**Solutions des exercices**

**Exercice : Créez un son de type 'Donkey-Kong'**

Solution :

int p=1000; // on prépare une variable pour la période

void setup() {

pinMode(3,OUTPUT); //on prépare le pin en mode sortie

}

void loop() {

p-=10; // on incémente p à chaque boucle de loop();

if (p<100){ // teste la limite de p

p=1000; // on le remet à 1000

delay(500); // on attend un peu

}

digitalWrite(3,0); // état bas

delayMicroseconds(p); //on attend p milli-secondes

digitalWrite(3,1); // état haut

delayMicroseconds(p); // on attend p millisecondes

}

 Ici le son va du grave vers l'aigu.

**Exercice : Affichez un tableau des codes ASCII de 32 à 127 et leurs caractères correspondants (variante 1)**

Solution  :

void setup() {

Serial.begin(9600);

for (int c=32;c<128;c++){ // boucle de 32 à 128

Serial.print(c); // affichage du nombre

Serial.print(" "); // espace

Serial.println(char(c)); // affichage du caractère

}

}

void loop() {

}

Ce petit code est toujours pratique si on a besoin de retrouver la table ASCII de l'Arduino.

**Exercice : Affichez un tableau des codes ASCII de 32 à 127 et leurs caractères correspondants (variante 2 avec tabulation)**

 Solution :

void setup() {

Serial.begin(9600);

for (int c=32;c<128;c++){ // boucle de 32 à 127

Serial.print(c); // on affiche le code

Serial.print('\t'); // on affiche une tabulation

Serial.print(char(c));// on affiche le caractère

Serial.print('\n'); // on finit la ligne

}

}

void loop() {

}

**Exercice : Concevez un programme qui inverse les lettres d'un mot**

Solution :

char mot[]="endroit"; // chaîne de 7 caractère + NULL

void setup() {

Serial.begin(9600); // début communication série

Serial.println(); // on saute une ligne

char resultat[8]; // on prépare un tableau de 8 (7+NULL)

for (int l=6;l>=0;l--) // on parcours mot[] à l'envers (7 lettres donc de 0 à 6)

resultat[6-l]=mot[l]; // on inscrit dans resultat[] en inversant les positions (6-6=0, 6-5=1, ...)

resultat[7]='\0'; // on ajoute le caractère NULL en fin de chaîne

Serial.println(resultat); // on affiche le résultat.

}

void loop() {

}

 Le programme est légèrement plus délicat si on ne connaît pas la fin de la chaîne. Je vous donne un exemple de code (ce n'est pas forcément la seule façon de procéder) :

char mot[]="gloubiboulga"; // chaine de taille inconnue

void setup() {

Serial.begin(9600); // initialisation de la communication série

Serial.println(); // on saute une ligne

int t=0; // variable pour compter la taille de la chaîne

while (mot[t]) // tant qu'on n'est pas au bout (caractère NULL)

t++; // on incrémente le comptage

char resultat[t+1]; //on prépare une chaîne de la bonne taille (taille de la chaîne + caractère NULL)

for (int l=0;l<t;l++) // on parcourt la chaîne (de 0 à t-1)

resultat[l]=mot[t-1-l]; // on place les caractères dans le sens inverse

resultat[t]='\0'; // on fixe le dernier caractère à NULL

Serial.println(resultat); // on inscrit le résultat

}

void loop() {

}

Alors je vous conseille de bien chercher à comprendre les questions de taille de chaîne (longueur du mot), taille du tableau (longueur du mot + 1 pour le caractère NULL), de position dans le tableau (le tableau commence à 0 et finit à taille-1), la position du caractère NULL (situé à la position taille, puisque le tableau commence à 0).

Je dirais que c'est le plus délicat à réaliser dans le traitement des chaînes de caractères.

Les fonctions Serial.print() et Serial.println() sont des fonctions qui envoient par le port série une chaîne de caractères. Elles procèdent comme on vient de le faire, c'est-à-dire qu'elles envoient un caractère par un caractère !

Le port série récupère les données une à une et les écrit. Le saut de ligne est un ajout d'un '\l' (et parfois d'un '/r' ) en bout de chaîne.

**Exercice  : Faites clignoter une LED avec la fonction millis()**

La solution avec un petit plus : j'utilise un booléen pour l'état de la LED.

/\*

\* Matrice de LED 3 lignes, 3 colonnes

\* pin 2 = ligne 0, pin 3 = ligne 1, pin 4 = ligne 2

\* pin 5 = colonne 0, pin 6 = colonne 1, pin 7 = colonne 2

\*/

boolean etat = 1; // boléen pour l'état de la LED

unsigned long tpsDep = millis(); // variable pour le temps de départ

void setup() {

//initialisation des pins en OUTPUT et mise à LOW

for (int l = 2; l < 8; l++) {

pinMode(l, OUTPUT);

digitalWrite(l, LOW);

}

//on passe les pins de colonne à HIGH

for (int l = 5; l < 8; l++) {

digitalWrite(l, HIGH);

}

}

void loop() {

unsigned long tpsAct = millis(); // variable pour le temps actuel

if (tpsAct - tpsDep > 500) { // si le temps dépasse 500ms

etat = !etat; //on inverse l'état de la LED

tpsDep = tpsAct; //on réinitialise le temps de départ

digitalWrite(2, etat); //on place le pin ligne de la LED comme etat

digitalWrite(5, !etat); //on place le pin colonne de la LED à l'opposé d'état

}

}

Là encore, il est important de comprendre le programme.

Notez que j'aurais très bien pu me contenter de faire passer seulement le pin 2 de LOW à HIGH à chaque fois. Mais le principe de gestion des deux pins (ligne et colonne) a toute son importance pour la suite.

**Exercice : Gérez la matrice de LED avec un tableau à simple entrée**

Solution :

char matrice[10]="eeaeeeaee"; //tableau de char

byte ligne[3] = {4, 3, 2}; //stockage des pins ligne

byte colonne[3] = {5, 6, 7}; // stockage des pins colonne

void setup() {

//initialisation des pins en OUTPUT et mise à LOW ou HIGH

for (byte l = 2; l < 8; l++) {

pinMode(l, OUTPUT);

if (l > 1 && l < 4)

digitalWrite(l, LOW);

else

digitalWrite(l, HIGH);

}

}

void loop() {

affiche(); //appel de la fonction d'affichage

}

// fonction d'affichage

void affiche() {

for (byte p=0;p<9;p++){//parcours du tableau matrice

if (matrice[p]=='a'){ //test si 'a'

//allumage

digitalWrite(ligne[p/3],HIGH);

digitalWrite(colonne[p%3],LOW);

delay(2);

//exctinction

digitalWrite(colonne[p%3],HIGH);

digitalWrite(ligne[p/3],LOW);

}

}

}

 Ceux qui sont observateurs, remarquerons que ce programme ne gagne du temps que si peu de LED sont allumées en même temps. En effet, si toutes le sont, on se retrouve avec un appel à la fonction digitalWrite() de 9\*4=36 fois...