D A E M O N L A B

Journée 6

Macro, en-tête, analyse grammaticale

- DaemonLab -

Durant cette journée, vous découvrirez un nouvel aspect du C : les en-têtes de fichier. En ouvrant le sujet des en-têtes, nous aborderons aussi les macros et les prototypes.

INDEX

01 – Avant-propos

Votre travail doit être rendu via le dossier **~/hyperespace/j6/** dans votre espace personnel.

Si vous faites erreur et que le dossier que vous utilisez pour votre rendu est différent, vous ne serez pas évalué faute d'avoir pu trouver votre travail.

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.

Votre rendu doit respecter **strictement** l'ensemble des règles suivantes :

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

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

Votre programme doit respecter les Tables de la Norme dans leur intégralité. Vous êtes invité à les observer depuis **l'Infosphère**. Elles sont disponibles comme ressource de cette activité.

02 – Fonctions autorisées

La bibliothèque logicielle venant avec le C est vaste et disponible. Nous avons cependant fait le choix de vous interdire son utilisation, afin de vous amener progressivement à reprogrammer vous même ses fonctionnalités les plus utiles.

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.

Vous n'avez le droit d'utiliser aucune fonction issue de la LibC à l'exception de celles que nous vous autoriserons explicitement.

Pour cette activité, vous avez le droit à l'appel système **write** ainsi qu'aux fonctions **malloc** et **free**.

03 – Macro ABS

Mettre dans le fichier : macro.h

Programmez la macro ABS.

```
#define ABS(a)     ...
```

La macro ABS renvoi la valeur absolue de a.
Remplacez ... par le contenu de la macro.

04 – Macro MIN

Mettre dans le fichier : macro.h

Programmez la macro MIN.

```
#define MIN(a, b)     ...
```

La macro MIN renvoi la plus petite valeur entre a et b.

05 – Macro MAX

Mettre dans le fichier : macro.h

Programmez la macro MAX.

```
#define MAX(a, b)     ...
```

La macro MAX renvoi la plus grande valeur entre a et b.

06 – Macro CLAMP

Mettre dans le fichier : macro.h

Programmez la macro **CLAMP**.

```
#define CLAMP(a, b, c)     ...
```

La macro **CLAMP** renvoi :
- **b** si **a** est plus petit que **b**
- **c** si **a** est plus grand que **c**
- sinon **a**

07 – Macro RCLAMP

Mettre dans le fichier : macro.h

Programmez la macro **RCLAMP**.

```
#define RCLAMP(a, b, c)     ...
```

La macro **RCLAMP** renvoi :
- **c** si **a** est plus petit que **b**
- **b** si **a** est plus grand que **c**
- sinon **a**

08 – Macro DIFF

Mettre dans le fichier : macro.h

Programmez la macro **DIFF**.

```
#define DIFF(a, b)     ...
```

La macro **DIFF** renvoi la valeur absolue de la différence entre **a** et **b**.

09 – Macro ARRAY_LENGTH

Mettre dans le fichier : macro.h

Programmez la macro **ARRAY_LENGTH**.

```
#define ARRAY_LENGTH(a)     ...
```

La macro **ARRAY_LENGTH** renvoi la longueur en case du tableau **a**.

Comment mesurer la taille du tableau en entier avec l'opérateur sizeof ?
Comment mesurer la taille d'une case unique ?

10 – print_split

Fichier : print_split.c

Programmez la fonction **print_split**. Cette fonction vous aidera à débugger les travaux à suivre.

```
int std_print_split(const char **wordtab);
```

La fonction **print_split** affiche toutes les chaînes de caractères dans **wordtab**. Chaque chaîne est suivie d'un saut de ligne. Le paramètre **wordtab** est un tableau de chaînés de caractères terminé par le pointeur **NULL**. La fonction renvoi le nombre de caractères écrits.

Ci-dessous, un exemple de main de test pour vous aider à générer le fameux **wordtab**.

```
int main(void)
{
	const char *wt[4];

	wt[0] = "abc";
	wt[1] = "def";
	wt[2] = "ghi";
	wt[3] = NULL;
	print_split(wt);
	return (0);
}
```

```
$> gcc print_split.c
$> ./a.out
abc
def
ghi
$>
```

11 – split

Fichier : split.c

Programmez la fonction **split** et son test.

```
char **split(const char *str, char token);
```

La fonction **split** éclate la chaîne **str** en fonction du caractère token. Par exemple, si **token** vaut ';' et que **str** vaut "abc;def;;ghi;", alors le tableau renvoyé par **split** sera ainsi :

- Sa case 0 contiendra "abc"
- Sa case 1 contiendra "def"
- Sa case 2 contiendra ""
- Sa case 3 contiendra "ghi"
- Sa case 4 contiendra ""
- Sa case 5 contiendra NULL

Si une erreur advient, la fonction renverra **NULL**.

Vous devez vérifier que malloc a bien fonctionné.
Ne pas vérifier malloc ouvre votre programme à certains risques.
Ne pas vérifier malloc est une faute de norme.

12 – free_split

Fichier : free_split.c

Programmez la fonction **free_split** et son test.

```
void free_split(const char **wt);
```

La fonction **free_split** libère l'ensemble envoyé situé dans wt.

14 – merge

Fichier : merge.c

Programmez la fonction **merge** et son test.

```
char *merge(const char **wt, char token);
```

La fonction **merge** génère une nouvelle chaîne de caractère contenant l'ensemble des chaînes situés dans **wt**, relié par le symbole situé dans **token**. Si **token** vaut '\0', aucun caractère n'est ajouté entre les chaînes.

14 – reverse_split

Fichier : reverse_split.c

Programmez la fonction **reverse_split** et son test..

```
void reverse_split(const char **wt);
```

La fonction **reverse_split** inverse l'ordre des éléments dans **wt**. Si l'on appelle **print_split** après modification, les chaînes seront affichées dans l'ordre inverse. (Les caractères dans les chaînes restent identique!)