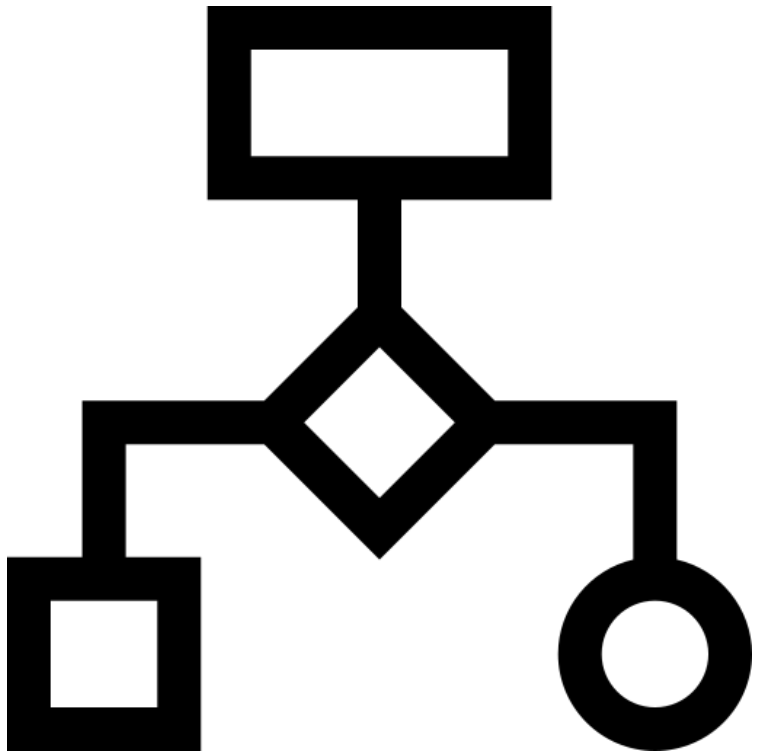

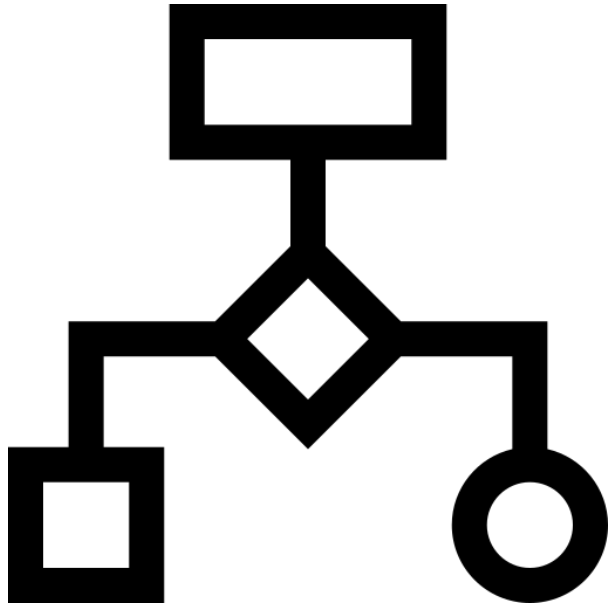

L’algorithmique

Les expressions :
Opérations de base, syntaxe,….

Il existe **plusieurs** **types** d'opérateurs :

- ✓ Les opérateurs **arithmétiques** ;
- ✓ Les opérateurs **de comparaison** ;
- ✓ Les opérateurs **logiques** ;
- ✓ Les opérateurs **divers**.

Les opérateurs arithmétiques :

Opérateur	Opération
+	L'addition
-	La soustraction
*	La multiplication
:	La division entière
/	La division réelle

Les opérateurs de comparaison :

Opérateur	Opération
=	Egal à
<>	Différent de
<	Inférieur à
<=	Inférieur ou égal à
>	Supérieur à
>=	Supérieur ou égal à

Les opérateurs logiques :

Opérateur	Opération
ET	ET logique
OU	OU logique inclusif
NON	Négation logique
XOR	OU logique exclusif

Les opérateurs divers :

Opérateur	Opération
<-	L'affectation
&	La concaténation

Syntaxe d'une expression :

Une expression est soit:

- ✓ une **valeur** (25, 3.8, 'Bonjour', la variable x,…) ;
- ✓ de la forme (**expression**) **opération** (**expression**) :

où les parenthèses sont **facultatives** est sont **uniquement** là pour **lever des ambiguïtés**….

- ➢ Ainsi, tout ce qui suit représente des expressions **valides** (on suppose que x est une variable)
  - ❖ 15
  - ❖ 3 + 8
  - ❖ 1 + (2 * x - 1)
  - ❖ 'Bon'+'jour'
  - ❖ (5*3) + (12 -8)
  - ❖ 1 + 2 + 3 + 4 + 5 * 2
  - ❖ 3 + Longueur('Bonjour')

Syntaxe d'une expression :

Les **comparaisons** (**ex**: en PHP et JS) :

- le test d'égalité

  « **=** » s'écrit **==** (**ex**: « x == 5 ») ;

- différent

  « **≠** » s'écrit **!=** (**ex**: x != 5) ;

- « ou **≤** et **≥**»

  s'écrivent **<=** et **>=** (**ex**: x <= 5).

Syntaxe d'une expression :

Les **opérateurs logiques** (**ex**: en PHP) :

- le « **et** logique »

  s'écrit **&&** (**ex**: « (5 < x) && (x<10)») ;

- le « **ou** logique »

  s'écrit || (ex: « (x < -1) || (x > 1) ») ;

- le « **non** logique » ou **contraire**

  s'écrit **!** (ex: « !(x < -1) »)

Instructions de base, séquentialité,….

L’exécution d’un algorithme **consiste** en l’**exécution** d’une **suite d’instruction** dans le **but** de **changer** l’**état** de la mémoire.

- Il y a **trois** instructions de base:
  - ★ Les **affectations**
  - ★ Les **saisies**
  - ★ Les **affichages**

Les instructions décrivent les **changements** de l'**état** de la mémoire :

- Elles s'écrivent :
  - ✓ **nom_variable** ← **expression**

  ex:

  - ❖ **x** ← **5**

    où **x** est une variable de type **numérique**

  - ❖ ou **nom** ← "Bourdon"

    où **nom** est une variable de type **chaîne de caractères**.

Les saisies :

- Ce sont les instructions qui permettent d'**interagir** en **entrée** avec l'utilisateur :
  - ✓ Ce dernier peut **saisir** des **valeurs**.
- Elles s'écrivent :

  nom_variable ← Saisie()

  ex:

  - **x ← Saisie()**
  - **x ← Saisie(''Veuillez entrer un chiffre : '')**

Les affichages :

- Ce sont les instructions qui permettent d'**interagir** en **sortie** avec l'utilisateur :
  - ✓ affichage d'un **résultat**.
- Elles s'écrivent :

  Ecrire(expression)

ex:

- **Ecrire**('Bonjour le monde')

La séquentialité

Un algorithme comporte une **suite d'instructions** qui s'exécutent (**par défaut**) l'une **après** l'autre dans l'**ordre** d'écriture.

- on parle de **séquentialité.**

Afin de marquer qu'une instruction succède à une autre instruction :

- on utilise le symbole « ; »

- et par **convention** et **clarté** d'écriture, on passe à la ligne.

Ex:

- x ← 5;
- y ← x+10; //La valeur de x utilisée ici est 5

Instruction Si-Alors-Sinon (If-Then-Else)

Si condition **p** vérifiée, exécuter action :

Si p **alors**
action

Si condition p vérifiée, exécuter action 1. **Sinon**, exécuter action 2.

Si *p* **alors**

action 1;

Sinon

action 2;

Bloc de conditions entre **Début** et **Fin**.

```
Si (x ≥ 0)  alors
 Début
 x = x-1;
 a = b+c;
 FinSi
```

Instruction Tant que

Tant que p est vrai **exécuter** action

Tant que p **Faire**
action;

Ex:

- **Tant que** x > 2 **Faire**
  x <- 3;

Instruction Pour (For)

Pour var = **init** à **limite** **Faire**
action;

Ex:

- **Pour** i=1 à 8 **Faire**
  x += i;
  x = i + 1;

Bilan

En résumé :

- ✓ Un algorithme est une suite **séquentielle** d'instructions qui **modifient** l'**état** de la mémoire.
- ✓ On **accède** à la mémoire par le biais de **variables** qui possèdent un **nom**, un **type** et une **valeur courante**.
- ✓ Un **historique** d'**exécution** est un outil **primordial** pour **comprendre** et **étudier** un algorithme.

Questions ??

Prochain chapitre: Exercices.