

# Compte rendu SAE 2.02

Vincent Miqueu-Denjean

Groupe 1\_B

## Table des matières

Introduction :	1
Simplicité 149 :	2
Simplicité 140 :	2
Simplicité 116 :	3
Simplicité 99 :	3
Efficacité 9 :	4
Efficacité 117 :	4
Efficacité18 :	7
Efficacité121 :	7
Sobriété164 :	8
Sobriété44 :	8
Sobriété145 :	9

## Introduction :

Notre texte de test est composé de 10 espaces consécutifs, de lettres et séparés par 1 – 2 ou 3 espaces aléatoirement afin de tester la fonctionnalité des algorithmes.

```
"  Cou c ou   T ex te   d e   Te st   : )  "
```

### Simplicité 149 :

Résultat : 

```
Coucou   Texte   de   Test   :)   .
```

Sans compter les commentaires, la déclaration de la fonction etc... l'algo ne contient qu'une seule ligne de code.

Ici les expressions régulières ont été utilisées (replaceAll) pour répondre au problème.

Bien que la requête avec les regex ne soit pas forcément évidente au premier coup d'œil, elle reste très simple et facilement compréhensible.

### Simplicité 140 :

Résultat : 

```
Coucou   Texte   de   Test   :)   .
```

Ce programme reste assez court (environ 25 lignes de codes) et fonctionne parfaitement.

Il perd néanmoins des points en lisibilité et en qualité car l'indentation n'est pas bien effectuée, il a des lignes sautées sans aucunes raisons et il n'y a pas le moindre commentaire.

Néanmoins il a plusieurs lignes qui ne servent à rien et auraient pu être supprimées du programme car celui-ci fonctionne très bien sans. Ces lignes rajoutent inutilement de la complexité au programme :

```
if (str == ""){  
    return str;  
}
```

```
if (string.charAt(string.length()-1)==' ') {  
    if (string.charAt(string.length()-2)!=' ') {  
        string.deleteCharAt(string.length()-1);  
    }  
}
```

Pour ce qui est du fonctionnement du programme, il est constitué d'une boucle « for » et de plusieurs « if ». Cette boucle qui utilise "charAt" passe 1 par 1 chaque caractère du texte jusqu'à tomber sur un espace. Puis elle vérifie s'il est précédé et suivi d'un espace. Si ce n'est pas le cas il est supprimé.

Cet algorithme est donc simple et résout le problème posé mais il est inutilement plus long que nécessaire, il perd donc des points dans le classement pour cette raison.

## Simplicité 116 :

Résultat :  .

Cet algo est très bien commenté et très bien indenté.


Il reste assez facile à lire et à comprendre et est court.

Il fonctionne parfaitement et répond au problème posé.

Il est composé d'une boucle « for » et 2 « if ». Son fonctionnement est quasiment similaire à l'algo 'Simplicité 140' juste au-dessus, il est donc inutile que je me répète ici.

Ce programme est donc parfait au niveau fonctionnel mais aussi au niveau des contraintes de la simplicité algorithmique même s'il n'est pas le plus simple de tous les algos de cette catégorie.

## Simplicité 99 :

Résultat : 

CE PROGRAMME NE COMPILE PAS !!!

Il manque les lignes des imports pour utiliser les ArrayList du code :

```
import java.util.ArrayList;
import java.util.Arrays;
```

Mais même en rajoutant ces imports, le code ne s'exécute pas correctement :

```
Exception in thread "main" java.lang.IndexOutOfBoundsException: Index 49 out of bounds for length 49
    at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:64)
    at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Preconditions.java:70)
    at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.java:266)
    at java.base/java.util.Objects.checkIndex(Objects.java:359)
    at java.base/java.util.ArrayList.get(ArrayList.java:427)
    at Simplicite99.erase(Simplicite99.java:15)
    at Main.main(Main.java:7)
```

Les ArrayList ne sont pas très compliqué à comprendre mais en termes de simplicité ce n'est pas le plus idéal. De plus il y a 2 méthodes, 2 boucles « for », 3 « if » et un « while » dans cet algo, ce qui le complexifie encore un peu plus.

De surcroît il n'y a aucun commentaire pour expliquer le code.

En résumé ; il n'y a rien de positif dans ce code.

## Efficacite 9 :

Résultat : **ERROR**

Les résultats sont donnés en nanosecondes et test la rapidité d'exécution.

```
36 print("efficacite-117 : ")
37 resultat = erase("      Cou c ou   T ex te   d e   Te st   : ) ")
38 print(resultat)
```

PROBLÈMES   SORTIE   CONSOLE DE DÉBOGAGE   TERMINAL   JUPYTER: VARIABLES

Windows PowerShell  
Copyright (C) Microsoft Corporation. Tous droits réservés.

Testez le nouveau système multiplateforme PowerShell <https://aka.ms/pscore6>

PS C:\Users\Utilisateur> & C:/Users/Utilisateur/AppData/Local/Programs/Python/Python39/python.exe  
cacite-9.py"  
efficacite-117 :  
Traceback (most recent call last):  
  File "c:\Users\Utilisateur\Desktop\Nouveau dossier\efficacite-9.py", line 37, in <module>  
    resultat = erase(" Cou c ou T ex te d e Te st : ) ")  
  File "c:\Users\Utilisateur\Desktop\Nouveau dossier\efficacite-9.py", line 26, in erase  
    res = res + cc[i]  
UnboundLocalError: local variable 'res' referenced before assignment  
PS C:\Users\Utilisateur>

Les 3 dernières lignes de code permettent de print ma phrase de test et de vérifier que le programme fonctionne correctement.

Ici on peut voir que le terminal affiche une erreur et que le programme ne fonctionne pas.

## Efficacite 117 :

Résultat : **Coucou   Texte   de   Test   :)   .**

Complexité constante :

**Result:**

**O(1)**

Contrairement au programme précédent, on peut voir que celui-ci fonctionne et nous retourne notre texte correctement. Le texte est bien indenté est bien commenté.

```

56 print("efficacite-117 : ")
57 resultat = erase("      Cou c ou   T ex te   d e   Te st   : ) ")
58 print(resultat)

```

PROBLÈMES   SORTIE   CONSOLE DE DÉBOGAGE   TERMINAL   JUPYTER: VARIABLES

Windows PowerShell  
Copyright (C) Microsoft Corporation. Tous droits réservés.

Testez le nouveau système multiplateforme PowerShell <https://aka.ms/pscore6>

PS C:\Users\Utilisateur> & C:/Users/Utilisateur/AppData/Local/Programs/Python/Python39/python.exe  
cacite-117.py"  
efficacite-117 :  
 Coucou Texte de Test :)  
PS C:\Users\Utilisateur> █

On vérifie ensuite sa rapidité d'exécution qui permet de voir la vitesse en nanosecondes :

```

efficacite-152 chaine de caracteres de 10 caracteres : 0.0
efficacite-152 chaine de caracteres de 100 caracteres : 0.0

```

```

efficacite-152 chaine de caracteres de 10 caracteres : 0.0
efficacite-152 chaine de caracteres de 100 caracteres : 9971.0

```

On peut obtenir ces résultats grâce au code suivant :

```

import random
import string
import time
import timeit

def efficacite_152(cc):
    stringList = []
    spaces = 0
    for char in cc:
        # espace : on le compte
        if char == " ":
            spaces+=1
        else:
            # lettre : si on a compté plus d'un espace de suite, on les met d'abord
            if spaces > 1:
                stringList.append(" "*spaces)
            # on met la lettre dans tous les cas ensuite
            stringList.append(char)
            # on remet à 0 les espaces consécutifs
            spaces = 0
    # finir d'écrire les espaces à la fin de la chaine si besoin
    if spaces > 1:
        stringList.append(" "*spaces)
    return "".join(stringList)

def test_efficacite_152(cc : str, nb : int):
    times = []

    for i in range(nb):
        start = time.time_ns();

        efficacite_152(cc)

        end = time.time_ns();

        times.append(end-start)

    return avg(times)

def avg(times : list):
    return sum(times)/len(times)

s10 = ""
for i in range(10):
    s10+=random.choice(string.ascii_letters+" ")

print("efficacite-152 chaine de caracteres de 10 caracteres : ", test_efficacite_152(s10, 100))
print('\n')

s100 = ""
for i in range(100):
    s100+=random.choice(string.ascii_letters+" ")

print("efficacite-152 chaine de caracteres de 100 caracteres : ", test_efficacite_152(s100, 100))
print('\n')

for j in range(1000, 10001, 1000):
    s = ""
    for i in range(j):
        s+=random.choice(string.ascii_letters+" ")

    print("", test_efficacite_152(s, 100))
    print(['\n'])

```

## Efficacite18 :

Résultat : **ERROR**

```
▼ C efficacite-18.c C:\Users\Utilisateur\Desktop\Nouveau dossier 2
  ✖ Erreurs #include détectées. Mettez à jour includePath. Les tildes sont désactivés pour cette unité de tradu... C/C++(1696) [Ln 1, Col 1]
  ✖ impossible d'ouvrir le fichier source "stdlib.h" C/C++(1696) [Ln 1, Col 1]
```

J'ai essayé d'ouvrir le programme sur 3 logiciels différents (VS Code, Replit, etc...)

Le programme ne fonctionne pas.

## Efficacite121 :

Résultat : **ERROR**

**Result:**

**O(1)**

**Code Analysis:**

Complexité constante

NE FONCTIONNE PAS

```
C: > Users > Utilisateur > Desktop > Nouveau dossier > C efficacite-121.c > ...
1 char* erase(char* chaine) {
2
3     char* chaine2 = calloc(strlen(chaine), sizeof(int));
4
5     int i = 0;
6     int j = 0;
7
8     while (chaine[i] != '\0') {
9         if ((chaine[i] == ' ' && chaine[i+1] != ' ' && chaine[i-1] != ' ') != 1) {
10             chaine2[j] = chaine[i];
11             j++;
12         }
13         i++;
14     }
15
16     chaine2[j] = '\0';
17
18     printf("Après la suppression des espaces : '%s' ", chaine2);
19
20
21     free(chaine2);
22 }
23
```

### Sobriete164 :

Résultat : `Coucou Texte de Test :) .`

```
efficacite-152 chaine de caracteres de 10 caracteres : 0.0
```

```
efficacite-152 chaine de caracteres de 100 caracteres : 9641.0
```

```
efficacite-152 chaine de caracteres de 10 caracteres : 0.0
```

```
efficacite-152 chaine de caracteres de 100 caracteres : 0.0
```

On peut voir que ce code s'exécute très vite.

De plus sa complexité est faible.

### Sobriete44 :

Résultat : `Coucou Texte de Test :) .`

Le programme fonctionne bien, est lisible est simple à comprendre.

On peut voir ici la vitesse d'exécution du programme :

```
<terminated> Analyse [Java  
Sobriete44 : 997500  
Sobriete44 : 997000
```

Obtenu grâce au code suivant :



```

1 package algos;
2 import java.time.Duration;
6
7 public class Analyse {
8     public static void main(String[] args) {
9         String alphabet = "Cou c ou T e x t e d e T e s t : ) ";
10        StringBuilder sb = new StringBuilder();
11        Random rand = new Random();
12
13        int longueur = 500;
14        for (int i = 0; i < longueur; i++) {
15            sb.append(alphabet.charAt(rand.nextInt(alphabet.length())));
16        }
17        String chaine = sb.toString();
18
19        Instant inst1, inst2;
20
21        inst1 = Instant.now();
22        Sobriete44.erase(chaine);
23        inst2 = Instant.now();
24
25        System.out.println("Sobriete44 : "+ Duration.between(inst1, inst2).toNanos());
26
27        inst1 = Instant.now();
28        Sobriete44.erase(chaine);
29        inst2 = Instant.now();
30
31        System.out.println("Sobriete44 : "+ Duration.between(inst1, inst2).toNanos());
32
33    }
34 }

```

## Sobriete145 :

Résultat : **ERROR**

### Code Analysis:

for(int i = 0; chaine[i] != '\0'; i++) **Error: bad syntax or infinite loop**

Le programme ne compile pas et ne s'exécute pas