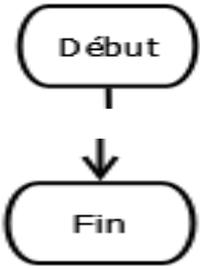
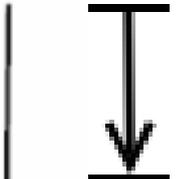
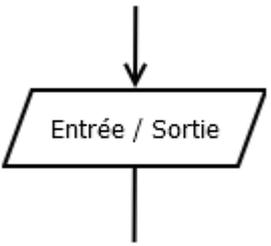
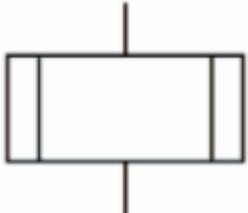
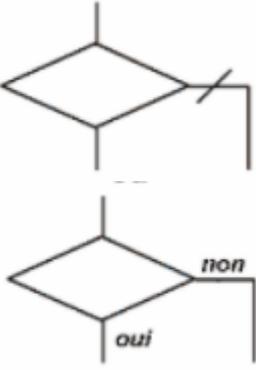
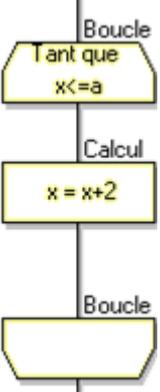


1. Fonction

Le rôle d'un algorithme est de représenter un algorithme.
La représentation graphique utilisée permet d'en simplifier la lecture.
Les algorithmes ne peuvent être utilisés que pour représenter des programmes relativement courts

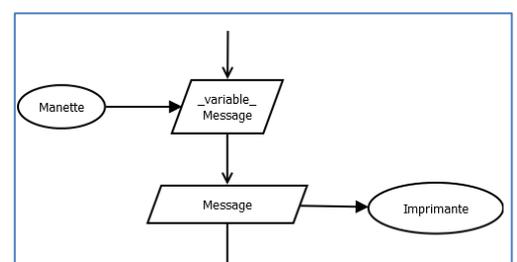
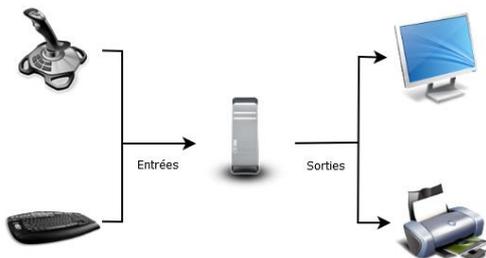
2. Symboles utilisés

Pour construire un algorithme, on utilise des symboles normalisés

			
<p>Début / fin Il n'y a qu'un seul « Début » mais il peut y avoir plusieurs « Fin »</p>	<p>Liaison, avec ou sans flèche, permettent de guider la lecture de l'algorithme</p>	<p>Instructions : elles permettent de réaliser les traitements : calculs, comparaisons, ...</p>	<p>Lecture ou écriture d'une donnée</p>
			
<p>Sous-programme</p>	<p>Test</p>	<p>Boucle</p>	<p>Commentaire</p>

Les périphériques. Ils sont représentés dans des formes ovales. Une flèche indique s'ils fournissent une valeur en entrée ou s'ils constituent une sortie.

Les entrées sont généralement mises à gauche et les entrées à droite.



3. Structures

Exemples de structures :

Structure	Algorithme	Algorithmme
<p>Structure linéaire</p> <p>La structure linéaire se caractérise par une suite d'actions à exécuter successivement dans l'ordre de leur énoncé.</p>	<pre> graph TD T1[Traitement 1] --> T2[Traitement 2] T2 --> T3[Traitement 3] </pre>	<p>FAIRE « traitement 1 » FAIRE « traitement 2 » FAIRE « traitement 3 »</p>
<p>Structure alternative ou conditionnelle</p> <p>Une structure alternative n'offre que deux issues possibles s'excluant mutuellement. Une condition est testée et en fonction du résultat du test soit le traitement 1, soit le traitement 2 est réalisé.</p>	<pre> graph TD Start(()) --> Cond{Condition} Cond -- oui --> T1[Traitement 1] Cond -- non --> T2[Traitement 2] T1 --> Join(()) T2 --> Join Join --> End(()) </pre>	<p>SI « condition » vraie</p> <p>ALORS FAIRE « traitement 1 »</p> <p>SINON FAIRE « traitement 2 »</p> <p>FIN SI</p>
<p>Structure répétitive ou itérative (boucle avec pré-test)</p> <p>Dans cette structure on commence par tester la condition, si elle est vraie alors le traitement est exécuté.</p>	<pre> graph TD Start(()) --> Cond{Condition} Cond -- oui --> T[Traitement] T --> Cond Cond -- non --> End(()) </pre>	<p>TANT QUE « condition » vraie</p> <p>FAIRE « traitement »</p> <p>FIN TANT QUE</p>
<p>Boucle avec comptage</p> <p>On initialise la variable N avec une valeur x. On teste si N est égal à 0, si ce n'est pas le cas, on exécute le traitement et on décrémente la variable N puis on teste à nouveau la variable N, et ainsi de suite jusqu'à ce que N=0.</p>	<pre> graph TD Start(()) --> Init[N = x] Init --> Cond{N = 0} Cond -- non --> T[Traitement] T --> Dec[N = N - 1] Dec --> Cond Cond -- oui --> End(()) </pre>	<p>POUR N = x A 0 REPETER</p> <p>« traitement »</p> <p>FIN POUR</p>