CODING CLUB

EPITECH

Cpoolday

DIVERSITY by EPITECH

Avant Tout

L'objectif de cet atelier est de vous faire découvrir les bases du langage C, mais aussi l'ambiance et la philosophie de la piscine à Epitech.

Vous allez devoir faire preuve de rigueur et de persévérance. Si vous bloquez sur un problème, ici on ne passe pas au suivant, on demande à son voisin de droite, celui de gauche et en face de soi, puis internet. Si vous n'avez toujours pas trouvé, c'est là qu'interviennent les **Cobras**.

Règles :

- Les différents exercices doivent être réalisés sans utiliser aucune bibliothèque. Deux fonctions ont été créées pour vous assister :

```c
void print_char(char c);

void print_number(int number);
```

Elles ne sont pas disponibles pour chaque exercice (ce sera précisé pour chacun).

- Pour chaque exercice, nous attendons de vous une **fonction** respectant le prototype imposé. Vous devez donc le respecter, sinon vous ne serez pas corrigé par la moulinette.
- Il est important de respecter les noms de fichiers ainsi que les prototypes donnés. Pour chaque exercice, vous créerez un nouveau fichier .c portant le nom de l'exercice. Par exemple, pour un exercice my_print_name :

```
touch my_print_name.c
```

```c
void my_print_name(void);
```

Allons-y !

Maintenant que vous avez bien lu ce qui précède, il ne vous reste plus qu'à vous lancer.

Bonne chance !

My_Print_Ascii

Fichier de rendu: `my_print_ascii.c`

Sujet :

L'objectif de cet exercice est d'écrire une fonction qui affichera l'ensemble des caractères affichables de la table ascii.

Pour cet exercice vous pouvez utiliser print_char

La fonction devra être prototypée de la façon suivante :

```
void  my_print_ascii(void);
```

Exemple :

Un appel de votre fonction devra afficher ces résultats :

```
int main(void)
{
    my_print_ascii();
    return 0;
}
```

```
$> !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghij
   klmnopqrstuvwxyz{|}~
```

My_Print_N_Ascii

Fichier de rendu: `my_print_n_ascii.c`

Sujet :

L'objectif de cet exercice est d'écrire une fonction qui affichera les n premiers caractères affichables de la table ascii.

S'il y a un problème la fonction doit s'arrêter sans rien faire.

Pour cet exercice vous pouvez utiliser print_char

La fonction devra être prototypée de la façon suivante :

```
void my_print_n_ascii(int howMany);
```

Exemple :

Un appel de votre fonction devra afficher ces résultats :

```
int main(void)
{
    my_print_n_ascii(5);
    return 0;
}
```

```
$> !"#$%
```

```
int main(void)
{
    my_print_n_ascii(-2);
    return 0;
}
```

5

```
$>
```

```
int main(void)
{
    my_print_n_ascii(100);
    return 0;
}
```

```
$>
```

My_Print_Revert

Fichier de rendu: `my_print_revert.c`

Sujet :

L'objectif de cet exercice est d'écrire une fonction qui affichera une string à l'envers.

> Pour cet exercice vous pouvez utiliser print_char

La fonction devra être prototypée de la façon suivante :

```
void my_print_revert(char *toRevert);
```

Exemple :

Un appel de votre fonction devra afficher ces résultats :

```
int main(void)
{
    my_print_revert("codingclub");
    return 0;
}
```

```
$> bulcgnidoc
```

My_Char_Replace

Fichier de rendu: `my_char_replace.c`

Sujet :

L'objectif de cet exercice est d'écrire une fonction qui renverra une chaine de caractères avec dans laquelle un caractère est remplacé par un autre à chaque apparition.

La fonction devra être prototypée de la façon suivante :

```
char *my_char_replace(char *origin, char toFind, char toReplace);
```

Exemple :

Un appel de votre fonction devra afficher ces résultats :

```
#include <stdio.h>

int main(int ac, char **av)
{
    char str[] = "bienvenue à epitech !";
    printf("%s\n", my_char_replace(str, 'e', 'a'));
    return 0;
}
```

```
$> bianvanua à apitach !
```

My_Get_Char_Repeat

Fichier de rendu: `my_get_char_repeat.c`

Sujet :

L'objectif de cet exercice est d'écrire une fonction capable de compter le nombre d'occurrences d'un caractère dans une string.

La fonction devra être prototypée de la façon suivante :

```
int  my_get_char_repeat(char to_find, const char *to_search);
```

Exemple :

Un appel de votre fonction devra afficher ces résultats :

```
#include <stdio.h>

int main(void)
{
    char str[] = "bienvenue à epitech";
    printf("%d\n", my_get_char_repeat('e', str));
    return 0;
}
```

```
$> 5
```

```
#include <stdio.h>

int main(void)
{
    printf("%d\n", my_get_char_repeat('e', NULL));
    return 0;
}
```

```
$> 0
```

My_Absolute

Fichier de rendu: `my_absolute.c`

Sujet :

L'objectif de cet exercice est d'écrire une fonction qui renverra la valeur absolue d'un nombre passé en argument.

La fonction devra être prototypée de la façon suivante :

```
int my_absolute(int number);
```

Exemple :

Un appel de votre fonction devra afficher ces résultats :

```
#include <stdio.h>

int main(void)
{
    printf("%d\n", my_absolute(5));
    return 0;
}
```

```
$> 5
```

```
#include <stdio.h>

int main(void)
{
    printf("%d\n", my_absolute(-5));
}
```

```
$> 5
```

My_Square

Fichier de rendu: `my_square.c`

Sujet :

L'objectif de cet exercice est d'écrire une fonction qui renverra un nombre au carré.

La fonction devra être prototypée de la façon suivante :

```
int my_square(int number);
```

Exemple :

Un appel de votre fonction devra afficher ces résultats :

```
#include <stdio.h>

int main(void)
{
    printf("%d\n", my_square(5));
    return 0;
}
```

```
$> 25
```

My_Little_Bistro

Fichier de rendu: `my_little_bistro.c`

Sujet :

L'objectif de cet exercice est d'écrire une fonction qui prend 3 arguments :

- un premier nombre sous forme d'un int.
- un caractère qui représente un symbole d'opération (+, -, *, /, %) sous forme d'un char.
- un deuxième nombre sous forme d'un int

Votre fonction doit appliquer l'opération donnée par le caractère sur les deux nombres et renvoyer le résultat sous forme d'un int.

S'il y a un problème la fonction renvoie 0.

Voici donc son prototype :

```
int my_little_bistro(int value1, char op, int value2);
```

Exemple :

Un appel de votre fonction devra afficher ces résultats :

```
#include <stdio.h>

int main(void)
{
    printf("%d\n", my_little_bistro(40, '+', 2));
    return 0;
}
```

```
$> 42
```

```c
#include <stdio.h>

int main(void)
{
    printf("%d\n", my_little_bistro(85, '/', 2));
    return 0;
}
```

```
$> 42
```

```c
#include <stdio.h>

int main(void)
{
    printf("%d\n", my_little_bistro(150, '%', 54));
    return 0;
}
```

```
$> 42
```

My_Print_Square

Fichier de rendu: `my_print_square.c`

Sujet :

L'objectif de cet exercice est d'écrire une fonction qui prend 2 arguments :

- un premier nombre sous forme d'un int > 1.
- le second sous la forme d'un caractère.

Votre fonction doit afficher un carré de coté de la taille du premier argument avec le caractère du second.

S'il y a un problème la fonction ne fait rien.

> Pour cet exercice vous pouvez utiliser print_char

Voici donc son prototype :

```
void my_print_square(int size, char c);
```

Exemple :

Un appel de votre fonction devra afficher ces résultats :

```
int main(void)
{
    my_print_square(2, 'g');
    return 0;
}
```

```
$> gg
   gg
```

```
int main(void)
{
    my_print_square(-2, 'g');
    return 0;
}
```

```
$>
```

My_Average

Fichier de rendu: `my_average.c`

Sujet :

Pour cet exercice vous devez rendre une fonction qui prend en argument un pointeur sur une liste de nombre (int *) ainsi que sa taille. Vous devez renvoyer la moyenne de ses valeurs sous forme d'un float.

Voici donc son prototype :

```
float my_average(int *grades, int count);
```

Exemple :

Un appel de votre fonction devra afficher ces résultats :

```
#include <stdlib.h>
int main(void)
{
    int list[5] = {10, 17, 6, 14, 20};
    printf("%f\n", my_average(list, 5));
    return 0;
}
```

```
$> 13.4
```

My_Get_Words

Fichier de rendu: `my_get_words.c`

Sujet :

Cet exercice a pour but de print chaque mot présent dans une string, donc séparés par: ' ' ou '\t'. Avec un \n entre chaque mot.

Pour cet exercice vous pouvez utiliser print_char

Voici donc son prototype :

```
void my_get_words(char *sentance);
```

Exemple :

Un appel de votre fonction devra afficher ces résultats :

```
#include <string.h>

int main(void)
{
    char str[] = "bienvenue                    à    epite ch";
    my_get_words(str);
    return 0;
}
```

```
$> bienvenue
   à
   epite
   ch
```

My_Rotate_Aplha

Fichier de rendu: `my_rotate_alpha.c`

Sujet :

La consigne de cet exercice est de créer une fonction appelée "my_rotate_alpha" qui prend en entrée une chaîne de caractères. La fonction doit effectuer une rotation alphabétique sur chaque caractère de la chaîne. Chaque caractère sera décalé de sa position dans l'alphabet (a -> 0, b -> 1...). La rotation doit être circulaire, c'est-à-dire que si un caractère atteint la fin de l'alphabet, il doit revenir au début. Finalement. La fonction doit retourner la chaîne de caractères modifiée. Si l'un des caractères de la string n'est pas une lettre minuscule la fonction s'arrête.

Voici donc son prototype :

```
char *my_rotate_alpha(char *sentance);
```

Exemple :

Un appel de votre fonction devra afficher ces résultats :

```
#include <stdio.h>

int main(void)
{
    char str[] = "abcdefghijklmnopqrstuvwxyz";
    printf("%s\n", my_rotate_alpha(str));
    return 0;
}
```

```
$> acegikmoqsuwybdfhjlnprtvxz
```

My_Sort

Fichier de rendu: `my_sort.c`

Sujet :

La fonction de cet exercice prend un premier argument: une liste de nombres entiers, et un second: le nombre d'entiers. Cette dernière doit trier et afficher les nombres de la liste. Les techniques de tri sont nombreuses nous vous conseillons le Bubble sort.

Voici donc son prototype :

Pour cet exercice vous pouvez utiliser print_number

```
void my_sort(int *nlist, int size);
```

Exemple :

Un appel de votre fonction devra afficher ces résultats :

```
int main(void)
{
    char list[6] = {-84, 42, -21, 21, -42, 84};
    my_sort(list, 6);
}
```

```
$> -84
   -42
   -21
   21
   42
   84
```

My_Get_Value

Fichier de rendu: `my_get_value.c`

Sujet :

La fonction de cet exercice prend un premier argument: un dictionnaire, et un second: une clé dont vous devez retrouver et print la valeur.

Exemple de Dictionnaire:

```
"ville:Lyon,    pays:France, continent=europe"
```

Le format se traduit donc par une clé séparer par ':' ou '=' de sa valeur sans espace. Chacun des ses ensembles clé-valeur sont séparer par des ',', ' ' , '\t'.

> Pour cet exercice vous pouvez utiliser print_char

Voici donc son prototype :

```
void my_get_value(char *dictionary,  char *toFind);
```

Exemple :

Un appel de votre fonction devra afficher ces résultats :

```
int main(void)
{
    char str[] = "ville:Lyon,    pays:France, continent=europe";
    char str1[] = "pays";
    char str2[] = "Ville";

    my_get_value(str, str1);
    my_get_value(str, str2);
    return 0;
}
```

```
$> France
```

My_Fibonacci

Fichier de rendu: `my_fibonacci.c`

Sujet :

L'objectif de cet exercice est d'écrire une fonction qui prend 2 arguments :

- un premier nombre sous forme d'un int >= 0 étant le minimum de la suite de fibonacci.
- le second nombre sous forme d'un int, le maximum.

Cette dernière doit afficher la suite de fibonacci en partant du minimum jusqu'au maximum.

Pour cet exercice vous pouvez utiliser print_number

Voici donc son prototype :

```
void my_fibonacci(int minimum, int maximum);
```

Exemple :

Un appel de votre fonction devra afficher ces résultats :

```
int main(void)
{
    int minimum = 0;
    int maximum = 10;

    my_fibonacci(minimum, maximum);
    return 0;
}
```

```
$> 0
   1
```

```
1
2
3
5
8
```

```
int main(void)
{
    int minimum = -2;
    int maximum = 10;

    my_fibonacci(minimum, maximum);
    return 0;
}
```

```
$>
```