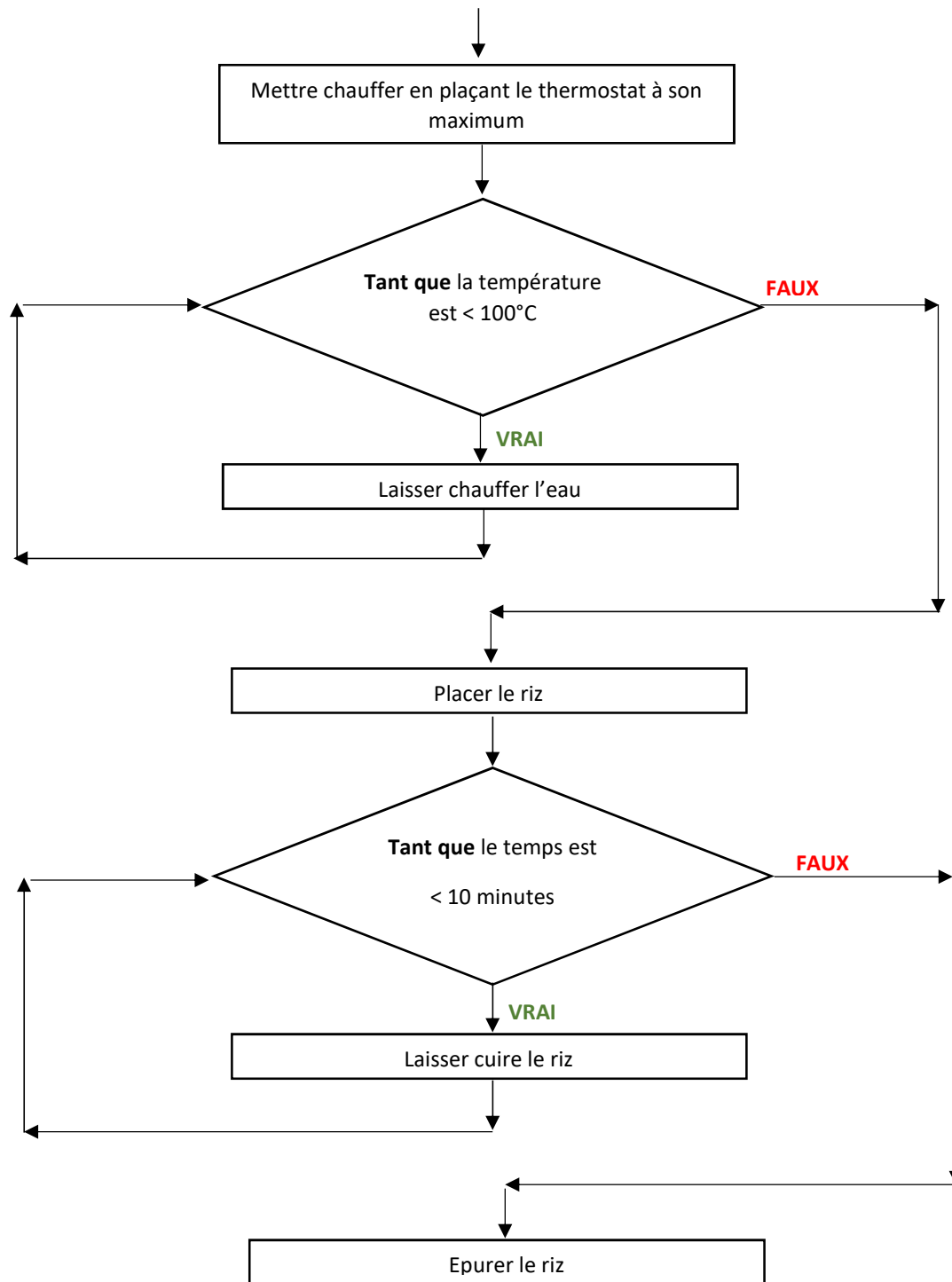
1. Prendre une casserole
2. Mettre de l'eau dedans
3. Poser la casserole sur une surface chauffante
4. La mettre chauffer
5. Mettre le riz
6. Laisser cuire
7. Epurer le riz

Ce sont les grandes étapes. Mais on pourrait encore expliquer avec plus de détails les différentes étapes.

1. Prendre une casserole
2. Mettre de l'eau dedans jusqu'à la moitié de sa capacité
3. Poser la casserole sur une surface chauffante
4. Mettre chauffer en plaçant le thermostat à son maximum
5. Lorsque la température atteint 100°C, placer le riz
6. Laisser cuire pendant 10 minutes
7. Epurer le riz.

On pourrait représenter cet algorithme comme ceci :





L'ordinateur fonctionne de cette manière, il faudra lui dire les tâches précises qu'il devra exécuter.

## Le pseudo-code

Pour cela, nous allons écrire du pseudo-code comme ceci :

Algorithme CuireRiz

Variable casserole, temps, temperature, sachetRiz : entier

Debut

Prendre(casserole)

TANT QUE (casserole.capacite < casserole.moitie) FAIRE

Remplir(casserole)

FINTQ

MettreChauffer(casserole)

temperature ← 15

TANT QUE (temperature < 100) FAIRE

temperature ← temperature + 1

FINTQ

Placer(sachetRiz, casserole)

temps ← 0

TANT QUE (temps < 10) FAIRE

temps ← temps + 1

FINTQ

Retirer(sachetRiz, casserole)

Fin

### Un algorithme est divisé en 3 parties :

1. Le **nom** de l'algorithme
2. La partie déclarative : on y indique toutes les **variables** dont on aura besoin.
3. Les **instructions** de l'algorithme : délimitées par les mots-clés « Debut » et « Fin »

## Chapitre 1 : L'algorithmique : les bases

### Les variables

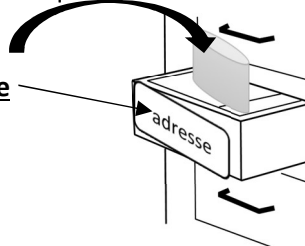
Un ordinateur doit retenir des informations temporaires pour exécuter des tâches.

Une variable peut être comparée à un tiroir avec une étiquette.

- L'étiquette permet de lui donner un nom
- L'intérieur du tiroir permet d'y ranger l'information

Ces variables se trouvent dans la RAM. Cette dernière est comme une immense armoire à tiroirs. En terme informatique :

- le contenu du tiroir est une **valeur**
- l'étiquette du tiroir est une **adresse**



Une variable est donc un élément de programmation auquel on **affecte** une valeur particulière pouvant être modifiée au cours de l'algorithme.

Une variable est définie par :

- Un **nom** : un identifiant **unique** permettant de la désigner
- Un **type** : définit quel type de valeur la variable peut contenir
- Un **emplacement** en mémoire RAM, représenté par son adresse
- Une **valeur** : correspondant au type

### Le typage des variables

Comme expliqué précédemment, une variable est entre autres définie par le type de valeur qu'elle peut contenir. Il existe de nombreux types :

<b>Entier</b>	1, 2, 5, 6, 42 ...
<b>Réel</b>	0.14, 157.1574, 3.1415 ...
<b>Caractère</b>	'a', 'B', '^' ...
<b>Texte</b>	'Bonjour', 'Je suis de type texte' ...
<b>Logique ou booléen</b>	Vrai ou faux (cf structures conditionnelles)

### Déclarer une variable

Déclarer une variable consiste à **réserver un emplacement en mémoire** afin de pouvoir l'utiliser par la suite. Pour déclarer une variable, il nous faut spécifier 2 choses :

- son *nom*
- son **type**

Il est important de choisir un nom cohérent qui permette de facilement comprendre l'utilité de la variable.

<b>En PHP : type dynamique</b>
\$age;
\$message ;

En PHP, toutes les variables commencent pas un dollar (\$).

## Les constantes

Dans nos algorithmes, nous avons parfois besoin de valeurs qui ne changent pas au cours de l'exécution.

Exemple : la valeur du nombre  $\pi$  (pi) ou encore le pourcentage de TVA.

Ces valeurs sont appelées des constantes, car elles **ne changent pas** durant l'exécution de l'algorithme.

Voici comment déclarer une constante :

En PHP
<pre>const PI = 3.141592 ; const TAUX TVA = 0.21 ;</pre>

## Bonnes pratiques

Le choix des **noms** de variable doit rester cohérent. À la relecture de l'algorithme, il doit être possible de **comprendre le sens** d'une variable en fonction de son nom. L'alphabet n'est pas une bonne idée.

**Eviter les accents** pour des raisons informatiques. Les langages sont écrits en anglais et cette langue ne comprend pas les accents.

Utiliser le « lowerCamelCase » ou le « UpperCamelCase ». Le choix dépend des conventions que certains langages utilisent. En PHP, c'est le « **lowerCamelCase** » qui est principalement utilisé.

Pour les constantes, c'est « UPPERCASE ».

## Affectation

Une fois une variable déclarée, il faut pouvoir lui **donner une valeur**. Cette opération s'appelle une affectation.

La première affectation est appelée initialisation. Il s'agit d'une opération importante : on ne peut pas utiliser une variable ne possédant pas de valeur.

En PHP
<pre>\$message = "Bonjour" ;</pre>

L'initialisation est **primordiale** :

```
$b ← $a + 2;
```

Quelle est la valeur de \$a ?

## Exercice 1.1

### Consigne

Trouvez une méthode permettant d'inverser le contenu de deux variables (du même type évidemment). Si \$a = 5 et \$b = 7, après ce traitement \$a = 7 et \$b = 5.



## Opérations entrées/sorties

Il est très régulièrement nécessaire d'interagir avec un algorithme. 2 opérations sont mises à notre disposition :

- `$_POST['name']/$_GET['url']` : opération d'entrée
- `echo` : opération de sortie

Attention !

Ces deux opérations doivent être comprises du point de vue de l'ordinateur. L'opération `echo` va donc écrire une information (que nous pourrons lire sur l'écran par exemple) et l'opération `$_POST` va enregistrer une valeur (écrite sur le clavier par exemple).

### Echo

Opération permettant de communiquer une information au « monde extérieur ». C'est l'écran de l'ordinateur

Déroulement de cette opération :

1. Lecture de la valeur de la variable spécifiée
2. Communication de cette valeur au monde extérieur

PHP
<pre>\$message = "Bonne journée" ; echo \$message ;</pre>

### `$_POST['name']`

Opération permettant de récupérer de l'information du « monde extérieur ». Ce qui peut être :

- Saisie au clavier
- informations récupérées d'un formulaire

Déroulement de cette opération :

1. Lecture de la valeur depuis la source de l'information
2. Affectation de cette valeur à une variable donnée

PHP
<pre>\$message = \$_POST("message") ;</pre>

En web, la seule manière d'interagir avec l'utilisateur, c'est via un formulaire. On récupère via la méthode `$_GET(l'url)` ou `$_POST(le nom du champ du formulaire)`.

## Exercice 1.2

### Consigne

Depuis un formulaire de contact , récupérez la valeur du champ « Nom » et « Prénom », et afficher une page qui indique « Message envoyé » suivi de « Merci » et les nom et prénom de la personne.

## Aperçu

# Formulaire de contact

Vos coordonnées

Nom :

Prénom :

Votre message a bien été envoyé ! Merci Luke Lucky

## Chapitre 2 : Les opérateurs

### Les opérations arithmétiques

Voici les opérations possibles sur des valeurs de types numériques (entiers ou réels) :

#### Les opérateurs arithmétiques en PHP

Symbole	Nom	Exemple d'opération	Résultat
+	Addition	2 + 3	5
-	Soustraction	3 - 1	2
/	Division	7 / 2	3.5
%	Modulo (reste d'une division d'entiers)	4 % 2 7 % 2	0 1
**	Exponentielle \$a ** \$b élévation de \$a à la puissance \$b	3 ** 2 (= 3 <sup>2</sup> )	9

/!\ Il n'y a pas de division entière en PHP.

#### Exercice 2.1

Dans cet exemple, nous utilisons des opérations entre des valeurs, mais également des variables !

```
$a = 10 ;
$b = 5 / 2 ;
$c = $b + $a ;
$d = 5 % 2 ;
$e = ($b - 0.5) * $a ;
```

#### Donnez les valeurs de :

Variables	Valeurs
\$a	
\$b	
\$c	
\$d	
\$e	

#### Exercice 2.2

##### Consigne

Vérifiez la valeur en exécutant le code dans PHP.

#### Exercice 2.3

##### Consigne

Déterminez la valeur des variables de ce code.

```
$a = 8 % 3 ;
$b = 4 + $a ;
$c = $b * a ;
$d = ($c - $a) * $b ;
$e = (($a + 7) * ($d / $a)) * 0 ;
```

#### Correctif

Variables	Valeurs
\$a	
\$b	
\$c	
\$d	
\$e	

#### Exercice 2.4

##### Consigne

Vérifiez la valeur en exécutant le code dans PHP.

## Chapitre 3 : Les opérateurs conditionnels

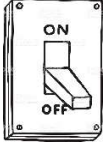

### Les booléens

Ce type de variable n'a que deux valeurs possibles :

- vrai : true
- faux : false

Ce type permet de connaître un état.

Exemples :

	
Considérons un interrupteur allumé ou éteint : Nous utilisons la variable \$interrupteurEteint : <ul style="list-style-type: none"> <li>- Vrai : l'interrupteur est éteint ;</li> <li>- Faux : l'interrupteur est allumé.</li> </ul>	Un panier est vide ou rempli : Nous utilisons la variable : \$panierRempli : <ul style="list-style-type: none"> <li>- Vrai : le panier est rempli</li> <li>- Faux : le panier est vide</li> </ul>

### Comparer des variables

Dans un algorithme, il est souvent intéressant de pouvoir comparer des variables.

Par exemple, pour savoir si un étudiant a réussi un examen, il faut savoir si des points sont supérieurs ou égaux à 10 (sur un total de 20).

Les opérateurs de comparaisons sont là pour nous aider.

=	égalité stricte	== (égal) === (égal et de même type)
≠	différence	!= ou <> !== (différent ou pas de même type)
>	strictement supérieur	>
<	strictement inférieur	<
≥	supérieur ou égal	>=
≤	inférieur ou égal	<=

Le résultat d'une comparaison sera un booléen.

### PHP

Algorithme de comparaison de points entre Jean et Paul pour savoir s'ils ont réussi.

```
$pointsJean = 15 ;
$pointsPaul = 9 ;
$reussiteJean = ($pointsJean >= 10) ;
$reussitePaul = ($pointsPaul >= 10) ;
```

#### Qui a réussi ?

Variables	Valeurs
\$reussiteJean	true
\$reussitePaul	false

## Combiner des comparaisons

Exemple :

Nous désirons démarrer la chaudière lorsque la température est comprise entre 5 et 15 degrés. En considérant la variable température, deux comparaisons apparaissent :

- (température ≤ 15)

- (température ≥ 5)

Pour éteindre la chaudière : (température ≥ 5) et (température ≤ 15) doivent être vrais. Il serait donc intéressant de pouvoir combiner ses deux comparaisons. Les opérateurs logiques vont nous permettre de le faire.

### En PHP

!	NON	Négation logique
&& ou AND	ET	Le 'et' logique
ou OR	OU	Le 'ou' inclusif logique

### Les tables de vérité : ET – OU – NON

Une table de vérité permet de modéliser le comportement d'un opérateur logique en fonction des valeurs des deux opérandes.

Voici les tables de vérité des trois opérateurs :

&&						!	
\$a && \$b	\$a = T	\$a = F	\$a    \$b	\$a = T	\$a = F		!\$a
\$b = T	T	F	\$b = T	T	T	TRUE	F
\$b = F	F	F	\$b = F	T	F	FALSE	T

### Exercice 3.1

#### Consigne

Donnez le résultat pour chacune de ces instructions :

**Exercice 1 :** Considérons \$a = 3, \$b = 9, \$c = false, \$d = !(\$c), \$e = 9.

N°	Exercice	Résultat
1	(\$a > 8)	
2	(\$b == 9)	
3	!(\$a != 3)	
4	!(\$c)	
5	(\$a < \$b)    \$c	
6	(((\$a + \$b) != 12)	
7	(((\$b == 5)    (((\$e > 10) && (\$a < 8))))	
8	(((\$b == 5)    (((\$e > 10) && (\$a < 8)))    (\$a < \$b)    \$c) && \$c	

**Exercice 2 :** Considérons \$a = 3, \$b = 9, \$c = FAUX, \$d = !(\$c), \$e = 9.

N°	Exercice	Résultat
1	\$a != 3	
2	!(\$d)    \$c	
3	(((\$a + \$b) == 12) && \$d	

## Chapitre 4 : Les structures conditionnelles

Souvent, un algorithme se retrouve face à plusieurs situations qui ne peuvent pas être traitées avec les mêmes instructions. Nous ne savons pas quel sera le cas de figure à l'exécution.

Nous devons alors prévoir tous les cas possibles. Nous allons donc utiliser les structures conditionnelles.

### SI... SINON

Reprenons l'exemple de la chaudière :

Nous désirons l'allumer si la température est comprise entre 5 et 15 degrés. Pour cela, nous avons besoin de dire :

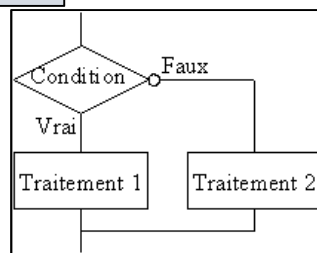
**SI** la température est comprise entre 5 et 15 degrés **ALORS** allume la chaudière

**SINON** éteint la chaudière.

Cette phrase illustre une structure conditionnelle.

#### En pseudo-code :

```
SI [condition] ALORS
    [instructions1]
SINON
    [instructions2]
FINSI
```



#### Exemple :

```
Variable age : Entier,
Debut
    Lire(age)
    SI age ≥ 18 ALORS
        Ecrire("Bienvenue")
    SINON
        Ecrire("Accès interdit")
    FINSI
Fin
```

#### En PHP :

```
if (conditions)
{
    instructions;
}
else
{
    instructions;
}
```

#### Exemple :

```
$age = $_POST["age"];
if ($age >= 18)
{
    echo "Bienvenue";
}
else
{
    echo "Accès interdit";
}
```

#### Instant de réflexion :



« Peut-on placer un SI... SINON dans un autre ? ».

## Beaucoup de conditions

Lorsque de nombreux traitements sont possibles en fonction de la valeur d'une variable, nous pouvons imbriquer des SI...SINON, mais cela devient rapidement illisible et laborieux à écrire.

Voyons ça avec un petit algorithme déterminant le jour de la semaine.

```
Variable jour : 1.. 7
Debut
  Lire(jour)
  Si jour = 1 Alors Ecrire("Lundi")
  Sinon
    Si jour = 2 Alors Ecrire("Mardi")
    Sinon
      Si jour = 3 Alors Ecrire("Mercredi")
      Sinon
        Si jour = 4 Alors Ecrire("Jeudi")
        Sinon
          Si jour = 5 Alors Ecrire("Vendredi")
          Sinon
            Si jour = 6 Alors Ecrire("Samedi")
            Sinon
              Ecrire("Dimanche")
            FinSi
          FinSi
        FinSi
      FinSi
    FinSi
  FinSi
Fin
```



## L'indentation

Nous commençons à voir apparaître, dans le pseudo-code, des blocs d'instructions. Afin de garder le code lisible, nous utilisons des espaces (ou des tabulations) pour structurer le code en fonction de sa logique. Ce concept se nomme l'indentation.

### Exemple :

```
Variable jour : 1.. 7
Debut
  Lire(jour)
  Si jour = 1 Alors Ecrire("Lundi")
  Sinon
    Si jour = 2 Alors Ecrire("Mardi")
    Sinon
      Si jour = 3 Alors Ecrire("Mercredi")
      Sinon
        Si jour = 4 Alors Ecrire("Jeudi")
        Sinon
          Si jour = 5 Alors Ecrire("Vendredi")
          Sinon
            Si jour = 6 Alors Ecrire("Samedi")
            Sinon
              Ecrire("Dimanche")
            FinSi
          FinSi
        FinSi
      FinSi
    FinSi
  FinSi
Fin
```



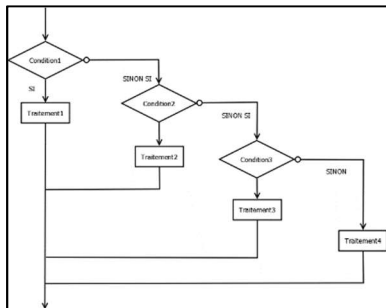
```
Variable jour : 1.. 7
Debut
  Lire(jour)
  Si jour = 1 Alors Ecrire("Lundi")
  Sinon
    Si jour = 2 Alors Ecrire("Mardi")
    Sinon
      Si jour = 3 Alors Ecrire("Mercredi")
      Sinon
        Si jour = 4 Alors Ecrire("Jeudi")
        Sinon
          Si jour = 5 Alors Ecrire("Vendredi")
          Sinon
            Si jour = 6 Alors Ecrire("Samedi")
            Sinon
              Ecrire("Dimanche")
            FinSi
          FinSi
        FinSi
      FinSi
    FinSi
  FinSi
Fin
```

## SI... SINON SI... SINON

Lorsque vous avons plus de deux conditions à vérifier, nous pouvons utiliser le SINON SI dans la structure conditionnelle que nous avons vue précédemment.

### En pseudo code :

```
SI condition ALORS
    instructions1
SINON SI condition2
    instructions2
SINON
    instructions3
FINSI
```



### Exemple :

```
Variable jour : 1.. 7
Debut
    Si jour = 1 alors Ecrire("Lundi")
    SinonSi jour = 2 alors Ecrire("Mardi")
    SinonSi jour = 3 alors Ecrire("Mercredi")
    SinonSi jour = 4 alors Ecrire("Jeudi")
    SinonSi jour = 5 alors Ecrire("Vendredi")
    SinonSi jour = 6 alors Ecrire("Samedi")
    Sinon : Ecrire("Dimanche")
FinSi
Fin
```

### En PHP :

```
if (condition)
{
    instructions;
}
elseif (condition)
{
    instructions;
}
else
{
    instructions;
}
```

### Exemple :

```
$jour = $_POST["jour"];
if ($jour == 1)
{
    echo "Lundi";
}
elseif ($jour == 2)
{
    echo "Mardi";
}
elseif ($jour == 3)
{
    echo "Mercredi";
}
elseif ($jour == 4)
{
    echo "Jeudi";
}
elseif ($jour == 5)
{
    echo "Vendredi";
}
elseif ($jour == 6)
{
    echo "Samedi";
}
else
{
    echo "Dimanche";
}
```





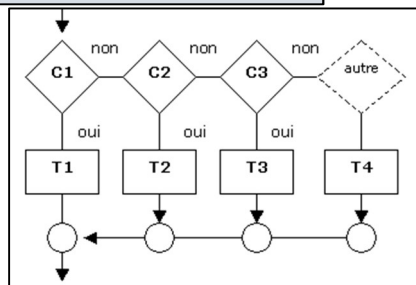
## SELON QUE

Bien que le SI... SINON SI... SINON soit plus compact, cela reste un peu lourd. Une autre instruction va nous permettre de soulager l'écriture de ce genre de tests : le SELON QUE.

Il permet de condenser une armée de SI... SINON SI... SINON de manière plus élégante.

### Pseudo-code :

```
SELON QUE expression VAUT
  valeur : instructions1
  valeur : instructions2
  [sinon : instructions3]
FINSQ
```



### Exemple :

```
Variable jour : 1.. 7
Debut
  Lire(jour)
  SelonQue jour vaut
    1 : Ecrire("Lundi")
    2 : Ecrire("Mardi")
    3 : Ecrire("Mercredi")
    4 : Ecrire("Jeudi")
    5 : Ecrire("Vendredi")
    6 : Ecrire("Samedi")
    Sinon : Ecrire("Dimanche")
  FinSQ
Fin
```

### En PHP :

```
switch (variable)
{
  case valeur :
    instructions;
    break;
  case valeur :
    instructions;
    break;
  default :
    instructions;
    break;
}
```

Le mot-clé break permet de délimiter la fin d'un bloc d'instructions.

### Exemple :

```
$jour = $_POST["jour"];
switch ($jour)
{
  case 1 :
    echo "Lundi";
    break;
  case 2 :
    echo "Mardi";
    break;
  case 3 :
    echo "Mercredi";
    break;
  case 4 :
    echo "Jeudi";
    break;
  case 5 :
    echo "Vendredi";
    break;
  case 6 :
    echo "Samedi";
    break;
  default :
    echo "Dimanche";
    break;
}
```



### Exercice 4.1

#### Consigne

##### **Année bissextile**

Réaliser un petit algorithme qui sur base d'une année donnée va déterminer s'il s'agit d'une année bissextile. Une année est bissextile si elle est divisible par 4, mais non divisible par 100. Ou si elle est divisible par 400.

### Exercice 4.2

#### Consigne

##### **Lanceur de balles de tennis**

Réaliser l'algorithme d'un lanceur de balles de tennis. Ce lanceur possède deux états :

- prêt : permet de savoir si le tennisman est prêt. Il ne faut pas lancer de balles dans le cas contraire
- panierVide : permet de savoir s'il y a encore des balles disponibles

Le lanceur de balle écrira « lancerBalle » qui, vous l'aurez compris, permet de lancer une balle.

### Exercice 4.3

#### Consigne

- **Distributeur de boissons**

Réaliser l'algorithme d'un distributeur de boissons. Ce dernier propose plusieurs boissons (eau et coca) et l'utilisateur choisit celle qu'il désire en entrant le numéro correspondant. Ne pas oublier de vérifier s'il y a encore des boissons en stock.

### Exercice 4.4

#### Consigne

- **Calculatrice**

Réaliser l'algorithme d'une calculatrice basique. L'utilisateur est invité à saisir un nombre, un opérateur et un deuxième nombre. La calculatrice affiche ensuite le résultat.

### Exercice 4.5

#### Consigne

- **Appréciations**

Ecrire un algorithme qui écrit une appréciation en fonction des points des étudiants.

Notes :

- < 10 : I
- 10-12 : S
- 13-15 : B
- 16-18 : TB
- >= 19 : Excellent

## Chapitre 5 : Les structures itératives

Nous pouvons maintenant faire pas mal de choses avec ce que nous avons appris.

Mais jusqu'à présent, nos algorithmes ne peuvent exécuter leurs instructions qu'une seule fois. Ce n'est pas très pratique !

Reprenons le lanceur de balle de tennis, le distributeur de boissons et la calculatrice :

- le lanceur de balle devrait pouvoir lancer des balles jusqu'à ce que son stock soit vide ;
- le distributeur de boissons pourrait proposer une autre boisson au client tant que ce dernier le désire ;
- pareil pour la calculatrice.

### Les boucles

Afin de répéter des instructions dans nos algorithmes, nous utilisons les boucles.

Une boucle est une structure itérative permettant de réitérer un comportement sur base d'une condition précise.

#### Exemples :

**TANT QU'**il y a des balles, lancer une balle.

Proposer une boisson **JUSQU'À** ce que le client soit comblé.

**FAIRE** un calcul **TANT QUE** l'utilisateur veut faire des calculs.

Pour utiliser une boucle, il est nécessaire de pouvoir identifier clairement :

- ce qu'il faut répéter
- pourquoi le répéter
- maîtriser cette répétition (principalement, comment l'arrêter)

Nous répétons des instructions afin d'atteindre un état défini. Il est donc primordial que les instructions répétées permettent d'atteindre l'état voulu sous peine de se retrouver dans une boucle infinie !

### Les types de boucles

Nous avons à notre disposition 3 types de boucles :

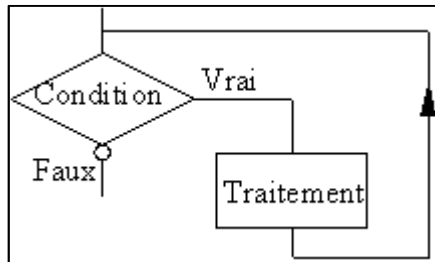
- Boucles à tests antérieurs (TANT QUE ... FAIRE)
- Boucles à tests postérieurs, décomposées en deux (FAIRE... TANT QUE et FAIRE... JUSQU'À)

## TANT QUE ... FAIRE

Dans cette boucle, nous répétons nos instructions tant que la condition définie est vraie. De plus, si cette condition n'est pas vérifiée au départ, nous n'exécutons pas du tout les instructions.

### En pseudo-code :

```
TANT QUE condition(s) FAIRE
    instructions
FINTQ
```



### Exemple :

```
Variable nombre : Entier
Debut
    nombre ← 0
    TANT QUE (nombre < 1) OU (nombre > 10)
    FAIRE
        Ecrire("Valeur incorrecte ! Réessayer")
        nombre ← nombre + 1
    FINTQ
Fin
```

### En PHP :

```
while (conditions)
{
    instructions;
}
```

### Exemple :

```
$nombre = 0;
while (($nombre < 1) || ($nombre > 10))
{
    echo "Nombre incorrect ! Réessayez.";
    $nombre = $nombre + 1;
}
```

## Exercice 5.1

### Consigne

À l'aide d'une boucle, afficher la table de multiplication par 2.

## Exercice 5.2

### Consigne

À l'aide de deux boucles, afficher les tables de multiplication de 1 à 9.

## Exercice 5.3

### Consigne

Adaptez l'exercice (4.2) du lanceur de balles pour qu'il travaille en continu.

## Exercice 5.4

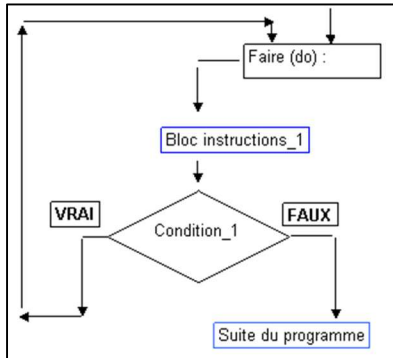
### Consigne

Adaptez l'exercice (4.4) de la calculatrice pour qu'elle travaille en continu.

## FAIRE... TANT QUE

### En pseudo-code :

```
FAIRE
    instructions
TANT QUE condition(s)
```



### Exemple :

```
Variable nombre : Entier
Debut
    nombre ← 0
    FAIRE
        Ecrire("Valeur incorrecte ! Ajoutons 1")
        nombre ← nombre + 1
    TANT QUE (nombre < 7) OU (nombre > 10)
Fin
```

### En PHP :

```
do {
    instructions;
} while (conditions);
```

### Exemple :

```
$nombre = 0;
do
{
    echo "Valeur incorrecte ! Ajoutons 1";
    $nombre = $nombre + 1;
} while (($nombre < 7) || ($nombre > 10))
```

## POUR...

La boucle POUR est plutôt utilisée pour parcourir des tableaux (Chapitre suivant).

Elle contient 3 éléments :

- départ : « Initialisation » : valeur que l'on donne au départ à la variable.
- condition : tant que la condition est remplie, la boucle est exécutée.
- pas : « incrémentation » : le nombre que l'on ajoute à chaque passage.

### En pseudo-code :

```
POUR indice DE valeurDebut À valeurFin [PAR p]
FAIRE
    instructions
FINP
```

### Exemple :

```
Debut
    POUR nombre DE 1 <10 [PAR 1] FAIRE
        Ecrire("Ceci est la ligne n°"+nombre)
    FINP
Fin
```

### En PHP :

```
for(départ; conditions; pas)
{
    instructions;
}
```

### Exemple :

```
for($nombre =1; $nombre <10; $nombre++)
{
    echo 'Ceci est la ligne n°'.$nombre;
}
```

## Chapitre 6 : Les tableaux

### Utilité

Dans beaucoup de cas, nous pouvons avoir besoin de nombreuses variables d'un même type :

- garder un ensemble de relevés de température pour fournir des statistiques ;
- conserver les notes d'un élève pour calculer sa moyenne.

Nous serions tentés de déclarer autant de variables que le nombre de valeurs dont nous avons besoin. C'est embêtant et laborieux !

Les informaticiens sont paresseux, ils n'aiment pas faire deux fois la même chose ☺ !

Les tableaux nous permettent d'obtenir un nombre donné d'éléments d'un même type.

Un tableau est une structure de données composée d'un ensemble de variables, du même type, accessibles par leur indice.

Indices							
0	1	2	3	4	5	6	7
12	512	15	4	-5	1	-88	782
Variables							

La déclaration d'un tableau est similaire à celle des variables avec deux éléments supplémentaires :

- le type de variables que va contenir le tableau ;
- la taille du tableau : représente le nombre d'éléments que peut contenir le tableau.

### Déclarer un tableau

Pour déclarer un tableau, rien de plus simple :

```
$tab = array() ;
```

### Affectation dans un tableau

Nous sommes capables de déclarer un tableau de N cases d'un certain type. Mais nous devons pouvoir accéder à chaque case du tableau !

Les cellules d'un tableau sont toutes numérotées à partir de 0 jusqu'à la taille du tableau -1.

Le numéro d'une cellule peut-être appelée l'indice de la cellule. Nous utilisons l'opérateur d'accès [i] qui va nous permettre d'accéder à la variable du tableau dont l'indice est i.

### Observons :

```
$tab[0] = 4 ;
$tab[1] = 2 ;
$tab[2] = 10 ;
```

### Complétez le tableau :

0	1	2

### En PHP :

#### Un tableau d'entiers :

```
$tableau = array(4, 2, 10) ;
```

```
$tableau[0] = 4 ;  
$tableau[1] = 2 ;  
$tableau[2] = 10 ;
```

#### Un tableau de texte :

```
$tableau = array("a", "b", "c") ;
```

```
$tableau[0] = "a" ;  
$tableau[1] = "b" ;  
$tableau[2] = "c" ;
```

### Trucs et astuces

Il est souvent utile de stocker la taille d'un tableau dans une constante. Cela rend l'algorithme plus facilement modifiable. Nous illustrerons cela un peu plus loin dans le chapitre.

Nous avons donc la taille du tableau en constante, mais comment savoir quelles sont la ou les cellules possédant déjà une valeur ou non ? Une solution simple consiste à utiliser une variable supplémentaire pour retenir le nombre d'éléments utiles dans le tableau.

De plus, l'utilisation de cette variable va nous permettre d'enregistrer les nouvelles valeurs au bon endroit sans nous tromper.

### Exemple :

```
const MAX_ELEMENT = 2 ;  
  
$nbElement = 0 ;  
$tab[$nbElement] = 9 ;  
$nbElement = $nbElement + 1 ;// ou $nbElement++  
$tab[$nbElement] = 79 ;  
$nbElement = $nbElement + 1 ;// ou $nbElement++  
  
for($i = 0 ; $i < MAX_ELEMENT ; $i++)  
{  
    Echo '$tab[$i]<br>' ;  
}
```

#### Complétez le tableau :

0	1

### Exercice 6.1

#### Consigne

Écrire un algorithme qui enregistre 6 entiers et les stocke dans un tableau, puis affiche le contenu de ce tableau une fois qu'il est rempli.

### Exercice 6.2

#### Consigne

Initialiser un tableau de 10 entiers avec les valeurs 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 à l'aide d'une boucle. Ensuite, à l'aide d'une boucle afficher la valeur de chaque cellule du tableau.

### Exercice 6.3

#### Consigne

Ecrivez un algorithme qui trouve qui est le nombre le plus grand dans un tableau de 5 données (1, 4, 3, 5, 2)

### Exercice 6.4

#### Consigne

Ecrivez un algorithme qui trouve qui est le nombre :

- le plus grand

- le plus petit

dans un tableau de 5 données (1, 4, 3, 5, 2)

### Exercice 6.5

#### Consigne

Ecrivez un algorithme qui calcule la somme des valeurs du tableau

### Exercice 6.6

#### Consigne

Ecrivez un algorithme qui calcule la moyenne des valeurs du tableau

### Exercice 6.7

#### Consigne

Ecrivez un algorithme qui contient un tableau avec les mois de l'année et qui les affichent dans l'ordre chronologique (janvier, février, ...)

### Exercice 6.8

#### Consigne

Ecrivez un algorithme qui contient un tableau avec les mois de l'année et qui les affichent dans l'ordre inverse (décembre, novembre, ...)

### Exercice 6.9

#### Consigne

Ecrivez un algorithme qui en fonction du nombre de jour par mois affiche le calendrier.

### Exercice 6.10

#### Consigne

Voici les températures maximales pour le mois de septembre :

- enregistrer les données dans le tableau
- afficher la date suivie de la température

EX:

1 septembre : 20°C

2 septembre : 20°C



### Exercice 6.11

#### Consigne

Voici les températures pour le mois de septembre :

- afficher la moyenne de température pour ce mois.

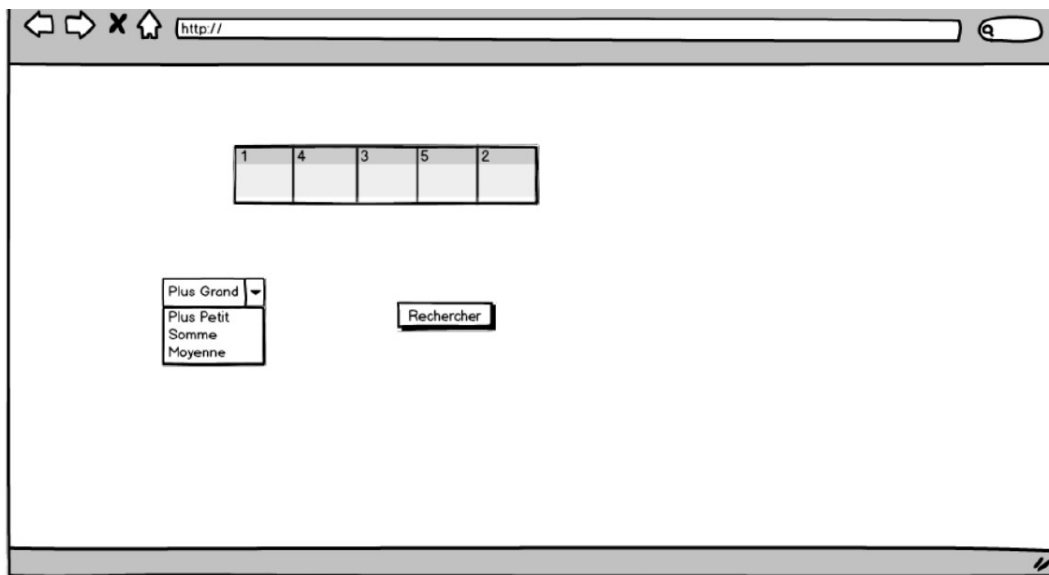
### Exercice 6.12

#### Consigne

À l'aide des boucles, réalisez un algorithme permettant de trier un tableau d'entiers dans l'ordre croissant. Mettez-le ensuite en pratique en PHP.

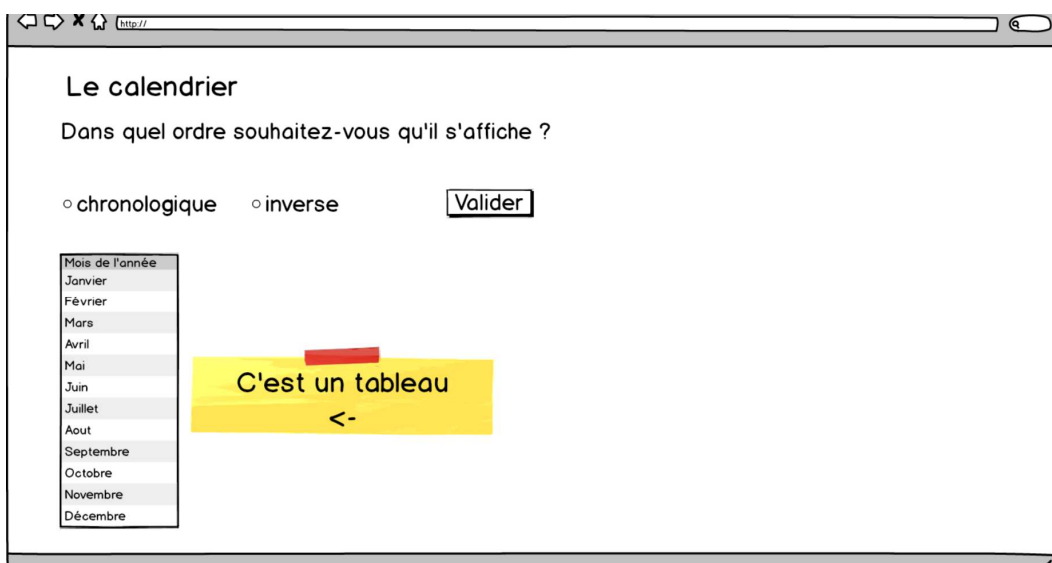
### Projet 1 : Le tableau

Ce projet est en lien avec les exercices 6.3 à 6.6.



### Projet 2 : Le calendrier ordonné

Ce projet est en lien avec les exercices 6.7 à 6.8.



### Projet 3 : Le calendrier jour

Ce projet est en lien avec les exercices 6.7 et 6.9

The screenshot shows a web browser window with the title 'Le calendrier'. The page content is divided into two main sections. The left section, titled 'Voici le calendrier avec le nombre de jours par mois', contains a table with the following data:

Mois de l'année	Jours par mois
Janvier	31
Février	28
Mars	31
Avril	30
Mai	31
Juin	30
Juillet	31
Aout	31
Septembre	30
Octobre	31
Novembre	30
Décembre	31

The right section, titled 'Rechercher le nombre de jours par mois :', contains a dropdown menu with 'Janvier' selected, a 'Rechercher' button, and the result '31'. The dropdown menu also lists the other months of the year.

### Projet 4 : Les températures

Ce projet est en lien avec les exercices 6.10 à 6.11.

The screenshot shows a web browser window with the title 'Les températures du mois de septembre 2017'. The page content includes the text 'Voici le calendrier avec leur température :'. Below this is a calendar grid for September 2017. The grid shows the days of the month and the temperature for each day. The temperatures are: 20, 20, 20, 20, 24, 19, 18, 17, 16, 17, 17, 18, 17, 17, 14, 15, 16, 16, 16, 17, 17, 19, 17, 20, 19, 20, 21, 21, 24, 17. The temperatures 24 and 14 are highlighted in red and blue respectively.

Il a fait le plus froid le : 15 avec une température de 14°C

Il a fait le plus chaud le : 5 et le 24 avec une température de 24°C

La température moyenne a été de : 18.3°C