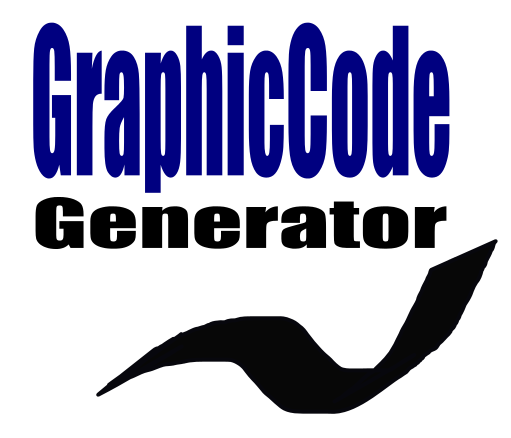
****

****

**Etudiants :** **Schaffo Raphaël et Gygi Damien**

**Date :** **03.02.2017**

**Table des matières**

[**1. Introduction** 3](#_Toc473975199)

[**2. Objectifs du projet** 3](#_Toc473975200)

[**2.1 Liste des objectifs** 3](#_Toc473975201)

[**3. Langages choisis** 3](#_Toc473975202)

[**4. Fonctionnalités implémentées** 4](#_Toc473975203)

[**4.1 Exemple de fonctionnalités** 4](#_Toc473975204)

[**4.2 Correction de code** 5](#_Toc473975205)

[**4.3 Exemple de code** 6](#_Toc473975206)

[**5. Analyse Lexicale** 7](#_Toc473975207)

[**6. Analyse Syntaxique** 7](#_Toc473975208)

# **1. Introduction**

Dans le cadre du cours de « Compilateur » de 3ème année de l’école d’ingénieur de Neuchâtel, en tant que « mini projet », il nous est demandé de créer notre propre interpréteur et/ou compilateur soit en utilisant un langage existant ou un langage qu’on a créé par nos propres soins. Ce projet devra respecter et aussi appliquer les différentes techniques, règles syntaxiques qui ont été enseigner durant les cours. On a décidé de créer un interpréteur qui génère des images formées de caractères.

# **2. Objectifs du projet**

Plus précisément, on désire traduire un code source fourni (une suite d’instruction respectant des règles prédéfinis dans notre implémentation) en une image finale. Cela à l’aide de matrices. Celle-ci seront créées et adaptées d’après les opérations du fichier source. L’image finale sera en noir et blanc.

## **2.1 Liste des objectifs**

* Création d’un dictionnaire de matrices (un caractère possédera sa propre matrice)
* Interpréter les instructions pour initialiser les matrices
* Création de fonctions pour modifier les matrices
* Utiliser les matrices pour « générer », afficher une image 2D (noir/blanc)

# **3. Langages choisis**

Plusieurs langages ont été impliqué dans notre projet. Pour commencer, on a créé l’interpréteur en python. Ensuite on avait le choix entre l’interprétation d’un langage existant ou un qu’on créer nous-même. On a décidé d’interpréter un langage que nous avons créé nous-même. C’est-à-dire qu’on a dû fixer nous même les règles de grammaires de ce nouveau langage. On a nommé ce langage le « GScode » (G pour Gygi et S pour Schaffo).

Comme entrée (input) de notre programme nous recevons du GScode que nous allons interpréter et ensuite traduire en une image de sortie(output) sous forme de matrices. Cette « image » sera affichée dans la console.

Notre langage (le GScode) s’inspire énormément des bases C/C++ comme les variables ou les boucles. De par la complexité et le temps mis à disposition, nous n’avons pas traité des aspects tels que les objets mais nous avons créé des fonctions.

# **4. Fonctionnalités implémentées**

Ce chapitre liste toute les fonctionnalités qui ont été implémenté dans notre interpréteur :

1. Assignation d’une matrice à une variable
2. Assignation d’une variable à une variable
3. Addition entre plusieurs matrices
4. Addition entre une ou plusieurs matrices et une ou plusieurs variables
5. Affichage de l’image d’une matrice
6. Affichage de l’image d’une variable
7. Répétition de matrice
8. Répétition de variable
9. Ajout de priorité avec les parenthèses
10. Utilisation de condition (if)
11. Utilisation de boucle (for)
12. Fonction de remplacement d’une matrice ou variable par une autre matrice ou variable
13. Fonction de suppression d’une matrice ou une variable dans une matrice ou variable

## **4.1 Exemple de fonctionnalités**

1. *x=’a’ ;*  Assignation de la matrice « a » à x
2. *y=x ;*  Assignation de la variable x à y
3. *z=’b’+’c’+’d’ :* Assignation d’une suite de matrice à y
4. *w=x+y+’a’ +’b’;*  Assignation de plusieurs matrices et variables à w
5. *line ‘f’ ;*  Affichage de l’image de la matrice « f »
6. *line w ;*  Affichage de l’image de la variable w
7. *v= ‘a’ mul 3 ;*  Répétition trois fois de la matrice ‘a’ et assignation à v
8. *u= v mul 2 ;* Répétition 2 fois de la variable v et assignation à u
9. *t= (u+v) mul 2 ;*  Les parenthèses traitées en premier, répétition 2 fois de u+v
10. *if(5<10){line t} ;*  Si 5 est plus petit que 10 on affiche l’image de t
11. *for(ab=0 ;ab<2 ;1){line t} ;* Assignation de 0 à ab, tant que ab plus petit que 2, on affiche t cela par pas d’incrémentation de 1
12. *x del ‘a’;* Suppression dans la variable x de la dernière occurrence de ‘a’ trouvé
13. *x rep ‘a’;’l’* Remplacement de la dernière occurrence de la matrice ‘a’

trouvée dans la variable x par la matrice ‘l'

Il est important de remarqué que chaque instruction se terminent par un point-virgule mise à part la dernière et celle se trouvant à l’intérieur d’accolade. La dernière instruction d’un programme en GScode ne doit pas contenir de point-virgule et si entre accolade dans une boucle ou une condition il y a une seule instruction elle ne doit elle non plus pas avoir de point-virgule à la fin. S’il y a plusieurs instructions dans une boucle ou une condition la dernière ne doit pas prendre de point-virgule mais les premières oui. Si une instruction se trouve après une condition ou une boucle, après l’accolade fermante de cette condition ou boucle devra figurer un point-virgule.

## **4.2 Correction de code**

Le code ci-dessous ne fonctionne pas car la dernière ligne de doit pas finir par un point-virgule comme il s’agit de la dernière instruction :

x = 'a'+'a'+'b';  
y = 'a'+'b';

Une boucle imbriquée dans une autre, ne nécessite pas de point-virgule à sa fermeture (après l’accolade fermante) comme elle est comptée comme une seule instruction :

for(abc=0; abc<2;1)  
{  
 for(def=0; def<2;1)  
 {  
 line 'd'+'c'  
 }  
};  
line 'a'

Une boucle imbriquée dans une autre nécessite un point-virgule si une deuxième instruction la suit. La boucle principale ne nécessite pas de point-virgule comme c’est la dernière instruction du programme :

for(abc=0; abc<2;1)  
{  
 for(def=0; def<2;1)  
 {  
 line 'd'+'c'  
 }  
 line 'a'  
};

Lors d’un remplacement, le caractère séparant la matrice ou la variable à remplacer et la nouvelle matrice ou variable à insérer, doit être un point-virgule, par exemple la première ligne ci-dessous fonctionnerait et le deuxième non :

x rep 'b';'e';  
x rep 'e':'j'

Le code ci-dessus ne fonctionnerait pas pour deux raisons. La première, l’instruction à l’intérieur de la boucle ne nécessite pas de point-virgule- La deuxième, la boucle étant la dernière instruction du programme ne nécessite pas de point-virgule non plus :

x=’a’+’b’;

for(bla=0; bla<3;1)  
{  
 x del 'a';  
};

Il n’est pas possible de remplacer ou supprimer plus d’un caractère (donc une matrice). Par exemple, on ne peut pas supprimer la variable x à y comme x est une matrice contenant 2 caractère :

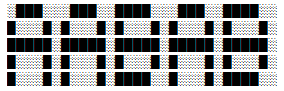
y = 'a'+'a'+'b';  
x = 'a'+'b';

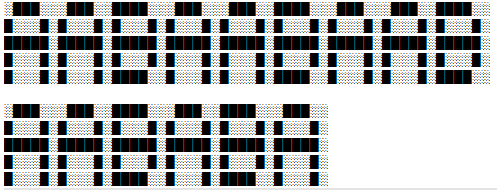
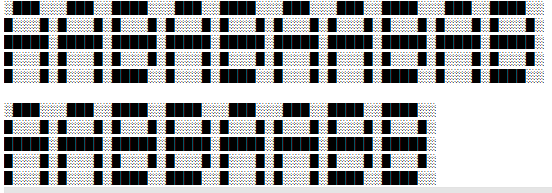
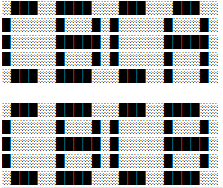
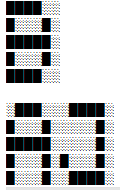
y del x

## **4.3 Exemple de code**

Voici un exemple de code fonctionnel regroupant toutes les informations citées précédemment :

x = 'a'+'a'+'b';  
y = 'a'+'b';  
line x+y;  
  
h= x mul 3;  
line h;  
  
z= x+y;  
line z + 'a';  
line z mul 2;  
  
line (x + 'b') mul 2;  
  
j= 'c'+'b'+'c'+'a';  
line j;  
line ('c'+'b') mul 2;  
  
if (500 < 300){  
 line x  
};  
  
for(abc=2; abc<2;1)  
{  
 for(def=2; def<2;1)  
 {  
 line 'd'+'c'  
 }  
};  
  
for(bla=0; bla<3;1)  
{  
 x del 'a'  
};  
  
line x;  
x rep 'b';'e';  
x rep 'e';'j';  
x rep 'j';'a';  
x=x+'j';  
line x

Il est aussi important de noté que seul les caractères « a », « b », « c », « d », « e », « f », « i », « j », « k », « l » sont disponible et utilisable dans le compilateur. Pourquoi cela, tout simplement car nous avons porté une plus grande importance à la création des fonctions, des boucles et des conditions qu’à la création d’un alphabet complet. Voici la sortie du programme ci-dessus :



# **5. Analyse Lexicale**

# **6. Analyse Syntaxique**