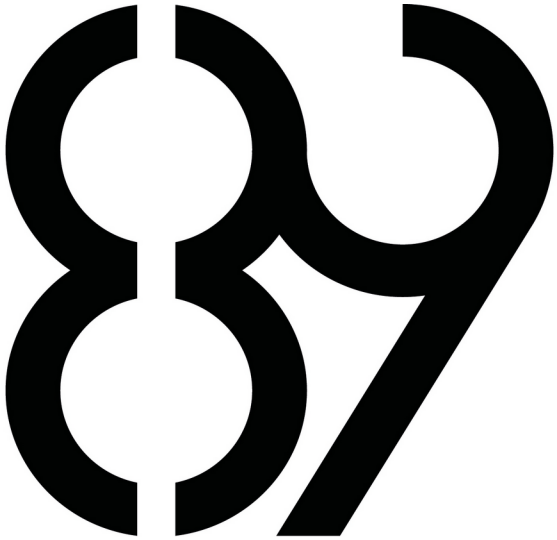

Journée 01

Boucles avancées, conversion mathématique, erreurs

- DaemonLab -
daemonlab@ecole-89.com

Durant cette journée, vous construirez des boucles plus complexes et programmerez des conversions à la complexité inattendue.

Nom de code : !pac1j01
Clôture du ramassage : 20/11/2019 23:59

Médailles accessibles :

INDEX

01 – Détails administratifs

Votre travail doit être envoyé via l'interface de ramassage du **TechnoCentre** :

http://technocentre.ecole89.com/ramassage

Le numéro de code présent sur ce sujet vous est propre : vous devrez le renseigner en rendant votre travail. En cas d'erreur, votre travail ne sera pas associée à l'activité et votre travail ne sera pas ramassé.

Pour cette activité, vous rendrez votre travail sous la forme d'une archive au format tar.gz. Cette archive devra contenir l'ensemble de votre travail tel que demandé dans la section 4.

Pour créer cette archive .tar.gz, il vous suffit d'utilliser la commande suivante :

```
$> tar cvfz mon_fichier.tar.gz fichier1 fichier2 fichier3
```

Le nom « mon_fichier.tar.gz » étant à remplacé par le nom que vous souhaitez donner votre fichier archive, et « fichier1 », « fichier2 », « fichier3 » par les fichiers ou dossiers que vous souhaitez mettre dans cette archive. Vous pouvez vérifier le contenu de votre archive à l'aide de la commande « **`tar -t mon_archive.tar.gz`** ».

Ce travail est a effectuer seul. Vous pouvez bien sur échanger avec vos camarades, néanmoins vous devez être l'auteur de votre travail. Utiliser le code d'un autre, c'est **tricher**. Et tricher annule **toutes** les médailles que vous avez reçu sur l'activité. La vérification de la triche est réalisée de la même manière que la correction : de manière **automatique**. Prenez garde si vous pensez pouvoir passer au travers.

Médailles accessibles :

Réussir à rendre :
Vous avez réussi à envoyer votre travail au système de correction.

02 – Propreté de votre rendu

Votre rendu, c'est à dire le contenu de l'archive ou du dépôt que vous entrez sur l'interface du TechnoCentre, doit respecter **strictement** l'ensemble des règles suivantes :

- Il ne doit contenir **aucun** fichier objet. (*.o)
- Il ne doit contenir **aucun** ichier tampon. (*~, #*#)
- Il ne doit pas contenir votre production finale (programme ou bibliothèque)

La présence d'un fichier interdit mettra immédiatement fin à votre évaluation.

Médailles accessibles :

Rendu propre
Votre rendu respecte les règles de propretés imposées.

03 – Règlement quant à la rédaction du code C

En général, le code source de votre programme doit impérativement respecter un ensemble de règles de mise en page définie par les Table de la Norme.

Pour cette première activité, nous vous libérons de cette contrainte qui vous sera néanmoins très bientôt imposée pour l'ensemble de vos programmes écrits en C.

04 – Construction partielle de votre rendu

Le programme de correction va construire une sous-partie déterminée de votre rendu afin d'effectuer des tests dessus. En voici les paramètres :

- Les fichiers qui seront compilés sont ceux qui auront l'extension *.c.
- Seuls les fichiers dans le(s) dossier(s) ./ seront compilés.
- Les fichiers seront compilés sans règle de compilation particulière.

- Tous les fichiers seront compilés **ensemble**, cela signifie donc que chaque fonction doit être unique, et que vous pouvez utiliser les fonctions des autres exercices dans chaque exercice.

L'ensemble du code que vous rendez doit pouvoir être compilé.
En cas d'échec de la compilation, vous ne serez pas évalué.

Médailles accessibles :

Construction partielle
Les éléments requis de votre projet se construisent séparement.

05 – Fonctions interdites

Vous n'avez le droit à aucune autre fonction que celle précisée dans la liste ci-dessous :

- putchar

L'utilisation d'une fonction interdite est assimilée à de la triche.
La triche provoque l'arrêt de l'évaluation et la perte des médailles.

06 – print_square

Fichier à rendre : print_square.c

Ecrivez la fonction suivante :

```
void e89_print_square(int size);
```

Cette fonction affiche un carré constitué de '-' sur le terminal. Chaque ligne affiche autant de '-' que la valeur de size et autant de ligne que size. Ci-dessous, un exemple de sortie avec size valant 5 :

```
$> ./a.out
-----
-----
-----
-----
-----
$>
```

Médailles accessibles :

Double boucle
Vous êtes parvenu à programmer une boucle dans une boucle.

07 – is_upper

Fichier à rendre : is_upper.c

Ecrivez la fonction suivante :

```
bool e89_is_upper(char c);
```

Cette fonction renvoit vrai si le caractère envoyé est en majuscule.

N'oubliez pas de joindre votre fonction de test.

08 – sum

Fichier à rendre : sum.c

Ecrivez la fonction suivante ainsi que son test :

```
int e89_sum(int c);
```

Cette fonction renvoit la somme de tous les entiers allant de 0 à **c** inclus, que c soit negatif ou positif.

09 – factoriel

Fichier à rendre : factoriel.c

Ecrivez la fonction suivante ainsi que son test :

```
int e89_factoriel(int c);
```

Cette fonction renvoit la multiplication de tous les entiers allant de 1 à **c** inclu.

Un int ne peut compter que jusqu'à 2147483647.
Aller au-delà produit un résultat indeterminé.

Vous devez être en mesure d'éviter le dépassement et renvoyer -1 si vous n'êtes pas en mesure de calculer la valeur attendue. N'oubliez pas de tester les cas d'erreur **aussi**.

Si la valeur passée en paramètre génère une erreur, en plus de renvoyer le code d'erreur -1, vous devez également mettre dans **errno**, variable globale accessible si vous avez inclus <errno.h>, la valeur **ERANGE** qui signifie « Capacité de variable dépassée ».

N'oubliez pas dans votre test, de vérifier aussi **errno**.

10 – to_digit

Fichier à rendre : to_digit.c

Vous vous êtes entrainé à manipuler des caractères dans la précédente journée, mais êtes vous tout à fait clair de la relation qui existe entre la table ASCII et les entiers ?

Ecrivez la fonction suivante, qui transforme un int dont la valeur est comprise entre 0 et 9 compris en son chiffre, caractère ASCII, correspondant.

```
int e89_to_digit(int nbr);
```

N'oubliez pas d'écrire également le test associé, et en cas d'erreur, de renvoyer le code d'erreur -1 et de mettre EINVAL dans errno. EINVAL signifie « paramètre invalide ».

11 – roll_2d6

Fichier à rendre : roll_2d6.c

Ecrivez la fonction suivante :

```
void e89_roll_2d6();
```

Cette fonction affiche toutes les combinaisons possible du score de deux dés à 6 faces, dans l'ordre croissant. Par exemple : 11, 12, 13, 14, 15, 16, 22, 23, etc. Notez l'absence de 21, qui n'est pas présent car 12 a déjà été tiré.

Médailles accessibles :

Double boucle
Vous êtes parvenu à programmer une boucle dans une boucle.

12 – get_numeral

Fichier à rendre : get_numeral.c

Ecrivez la fonction suivante :

```
int e89_get_numeral(int nbr, int mul);
```

Cette fonction récupère dans nbr le chiffre correspondant à la puissance de 10 situé dans mul. Par exemple :

- Pour nbr valant 12345 et mul valant 0, la fonction renverra 5.

- Pour nbr valant -12345 et mul valant 1, la fonction renverra 4.

- Pour nbr valant 12345 et mul valant 2, la fonction renverra 3.

- Pour nbr valant -12345 et mul valant 3, la fonction renverra 2.

- Pour nbr valant 12345 et mul valant 4, la fonction renverra 1.

- Pour nbr valant 12345 et mul valant 5, la fonction renverra -1 et mettra ERANGE dans errno.

- Si mul est negatif, la fonction renverra -1 et mettra EINVAL dans errno.

13 – pow

Fichier à rendre : pow.c

Écrivez la fonction suivante :

```
int e89_pow(int a, int b);
```

Cette fonction renvoi le nombre a mit à la puissance b. Par exemple, si a vaut 3 et b vaut 4, alors la valeur de retour de la fonction vaudra 81. N'oubliez pas de tester votre fonction.

Concernant les valeurs autorisées de b, seules des valeurs positives seront testés.

14 – print_base10

Fichier à rendre : print_base10.c

Ecrivez la fonction suivante :

```
int e89_print_base10(int nbr);
```

Cette fonction affiche nbr sur le terminal (en base 10) et renvoit le nombre de caractère qui ont été écrit. Attention : ces nombres peuvent être positifs comme négatif. Toutes les valeurs de nbr doivent être gérées.

Nous attendons une fonction de test où vous testerez au moins la valeur de retour de cette fonction, à défaut de tester ce qui est affiché.

15 – roll_3d10

Fichier à rendre : roll_3d10.c

Ecrivez la fonction suivante, fonctionnant sur le même principe que roll_2d6.

```
void e89_roll_3d10();
```

Les valeurs des dés lancés sont comprises dans [0 ; 9]. Voici le début et la fin de la sortie attendue : 111, 112, 113, 114, 115, 116, 117, 118, 119, 121, 122, 123 ... 999.

16 – print_base2

Fichier à rendre : print_base2.c

Ecrivez la fonction suivante :

```
int e89_print_base2(int nbr);
```

Cette fonction affiche nbr (**en base 2, en binaire**) sur le terminal et renvoit le nombre de caractère qui ont été écrit. Seul des nombres positifs et 0 seront testés.

Nous attendons une fonction de test où vous testerez au moins la valeur de retour de cette fonction, à défaut de tester ce qui est affiché.

17 – from_digit

Fichier à rendre : from_digit.c

Ecrivez la fonction suivante :

```
int e89_from_digit(char c);
```

Cette fonction renvoit l'entier symbolé par le caractère c. Par exemple, si '4' est envoyé à cette fonction, elle renverra l'entier 4. N'oubliez pas d'écrire le test et de gérer les erreurs avec **errno** et **EINVAL**. -1 est le code d'erreur de cette fonction

18 – is_lower

Fichier à rendre : is_lower.c

Ecrivez la fonction suivante ainsi que son test :

```
bool e89_is_lower(char c);
```

Cette fonction renvoit vrai si le caractère envoyé en paramètre est en minuscule.

19 – to_lower

Fichier à rendre : to_lower.c

Ecrivez la fonction suivante ainsi que son test :

```
char e89_to_lower(char c);
```

Cette fonction renvoit le caractère c passé en minuscule, si c'est pertinent. Sinon il renvoi le caractère c.

20 – to_upper

Fichier à rendre : to_upper.c

Ecrivez la fonction suivante ainsi que son test :

```
char e89_to_upper(char c);
```

Cette fonction renvoit le caractère c passé en majuscule, si c'est pertinent. Sinon il renvoit le caractère c.

21 – is_space

Fichier à rendre : is_space.c

Ecrivez la fonction suivante ainsi que son test :

```
bool e89_is_space(char c);
```

Cette fonction renvoit vrai si le caractère envoyé en paramètre est un « espace ». Afin de vous renseigner sur ce qu'est un « espace » dans le contexte actuel, vous êtes invités à rechercher dans le manuel de isspace ou sur internet ce dont il s'agit.

22 – is_blank

Fichier à rendre : is_blank.c

Ecrivez la fonction suivante ainsi que son test :

```
bool e89_is_blank(char c);
```

Cette fonction renvoit vrai si le caractère envoyé en paramètre est un « blanc ». Afin de vous renseigner sur ce qu'est un « blanc » dans le contexte actuel, vous êtes invités à rechercher dans le manuel de isblank ou sur internet ce dont il s'agit.

23 – is_digit

Fichier à rendre : is_digit.c

Ecrivez la fonction suivante ainsi que son test :

```
bool e89_is_digit(char c);
```

Cette fonction renvoit vrai si le caractère envoyé en paramètre est un chiffre.

24 – is_alpha

Fichier à rendre : is_alpha.c

Ecrivez la fonction suivante ainsi que son test :

```
bool e89_is_alpha(char c);
```

Cette fonction renvoit vrai si le caractère envoyé en paramètre est une lettre.