LE LANGAGE "C"

Exemple

```
char val1=0xA5;                 // Déclaration d'une variable "caractère" avec valeur initiale
int val2;                       // Déclaration d'une variable "nombre entier"
...

void tempo(char temps)          // Fonctions et procédures appelées plusieurs fois dans le programme
                                   principal
        {
        ...
        }


void main(void)                 // Programme principal
        {
        DDRBA=0xFF              // initialisation et configuration
        ....
        while (1)               // Boucle principale
                {
                ...
                tempo(100);
                ...
                val2=bintobcd(val1);
                ...
                }
        }
```

Chaque ligne d'instruction se termine par un ";".

Le début d'une séquence est précédé du symbole "{".

La fin d'un séquence est suivie du symbole "}".

La notation des nombres peut se faire en décimal de façon normale ou en hexadécimal avec le préfixe "0x".

le symbole "//" est suivi d'un commentaire

le symbole "#" est suivi d'une directive de compilation,.

Les types de données du langage "C".

Il existe, dans le langage "C", plusieurs "types" de données classés selon leurs tailles et leurs représentations

On en rencontre généralement trois types qui peuvent être signés ou non signés. Dans ce dernier cas la déclaration sera précédée du mot clé "**unsigned**"

TYPE	DEFINITION	TAILLE	DOMAINE NON SIGNE	DOMAINE SIGNE
char	Variable de type caractère ascii mais utilisée pour les nombres entiers	8 bits	0 à 255	-128 à +127
int	Variable de type nombre entier	16 bits	0 à 65536	-32768 à 32767
float	Variable de type nombre réel	32 bits	+/- $3{,}4.10^{-38}$ à $3{,}4.10^{38}$	X

Les opérateurs

Les opérateurs arithmétiques.

Ces opérateurs permettent d'effectuer les opérations arithmétiques traditionnelles :
Addition, soustraction, multiplication et division entière.

OPERATEUR	FONCTION
+	Addition
-	Soustraction
*	Multiplication
/	Division entière
%	Reste de la division entière

Les opérateurs d'affectation.

L'opérateur indispensable au langage "C" est l'affectation défini principalement par le signe "**=**".
Il permet de charger une variable avec la valeur définie par une constante ou par une autre variable.
Il en existe d'autres qui, en plus de l'affectation, effectue une opération arithmétique.

OPERATEUR	FONCTION	EQUIVALENCE
=	Affectation ordinaire	`X=Y`
+=	Soustraire de _	`X+=Y équivalent à X=X+Y`
*=	Multiplier par _	`X*=Y équivalent à X=X*Y`
/=	Diviser par _	`X/=Y équivalent à X=X/Y`
%=	Modulo	`X%=Y équivalent à X=X%Y`
--	Soustraire de 1 (Décrémentation)	`X-- équivalent à X=X-1`
++	Ajouter 1 (Incrémentation)	`X++ équivalent à X=X+1`

Les opérateurs logiques.

Ces opérateurs s'adresse uniquement aux opérations de test conditionnel. Le résultat de ces opérations est binaire : "0" ou "1".

OPERATEUR	FONCTION
&&	ET logique
||	OU logique
!	NON logique

Les opérateurs logiques bit à bit.

Ces opérateurs agissent sur des mots binaires. Ils effectuent, entre deux mots, une opération logique sur les bits de même rang.

OPERATEUR	FONCTION
&	ET
|	OU
^	OU exclusif
~	NON
>>	Décalage à droite des bits
<<	Décalage à gauche des bits

Opérateurs de comparaison.

Ces opérateurs renvoient la valeur "0" si la condition vérifiée est fausse, sinon ils renvoient "1".

OPERATEUR	FONCTION
==	Egale à
!=	Différent de
>	Supérieur à
<	Inférieur à
>=	Supérieur ou égal à
<=	Inférieur ou égal

Les structures répétitives.

Le langage "C" possède des instructions permettant de répéter plusieurs fois une même séquence en fonction de certaines conditions.

Structure "while" : tant que ... faire ...

Avec ce type d'instruction le nombre de répétitions n'est pas défini et dépend du résultat du test effectué sur la condition. Si cette dernière n'est jamais vérifiée, la séquence n'est pas exécutée.

```
while (int x!=0)
      {
      ... ;
      }
```

La structure précédente répète la suite d'instruction comprises entre crochets tant que la variable entière "x" est différente de 0.

Structure "do ... while" : faire ... tant que...

Cette structure ressemble fortement à la précédente à la seule différence que la séquence à répéter est au moins exécuter une fois même si la condition n'est jamais vérifiée.

```
      do {
          ... ;
         }

      while (int x!=0);
```

Structure "for" : Pour <variable> allant de <valeur initiale> à <valeur finale> faire...

Cette instruction permet de répéter, un nombre de fois déterminé, une même séquence.

```
for (i=0;i<5;i++)
      {
      ... ;
      }
```

La structure précédente répète 5 fois la suite d'instruction comprise entre crochets. La variable "i" prendra les valeurs successives de : 0, 1, 2, 3 et 4.

Les structures alternatives.

Ces structures permettent d'exécuter des séquences différentes en fonction de certaines conditions.

Structure "if ... Else" : Si <condition> faire ... sinon faire ...

Avec cette structure on peut réaliser deux séquences différentes en fonction du résultat du test sur une condition.

```
if (a<b) c=b-a;
else c=a-b;
```

La structure précédente affecte la valeur "b-a" à "c" si "a" est inférieur à "b" sinon "c" est affecté par la valeur "a-b".

```
if (RB0==0)          //test entrée
{
  RB1=1 ;                  //affectation sortie à 1
  RB3=0 ;                  //affectation sortie à 0
}
else RB3=1;
```

Structure "switch ... case".

Cette structure remplace une suite de "if ... else if ...else" et permet une de réaliser différentes séquences appropriées à la valeur de la variable testée.

```
switch (a)
      {
      case '1' : b=16;
      case '2' : b=8;
      case '3' : b=4;
      case '4' : b=2;
      }
```

Dans la structure précédente "b=16" si "a=1", "b=8" si "a=4" etc.

REPRESENTATION STRUCTUREE

1 - Définition

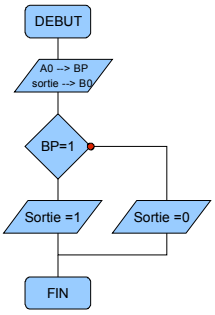
Le langage algorithmique est l'outil de description temporelle du fonctionnement d'un automatisme comme le GRAFCET. L'algorithme est une suite ordonnée d'actions permettant d'obtenir un résultat déterminé.

2 – Représentation des algorithmes

Deux modes de représentations sont possibles :

- Littérale
- Graphique

Exemple de ces deux modes de représentations :

littérale	Graphique : Algorigramme
DEBUT SI BP = 1 ALORS MOTEUR = 1 FIN SI FIN	

3 – Représentation littérale

Le langage algorithme comporte trois familles de mots :

3 – 1 Les mots délimitateurs

Ils fixent les bornes d'entrée et de sortie de l'algorithme et des structures utilisées dans l'algorithmes**. DEBUT** et **FIN** sont les deux seuls mots déliminateurs autorisés ; ils peuvent être éventuellement suivis d'un mot clé.

3 – 2 Les mots clés

Ils précisent le type de structure algorithmique utilisée. Exemple de mots clés

structure alternative : **SI ... ALORS … SINON**
structure itérative : **REPETER … JUSQU'A**

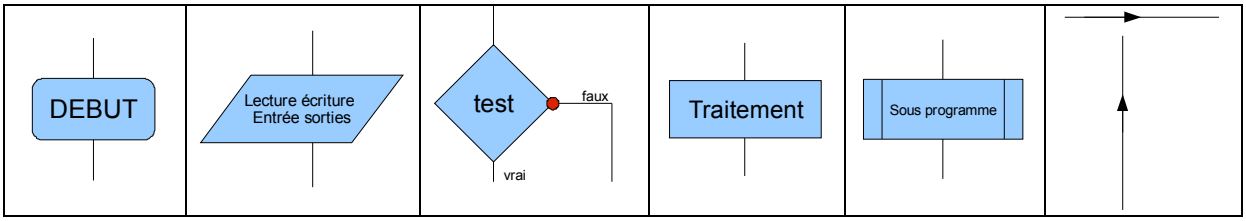
Un mot clé est toujours suivi :
soit d'une expression conditionnelle
soit d'un ou plusieurs mots instructions.

3 – 3 Les mots instructions

Ils caractérisent l'action à appliquer à l'objet et sont toujours suivis de la désignation de l'objet sur lequel l'action s'applique.

4 – Représentation graphique : Algorigramme

L'ensemble des structures de l'organigramme se représente sous la forme de symboles normalisés.

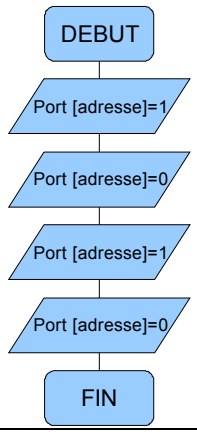


5 – Les structures algorithmiques de bases

5 – 1 Structure linéaire

La structure linéaire est simplement décrite par une suite d'actions à exécuter successivement dans l'ordre énoncé.

Exemple : rotation d'1 pas du moteur dans le sens horaire

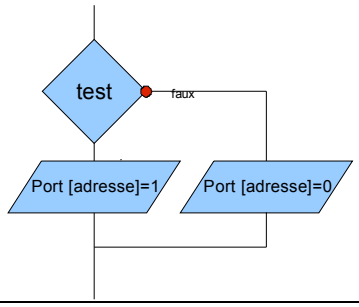
Littérale	**Algorigramme**
DEBUT Port [adresse] ← 1 Port [adresse] ← 0 Port [adresse] ← 1 Port [adresse] ← 0 FIN	

5 – 2 Structure alternative

Par alternative on désigne toute situation n'offrant que deux issues possibles s'excluant mutuellement.

Dans cette structure, l'exécution d'un des deux traitements ne dépend que du résultat d'un test effectué sur une grandeur variable ou un événement appelé **condition**.

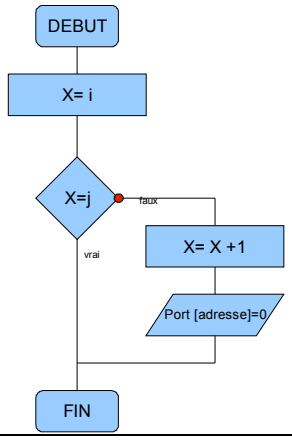
Si la condition est vérifiée, le premier traitement est effectué, dans le cas contraire c'est le deuxième traitement qui est exécuté.

Littérale	Organigramme
SI condition ALORS traitement 1 SINON traitement **2** IF condition THEN action ELSE action ENDIF	

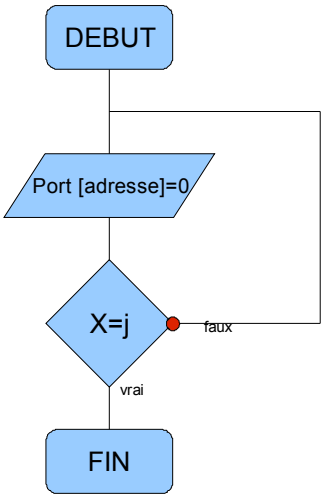
5 – 3 Structure itérative

Par itération, on désigne toute répétition de l'exécution d'un traitement (on utilise aussi le terme de boucle), il existe trois types de structures itératives.

Structure POUR … REPETER : on effectue un nombre de fois connu le traitement.

Littérale	Organigramme
DEBUT Initialisation X = i **POUR** X de i **JUSQU'**A j par n **REPETER** (action ou traitement) FIN FOR X = … TO … DO action ENDFOR	

Structure REPETER … JUSQU'A : le traitement est effectué une première fois puis sa répétition se poursuit jusqu'à ce que la condition d'entrée soit vraie.

Littérale	Organigramme
DEBUT **REPETER** (action ou traitement) **JUSQU'A** (condition réalisée) FIN REPEAT action UNTIL condition ENDREPEAT	

Structure TANT QUE … REPETER : On effectue le traitement tant que la condition d'entrée est vérifiée. Contrairement à la structure « répéter jusqu'à » le test de la condition d'entrée est effectué avant de réaliser le premier traitement.

Littérale	**Organigramme**
DEBUT **TANT QUE** (conditions à réaliser) **REPETER** (action ou traitement) FIN WHILE condition DO action ENDWHILE