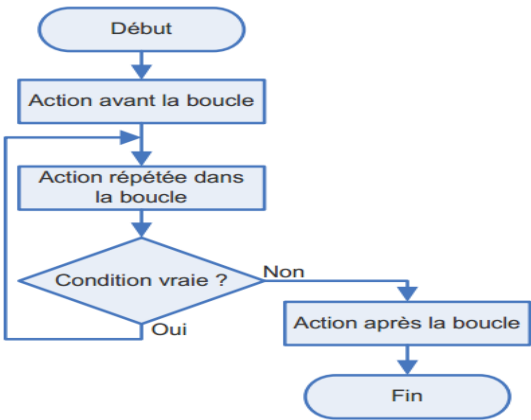


	Innovation et Développement Durable		1 ^{ère} STI2D
	Structures algorithmiques		
	Séquence 3 : Solution constructive	Cours	I2D

1. Structures algorithmiques.

L'en-tête	{ <i>Algorithme</i> NomAlgorithme
La partie déclarative	{ Constantes Identificateur=valeur ordonnant à un processeur de
	{ Variables Identificateur : type	réaliser un nombre de tâches dans un ordre précis pour
Le corps de l'algorithme	Début	résoudre un problème technique donné.
	Instruction1	La structure des langages de programmation connus comme Arduino, C, C++ , Python,
	Instruction2	
	
	Instruction2	
	Fin

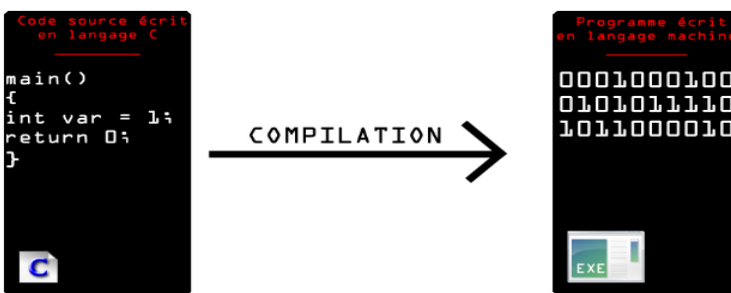
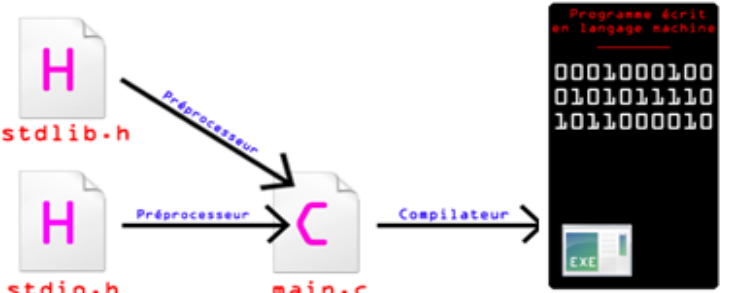
2. Définition Algorithme

	<p>L'algorithme</p> <p>.....</p> <p>Le rôle d'un algorithme est pour les personnes qui ne comprennent pas le code ou l'algorithme.</p>
--	--

3. Langage machine

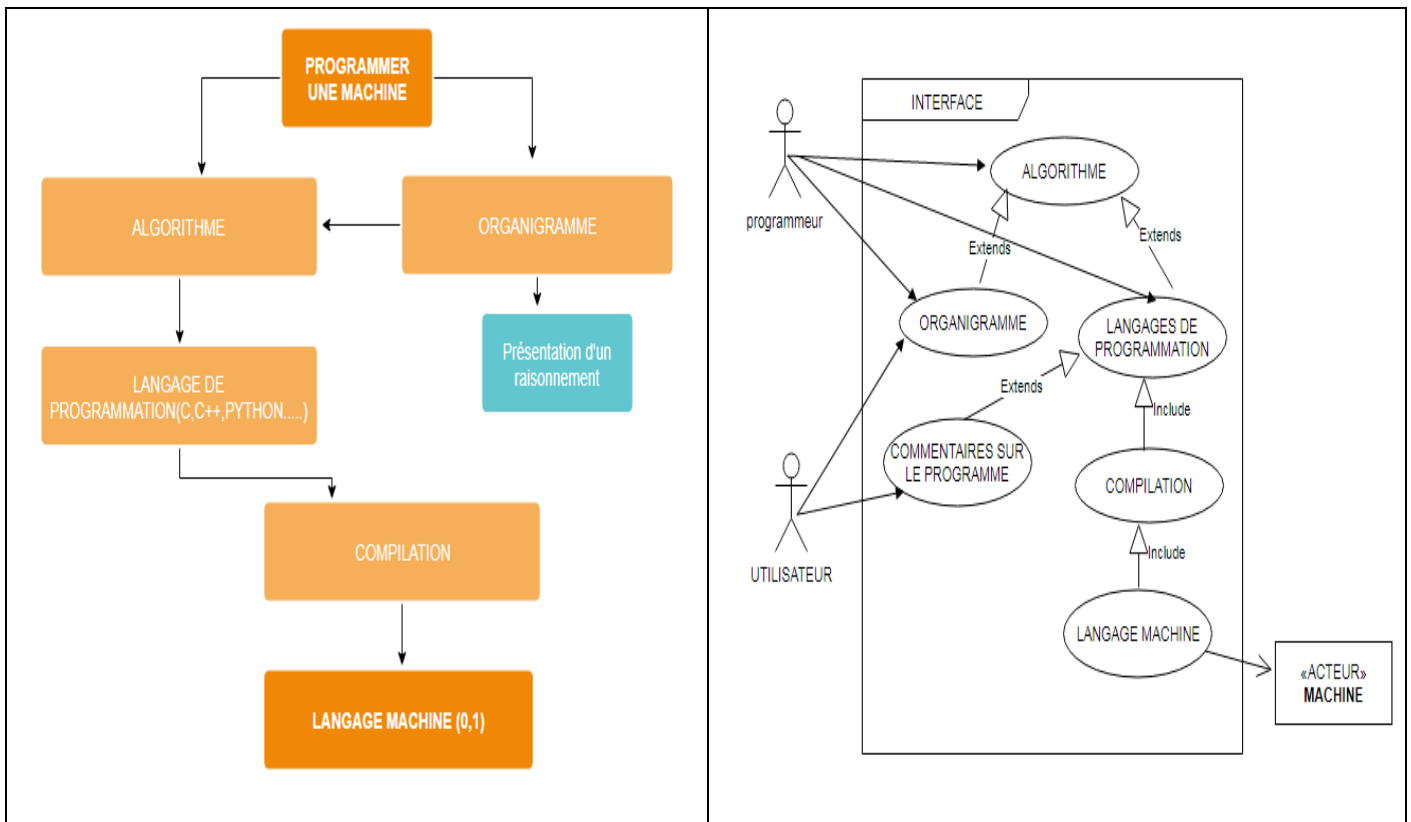
..... comme par exemple **00110000110000** .

Nous, en tant qu'humain, nous utilisons des langages qui permettent de parler à l'ordinateur,

	<p>A gauche, le c'est le contenu d'un fichier.</p> <p>A droite le code exécutable</p> <p>.....</p> <p>.....</p>
	<p>On peut aussi ajouter les</p> <p>comme le cas utilisés dans la programmation ARDUINO</p>

4. Présentation d'un programme informatique .

Le programmeur d'une machine doit forcément rédiger pour présenter le fonctionnement de son programme. Il doit donc utiliser



5. Structure d'un programme Linéaire.

<p>Algorithme</p> <pre> graph TD Debut([Début]) --> Action1[Action 1] Action1 --> Action2[Action 2] Action2 --> Fin([Fin]) </pre>	<p>Notation algorithmique</p> <pre> Début ↓ Action 1 ↓ Action 2 ↓ FIN </pre>	<p>Dans un programme dans l'ordre de leurs énoncés.</p>
---	---	--

6. Structure d'un programme itérative.

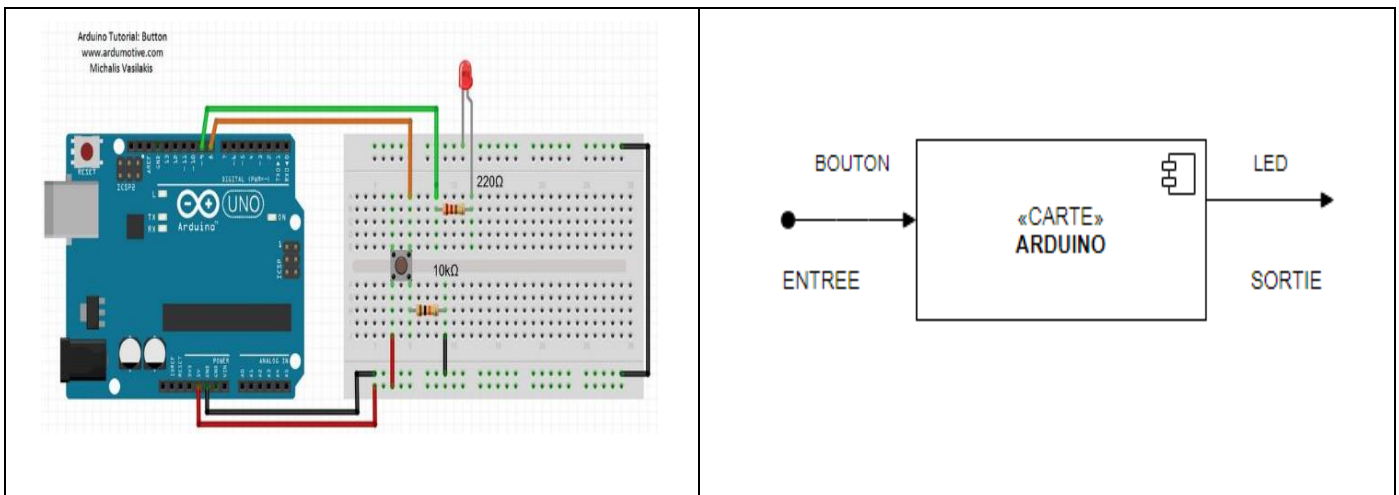
Il existe plusieurs façons de réaliser une Itérative ou alternative:

6.1. Structure SI->ALOR->SINON

ALGORITHME	ALGORIGRAMME	LANGGAGE ARDUINO
<p>Si condition Alors instruction 1 Sinon instruction 2 Fin Si</p>		<pre> if(condition1){ //instruction1 } else if(condition2){ //instruction2 } </pre>

Exemple1 : Soit le schéma structuel page suivante. Lorsqu'on appuie sur l'interrupteur, la diode électroluminescente doit s'allumer sinon la diode reste éteinte.

L'interrupteur est relié a la broche 8 . La LED est reliée a la broche 9



ALGORITHME	ALGORIGRAMME	LANGGAGE ARDUINO
Début Déclarer la broche 8 en entrée Déclarer la broche 9 en sortie Début de la boucle loop Lire la valeur de la broche 8 Si la broche 8=1 Alors envoyer 1 a la broche 9 Sinon envoyer 0 a la broche 9 Fin SI Fin loop Fin	<pre> graph TD Start(()) --> Decl8([Déclarer la broche 8 en entrée]) Decl8 --> Decl9([Déclarer la broche 9 en sortie]) Decl9 --> Read8([Lire la broche 8]) Read8 --> Cond8{Si la broche 8=1} Cond8 -- oui --> Send1([Envoyer 1 à la Broche 9]) Cond8 -- non --> Send0([Envoyer 0 à la Broche 9]) Send1 --> End(()) Send0 --> End </pre>	<pre> void setup() { pinMode(8, INPUT); pinMode(9, OUTPUT); } void loop() { int broche8 = digitalRead(8); if (broche8 == 1) {digitalWrite(9,HIGH);} else{ digitalWrite(9,LOW);} } </pre>

Q1- Réaliser un algorithme qui permet de faire clignoter la led chaque 1000 millisecondes si on appuie sur le bouton, sinon le clignotement se fait chaque 2000 millisecondes.

6.2. Structure TANT QUE ->FAIRE.

ALGORITHME	ALGORIGRAMME	LANGGAGE ARDUINO
Tant que condition vraie Séquence Fin tant que	<pre> graph TD Entry(()) --> Cond{Condition} Cond -- vraie --> Seq[Sequence] Seq --> Cond Cond -- fausse --> Exit(()) </pre>	<pre> while (condition) { Séquence ; } </pre>

Exemple : Réalisation de l'exemple 1 en utilisant la boucle while au lieu de la condition if

ALGORITHME	ALGORIGRAMME	LANGGAGE ARDUINO
Début de la boucle loop Lire la broche 8 Tant que la broche 8=1 Envoyer 1 à la broche 9 Lire la broche 8 Fin tant que Envoyer 0 a la broche 9 Fin de la boucle loop	<pre> graph TD Start([Début de loop]) --> Read8([Lire la valeur de la broche 8]) Read8 --> Decision{Broche 8=1} Decision -- oui --> Send1([Envoyer 1 a la broche 9]) Send1 --> Read8 Decision -- non --> Send0([Envoyer 0 a la broche 9]) Send0 --> End([Fin de loop]) </pre>	<pre> void loop() { broche8 = digitalRead(8); while (bouton == 1) { digitalWrite(9, HIGH); bouton = digitalRead(8); } digitalWrite(9, LOW); } </pre>

Q2- Réaliser un algorithme qui permet de faire clignoter la led chaque 1000 millisecondes si on appuie sur le bouton, sinon le clignotement se fait chaque 2000 millisecondes avec la boucle while.

6.3. Structure POUR->FAIRE.

ALGORITHME	ALGORIGRAMME	LANGGAGE ARDUINO
Pour i = 0 à N Faire Séquence Fin pour.	<pre> graph TD Init[i = 0] --> Decision{i <= N} Decision -- vraie --> Seq[Séquence] Seq --> Inc[i = i + 1] Inc --> Decision Decision -- fausse --> Exit[] </pre>	<pre> for(initialisation; condition; incrementation){ //instruction } </pre>

Exemple2: Le programme ci-dessous affiche sur le moniteur série les valeurs de 0 à 10 chaque 1000 millisecondes.

ALGORITHME	ALGORIGRAMME	LANGGAGE ARDUINO
<p>Début de la boucle loop</p> <p>Pour i = 0 à 10</p> <p> Afficher i</p> <p> Attendre 1000 ms</p> <p>Fin pour.</p> <p>Fin de la boucle loop</p>	<pre> graph TD Start([Début de loop]) --> Init(i=0) Init --> Decision{i <= 10} Decision -- oui --> Display[Afficher i] Display --> Increment[i=i+1] Increment --> Delay[Attendre 1000 ms] Delay --> Decision Decision -- non --> End([Fin de loop]) </pre>	<pre> void loop() { for(int i=0; i<=10;i=i+1){ Serial.println(i); delay(1000); } } </pre>

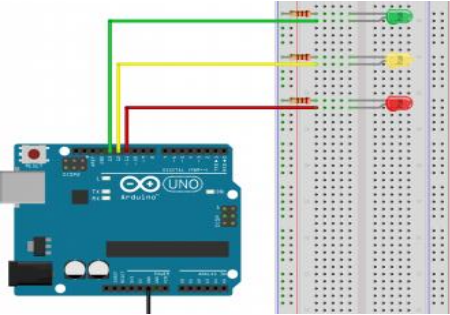
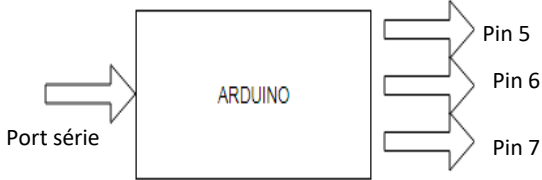
Q3 : Réaliser un programme qui affiche dix fois « Bonjour ». Avec la boucle for ensuite la boucle While.

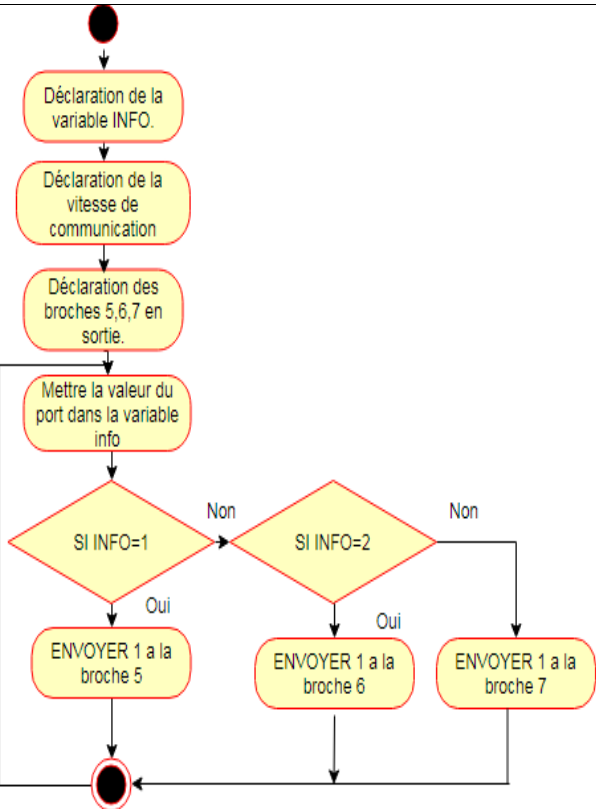
6.4. Structure MULTI-DECISIONS

Dans le cas d'un « switch ... case », la condition est remplacée par une expression évaluée par le switch. Chaque case mentionne la valeur de l'expression pour laquelle ses instructions doivent être exécutées

ALGORITHME	ALGORIGRAMME	LANGGAGE ARDUINO
<p>Si condition1 alors instruction1</p> <p>Si condition2 alors instruction2</p> <p>Sinon condition3 alors instruction3</p>		<pre> void loop() { switch (expression) { case valeur1: // instructions si expression est égale à valeur1 break; case valeur2: // instructions si expression est égale à valeur2 break; case valeur3: // instructions si expression est égale à valeur3 break; default: // instructions si expression est différente break; } } </pre>

Exemple : On allume les diodes (rouge, jaune) selon les chiffres 1, 2 que l'on envoie à la carte arduino sur le port sérié et si on envoie rien, on allume la diode verte.

Schéma Arduino	Entrées Sorties
	

Algorithme	Programme Arduino
 <pre> graph TD Start(()) --> DeclInfo[Déclaration de la variable INFO.] DeclInfo --> DeclSpeed[Déclaration de la vitesse de communication] DeclSpeed --> DeclPins[Déclaration des broches 5,6,7 en sortie.] DeclPins --> SetInfo[Mettre la valeur du port dans la variable info] SetInfo --> Dec1{SI INFO=1} Dec1 -- Oui --> Send5[ENVOYER 1 a la broche 5] Dec1 -- Non --> Dec2{SI INFO=2} Dec2 -- Oui --> Send6[ENVOYER 1 a la broche 6] Dec2 -- Non --> Send7[ENVOYER 1 a la broche 7] Dec2 -- Non --> Dec3{SI INFO=0} Send5 --> End(()) Send6 --> End Send7 --> End Dec3 --> End </pre>	<pre> int info; void setup() { Serial.begin(9600); pinMode(5, OUTPUT); pinMode(6, OUTPUT); pinMode(7, OUTPUT); } void loop() { info=Serial.parseInt(); switch (info) { case 1: digitalWrite(5, HIGH); break; case 2: digitalWrite(6, HIGH); break; default: digitalWrite(7, HIGH); break; } } </pre>

Q4. Ecrire l'algorithme de ce programme.

Q5. Modifier l'algorithme pour éteindre les leds si l'on n'envoie rien à la carte.

7. Les fonctions.

Une fonction (également désignée sous le nom de procédure ou de sous-routine) est un bloc d'instructions que l'on peut appeler à tout endroit du programme.

Le langage Arduino est constitué d'un certain nombre de fonctions, par exemple `analogRead()`, `digitalWrite()` ou `delay()`.

Il est possible de déclarer ses propres fonctions, par exemple :


```
void clignote() {  
    digitalWrite (brocheLED, HIGH);  
    delay (1000);  
    digitalWrite (brocheLED, LOW);  
    delay (1000);  
}
```

Pour exécuter cette fonction, il suffit de taper la commande :

```
clignote();
```

Le programme aura la forme suivante :

```
int brocheLed=13;  
  
void setup() {  
    pinMode (13, OUTPUT);  
}  
  
void loop() {  
    clignote();  
}  
  
void clignote() {  
    digitalWrite (brocheLED, HIGH);  
    delay (1000);  
    digitalWrite (brocheLED, LOW);  
    delay (1000);  
}
```



Appel de la fonction clignote

7.1. Paramètres des fonctions

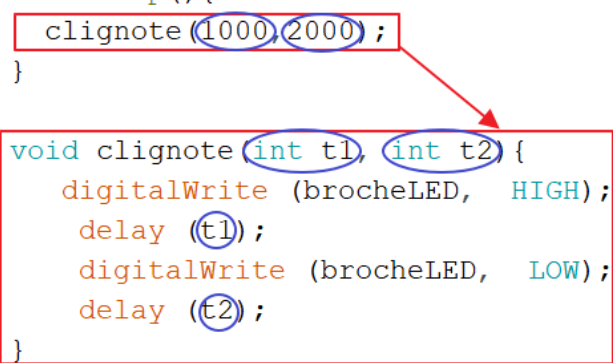
On peut également faire intervenir un ou des **paramètres** dans une fonction.
Sur la fonction ci-dessous on fait intervenir deux valeurs de temps sur la fonction clignote.


```
int brocheLed=13;

void setup() {
  pinMode(13,OUTPUT);
}

void loop() {
  clignote(1000,2000);
}

void clignote(int t1, int t2) {
  digitalWrite(brocheLED, HIGH);
  delay(t1);
  digitalWrite(brocheLED, LOW);
  delay(t2);
}
```



Q6 : De combien de temps la led s'allume et s'éteint ?

Q7 : Ecrire une fonction appeler afficher avec un paramètre nombre entier Int, appelée nombre.

Appeler cette fonction dans le fonction principal loop une fois pour afficher « 1 » et une deuxième fois pour afficher « 2 ».

7.2. Variable globale et variable locale

Une variable globale est une variable qui peut être "vue" et utilisée par n'importe quelle fonction dans un programme. Les variables locales ne sont visibles que dans la fonction dans laquelle elles ont été déclarée. Dans le langage Arduino, toute variable déclarée en dehors d'une fonction (telle que `setup()` ou `loop()`), au niveau de l'entête du programme par exemple, est une variable globale.

```
int a=5; // n'importe quelle fonction pourra voir cette variable qui est globale

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int b=6; //Uniquement la fonction loop peut voir cette variable qui est locale dans loop()
  affiche();
}

void affiche(){
  Serial.println(a); //cette ligne ne présente pas d'erreur ✓
  Serial.println(b); //cette ligne présente une erreur ✗
}
```

8. Opérateurs logiques sur Arduino.

Ces opérateurs peuvent être utilisés à l'intérieur de la condition d'une instruction **if** pour associer plusieurs conditions.

8.1. && (ET logique)

VRAI seulement si deux opérandes sont VRAI ,par exemple.

```
int a, b, c;

if( (a==100) && (b==100) && (c==100) )
{
  Serial.print("les trois valeurs sont à 100");
}
```

1 2 3

Il est recommandé d'utiliser les parenthèses : **(a>100) && (b>100) && (c>100)**

8.2. || (OU logique)

VRAI si un des opérateurs est vrai, par exemple

```
int a, b, c;  
  
if((a==100) || (b==100) || (c==100))  
{  
  Serial.println("en moins une valeur est à 100")  
}
```

Q8. Simplifier la définition des conditions

```
if (a>100){  
  if b>100{  
    if c<50{  
      Serial.println("Opération Valide")  
    }  
  }  
}
```

9. Les opérations de comparaison.

!= : non égal
< > : inférieur à / supérieur à
<= : inférieur ou égal
>= : supérieur ou égal
== : égal à