

# Comparaison des algorithmes

## Table des matières

Instruction de d'exécuter des testes .....	2
Pour java : .....	2
Pour C : .....	2
Pré-requis .....	2
Reproductibilité .....	2
Références .....	2
Catégorie « Simplicité » .....	2
Lisibilité du code .....	3
Qualité du code .....	3
Efficacité .....	4
Sobriété numérique .....	4
Temps d'exécution .....	4
Classement final .....	5
Catégorie « Efficacité » .....	5
Lisibilité du code .....	5
Qualité du code .....	5
Efficacité .....	6
Sobriété numérique .....	6
Temps d'exécution .....	7
Classement final .....	7
Catégorie « Sobriété » .....	7
Code pénalisé : .....	7
Lisibilité du code .....	7
Qualité du code .....	8
Efficacité .....	9
Sobriété numérique .....	9
Temps d'exécution .....	9
Classement final .....	9

## Instruction de d'exécuter des testes

Pour java :

- Créer un nouveau projet dans Eclipse
- Ajouter les trois fichier java dans un packet
- Changer le nom de fichier java à tester en tête de fichier test.java
- Exécuter le programme test.java

Pour C :

- Accéder à replit.com
- Charger les trois fichiers .c
- Charger le programme c à tester
- Inclure le programme à tester dans fichier main.c

A noter : le string des testes sont identiques pour tous les deux langages.

## Pré-requis

- Java v1.8
- Microsoft office 365
- Eclipse 2022-03

## Reproductibilité

- Pour reproduire mon analyse :
  1. Utiliser le MS Office pour rédiger
  2. Lancez Eclipse pour tester des codes et mésuser temp d'exécution
  3. Lancer Codacy pour évaluer la qualité
  4. Accéder à replit.com pour tester des programmes de C et pour mesurer la consommation des ressources de tous les programmes

## Références

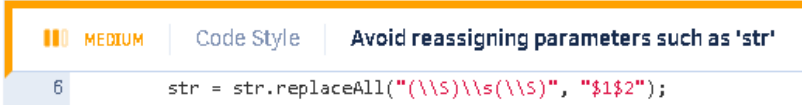
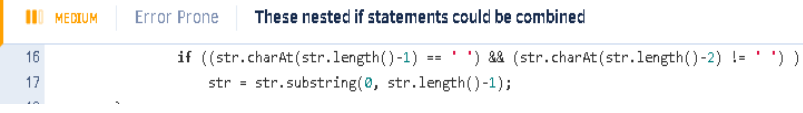
- Replit.com
- Codacy.com
- Libraire <time.h>pur mesurer de temps en C

## Catégorie « Simplicité »

## Lisibilité du code

Nom du fichier du code	Analyse	Classement
Simplicité-2.java	Code court mais peu « aéré », pas d'espaces entre les parties, bien détaillé, on comprend facilement ce que le code permet de faire.	4
Simplicité-28.java	Code court mais peu « aéré », pas d'espaces entre les parties, pas de signature, on comprend facilement ce que le code permet de faire.	2
Simplicité-49.java	Code court mais peu « aéré », pas d'espaces entre les parties, avec signature, on comprend facilement ce que le code permet de faire.	1
Simplicité-47.c	Code court mais peu « aéré », pas d'espaces entre les parties, avec signature, on comprend facilement ce que le code permet de faire.	3

## Qualité du code

Nom du fichier du code	Analyse	Classement
Simplicité-2.java	<p>1 suggestion de Codacy :</p> <ul style="list-style-type: none"> <li>- Il vaut mieux ne pas modifier la variable d'entrée comme « str », il est recommandé d'utiliser une locale.</li> </ul>  <ul style="list-style-type: none"> <li>- Des condition « if » peut être combiné.</li> </ul> 	3
Simplicité-28.java	Pas de suggestion de Codacy par rapport à l'erreur d'indentation.	2
Simplicité-49.java	Pas de suggestion de Codacy	1
Simplicité-47.c	1 suggestion de Codacy : il faut prudent avec des strings qui est terminé par « \0 » car il peut causer un over-read.	4

1	char* erase(char* chaîne) {	
2	if (strlen(chaîne) > 0) {	
3	int size = strlen(chaîne);	

## Efficacité

Nom du fichier du code	Analyse	Classement
Simplicite-2.java	Class : O(n) Double longueur->double de temp : Non	2
Simplicite-28.java	Class : O(n) Double longueur->double de temp : Non	4
Simplicite-49.java	Class : O(n) Double longueur->double de temp : Non	3
Simplicite-47.c	Class : O(n) Double longueur->double de temp : Non	1

## Sobriété numérique

Nom du fichier du code	Analyse	Classement
Simplicite-2.java	CPU :3% RAM :100MB	2
Simplicite-28.java	CPU :6% RAM :110MB	4
Simplicite-49.java	CPU :6% RAM :105MB	3
Simplicite-47.c	CPU :4% RAM :0.5MB	1

## Temps d'exécution

Nom du fichier du code	Analyse	Classement
Simplicite-2.java	Longueur standard :84 ms Double longueur : 52 ms	1
Simplicite-28.java	Longueur standard :39256 ms Double longueur : 174436 ms	2
Simplicite-49.java	Longueur standard :39524 ms Double longueur : 196589 ms	4
Simplicite-47.c	Longueur standard :39240 ms Double longueur : 184563 ms	3

## Classement final

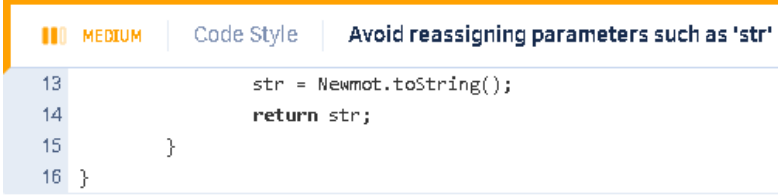
Nom du fichier du code	Classement
Simplicite-2.java	1
Simplicite-28.java	2
Simplicite-49.java	4
Simplicite-47.c	3

## Catégorie « Efficacité »

### Lisibilité du code

Nom du fichier du code	Analyse	Classement
Efficacité -3.java	Le code est court, simple et fonctionnelle mais manque de signature.	2
Efficacite -32.java	Le code est court, simple et fonctionnelle mais manque de signature.	2
Efficacite -48.c	Code court, manque d'espace, on comprend facilement ce que le code permet de faire.	4
Efficacite -52.java	Code aéré, bien détaillé, décrit et expliqué, on comprend facilement ce qu'il permet de faire.	1

### Qualité du code

Nom du fichier du code	Analyse	Classement
Efficacite-32.java	<p>1 suggestion moyenne de Codacy : Il vaut mieux ne pas modifier la variable d'entrée comme « str », il est recommandé d'utiliser une locale.</p> 	2
Efficacite -3.java	1 suggestion moyenne de Codacy : il vaut mieux de combiner ses condition « if ».	4

	<div><div><div><div><div></div><div></div><div></div></div><div>MEDIUM</div></div><div>Error Prone</div><div>These nested if statements could be combined</div></div><div><pre>8         if(str.charAt(i+1) != ' ' &amp;&amp; str.charAt(i-1) != ' ') { 9             str.deleteCharAt(i); 10        } 11    } 12    } 13    System.out.println(str); 14 } 15 }</pre></div></div>	
Efficacite -48.c	<p>1 suggestion de Codacy : il faut prudent avec des strings qui est terminé par « \0 » car il peut causer un over-read.</p> <div><div><div><div><div></div><div></div><div></div></div><div>MEDIUM</div></div><div>Security</div><div>Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).</div></div><div><pre>1 char* erase(char* chaîne) { 2     int size = strlen(chaîne); 3 }</pre></div></div>	1
Efficacite -52.java	<p>1 suggestion moyenne de Codacy : Il vaut mieux ne pas modifier la variable d'entrée comme « str », il est recommandé d'utiliser une locale.</p> <div><div><div><div><div></div><div></div><div></div></div><div>MEDIUM</div></div><div>Code Style</div><div>Avoid reassigning parameters such as 'str'</div></div><div><pre>37     str = texte; // la chaîne devient le nouveau texte 38     return str; //on retourne la nouvelle chaîne sans les esq 39 } 40 }</pre></div></div>	2

Nom du fichier du code	Analyse	Classement
Efficacite-3.java	Class : O(n) Double longueur->double de temp :	2
Efficacite -32.java	Class : O(n) Double longueur->double de temp : Non	1
Efficacite -48.c	Class : O(n) Double longueur->double de temp : Non	3
Efficacite -52.java	Class : O(n) Double longueur->double de temp : Non	4

## Sobriété numérique

Nom du fichier du code	Analyse	Classement
Efficacite-3.java	CPU :3% RAM :80MB	1
Efficacite -32.java	CPU :2% RAM :100MB	2

Efficacite -48.c	CPU :4.5% RAM :0.5MB	4
Efficacite -52.java	CPU :2% RAM :100MB	2

### Temps d'exécution

Nom du fichier du code	Analyse	Classement
Efficacite-3.java	Longueur standard :126 ms Double longueur : 2706 ms	2
Efficacite -32.java	Longueur standard :119 ms Double longueur : 503 ms	1
Efficacite -48.c	Longueur standard :39240 ms Double longueur : 184563 ms	3
Efficacite -52.java	Longueur standard :61100 ms Double longueur : 364810 ms	4

### Classement final

Nom du fichier du code	Classement
Efficacite-3.java	2
Efficacite -32.java	1
Efficacite -48.c	4
Efficacite -52.java	3

### Catégorie « Sobriété »

#### Code pénalisé :

Le code Sobriete-166.c a été mal écrit. Il est un main (pas un fonction comme attente) avec deux variables à entrer (au lieu de 1 comme attente). Donc si on veut le tester, il faut écrire un unique programme et il faut modifier beaucoup.

#### Lisibilité du code

Nom du fichier du code	Analyse	Classement
Sobriete-57.c	Code court, aéré, a facilement compréhensible, on comprend facilement ce que le code permet de faire.	1
Sobriete-67.java	Code aéré, signature mal écrite, on comprend facilement ce que le code permet de faire.	2

Sobriete-166.c	Code long, peu aéré, on a de difficulté à comprendre ce que le code permet de faire.	3
----------------	--	---

## Qualité du code

Nom du fichier du code	Analyse	Classement
Sobriete-57.c	<p>2 suggestion de Codacy :</p> <ul style="list-style-type: none"> <li>- il faut prudent avec des strings qui n'est pas « \D-terminated » car il peut causer un over-read.</li> </ul> <pre> MINOR Security Does not handle strings that are not \D-terminated; if given one it may per 5   int taille = strlen(chaine); 6   char* newChaine = malloc( sizeof(char) * taille); 7 8   while(chaine[i] != '\0'){ 9       if(chaine[i] != ' '){ 10          newChaine[j] = chaine[i]; 11          j++; </pre> <ul style="list-style-type: none"> <li>- la condition chaine[i]==' ' est toujours vrai</li> </ul> <pre> MINOR Code Style Condition 'chaine[i]==' ' is al 13         }else if(chaine[i] == ' ' &amp;&amp; chaine[i+1] == 14             newChaine[j] = chaine[i]; 15             j++; 16 17         }else if( i &gt; 0 &amp;&amp; chaine[i] == ' ' &amp;&amp; chai 18             newChaine[j] = chaine[i]; 19             j++; 20 21         } 22         i++; 23 24     } 25 26     newChaine[j] = '\0'; 27 28     return newChaine; 29 } </pre>	3
Sobriete-67.java	Pas de suggestion de Codacy.	1



Sobriete-166.c	Pas de suggestion de Codacy.	1
----------------	------------------------------	---

## Efficacité

Nom du fichier du code	Analyse	Classement
Sobriete-57.c	Class : O(n) Double longueur->double de temp : Oui	1
Sobriete-67.java	Class : O(n) Double longueur->double de temp : Non	2
Sobriete-166.c	Code inexécutable	0

## Sobriété numérique

Nom du fichier du code	Analyse	Classement
Sobriete-57.c	CPU :4% RAM :0.5MB	1
Sobriete-67.java	CPU :3% RAM :80MB	2
Sobriete-166.c	Code inexécutable	0

## Temps d'exécution

Nom du fichier du code	Analyse	Classement
Sobriete-57.c	Longueur standard :321 ms Double longueur : 653 ms	2
Sobriete-67.java	Longueur standard :100 ms Double longueur : 127 ms	1
Sobriete-166.c	Code inexécutable	0

## Classement final

Nom du fichier du code	Classement
Sobriete-57.c	1
Sobriete-67.java	2
Sobriete-166.c	0