ICN	**Algorithmie et Programmation** Document ressource		**Informatique et création numérique**

1. Avant propos

Avant de nous lancer dans l'apprentissage d'un langage de programmation, il faut tout d'abord définir un peu de vocabulaire :

Un programme, est formé par un ensemble d'instructions (simples) qui sont exécutés successivement par une machine séquentielle (microprocesseur d'un ordinateur). Un programme peut donc réaliser des tâches de calculs afin de réaliser des jeux, des simulateurs … Les processeurs sont capables d'effectuer plusieurs **M**illions d'**I**nstructions **P**ar **S**econde (MIPS), pour les moins rapides, à plusieurs dizaines de milliards pour les composants récents (>100.10^9)

Le code Source, est un texte (écrit par exemple en langage Cou C++ ou C#), compréhensible par le programmeur, qui est la forme sous laquelle est crée le programme. Cependant, des représentations graphiques de programmes voient de plus en plus le jour (***AppInventor*** _ Scratch_ pour Androïd, ***Flowcode*** pour les systèmes à microcontrôleurs …). Ces derniers offrent une grande simplicité dans la création mais n'autorise pas tous les développements que le « code » permet. Les professionnels de l'informatique se tournent généralement vers des solutions «écrites » plutôt que graphiques. Cependant, ces « passerelles » sont très intéressantes, et offrent beaucoup de ressources…

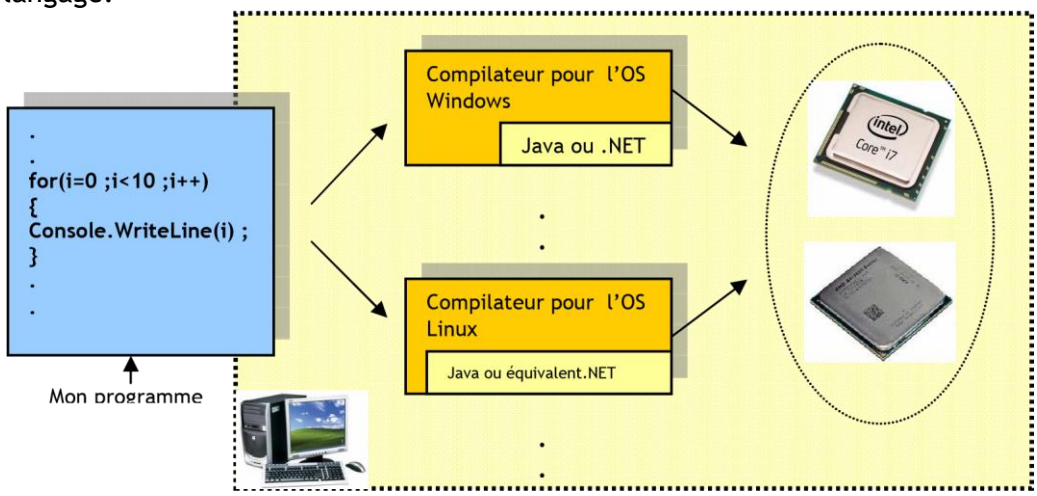
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World");
            Console.Read();

        }
    }
}
```

Le compilateur : est un programme utilitaire qui permet de **traduire** le code source (ou le graphique représentant le programme) en une suite de codes binaires, image du programme écrit, mais compréhensible par le microprocesseur. (En fait, bien souvent, le compilateur fournit une suite de code à un « exécuteur » intermédiaire _ « machine java » par exemple ou framework .NET)

Dès lors on peut introduire une notion très importante en programmation : la **portabilité** d'un langage.



Un programme écrit par exemple en **langage C,** pourra être « compris » par des machines équipées d'**OS** (Operating System _ **Système d'exploitation**_) différents, voire même de processeurs différents (**A**dvanced **M**icro **D**evice et Intel par exemple). Il suffit que le compilateur soit adapté… *la participation de AMD et Intel à la conception de tels compilateurs représente également un enjeu économique …*

En tout cas le travail du programmeur ne changera pas d'une machine à l'autre. On parle alors de ***portabilité***.

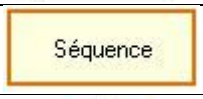
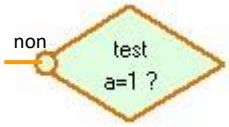
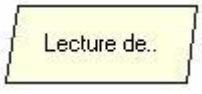
2. Algorithmie

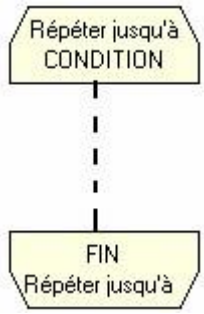
2.1 D'abord qu'est-ce que l'algorithmie (ou algoritmique) ?

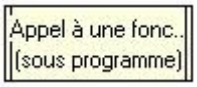
C'est l'ensemble des règles qui permettent d'écrire une suite finie d'opérations permettant de résoudre un problème.
La suite d'opérations ainsi définie se nomme **algorithme**.

Un algorithme peut être représenté sous une forme graphique, il s'agit alors de **l'algorigramme**.

2.2 Les symboles graphiques, l'écriture d'un algorithme (norme ISO 5807)

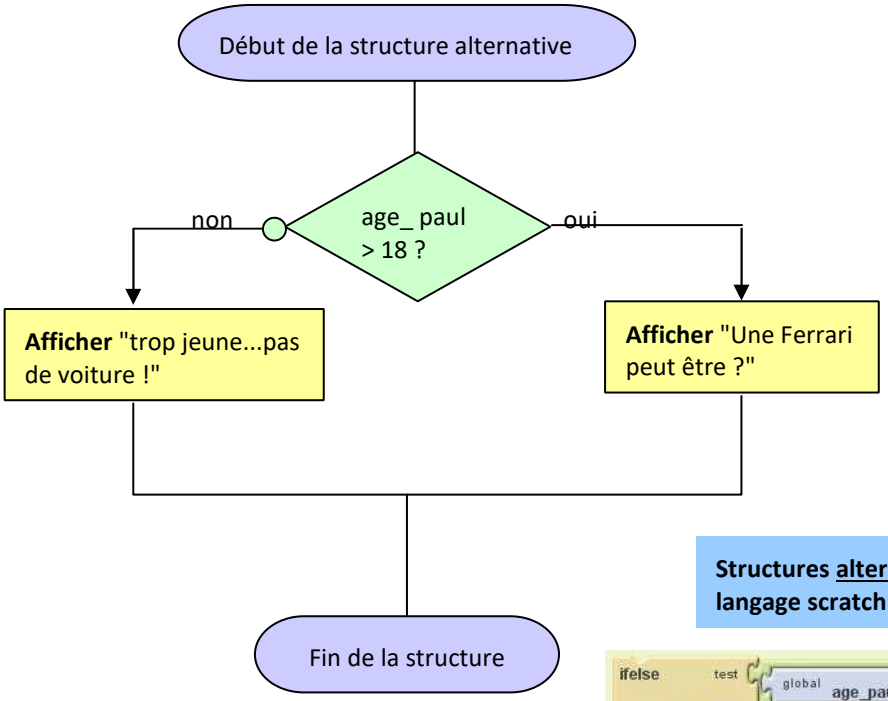
Symboles	Signification	Exemple algorithme (texte)	Exemple algorithme (langage C#)
	Indique le début d'un processus (programme ou sous-programme)	Début du programme	**int main()** **{**
	Indique une suite d'actions.	variable2 = variable1+3 ; variable3 = variable2*5 ;	variable2 = variable1+3 ; variable3 = variable2*5 ;
	Indique un **test conditionnel** permettant au processus d'opérer un **choix.**	**Si** age_paul plus grand ou égal à 18 **alors** **afficher** une Ferrari c'est bien ! mais les assurances sont chères **sinon** **afficher** la voiture attendra, et mon vélo est sympa **fin si**	**If** (age_paul>=18) { Console.WriteLine(«une Ferrari c'est bien ! mais les assurances sont chères ») ;} **else** {printf(«la voiture attendra, et mon vélo est sympa») ;}
	Indique des étapes d'initialisations. Par exemple, attributions de valeurs à des variables. Et/ou appel d'une méthode d'initialisation.	variable1=0 variable2=0 initialisation afficheur LCD	variable1=0 ; variable2=0 ; LCD_init();
	Indique une opération permettant une acquisition (lecture d'un clavier….)	Lire une entrée au clavier.	`i=Console.Read();/*permet de placer une lettre tapée au clavier dans la variable i.*/`

	Indique une **boucle.** Cette boucle (processus itératif) permet de **répéter une séquence jusqu'à ce que la condition soit atteinte.**	Boucle **TANT QUE**
		Tant que (***condition est vraie***) faire
		Afficher « bonjour »
		Fin faire
		
		Boucle **REPETER JUSQU'A**
		Répéter
		Afficher « bonjour »
		Jusqu'à (***condition d'arrêt***)
		
		Ou encore…
		Répéter
		Afficher « bonjour »
		Tant que (***condition est vraie***)
			(TANT QUE …FAIRE : while…)
			while (i<10)
			{
			Console.WriteLine ("bonjour") ;
			}
			
			//
			//
			//
			(FAIRE … TANT QUE : do …while) do
			{
			Console.WriteLine ("bonjour");
			}
			while (i<10)
			Voir en page suivante

	Indique l'appel d'un processus (méthode, fonction ou sous-programme)	Afficher « bonjour »	**Console.WriteLine** (« bonjour »)
	Indique la fin d'un processus.	Fin du programme	**}**

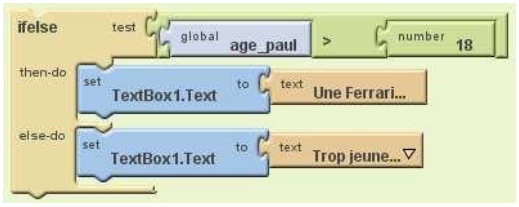
Les structures alternatives (de décisions)

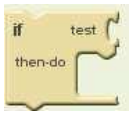
Ces structures sont employées à chaque fois qu'il est nécessaire qu'un choix soit effectué. On teste la valeur d'une variable, et suivant le résultat du test, une séquence est exécutée plutôt qu'une autre...

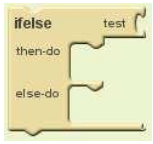



```
//début
if (age_paul >18)
 {
        Console.WriteLine («Une
Ferrari... ») ;
  }
else
{
 Console.WriteLine («trop jeune ») ;
}
//fin
```

Structures alternatives avec langage scratch (AppInventor)



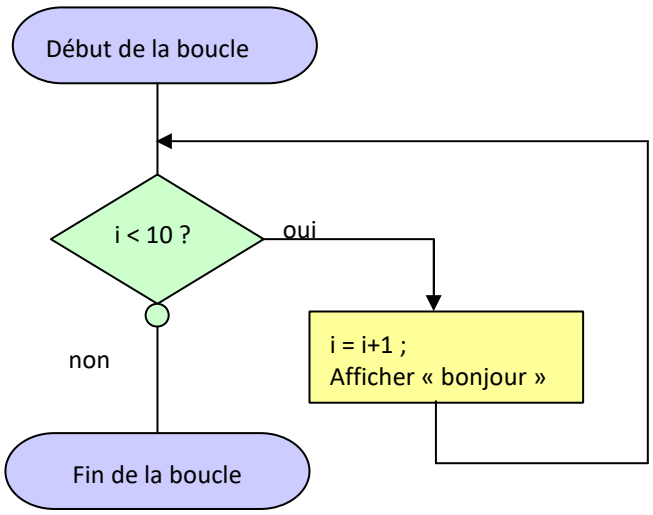




Les structures itératives (boucles)

Ces structures sont très employées en informatique pour répéter une suite d'instructions jusqu'à ce qu'une condition devienne vraie.

A titre d'exemple la boucle ***while***, peut être représentée ainsi :



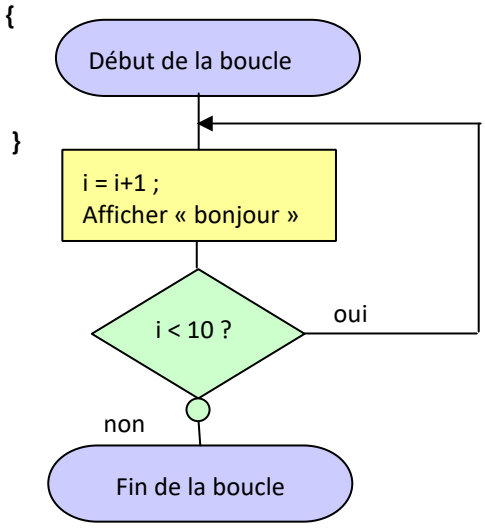

```
//début
while (i < 10)
 {
        i=i+1 ;
        Console.WriteLine (« bonjour ») ;
  }
//fin
```

Cette boucle (**à condition que i soit à zéro au départ**) permettra d'afficher 10 fois le mot « bonjour »

La dernière valeur « atteinte » par i sera 10.

Structure de la boucle while avec langage scratch (AppInventor)

…et la boucle ***do … while***



//début **do**

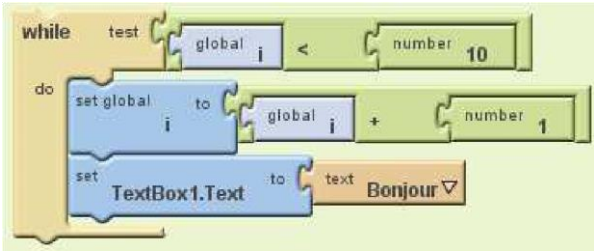
i=i+1 ;

Console.WriteLine (« bonjour ») ;

while (i < 10)

//fin

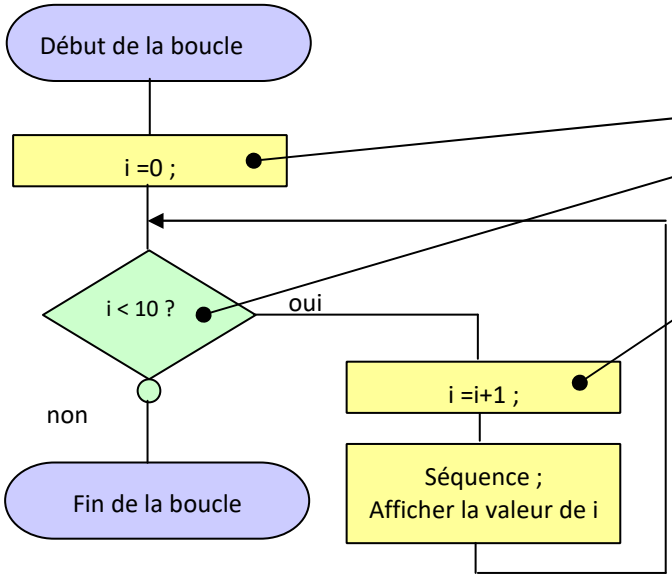
Cette boucle (**au départ**) permettra d'afficher 10 fois le **à condition que i soit à zéro** mot « bonjour »



La dernière valeur « atteinte » par i sera 10.

A la différence de la boucle while, la séquence est au moins exécutée une fois.

Un autre type de boucle très employée existe également, il s'agit de la boucle « ***for…*** »

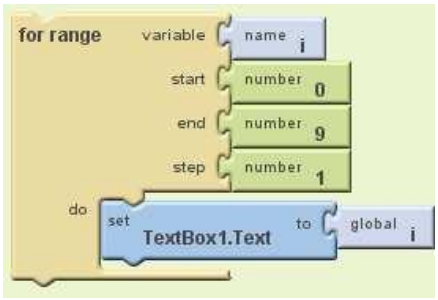



```
//début
for (i=0 ; i<10 ;i=i+1)
{
Console.WriteLine («Bonjour » );
}
//fin
```

Cette boucle permettra d'exécuter dix f ois l'affichage « Bonjour »

La dernière valeur « atteinte » par i sera 10.

Structure de la <u>for…next</u> avec langage scratch (AppInventor)



3. Classe, Objet, Propriétés, Méthodes, Evénements…

Voilà toute une terminologie qui n'aide pas, a priori, et pourtant…

En **programmation orientée objet** (**POO**), une ***classe représente la structure d'un objet***, c'est en quelque sorte son plan (au même titre que celui d'une maison). Un **<u>objet</u>** est en fait une ***instanciation*** d'une classe (c'est la maison !). Une même classe peut donc donner naissance à plusieurs objets ***pouvant se comporter différemment et avoir des <u>propriétés</u> différentes*** (un même plan peut donner naissance à tout un quartier…).

Le comportement ainsi que les propriétés sont ici importantes, en effet, une « maison 1 » pourra être bleue (c'est une maison bleue adossée …) et une « maison 2 » avoir des volets roses. De même l'événement « ouverture porte » de la maison 1 n'est pas obligé de se produire en même temps que l'événement « ouverture porte » de la maison 2 (en C#, créer un nouvel objet à partir d'une classe se fait avec le mot clef ***new***).

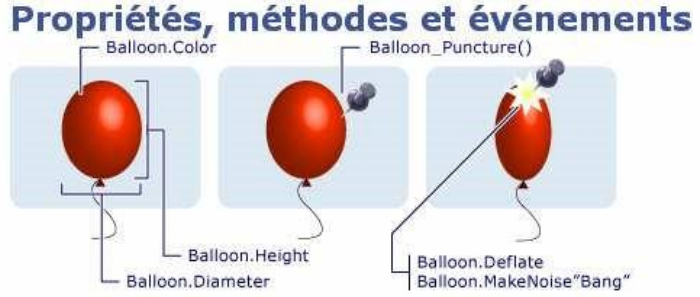


Image : http://msdn.microsoft.com

Extrait adapté de Microsoft.com

« Un ***objet*** ballon possède des ***propriétés*** (Color, Height et Diameter), répond à des ***événements*** (Puncture) et peut exécuter des ***méthodes*** (Deflate, MakeNoise).

Propriétés

Si vous pouviez programmer un ballon, son code Visual C# ressemblerait au « code » suivant, qui définit les propriétés d'un ballon.

```
Ballon.Color = Red; Ballon.Diameter
= 10;
Ballon.Inflated = True ;
```

Remarquez l'ordre du code : l'objet (Ballon), suivi par la propriété (Color), suivie par l'assignation de la valeur (= Red). Vous pourriez modifier la couleur du ballon en remplaçant sa valeur.

Méthodes

Les méthodes d'un ballon sont *appelées* de la manière suivante :

```
Ballon.Inflate();
Ballon.Deflate();
Ballon.Rise(5);
```

L'ordre est similaire à celui d'une propriété : l'objet (un nom), suivi de la méthode (un verbe). Dans la troisième méthode, un élément supplémentaire, appelé ***argument***, spécifie la hauteur d'élévation du ballon. Certaines ***méthodes*** auront un ou plusieurs arguments pour décrire plus en avant l'action à effectuer.

Événements

Le ballon peut réagir à un événement de la manière suivante :

```
Ballon_Puncture()
        {
        Ballon.MakeNoise("Bang");
        Ballon.Deflate();
        Ballon.Inflated = False;
        }
```

Lorsque l'évènement "crevaison" arrive au ballon, la séquence de programme correspondante est alors exécutée.

fin d'extrait

A titre d'analogie :
Si un **objet** est un "robot culinaire tout en un"...
Une de ses **propriétés** peut être le type de lame choisi par le cuisinier.
Une de ses **méthodes** peut être le type de mouvement de la lame (vitesse...etc..) Un des **évènements** lié à l'objet peut être un signal annonçant "la soupe est prête !" quand la température de cuisson est atteinte depuis assez longtemps.

Une **propriété** configure l'objet
Une **méthode** demande à l'objet d'exécuter une action
Un **évènement** est une action qui arrive à l'objet à un instant non connu.