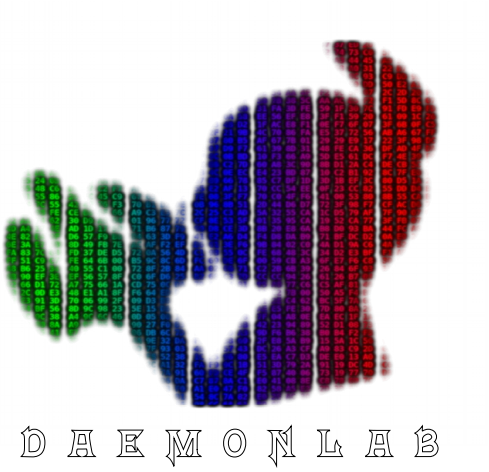

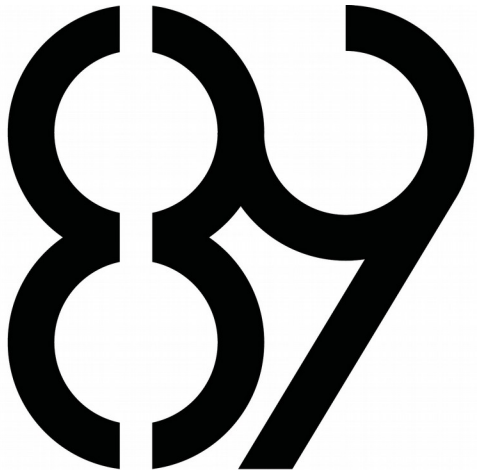

BABL

Le premier langage de programmation,
Si! Si! Le premier dont vous êtes **l'auteur**

- Le Laboratoire aux Lapins Noirs -
pedagogie@ecole-89.com

Ce mini-projet consiste à réaliser un langage de programmation très simple, basé sur des expressions mathématiques.

Ce projet est à rendre le 7 Février 2021.

INDEX

01 – Détails administratifs

Votre travail doit être envoyé via l'interface de ramassage de l'Infosphère.

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

Vous devez respecter l'intégralité des tables de norme pour ce projet.

04 – Construction 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 avec -W -Wall -Werror.

- 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.

Le nom de votre fichier exécutable sera babl.

Médailles accessibles :

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

05 – Fonctions interdites

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

Fonctions systèmes

- open
- close
- write
- read
- ioctl
- malloc
- free

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 – Table des symboles

Votre programme disposera d'une table des symboles globale, et une locale dépendant du contexte de la fonction.

Un symbole peut-être trois choses : une valeur, une adresse ou une fonction.

Une valeur est un **char**. Une adresse est un pointeur (sur **char** ou sur un autre pointeur, la responsabilité est laissé à l'auteur du programme que vous executerez). Une fonction est un symbole comportant 0 à 8 paramètres ainsi **qu'une valeur de retour obligatoire**.

Votre programme gardera en mémoire tous les symboles du programme que vous exécuterez afin de pouvoir modifier ces valeurs lors de l'exécution.

En plus de la table des symboles, vous disposerez également d'une table des éléments anonymes, qui vous permettra de stocker les chaînes de caractères en dur de votre programme.

07– Expression

Votre langage de programmation gérera les opérateurs binaires suivants, +, -, *, /, %, ==, <> (Equivalent à l'opérateur != du C), <=, <, >=, > ainsi que, c'est très important, l'opérateur **ternaire ? :**. Votre programme gérera également l'opérateur mathématique parenthèse ( ) qui permettra de créer une sous expression.

Votre langage de programmation gérera l'opérateur d'assignation = permettant de modifier un symbole dans votre table des symboles.

Votre programme gérera également l'opérateur unaire suivant : [ ], l'opérateur crochet, permettant d'accéder à l'adresse contenu par un pointeur.

Votre programme gérera également l'opérateur unaire suivant : l'opérateur parenthèse ( ) qui permettra d'appeler une fonction en lui passant des paramètres. Contrairement à l'opérateur parenthèse mathématique, cet opérateur est **unaire**, cela veut dire qu'il est après une opérande et non après un opérateur.

Une opérande de votre expression peut-être deux choses : une v**aleur directe** pouvant être soit sous la forme entière (par exemple, 47) soit sous la forme d'une chaîne de caractère entre guillemets… ou un symbole de votre table.

08– Exemple

Voici un exemple de BABL, qui vous permettra de comprendre les subtilités restantes concernant, par exemple, la syntaxe de déclaration des fonctions. Un $ indique une adresse, un # une valeur directe, soit dans les paramètres, soit dans la valeur de retour.

```
$> cat program.babl

hello="bonjour"
#strlen($str) str[0] <> 0 ? strlen($str + 1) + 1 : 0
#main() strlen(hello)

$> ./babl program.babl
$> echo $?
7
```

09– Bindings

Un « binding » est la liaison faite entre votre langage de programmation et une fonction issu de votre système d'exploitation. Il consiste par exemple à permettre d'appeller « malloc » depuis votre langage de programmation.

Permettez d'appeller open, close, read, write, malloc et free depuis votre langage de programmation.

Implémentez ensuite les fonctions putchar, strcpy, strdup et puts.