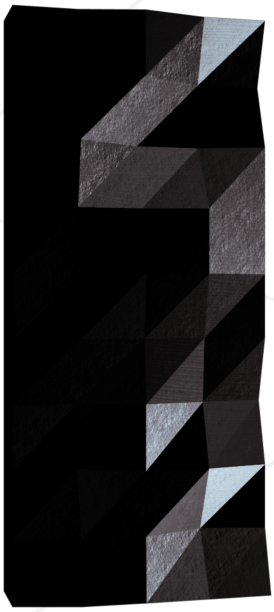

# Projet d'Adaptation

## yasl

42 staff staff@42.fr

*Résumé: YASL is an easy multiple-reversed-stacked language.*

# Table des matières

# Chapitre I

# Préambule

C'est un projet théoriquement calibré pour une seule journée. Mais comme les piles de YASL sont alcalines, ça risque bien de durer deux foix plus longtemps.

# Chapitre II

# L'Apéro

Le but de ce projet YASL est de :

- Vous faire découvrir un langage exotique, que vous ne verrez peut-être plus jamais.
- Résoudre un problème plausible d'entreprise en un temps limité.
- Vous persuader que vos capacitś d'adaptation rapide sont bien présentes.
- Vous confronter à une façcon de penser et programmer moins académique que ce que vous avez connu (en C, C++, ..).

YASL est un langage plutôt ancien, qui fonctionne avec plusieurs piles inversées. Le détail de ce concept est présenté dans le man fournit avec l'interpréteur yasl. Vous allez donc devoir repenser votre façon de stocker et de représenter les données d'un programme.

Dans un premier temps, vous avez plusieurs petits exercices de nature assez académique afin de vous familiariser avec le langage. Une faible part de la notation mais indispensable pour la suite.
Ensuite, le gros morceau est l'évolution d'un code yasl existant dans une entreprise qui a besoin d'ajuster les fonctionnalités du traitement effectué par le programme.

# Chapitre III

# L'entrée

Pour ces premiers exercices, prévoyez une gestion des erreurs pour ne pas tomber sur un message de l'interpréteur et une interruption de votre programme.

Exo 0 : yasl_hw
Affiche le message de bienvenue. L'exemple du man est probablement un bon point de départ.

```
$> ./yasl_hw
Hello world
$>
```

Exo 1 : yasl_aff_param
Affiche les paramètres du programme, un par ligne, dans l'ordre de la ligne de commande.

```
$> ./yasl_aff_param Hello World 42 "Be Cool"
Hello
World
42
Be Cool
$>
```

```
$> ./yasl_aff_param
$>
```

Exo 2 : yasl_do <value1> <operator> <value2>
Effectue l'opération ou la comparaison. L'opérateur peut être :
+ - * / % < > <= >= == !=

```
$> ./yasl_do 21 + 42
63
$>
```

```
$> ./yasl_do 84 ">" 24
1
$>
```

Exo 3 : yasl_repeat <start_num> <value1> [ <value2> [ ...]]
Affiche chaque paramètre un certain nombre de fois. La première valeur est affichée le nombre de fois indiqué, puis la suivante une fois de plus, et ainsi de suite. Un retour à la ligne est effectué entre chaque valeur.

```
$> ./yasl_repeat 4 Bonjour "comment ca va ?"
BonjourBonjourBonjourBonjour
comment ca va ?comment ca va ?comment ca va ?comment ca va ?comment ca va ?
$>
```

```
$> ./yasl_repeat 1 "************" "******" "****" "---"
************
************
************
------------
$>
```

Exo 4 : yasl_fact <num>
Classiquement, calcule la factorielle du nombre. Profitez-en pour faire un peu de récursivité, et voir à partir de quel moment il y a un problème.

```
$> ./yasl_fact 10
3628800
$>
```

Exo 5 : yasl_split <separator> <string>
Découpe une chaine de caractères selon le séparateur (un caractère unique) et affiche chaque partie séparement sur une ligne chacune.
Si vous trouvez les 2 façons de faire, c'est que vous commencez à être au point en yasl.

```
$> ./yasl_split a ljksdpoiapoipoialljlaiuoiu
ljksdpoi
poipoi
lljl
iuoiu
$>
```

```
$> ./yasl_split " " "   mais pourquoi tant   de haine et de   vilainies ? "
mais
pourquoi
tant
de
haine
et
de
vilainies
?
$>
```

Exo 6 : yasl_interactive

Bien que l'interpréteur yasl le fasse par défaut, il s'agit ici de lire une chaine de caractères sur l'entrée standard, puis d'interpréter le yasl qu'elle contient.

```
$> ./yasl_interactive
42
21
+ "\n" +
print
63
^D
$>
```

# Chapitre IV

# Le plat de résistance

*Evolution de goupil.yasl*

C'est dans la société Goupil et companie que ce programme a été utilisé. Il a vocation à effectuer un traitement statistique sur des flux de données en provenance de leur mainframe comptable.

Chaque nuit, un batch récupère toutes les nouvelles transactions effectuées dans la journée, et réalise des statistiques dessus. De même chaque nuit de dimanche à lundi, et enfin le premier de chaque mois.

Des exemples de fichiers de données sont fournis (dans data_goupil), et génèrent des statistiques générales.

La société se déployant à l'international, son nombre d'employés est en augmentation, et il devient nécessaire de pouvoir, en plus de la balance générale et de la balance par client/fournisseur, d'avoir une balance par employé. C'est la première évolution à effectuer dans le programme. Le formatage de sortie est présent en annexe (les fichiers output).

Dans un deuxième temps, afin de mieux analyser ces mêmes statistiques sur la durée, il est demandé de mettre en début d'affichage la période concernée, en mentionnant la plus ancienne et la plus récente date de l'échantillon traité. Le formatage de sortie est également en annexe.

La correction est à base de diff, donc respectez le formatage.
A noter qu'aucune tabulation n'est présente.
Enfin, on ne va pas mixer les 2 évolutions ensembles (voir les fichiers output).
Ne gérez pas la timezone.

Les 2 nouveaux programmes s'appeleront :

```
goupil_evolution_1.yasl
goupil_evolution_2.yasl
```

# Chapitre V

# Le dessert

Si ça ne vous suffit pas, faites-donc un ensemble de Mandelbrot en yasl / ascii-art, avec une font mini et un term très grand. Il y aura bien sûr des points pour ce bonus.