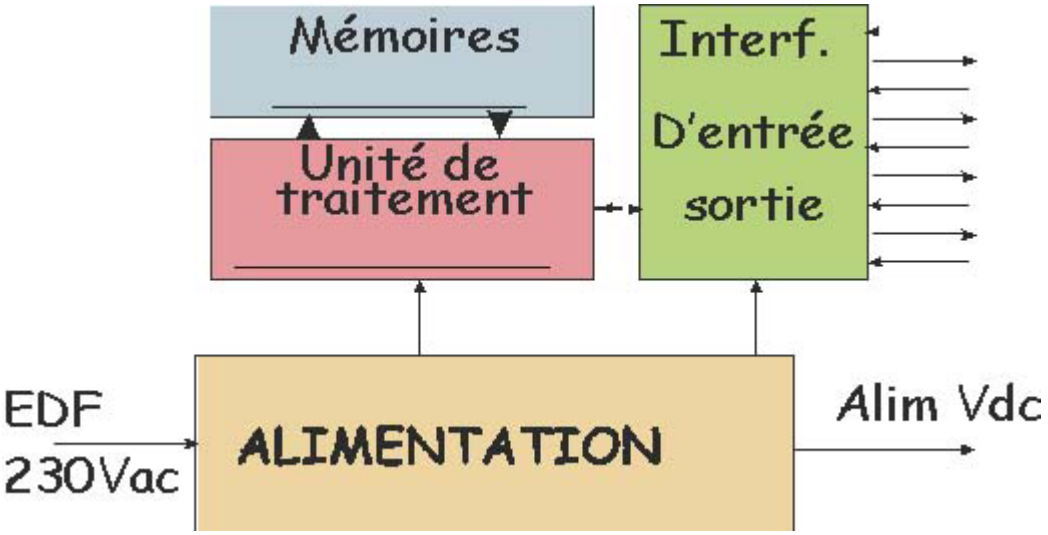
INITIATION À LA PROGRAMMATION ALGORITHME

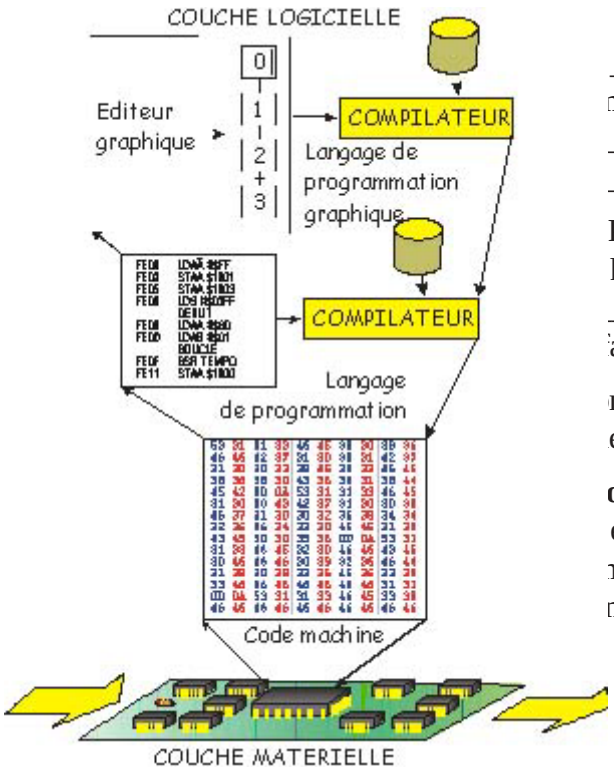
I/ NOTIONS PRÉLIMINAIRES ET VOCABULAIRE

Tout système de traitement numérique de l'information "moderne" se compose d'un couche matérielle et d'une couche logicielle.

La **couche matérielle** -partie visible du système- est centrée autour d'un microprocesseur.



C'est ce dernier qui réalise les opérations logiques et arithmétiques. Il est toujours associé à de la mémoire vive (RAM) dont le rôle est de stocker des données temporaires (_____ _____) et de la mémoire morte (ROM) qui contient les données permanentes _____ _____).



Pour permettre le dialogue avec les périphériques hes de clavier, ports de communication : _____....) il est lié à un système 'açage d'entrée/sortie.

mme dans toute structure électrique, l'alimentation e fournir l'énergie nécessaire au fonctionnement.

ouche logicielle n'est pas visible en tant que telle, elle qui détermine les opérations à réaliser par les natérielles. Le logiciel est composé de "0" et de "1" mations binaires qui commandent les états internes du microprocesseur, mais pour simplifier l'élaboration du logiciel (le programme) , on dispose de langages évolués

(_____, _____, _____, _____....) plus ou moins spécialisés qui permettent de décrire plus facilement la réaction que doit avoir le microprocesseur face aux données à gérer...).

Pour transformer le langage évolué (qui n'est que du texte ou du graphique) en "0" et en "1", on fait appel à un ***compilateur***.

II/ PROGRAMMATION OBJET

La programmation objet trouve sa force dans le fait qu'un développeur utilise des objets préprogrammés, utilisables directement et qui permettent de gagner un temps considérable dans le développement. Ces objets sont en partie fournis par le SDK de windows (System Développement Kit). Ainsi si l'éditeur du système d'exploitation change la caractéristique d'un objet (passage de windows 2000 à windows XP par exemple) les applications developpées en objet ne sont pas à redévelopper.

Par ailleurs, en programmation objet les objets profitent d'un héritage :

Exemple :

un point sur l'écran est défini par sa position (x,y) et sa couleur

un trait hérite du point car il est une succession de point entre un point initial et un point final. Le code de création d'un trait est simplement une surcharge du code de création de point

un rectangle est un ensemble de 4 traits. Il hérite donc des codes du trait mais la surcharge de ce code permet de créer le rectangle, etc...

Ainsi chaque objet possède des propriétés héritées et d'autres propres mais aussi des méthodes spécifiques à l'objet.

III/ ALGORITHME ET ALGORIGRAMME (NORME NF Z 61-000)

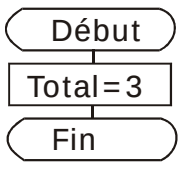
Afin de simplifier la compréhension d'un logiciel, les informaticiens décomposent les problèmes et utilisent des outils de description du logiciel.

L'algorithme est un outil textuel tandis que l'algorigramme est un outil graphique.

III.1/ Structure linéaire

Chaque portion de code de programme est délimité par un début et une fin. Entre ces deux mots sont décrit de la manière la plus simple les opérations à réaliser.

```
Début
    Le total est égal à 3
Fin
```




```
begin
    total:=3;
end;
```

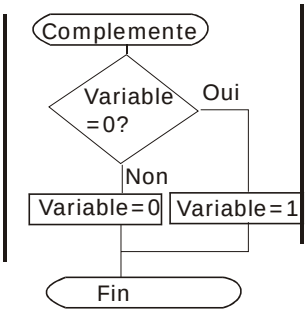
Remarque 1 : le mot début peut être remplacé parfois par le nom du programme ou sous-programme.

Remarque 2 : en langage pascal une ligne d'instruction se termine par un caractère ";"

III.2/ Structures alternatives : SI-ALORS-SINON

En fonction du résultat de la condition on réalise une opération ou l'autre.

```
Complemente
  SI variable est égale à 0
  ALORS
   variable devient égale à 1
  SINON
    Variable est égale à 0
  FINSI
Fin
```




```
Function complementer(variable:byte):byte;
Begin
      If variable=0
      Then variable:=1
      Else variable:=0;
      Result:=variable;
end;
```

Remarque1 : en cas de réponse négative à la condition, il n'est pas nécessaire d'avoir une opération (*SINON variable est égale à 0* peut ne pas exister)

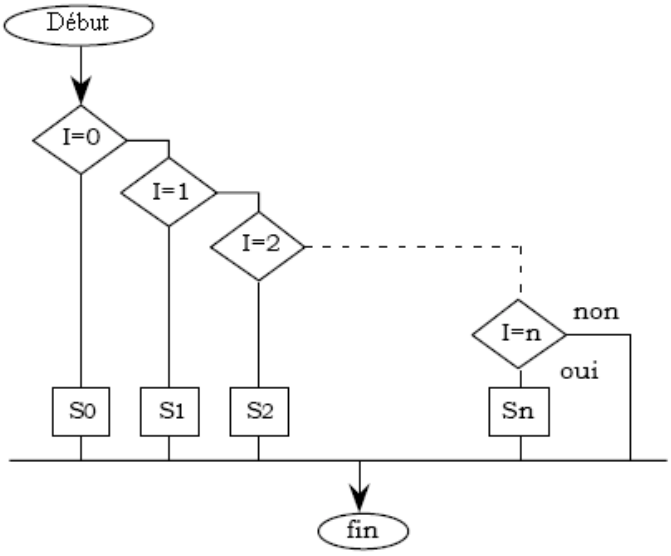
Remarque 2 : En PASCAL une ***function*** est un sous-programme qui restitue un résultat ***RESULT***. Dans notre exemple *complemente(1)* est égal à 0. Une procedure est également un sous programme mais qui ne rend pas de résultat.

Remarque 3 : En PASCAL, un paramètre est passé, entre parenthèse, dans une *function*. Dans notre exemple, supposons que J est égal à 0. En exécutant l'instruction *J:=complemente(J)*; J passera à "1" même si J n'est pas évoqué dans ***Function Complemente.***

III.3/ Structure alternative : SELON PARMI FAIRE

Cette forme de structure alternative n'est pas normalisée, mais existe dans beaucoup de langages informatique. Elle permet une écriture plus claire du SI ALORS SINON.

```
Début
      SELON I PARMI
            0: FAIRE S0
            1: FAIRE S1
            2: FAIRE S2
            ....
Fin
```




```
begin
      CASE I FOR
            0: S0;
            1: S1;
            2: S2;
            ....
      end;
end;
```

III.4/ Structure itérative POUR REPETER

Dans cette structure la sortie de la boucle d'itération s'effectue lorsque le nombre de répétitions est atteint.

```
Factoriel
      factoriel=1
POUR I=1 à X
 REPETER
  Factoriel=factoriel x I
FINPOUR
Fin
```

```
Function factoriel(x:integer):integer
     Var I,fact:integer;
Begin
     fact:=1;
     FOR I:=1 TO X
     DO fact:=fact*x;
     Result:=fact;
End;
```

III.5/ Structure itérative TANTQUE REPETER

La structure précédente sous entend que l'un connait le nombre de répétitions. Si ce n'est pas le cas on peut utiliser la boucle TANTQUE REPETER.

Le test se fait au début du bloc itératif.

```
DIV
      Quotient=0
      Total=0
TANTQUE Total<=Dividende
REPETER
   Total=Total+Diviseur
    Quotient=Quotien+1
FINREPETER
Fin
```

```
Function div(dividende,diviseur:integer):integer
     Var total,quotient:integer;
Begin
     Total:=0; Quotient:=0;
     WHILE Total<=dividende
     DO BEGIN
     Total:=Total+diviseur;
     Quotient:=Quotient+1;
     END;
     Result:=Quotient;
End;
```

III.6/ Structure itérative : REPETER JUSQU'A

Même principe que la boucle TANTQUE REPETER, masi cette fois ci le test se fait à la fin du bloc itératif

```
ATTENDRE_A

REPETER
  Afficher “Appuyez sur touche A”
 Lire(touche)
JUSQU’A touche=A
FINREPETER
Fin
```

```
Procedure Attendre_a;
    Var touche:char;
Begin
     REPEAT
          WRITELN(”Appuyer sur touche A”);
          READLN(touche)
     UNTIL touche=’A’
End;
```