Piscine - C - Tek1

Sujet Jour 12

Responsable Astek astek_resp@epitech.eu

Table des matières

Consignes

- Le sujet peut changer jusqu'à une heure avant le rendu.
- Vos exercices doivent être à la norme.
- Vous ne devez avoir de `main()` dans aucun fichier de votre repertoire de rendu.
- Pour chaque repertoire de chaque exercice nous allons compiler vos fichiers avec la commande `cc -c *.c`, ce qui va génerer tous les fichiers `.o` que nous allons ensuite linker un par un en y ajoutant notre `main.c` :

```
$> cd ex_01
$> cc -I/afs/epitech.net/users/group/login/rendu/include/ -c *.c
$> cc *.o ~moulinette/main_ex_01.o -o ex01 -L/afs/epitech.net/users/group/login/rendu/lib/ -lmy
$> ./ex01
[...]
```

- Vous ne devez laisser dans votre répertoire aucun autre fichier que ceux explicitement specifiés par les énoncés des exercices.
  Si un seul de vos fichiers empêche la compilation avec `*.c`, la moulinette ne pourra pas vous corriger et vous aurez `0`. Vous avez donc tout intéret à effacer vos rendus d'exercices ne fonctionnant pas.
- Vous n'avez le droit qu'à la fonction `my_putchar` pour faire les exercices qui suivent. Cette fonction sera fournie, donc :
  - vous ne devez pas avoir lors du rendu de fichier `my_putchar.c`
  - la fonction `my_putchar` ne doit être mise dans aucun des fichiers rendus
- Pensez à en discuter sur le forum piscine !
- Travaillez en local !
  C'est-à-dire que pour chaque exercice vous devez le compiler sur votre compte linux puis, une fois qu'il fonctionne, le copier sur votre compte AFS.
  Ceci dans le simple but de ne pas surcharger les serveurs car vous êtes nombreux.

Indices `Faites-vous un script shell pour copier vos fichiers sur l'AFS`

- Rendu :
  `/afs/epitech.net/users/group/login/rendu/piscine/Jour_12`
- Votre bibliothèque sera utilisée durant la phase de 'linkage'.
- Pour les exos sur les listes, on utilisera la structure suivante (légèrement différente de celle vue en cours) :

```
1  typedef struct s_list
2  {
3    void *data;
4    struct s_list *next;
5  } t_list;
```

- Vous devez mettre cette structure dans un fichier .h dans votre repertoire d'includes :
  `/afs/epitech.net/users/group/login/rendu/include/`

Exercice 01 : my_params_in_list

- Écrire la fonction `my_params_in_list` qui crée une nouvelle liste en y mettant les paramètres de la ligne de commande.
- L'adresse du premier maillon de la liste est renvoyée.
- Elle devra être prototypée de la façon suivante :

```
1   t_list *my_params_in_list(int ac, char **av);
```

- Exemple :

```
$> ./a.out test arg2 arg3
```

  - Si le `main` transmet directement ses arguments (`argc` et `argv`) à `my_param_in_list` cela devra mettre d'abord `./a.out` dans la liste puis `test` puis `arg2` puis `arg3` lors du parcours de la liste, on aura donc comme premier élément `arg3` puis `arg2` puis `test` puis `./a.out`

- Rendu :

`/afs/epitech.net/users/group/login/rendu/piscine/Jour_12/ex_01/my_params_in_list.c`

Exercice 02 : my_list_size

- Écrire la fonction `my_list_size` qui renvoie le nombre d'éléments dans la liste.
- Elle devra être prototypée de la façon suivante :

```
1   int my_list_size(t_list *begin);
```

- Rendu :

`/afs/epitech.net/users/group/login/rendu/piscine/Jour_12/ex_02/my_list_size.c`

Exercice 03 : my_rev_list

- Écrire la fonction `my_rev_list` qui inverse l'ordre des éléments de la liste. Seuls les jeux de pointeurs sont admis.
- Elle devra être prototypée de la façon suivante :

```
1  int my_rev_list(t_list **begin);
```

- Rendu :

`/afs/epitech.net/users/group/login/rendu/piscine/Jour_12/ex_03/my_rev_list.c`

Exercice 04 : my_apply_on_list

- Écrire la fonction `my_apply_on_list` qui applique une fonction donnée en paramètre à l'information contenue dans chaque maillon de la liste.
- Elle devra être prototypée de la façon suivante :

```
1   int my_apply_on_list(t_list *begin, int (*f)());
```

- La fonction pointée par `f` sera utilisée de la façon suivante :

```
1   (*f)(list_ptr->data);
```

- Rendu :

`/afs/epitech.net/users/group/login/rendu/piscine/Jour_12/ex_04/my_apply_on_list.c`

Exercice 05 : my_apply_on_eq_in_list

- Écrire la fonction `my_apply_on_eq_in_list` qui applique une fonction donnée en paramètre à l'information contenue dans certains maillons de la liste. Une information de référence ainsi qu'une fonction de comparaison nous permettent de sélectionner les bons maillons de la liste : ceux qui sont "egaux" avec l'information de référence.
- Elle devra être prototypée de la façon suivante :

```
1   int my_apply_on_eq_in_list(t_list *begin, int (*f)(), void *data_ref, int (*cmp)());
```

- Les fonctions pointées par `f` et par `cmp` seront utilisées de la façon suivante :

```
1   (*f)(list_ptr->data);
2   (*cmp)(list_ptr->data, data_ref);
```

- Rendu :

`/afs/epitech.net/users/group/login/rendu/piscine/Jour_12/ex_05/my_apply_on_eq_in_list.c`

Indices La fonction cmp pourrait être par exemple my_strcmp, c'est-à-dire que si cmp renvoie 0, les données sont ''égales'', donc dans ce cas seulement on appelle la fonction f

Exercice 06 : my_find_elm_eq_in_list

- Écrire la fonction `my_find_elm_eq_in_list` qui renvoie la donnée du premier maillon “égale” à la donnée de référence.
- Elle devra être prototypée de la façon suivante :

```
1   void *my_find_elm_eq_in_list(t_list *begin, void *data_ref, int (*cmp)());
```

- Rendu :

`/afs/epitech.net/users/group/login/rendu/piscine/Jour_12/ex_06/my_find_elm_eq_in_list.c`

Indices La fonction cmp pourrait être par exemple my_strcmp, c’est-à-dire que si cmp renvoie 0, les données sont ‘‘égales’’, donc dans ce cas seulement on appelle la fonction f

Exercice 07 : my_find_node_eq_in_list

- Écrire la fonction `my_find_node_eq_in_list` qui renvoie l'adresse du premier maillon dont la donnée est "égale" à la donnée de référence.
- Elle devra être prototypée de la façon suivante :

```
1   t_list *my_find_node_eq_in_list(t_list *begin, void *data_ref, int (*cmp)());
```

- Rendu :

`/afs/epitech.net/users/group/login/rendu/piscine/Jour_12/ex_07/my_find_node_eq_in_list.c`

Indices La fonction cmp pourrait être par exemple my_strcmp, c'est-à-dire que si cmp renvoie 0, les données sont ''égales'', donc dans ce cas seulement on appelle la fonction f

Exercice 08 : my_rm_all_eq_from_list

- Écrire la fonction `my_rm_all_eq_from_list` qui efface de la liste tous les élements dont la donnée est “égale” à la donnée de référence.
- Elle devra être prototypée de la façon suivante :

```
1   int my_rm_all_eq_from_list(t_list **begin, void *data_ref, int (*cmp)());
```

- Rendu :

`/afs/epitech.net/users/group/login/rendu/piscine/Jour_12/ex_08/my_rm_all_eq_from_list.c`

Indices La fonction cmp pourrait être par exemple my_strcmp, c’est-à-dire que si cmp renvoie 0, les données sont ‘‘égales’’, donc dans ce cas seulement on appelle la fonction f

Exercice 09 : my_add_list_to_list

- Écrire la fonction `my_add_list_to_list` qui met les élements d'une liste `begin2` à la fin d'une autre liste `begin1`.
- La création d'éléments n'est pas autorisée.
- Elle devra être prototypée de la façon suivante :

```
1   int my_add_list_to_list(t_list **begin1, t_list *begin2);
```

- Rendu :

`/afs/epitech.net/users/group/login/rendu/piscine/Jour_12/ex_09/my_add_list_to_list.c`

Exercice 10 : my_sort_list

- Écrire la fonction `my_sort_list` qui trie par ordre croissant le contenu de la liste, en comparant deux maillons grâce à une fonction de comparaison de données des deux maillons.
- Elle devra être prototypée de la façon suivante :

```
1   int my_sort_list(t_list **begin, int (*cmp)());
```

- Rendu :

`/afs/epitech.net/users/group/login/rendu/piscine/Jour_12/ex_10/my_sort_list.c`

Indices `La fonction cmp pourrait être par exemple my_strcmp`

- c'est-à-dire que si :
  - `cmp` renvoie `0`, les données sont "égales"
  - `cmp` renvoie une valeur negative, la première donnée et plus petite que la seconde donnée
  - `cmp` renvoie une valeur positive, la première donnée et plus grande que la seconde donnée

Exercice 11 : my_put_elem_in_sort_list

- Écrire la fonction`my_put_elem_in_sort_list` qui crée un nouvel élément, et l'insère dans une liste triée de sorte que la liste reste triée par ordre croissant.
- Elle devra être prototypée de la façon suivante :

```
1   int my_put_elem_in_sort_list(t_list **begin, void *data, int (*cmp)());
```

- Rendu :

`/afs/epitech.net/users/group/login/rendu/piscine/Jour_12/ex_11/my_put_elem_in_sort_list.c`

Indices `La fonction cmp pourrait être par exemple my_strcmp`

- c'est-à-dire que si :
  - `cmp` renvoie `0`, les données sont "égales"
  - `cmp` renvoie une valeur negative, la première donnée et plus petite que la seconde donnée
  - `cmp` renvoie une valeur positive, la première donnée et plus grande que la seconde donnée

Exercice 12 : my_add_sort_list_to_sort_list

- Écrire la fonction `my_add_sort_list_to_sort_list` qui intègre les élements d'une liste triée `begin2` dans une autre liste triée `begin1`, de sorte que la liste `begin1` reste triée par ordre croissant.
- Elle devra être prototypée de la façon suivante :

```
1   int my_add_sort_list_to_sort_list(t_list **begin1, t_list *begin2, int (*cmp)());
```

- Rendu :

`/afs/epitech.net/users/group/login/rendu/piscine/Jour_12/ex_12/my_add_sort_list_to_sort_list.c`

Indices `La fonction cmp pourrait être par exemple my_strcmp`

- c'est-à-dire que si :
  - `cmp` renvoie `0`, les données sont "égales"
  - `cmp` renvoie une valeur negative, la première donnée et plus petite que la seconde donnée
  - `cmp` renvoie une valeur positive, la première donnée et plus grande que la seconde donnée

`Attention les pointeurs peuvent être NULL !`