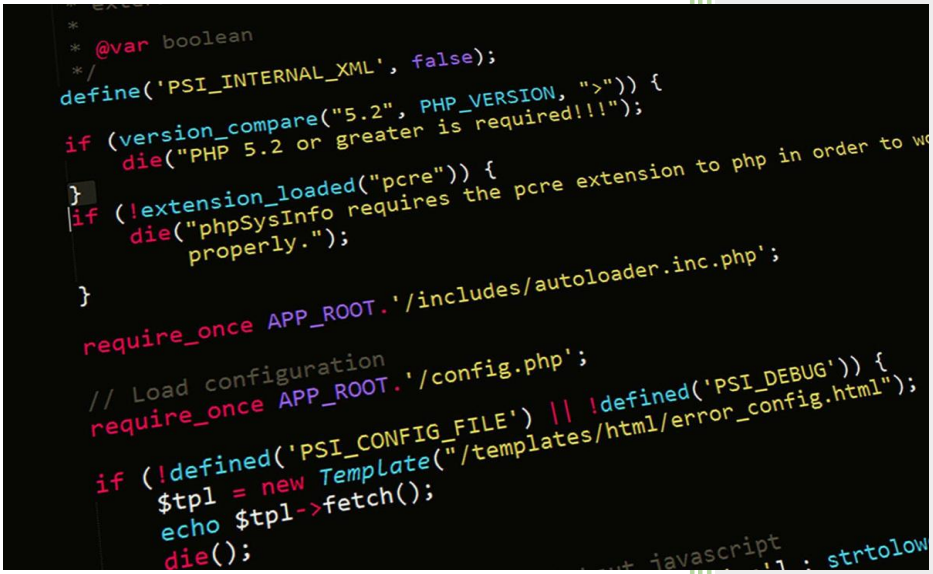
2020

HuFFman : Programmation C++



Réalisé par :

Rania BEN DHIA

Marwan ALOMARI

15/10/2020

Veuillez trouver ci-joint les fichiers texte.cpp, texte.h, main.cpp, et input.txt

1. Codage de Huffman

1.1 Préambule

Les lettres affichées par la suite, correspondent à des fréquences bien déterminées.

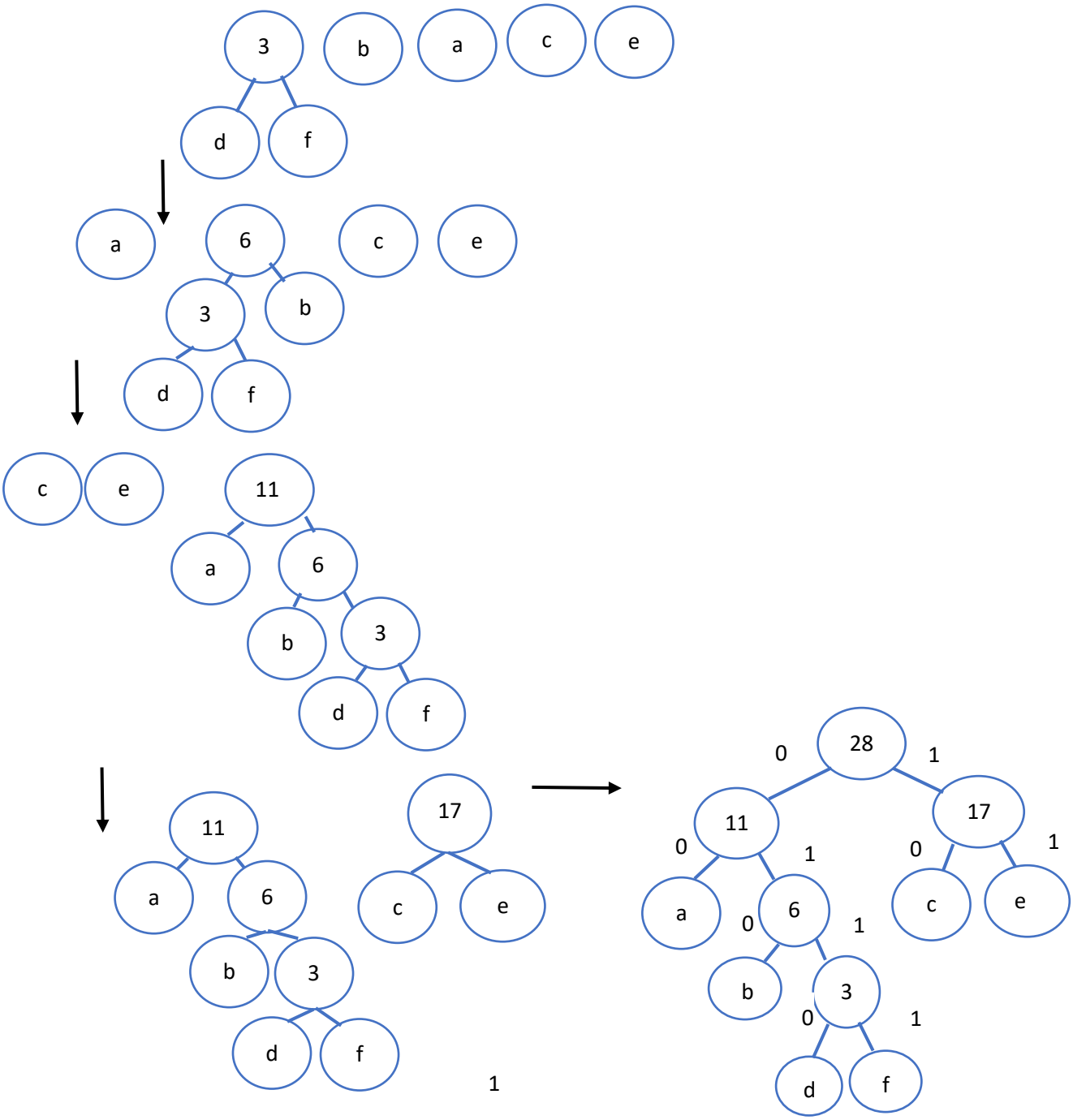
	a	b	c	d	e	f
Occurences	5	3	7	1	10	2

En les mettant en ordre croissant, on va obtenir le tableau suivant :

d	**f**	**b**	**a**	**c**	**e**
1	2	3	5	7	10

1.2 Construction de l'arbre

Afin de construire l'arbre de **Huffman**, on a pu suivre la démarche suivante :



Les six codes binaires de notre arbre seront :

- a = 00
- b = 010
- c = 10
- d = 0110
- e = 11
- f = 0111

En fait, **dans le fichier Texte.cpp**, on va initialiser tout d'abord le constructeur, le déconstructeur, les accesseurs et les mutateurs.

On va utiliser ai--nsi, la fonction **HuffManCodes** qui va parcourir un vecteur rempli de toutes les lettres (a,b,c,d,e,f), extraire le minimum entre deux nœuds en respectant la règle de priorité inclue dans STD, et enfin l'enlever pour passer au prochain traitement.

Pour cela, on a utilisé le 'top', 'pop' et le 'push' afin d'insérer le nœud dans Q (queue).

Ainsi, pour la conversion de la chaine à un tableau, on va utiliser la collection 'set'. Ceci afin d'éviter la redondance.

Dans le fichier Texte.h, on a utilisé la structure du nœud donné par l'énoncé ainsi la structure de comparaison entre les fréquences.

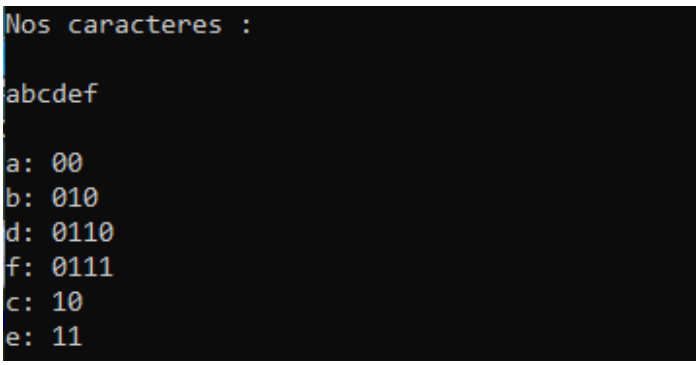
La classe Texte va appeler donc, toutes les fonctions membres utilisées.

Dans le fichier main.cpp, on a défini notre liste ainsi que les fréquences à traiter à la fonction Huffmancodes pour créer l'arbre et coder les nœuds.

1.3 Affichage des codes binaires

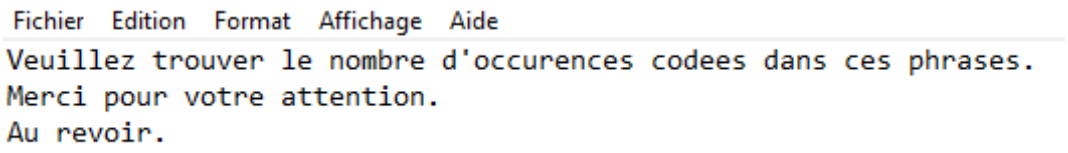
Pour l'affichage, on a utilisé la fonction **« afficherHuffman »,** qui affiche à la fin du programme l'arbre complet avec chaque lettre correspondante à son code binaire.

Le résultat sera :



2. Et pour le texte ?

Notre fichier texte est montré par la suite :



On va obtenir le résultat suivant :

```
notre chaine :
'Veuillez trouver le nombre d'occurences codees dans ces phrases.'
V: 00
 : 010
': 011
d: 100
a: 1010
z: 10110000
p: 10110001000
h: 1011000010010000
l: 101100010010001
m: 101100010010010
s: 101100010010011
u: 101100010010110
v: 101100010010111
r: 101100010011
i: 1011000101
n: 101100011
o: 1011001
b: 101101
.: 10111
e: 110
c: 111
```

En conclusion, on a pu manipuler les données à travers le codage de Huffman, en les compressant.

On peut dire donc, que les arbres binaires facilitent le traitement des données, ou autrement dit, le contenu de chaque nœud.

Cette gestion de données offre la recherche d'une clé, l'insertion d'une clé ou même la suppression d'une clé.