

Cours d'algorithmique illustré par des exemples pour le picbasic

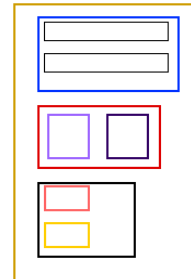
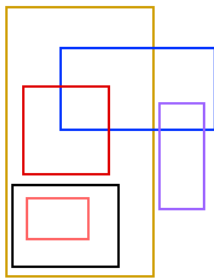
Même s'il est possible d'écrire un programme petit à petit par touches successives, le résultat est souvent décevant, la mise au point délicate et la maintenance difficile, un minimum de méthode et de rigueurs sont nécessaires.

La programmation structurée

Un programme doit être construit par niveaux du général au particulier. A l'intérieur de chaque grand logiciel, il existe un petit logiciel qui ne demande qu'à sortir

Structure d'un programme mal écrit. Tous les modules sont dépendants. l'écriture et la maintenance sont difficiles.

Programme structuré, chaque module est décomposé en modules plus petits et indépendants.



L'analyse descendante

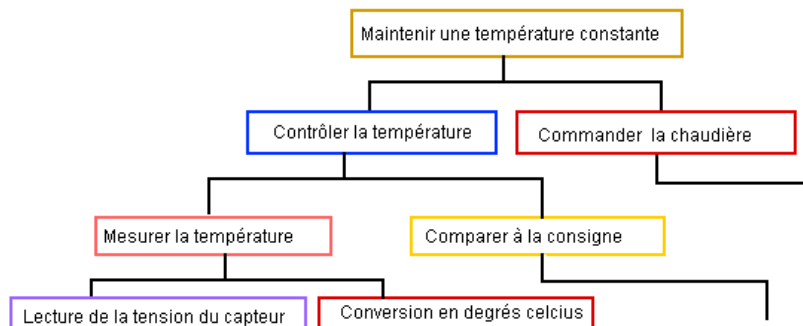
La programmation est avant tout une méthode d'analyse et non l'apprentissage d'un langage.

Une bonne méthode permet

- Une conception aisée
- Une meilleure efficacité du programme (moins d'erreurs)
- Une maintenance aisée

Un programme est un système complexe qu'il n'est pas possible d'aborder dans son ensemble. Il faut le décomposer en petits modules indépendants.

Exemple de la gestion d'un chauffage



Le programme est construit par niveaux, chaque niveau définit plus précisément le précédent. Le dernier niveau est celui de la machine.

Exemple d'algorithme

Programme principal

```
DEBUT
  initialiser           { initialisation du système }
  configurer          { configuration du système }
  REPETER
    Lireentrees        { lecture des entrées }
    traiterdonnees     { traitement des données }
  JUSQUA appui sur stop
FIN
```

sous programme initialiser

```
DEBUT
  {ici le sous programme initialiser}
FIN
```

sous programme lireentrees

```
DEBUT
  {ici le sous programme Lireentrees}
FIN
```

sous programme traiterdonnees

```
DEBUT
  {ici le sous programme traiterdonnees}
FIN
```

Les variables

Les variables sont des éléments fondamentaux d'un programme, elles décrivent l'état du système à un instant donné. Affecter une valeur à une variable consiste à lui associer cette valeur jusqu'à une nouvelle affectation. Si une variable doit toujours avoir la même valeur, nous utilisons une constante.

Soit la variable température de type entier, cette variable peut changer de valeur régulièrement au fil du temps

Temperature

Pour affecter une valeur à la variable température nous pouvons écrire **temperature prend pour valeur 21**

Pour simplifier nous écrivons **temperature ← 21**

Une variable peut aussi prendre pour valeur une expression écrite avec les opérateurs du langage

A ← (B + 2) / F

X ← X + 1 (*incréméntation*)

EnButee ← port5 and 64 (opération logique)

Notion de type

Les objets que nous utiliserons seront exploités de différentes manières selon leur type

Soit un objet de type **texte** nommé **couleur**, cet objet peut prendre les valeurs "rouge", "vert" ou "bleu"

Soit un objet de type **entier** nommé **longueur**, cet objet est une valeur numérique comprise entre 0 et 100

Nous pouvons écrire **vitesse** \leftarrow **longueur / temp**
Mais nous ne pouvons pas écrire **bidule = couleur + 2**

longueur est un nombre avec lequel on peut faire des calculs mais pas couleur.

Les types courants sont

Les bits ou booléens : 0 ou 1, vrai ou faux

Les octets (byte) : 0 à 255

Les entiers (integer) : 5, 10, -4, 456 etc.

Les réels (real) : -12.3, 4.5, 0.9 etc.

Les caractères (char) : A, b,), ? etc.

Les chaînes de caractères (string) : "bonjour", "rouge", "ceci est une chaîne" etc.

L'algorithme

C'est l'outil que nous utilisons pour décrire le comportement d'un programme. Un algorithme est la description du comportement à l'aide d'actions élémentaires sur des objets connus.

Algorithme CalculSurface

```
Début
| variables longueur, largeur, surface : entiers
|
| Lire longueur
| Lire largeur
| surface  $\leftarrow$  largeur * longueur
| afficher surface
Fin
```

Programmer c'est

- Définir précisément le problème à résoudre
- Décrire peu à peu la solution par un algorithme
- Traduire l'algorithme en utilisant un langage de programmation

Les actions sont : **Lire**, **Afficher** et \leftarrow (qui signifie prend pour valeur)

Les objets manipulés sont : **longueur**, **largeur** et **surface**

Dans le programme les actions correspondent à des instructions et les objets à des variables

Formalisme algorithmique (règles d'écriture)

L'algorithme est écrit en langage courant compréhensible par toute personne

Les commentaires {ceci est un commentaire}	Le commentaire ne décrit pas un comportement mais renseigne le lecteur sur ce qui se passe.
Les actions Afficher	Nomme une action qui doit avoir lieu, il est souhaitable de faire ressortir les actions en gras ou en souligné.
Les variables longueur, largeur : entiers	Deux variables de type entier sont ici déclarées.
Affectation $A \leftarrow B + 2$	Le symbole \leftarrow signifie prend pour valeur, ici A prend pour valeur B+2
Blocs Début { Description des actions } Fin	Un morceau de programme est logé dans un bloc avec un début et une fin. Un trait vertical peut faciliter la lecture.

Les commentaires

Pour être exploitable, un programme doit être commenté même si cela peut sembler superflu au premier abord. Les commentaires sont de trois ordres.

- L'algorithme en français doit être joint au programme pour en faciliter la lecture
- Des commentaires doivent apparaître dans le programme lui même pour définir précisément ce qui s'y passe (titre, remarques etc.). Indiquer à quoi correspondent les différentes variables et utiliser si possible des identificateur clairs. (**heure** est plus clair que **h**, **temperature_four** est plus clair que **tf**)
- Une notice d'utilisation peut être jointe au programme.

Rq : Un programme mal commenté est très difficile à lire par une autre personne et même par son auteur quelques semaines après sa rédaction.


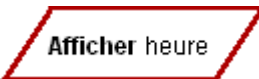
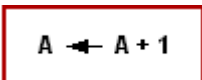
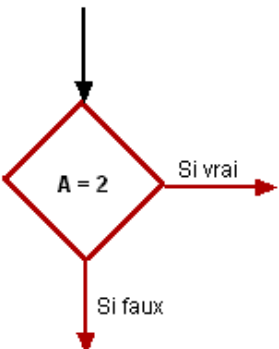
Les blocs

Un algorithme sera décrit sous forme de blocs qui décrivent les différentes parties du programme

Début
|
| { Description des actions }
|
Fin

On peut également décrire un algorithme avec un langage graphique mais cette description devient rapidement lourde à manipuler et ne facilite par une bonne structuration des programmes.

Les symboles utilisés dans ce cas sont les suivants.

	<p>Marque le début et la fin d'un bloc</p>
	<p>Description d'une opération d'entrée ou de sortie</p>
	<p>Description d'une action</p>
	<p>Exécution d'un test, selon le résultat, le programme se poursuit vers une branche ou vers l'autre.</p>

Les structures de contrôle

L'écriture d'un algorithme ou d'un programme consiste à décrire de façon statique (du texte) le comportement dynamique d'un système (microcontrôleur par exemple).

Nous disposons pour cela de trois possibilités

- Enchaîner les actions (exécution séquentielle)
- Choisir entre plusieurs solutions en faisant des tests
- Répéter des actions avec des boucles

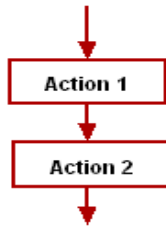
```

Répéter 1000 fois
|  afficher heure
fin

```

Ici trois lignes commandent des centaines d'opérations

L'enchaînement



Algorithme

```
Début  
| Action 1  
| Action 2  
Fin
```

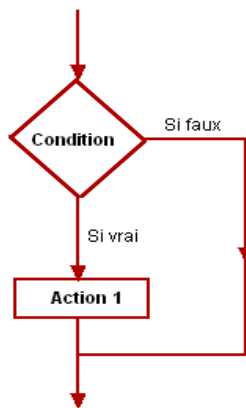
Les actions sont exécutées les unes à la suite des autres dans l'ordre d'écriture.

Exemple de programme

```
DIM tension AS BYTE  
DIM temperature AS BYTE
```

```
SET PICBUS HIGH  
LCDINIT  
tension = ADIN(4)  
temperature = tension / 5  
PRINT DEC (temperature)
```

La sélection



Algorithme

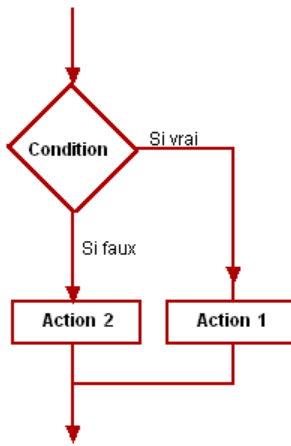
```
SI condition  
| ALORS action 1  
FIN SI
```

Si la condition est vraie alors l'action est exécutée, sinon elle ne l'est pas.

Exemple de programme

```
'Ce programme permet de tester le clavier 10 touches  
  
SET PICBUS HIGH 'Configure le "BUS" à 19200 bps  
LCDINIT 'initialise l'afficheur  
  
DIM touche AS BYTE 'déclaration de la variable de type octet  
  
PRINT "appuyer sur une touche"  
  
LOCATE 0,1  
touche = ADKEYIN(4) 'lecture touche sur l'entrée N°4  
IF touche <> 0 THEN 'une touche est enfoncée  
| PRINT "appui sur "  
| PRINT dec (touche,2,1) 'afficher son numéro  
END IF
```

Sélection (suite)



Algorithme

```

SI condition
  | ALORS action 1
  | SINON action 2
FIN SI

```

Si la condition est vraie alors l'action1 est exécutée, sinon c'est l'action 2 qui sera exécutée.

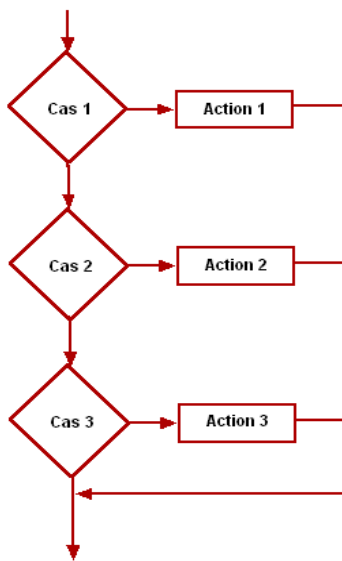
Exemple de programme

```

IF touche <> 0 THEN           'une touche est enfoncée
  | PRINT "appui sur "
  | PRINT dec (touche,2,1) 'afficher son numéro
ELSE
  | PRINT "          " 'efface le texte précédent
END IF

```

Les choix multiples



Algorithme

```

SELON condition
  | cas 1 : action 1
  | cas 2 : action 2
  | cas 3 : action 3
FIN SI

```

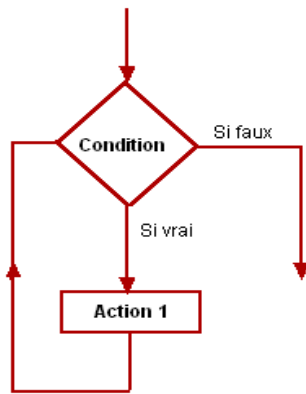
Cette structure de contrôle n'existe pas avec le picbasic, il faut donc la programmer avec les instructions de saut disponibles.

Exemple de programme

```
'SELON touche
IF touche = 1 THEN
  'appel du sous programme de configuration
  GOSUB configuration
ELSEIF touche = 2 THEN
  'appel du sous programme de marche rapide
  GOSUB marcherapide
ELSEIF touche = 3 THEN
  'appel du sous programme de marche lente
  GOSUB marchelente
ELSE
  'aucune touche valide
END IF
'FIN de SELON
```

La boucle

TANT QUE



Algorithme

```
TANT QUE condition
  | Action 1
Fin tant que
```

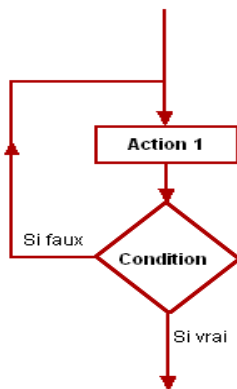
Rq : une boucle TANT QUE n'est pas exécutée si la condition est fausse au début

Cette structure de contrôle n'existe pas avec le picbasic, il faut donc la programmer avec les instructions de saut disponibles.

Exemple de programme

```
'TANT QUE A < 10
tantque10:
  IF A = 10 THEN GOTO fin
  'Action 1
  A = A + 1
  GOTO tantque10
fin:
```

La boucle REPETER



Algorithme

```
REPETER
  | Action 1
JUSQUA condition
```

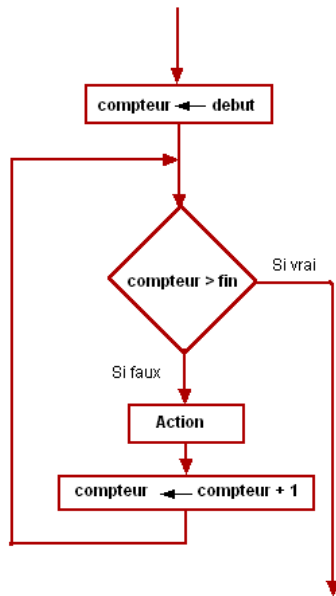
Rq : Contrairement à la boucle tant que, l'action est ici exécutée au moins une fois.

Cette structure de contrôle n'existe pas avec le picbasic, il faut donc la programmer avec les instructions de saut disponibles.

Exemple de programme

```
'REPETER jusqu'a A = 10
repeter10:
  'Action 1
  A = A + 1
  IF A <> 10 THEN GOTO repeter10
fin:
```

La boucle POUR



Algorithme

```
POUR compteur variant de debut à fin
  | action
FIN POUR
```

Cette boucle est utilisée quand le nombre de boucles à exécuter est connu avant.

Exemple de programme

```
dim i as integer
'exécution de 1000 mesures

for i = 1 to 1000
  | delay (100)
  | gosub MESURE
next i

*****
'sous programme de mesure
MESURE:
'debut
  | 'ici le programme de mesure
'fin
```