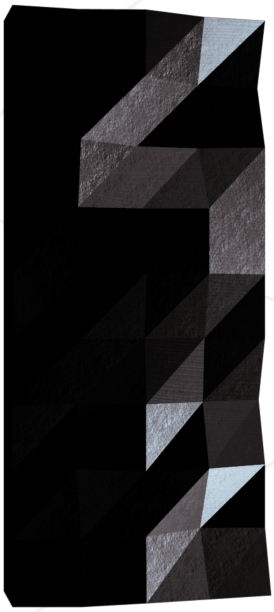

Piscine C

Jour 08

Staff 42 pedago@42.fr

Résumé: Ce document est le sujet du jour 08 de la piscine C de 42.

Table des matières

Chapitre I

Consignes

- Seule cette page servira de référence : ne vous fiez pas aux bruits de couloir.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- Attention aux droits de vos fichiers et de vos répertoires.
- Vous devez suivre la procédure de rendu pour tous vos exercices.
- Vos exercices seront corrigés par vos camarades de piscine.
- En plus de vos camarades, vous serez corrigés par un programme appelé la Moulinette.
- La Moulinette est très stricte dans sa notation. Elle est totalement automatisée. Il est impossible de discuter de sa note avec elle. Soyez d'une rigueur irréprochable pour éviter les surprises.
- La Moulinette n'est pas très ouverte d'esprit. Elle ne cherche pas à comprendre le code qui ne respecte pas la Norme. La Moulinette utilise le programme `norminette` pour vérifier la norme de vos fichiers. Comprendre par là qu'il est stupide de rendre un code qui ne passe pas la `norminette`.
- L'utilisation d'une fonction interdite est un cas de triche. Toute triche est sanctionnée par la note de `-42`.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.
- Si ft_putchar() est une fonction autorisée, nous compilerons avec notre `ft_putchar.c`.
- Vous ne devrez rendre une fonction main() que si nous vous demandons un programme.
- La Moulinette compile avec les flags -Wall -Wextra -Werror, et utilise `gcc`.
- Si votre programme ne compile pas, vous aurez `0`.
- Vous ne devez laisser dans votre répertoire aucun autre fichier que ceux explicitement specifiés par les énoncés des exercices.

- Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Votre manuel de référence s'appelle `Google / man / Internet / ...`.
- Pensez à discuter sur le forum Piscine de votre Intra !
- Lisez attentivement les exemples. Ils pourraient bien requérir des choses qui ne sont pas autrement précisées dans le sujet...
- Réfléchissez. Par pitié, par Odin ! Nom d'une pipe.

Chapitre II

Préambule

L'encyclopédie collaborative *Wikipédia* a ceci à dire sur l'ornithorynque :

L'ornithorynque (Ornithorhynchus anatinus) est une espèce de petits mammifères semi-aquatiques endémique de l'est de l'Australie, y compris la Tasmanie. C'est l'une des cinq espèces de l'ordre des monotrèmes, seul ordre de mammifères qui ponde des œufs au lieu de donner naissance à des petits complètement formés (les quatre autres espèces sont des échidnés). C'est la seule espèce survivante de la famille des Ornithorhynchidae et du genre Ornithorhynchus bien qu'un grand nombre de fragments d'espèces fossiles de cette famille et de ce genre aient été découverts.

L'apparence bizarre de ce mammifère pondant des œufs, muni d'aiguillons venimeux, à la mâchoire cornée ressemblant au bec d'un canard, à queue évoquant un castor, qui lui sert à la fois de gouvernail dans l'eau et de réserve de graisse, et à pattes de loutre a fortement surpris les premiers explorateurs qui l'ont découvert ; bon nombre de naturalistes européens ont cru à une plaisanterie. C'est l'un des rares mammifères venimeux : le mâle porte sur les pattes postérieures un aiguillon qui peut libérer du venin capable d'infliger de vives douleurs à un être humain. Les traits originaux de l'ornithorynque en font un sujet d'études important pour mieux comprendre l'évolution des espèces animales et en ont fait un des symboles de l'Australie : il a été utilisé comme mascotte pour de nombreux évènements nationaux et il figure au verso de la pièce de 20 cents australiens.

Jusqu'au début du XXe siècle, il a été chassé pour sa fourrure mais il est protégé à l'heure actuelle. Bien que les programmes de reproduction en captivité aient eu un succès très limité et qu'il soit sensible aux effets de la pollution, l'espèce n'est pas encore considérée comme en danger.

Ce sujet ne traite pas de l'ornithorynque.

Chapitre III

Exercice 00 : ft_split_whitespaces

	Exercice : 00
ft_split_whitespaces	
Dossier de rendu : *ex*00/	
Fichiers à rendre : `ft_split_whitespaces.c`	
Fonctions Autorisées : `malloc`	
Remarques : `n/a`	

42 - Classics : Ces exercices sont incontournables et ne rapportent aucun points, mais il est imperatif de les valider pour accéder aux véritables exercices du jour.

- Écrire une fonction qui découpe une chaîne de caractères en mots.
- Les séparateurs sont les espaces, les tabulations et les retours à la ligne.
- La fonction renvoie un tableau où chaque case contient l'adresse d'une chaîne de caractères représentant un mot. Le dernier élement du tableau devra être égal à 0 pour marquer la fin du tableau.
- Il ne doit pas y avoir de chaîne vide dans votre tableau. Tirez-en les conclusions qui s'imposent.
- La chaîne qui sera transmise ne sera pas modifiable.
- Elle devra être prototypée de la façon suivante :

```
char **ft_split_whitespaces(char *str);
```

Chapitre IV

Exercice 01 : ft.h

	Exercice : 01
ft.h	
Dossier de rendu : $ex01/$	
Fichiers à rendre : `ft.h`	
Fonctions Autorisées : `Aucune`	
Remarques : `n/a`	

- Écrire votre fichier `ft.h`
- Il contient tous les prototypages de vos fonctions de votre `libft.a`.

Chapitre V

Exercice 02 : ft_boolean.h

	Exercice : 02
ft_boolean.h	
Dossier de rendu : *ex*02/	
Fichiers à rendre : `ft_boolean.h`	
Fonctions Autorisées : `Aucune`	
Remarques : `n/a`	

- Écrire un fichier `ft_boolean.h` qui fera compiler et fonctionner correctement le main suivant :

```c
#include "ft_boolean.h"

void        ft_putstr(char *str)
{
        while (*str)
                write(1, str++, 1);
}

t_bool        ft_is_even(int nbr)
{
        return ((EVEN(nbr)) ? TRUE : FALSE);
}

int                main(int argc, char **argv)
{
        (void)argv;
        if (ft_is_even(argc - 1) == TRUE)
                ft_putstr(EVEN_MSG);
        else
                ft_putstr(ODD_MSG);
        return (SUCCESS);
}
```

- Ce programme devra afficher

```
I have an even number of arguments.
```

- ou

```
I have an odd number of arguments.
```

- suivi d'un retour à la ligne, dans le cas adéquat.

Chapitre VI

Exercice 03 : ft_abs.h

	Exercice : 03
ft_abs.h	
Dossier de rendu : $ex03/$	
Fichiers à rendre : `ft_abs.h`	
Fonctions Autorisées : `Aucune`	
Remarques : `n/a`	

- Écrire une macro `ABS` qui remplace son paramètre par sa valeur absolue :

```
#define ABS(Value)
```

Chapitre VII

Exercice 04 : ft_point.h

	Exercice : 04
ft_point.h	
Dossier de rendu : *ex*04/	
Fichiers à rendre : `ft_point.h`	
Fonctions Autorisées : `Aucune`	
Remarques : `n/a`	

- Écrire un fichier `ft_point.h` qui fera compiler le main suivant :

```
#include "ft_point.h"

void        set_point(t_point *point)
{
        point->x = 42;
        point->y = 21;
}

int             main(void)
{
        t_point             point;

        set_point(&point);
        return (0);
}
```

Chapitre VIII

Exercice 05 : ft_param_to_tab

	Exercice : 05
ft_param_to_tab	
Dossier de rendu : $ex05/$	
Fichiers à rendre : `ft_param_to_tab.c, ft_stock_par.h`	
Fonctions Autorisées : `ft_split_whitespaces, ft_show_tab, malloc`	
Remarques : `n/a`	

- Écrire une fonction qui stocke les paramètres du programme dans un tableau de structures et qui renvoie l'adresse de la première case du tableau.
- Tous les éléments du tableau devront être traités, y compris `av[0]`.
- Elle devra être prototypée de la façon suivante :

```
struct s_stock_par *ft_param_to_tab(int ac, char **av);
```

- Le tableau de structures devra être alloué et la dernière case contiendra 0 dans son élément `str` pour signaler la fin.

- La structure est définie dans le fichier `ft_stock_par.h` comme suit :

```
typedef struct s_stock_par
{
  int  size_param;
  char *str;
  char *copy;
  char **tab;
}            t_stock_par;
```

  - `size_param` étant la longueur du paramètre ;
  - `str` étant l'adresse du paramètre ;
  - `copy` étant la copie du paramètre ;
  - `tab` étant le tableau retourné par `ft_split_whitespaces`.

- Nous testons votre fonction avec notre `ft_split_whitespaces` et notre `ft_show_tab` (exercice suivant). Prenez les mesures nécessaires pour que cela fonctionne !

Chapitre IX

Exercice 06 : ft_show_tab

	Exercice : 06
ft_show_tab	
Dossier de rendu : *ex*06/	
Fichiers à rendre : `ft_show_tab.c, ft_stock_par.h`	
Fonctions Autorisées : `ft_putchar`	
Remarques : `n/a`	

- Écrire une fonction qui affiche le contenu d'un tableau créé par la fonction précédente.
- Elle devra être prototypée de la façon suivante :

```
void ft_show_tab(struct s_stock_par *par);
```

- La structure est définie dans le fichier `ft_stock_par.h` comme suit :

```
typedef struct s_stock_par
{
  int  size_param;
  char *str;
  char *copy;
  char **tab;
}            t_stock_par;
```

- Pour chaque case, on affiche (un élément par ligne) :
  - le paramètre
  - la taille
  - chaque mot (un par ligne)
- Nous testons votre fonction avec notre `ft_param_to_tab` (exercice précédent). Prenez les mesures nécessaires pour que cela fonctionne !