	Spécialité SIN	STI2D
		Terminale
Arduino – Exercices de programmation en langage C		TP

sNOM : _____ PRÉNOM : _____ DATE : _____

Compétences visées	Connaissances visées
CO3.1 - Être capable d'appliquer les structures de traitement (séquentielle, conditionnelles, itératives) pour réaliser le cahier des charges d'un programme CO4.1 - Être capable d'écrire un algorithme d'après un cahier des charges CO4.2 - Être capable de représenter l'algorithme d'une solution logicielle. CO5.8 - Être capable d'utiliser un langage compilé et/ou interprété (C/Python) et leur environnement de développement. CO5.8 - Être capable d'appliquer les structures de traitement (séquentielle, conditionnelles, itératives) pour réaliser le cahier des charges d'un programme	1.2.2c Analyse du besoin : besoin initial, mission principale, contexte, cas d'utilisations, scénarios d'utilisation, besoins des parties prenantes. 2.4.3b Algorithmique. 2.4.5 Organisation structurelle d'une application logicielle : (programme principal, interfaces, entrées-sorties, sous programmes, procédures, fonctions). 4.3.5b Codage dans un langage spécifique. Règles d'écriture (organisation du code, commentaires, documentation…). 4.3.5c Mise au point

1 – Tables de multiplication.

Écrivez un programme qui affiche les tables de multiplication de 0 à 10. Ce programme devra faire afficher, dans le moniteur série de l'IDE Arduino, les tables de multiplication sous le format ci-contre.

Pour cela, vous utiliserez deux boucles « for » imbriquées.

Testez votre programme pour valider son fonctionnement.

```
2 * 9 = 18
2 * 10 = 20
----------
3 * 0 = 0
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
----------
4 * 0 = 0
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
```

2 – Calcul de carrés.

2.1 – Après avoir consulté la documentation en ligne (adresse : https://www.arduino.cc/reference/fr/), **expliquez** à quoi sert la méthode Serial.parseInt () :

2.2 – Dans l'IDE Arduino, **saisissez** le squelette de programme ci-contre. **Complétez** le programme (cadre en jaune) pour obtenir l'affichage ci-contre (celui-ci dépend évidemment des entiers saisis par l'utilisateur lorsque le programme est exécuté ; ici, les valeurs saisies étaient 2, puis 22, puis 18).

```
1  // Calcul de carres
2
3  void setup() {
4    Serial.begin (9600) ;
5  }
6
7  void loop() {
8    int nombre ;
9    if (Serial.available ()) {
10     nombre = Serial.parseInt () ;
11
12
13
14   }
15 }
```

```
2 au carre = 4
22 au carre = 484
18 au carre = 324
```

2.3 – Testez votre programme avec le jeu de tests suivant :

Nombre saisi	Valeur attendue en réponse	Valeur obtenue en réponse
1	1	
-5	25	
128	16384	
500	250000	

Votre programme fournit-il les réponses attendues ? **Expliquez** pourquoi :

3 – Parité.

Écrivez un programme qui détermine si un entier saisi est pair ou impair. Vous utiliserez le même squelette de programme que pour l'exercice 2.

4 – Échange des valeurs de deux variables.

Saisissez le squelette de programme fourni ci-contre.

4.1 – Complétez la portion manquante (en jaune) par les deux lignes suivantes :

a = b ;
b = a ;

Testez le programme. **Indiquez** le résultat affiché :

```
// Echange de variables

void setup() {
  int a, b, tmp ;

  Serial.begin (9600) ;

  // Initialisation de deux variables entieres
  a = 1984 ;
  b = 1515 ;

  // Echange des valeurs de a et b


  // Affichage des valeurs des variables apres echange
  Serial.print ("a = ") ;
  Serial.println (a) ;
  Serial.print ("b = ") ;
  Serial.println (b) ;
}

void loop() {
}
```

Les valeurs de deux variables a et b ont-elles été échangées ? **Expliquez** pourquoi :

4.2 – Modifiez le programme (partie en jaune) pour que les valeurs des variables a et b soient correctement échangées. Pour cela vous utiliserez la variable tmp. **Écrivez** en langage C et en pseudo-code la solution que vous avez trouvée :

5 – Diviseurs.

Écrivez un programme qui affiche dans le moniteur série tous les diviseurs d'un nombre entier saisi (plus grand que 1). Vous utiliserez le même squelette de programme que pour l'exercice 2.

Rappels :

- *L'entier k est un diviseur de l'entier n si le reste de la division entière de n par k est égal à zéro.*
- *En langage C, l'opérateur '%' fournit le reste de la division entière du premier opérande par le deuxième.*

6 – Compte à rebours.

6.1 - Écrivez un programme qui demande à l'utilisateur de saisir un entier, celui-ci indiquant un nombre de secondes. Une fois cet entier saisi, votre programme devra afficher sur le moniteur série un compte à rebours seconde par seconde. Vous utiliserez le même squelette de programme que

pour l'exercice 2. Vous utiliserez la fonction delay() pour rythmer le décompte.

6.2 – Modifiez votre programme pour qu'il décompte :

- les dizaines de secondes s'il en reste plus de 10 à décompter ;
- les secondes s'il en reste 10 ou mois à décompter (voir exemple de décompte ci-contre).

Testez votre programme pour valider son fonctionnement.

7 – Bits de poids faible et fort.

Écrivez un programme qui affiche le bit de poids fort (MSB) puis le bit de poids faible (LSB) d'un entier positif (type unsigned int) saisi. Vous utiliserez le même squelette de programme que pour l'exercice 2, en modifiant le type de la variable nombre en ligne 8.

Testez votre programme avec le jeu de tests suivant pour valider son fonctionnement :

Nombre saisi	Valeur attendue en réponse	Valeur obtenue en réponse
1	MSB = 0 LSB = 1	
100	MSB = 0 LSB = 0	
32768	MSB = 1 LSB = 0	
32771	MSB = 1 LSB = 1	

8 – Moyenne.

8.1 - Écrivez un programme de calcul de moyenne de notes. Celui-ci devra :

- Lire des entiers saisis par l'utilisateur. Vous utiliserez le même squelette de programme que pour l'exercice 2.
- Après chaque lecture d'un entier, calculer somme de tous les entiers saisis, depuis le lancement du programme et stocker cette somme dans une variable globale nommée somme.
- Compter le nombre de notes saisies, et stocker ce compte dans une variable globale nommée coef.
- Après chaque lecture d'un entier, afficher la moyenne arithmétique de tous les entiers saisis depuis le lancement du programme. Un exemple d'affichage dans le moniteur série est fourni ci-contre.

```
10
Moyenne = 10.00
5
Moyenne = 7.50
15
Moyenne = 10.00
20
Moyenne = 12.50
0
Moyenne = 10.00
```

Rappel : pour c notes dont la somme vaut s, la moyenne est donnée par le quotient s / c.

8.2 – Nous supposons maintenant que les valeurs saisies par l'utilisateur sont des notes scolaires entre 0 et 20. **Calculez** le nombre de notes dont l'on pourra (à coup sûr) faire la moyenne dans les configurations suivantes (voir la page d'aide https://www.arduino.cc/reference/en/#data-types) :

- Si la variable somme est déclarée ainsi : int somme ;
- Si la variable somme est déclarée ainsi : long somme ;
- Si la variable somme est déclarée ainsi : unsigned long somme ;

Expliquez votre calcul. **Expliquez** ce qu'il risquerait de se produire si l'on tentait de faire une moyenne pour un nombre de notes supérieur à la limite que vous avez calculée.

9 – Présentation et commentaires.

Le document technique **DT1** fournit un programme d'affichage d'une table ASCII. Le **DT2** fournit un ensemble de commentaires explicatifs de ce programme, en anglais.

9.1 – Indentez correctement le programme fourni dans le DT1, pour le rendre plus lisible.

9.2 – Insérez les commentaires fournis dans le DT2 aux bons endroits, pour obtenir une version du programme correctement expliquée.

Testez votre programme pour valider son fonctionnement.

10 – Compte à rebours double (optionnel).

Pour les élèves qui ont terminé les exercices précédents, et souhaitent aller plus loin.

Votre objectif consiste à réaliser un programme qui fait fonctionner simultanément **deux** comptes à rebours à des rythmes différents et quelconques (choisis par l'utilisateur). L'un pourra par exemple décompter les secondes tandis que l'autre décomptera les minutes. **Attention** : il faut traiter le cas général (dans lequel aucun des deux délais de décompte n'est un multiple de l'autre). Vous pourrez vous limiter à des décomptes de délais supérieurs au dixième de seconde.

A vous d'investiguer pour relever ce défi. Pour l'affichage, vous avez le choix entre :
- afficher sur la console un « top » chaque fois qu'un des deux comptes-à-rebours décompte ;
- ou réaliser l'affichage grâce à un « shield » (afficheurs à segments ou à écran).

Pour le décompte, au moins trois options sont possibles. Voici des indices :
- une utilisation intelligente de la fonction delay()... ;
- l'utilisation de *timers* et d'interruptions... ;
- l'utilisation de la fonction millis()…

A quoi cela peut-il servir … ? La gestion de rythmes multiples est un besoin fréquent : par exemple dans un smartphone, qui va simultanément gérer l'horloge du système et un chronomètre de course.

Autre application : pour la musique électronique (voir historiquement l'EKO ComputeRhythm...)

Document Technique DT1 – Programme d'affichage d'une table ASCII.

```
1  void setup() {
2  Serial.begin(9600);
3  while (!Serial) {
4  ;
5  }
6  Serial.println("ASCII Table ~ Character Map");
7  }
8
9  int thisByte = 33;
10
11 void loop() {
12 Serial.write(thisByte);
13 Serial.print(", dec: ");
14 Serial.print(thisByte);
15 Serial.print(", hex: ");
16 Serial.print(thisByte, HEX);
17 Serial.print(", oct: ");
18 Serial.print(thisByte, OCT);
19 Serial.print(", bin: ");
20 Serial.println(thisByte, BIN);
21 if (thisByte == 126) {    while (true) {
22 continue;
23 }   }
24 thisByte++;
25 }
```

Document Technique DT2 – Commentaires du programme d'affichage d'une table ASCII.

```
/*
  ASCII table

  Prints out byte values in all possible formats:
  - as raw binary values
  - as ASCII-encoded decimal, hex, octal, and binary values

  The circuit: No external hardware needed.

  created 2006 by Nicholas Zambetti <http://www.zambetti.com>
  modified 9 Apr 2012 by Tom Igoe

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/ASCIITable
*/

  //Initialize serial and wait for port to open:

    // wait for serial port to connect.

  // prints title with ending line break
}

// first visible ASCIIcharacter '!' is number 33:

// you can also write ASCII characters in single quotes.
// for example, '!' is the same as 33, so you could also use this:
// int thisByte = '!';

  // prints value unaltered, i.e. the raw binary version of the byte.
  // The Serial Monitor interprets all bytes as ASCII, so 33, the first number,
  // will show up as '!'

  // prints value as string as an ASCII-encoded decimal (base 10).
  // Decimal is the default format for Serial.print() and Serial.println(),

  // prints value as string in hexadecimal (base 16):

  // prints value as string in octal (base 8);

  // prints value as string in binary (base 2) also prints ending line break:

  // if printed last visible character '~' or 126, stop:
    // you could also use if (thisByte == '~') {
    // This loop loops forever and does nothing

  }
  // go on to the next character
}
```

(Ces commentaires sont disponibles dans le fichier texte nommé table_ascii_com_seulement.txt)