

TABLE DE MATIERE

INTRODUCTION

- **PRESENTATION DU PROGRAMME.**
 - Fenêtre principale du programme
 - Section candidat
 - Section électeur
 - Partie Vote
 - Affichage des statistiques du vote
- **ORGANISATION DU TRAVAILLE.**
 - Recherche et organisation des d'idées
 - Description des fonctions principales du programme et de leurs différents modes de fonctionnements.
 - La fonction `int main (int argc, char* argv [])`.
 - La fonction `void GestionCandidat(void)`.
 - La fonction `void GestionElecteur(void)`.
 - La fonction `int Voter(void)`
 - La fonction `void AfficheStatistiques(void)`
 - Assemblage et liaison des fonctions
- **RECAPITULATIF DES FONCTIONS DU PROGRAMME LEUR ROLES RESPECTIFS.**
 - Quelques problèmes rencontrés.

CONCLUSION

INTRODUCTION

De jour en jour, l'Homme traverse de nombreux problèmes dont la résolution peut l'amener à penser aux solutions informatiques. Le programme ainsi proposé au terme de notre travail est un mini-système électoral programmé en langage c, fonctionnant en mode console et pouvant être utilisé pour gérer une élection. C'est un programme qui assure en gros des fonctionnalités telles que l'enregistrement des électeurs et des candidats, le vote des candidats par les électeurs et affiche les statistiques du vote. Ce document présente dans les détails notre travail et son évolution ; la démarche que nous nous sommes fixés à suivre pour arriver au résultat et les différentes difficultés que nous avons rencontrés durant la réalisation du projet. Il contient le fruit d'un très long travail qui a réuni un groupe de 8 étudiants de 3^{eme} années mathématique de l'Université de Yaoundé 1.

- **PRESENTATION DU PROGRAMME.**
 - Fenêtre principale du programme

```
*****
*                               *
*      MINI SYSTEME ELECTORALE  *
*                               *
*****

Menu du systeme:
1)-GESTION DES CANDIDATS
2)-GESTION DES ELECTEURS
3)-VOTER
4)-CONSULTER LES STATISTIQUES
5)-REINITIALISER
6)-QUITTER

Quelle est votre choix:
```

Ceci est la fenêtre principale du programme ou l'utilisateur peut faire le choix qu'il désire selon les options qui lui sont proposé. Ainsi il peut donc choisir une option en indiquant juste le numéro de l'option qu'il désire (Exemple **1** pour entrer dans la partie du programme qui gère les **candidats** ou **2** dans celle qui gère les **électeurs**).

- Section candidat

C'est elle qui gère tous ce qui concerne le candidat il suffit juste le numéro correspondant à ce qu'on veut faire. À partir d'ici on peut :

- Soit inscrire un nouveau candidat ("2") :
- Soit afficher la liste de tous les candidats inscrit ("1") :

```

Menu :
1)-Consulter la liste des candidats déjà inscrit
2)-Ajouter un nouveau candidat
3)-Modifier les informations d'un candidat
4)-Quitter

Votre choix: 1

-----> LISTE DES CANDIDATS INSCRITS
-----
| Candidat N° | Identifiant | Numero de CNI | Nom | P
-----
| 1 | 1 | 15Y839 | NGUEMECHIO | KESSEL
-----
| 2 | 2 | 12424 | NKS | SYLVIO
-----
| 3 | 3 | 15244 | FORCE | BRICE
-----
| 4 | 4 | 125444 | WAMBA | FLEURE
-----
| 5 | 5 | 111111 | PRESIDENT | 1
-----
Press any key to continue . . .

```

- Soit modifier un candidat ("3") :

```

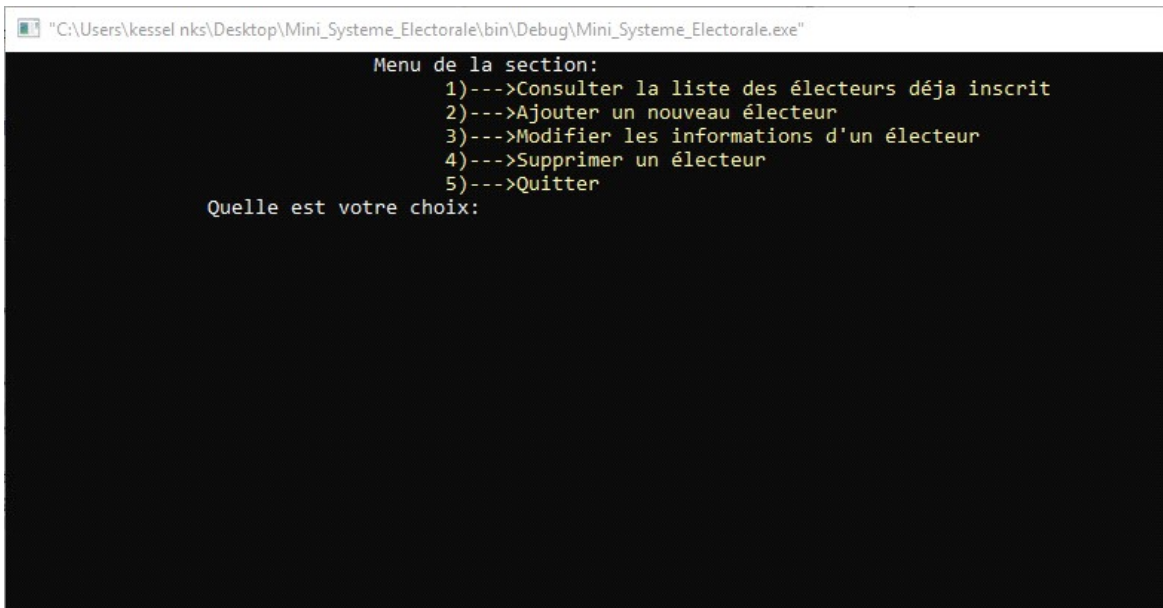
-----
| 1 | 55154 | NNN | DFJH
-----
| 2 | 1545RL | BMC | BERTHOLD
-----
| 3 | 15YNKS | NKS | KESSEL
-----
| 4 | 1475DD | NGUEME | NGUEME
-----
Entrez le numero de l'électeur: 2
Vous ete entreint de vouloir modifier les informations de l'electeur BERTHOLD B
êtê vous sûre de vouloir continuer? y/n ?

Info de l'electeur BERTHOLD BMC:
-----
Electeur N°:2
-->Nom:BMC
-->Prenom:BERTHOLD
-->Sexe:M
-->Numero de CNI:1545RL
-----
Nouvelles informations de l'électeur :
Nom: bertyhg

```

- Section électeur

Ici on peut faire tout comme pour les électeurs mais la seule différence qu'ici, on peut supprimer un électeur :



```
"C:\Users\kessel nks\Desktop\Mini_Systeme_Electorale\bin\Debug\Mini_Systeme_Electorale.exe"

Menu de la section:
1)--->Consulter la liste des électeurs déjà inscrit
2)--->Ajouter un nouveau électeur
3)--->Modifier les informations d'un électeur
4)--->Supprimer un électeur
5)--->Quitter

Quelle est votre choix:
```

- **LE VOTE**

Cette partie concerne les électeurs. Pour voter, ils doivent fournir leur numéro (reçu lors de l'enregistrement en tant qu'électeur) et une fois le numéro validé il devra choisir un candidat parmi tous ceux qu'on lui proposera. Il peut décider de voter le **bulletin Nulle** ou **d'amorcer** son vote et revenir une prochaine fois.

```
"C:\Users\kessel nks\Desktop\Mini_Systeme_Electorale\bin\Debug\Mini_Systeme_Electorale.exe"

----- Le vote !!!!! -----

Entrer votre numero: 1
Electeur:jhg bertyhg

Vous avez déjà attribué votre voie à un candidat. Desolé vous ne pouvez plus voter
Press any key to continue . . .
```

- **AFFICHER LES STATISTIQUES DU VOTE**

```
----- Statistique du vote -----
Nombres de personne ayant voté 1

----- Resultat du vote -----
-Candidat: KESSEL NGUEMECHIO      -Nombre de voies: 0   -Pourcentage: 0.00
-Candidat: SYLVIO NKS             -Nombre de voies: 0   -Pourcentage: 0.00
-Candidat: BRICE FORCE             -Nombre de voies: 1   -Pourcentage: 100.00
-Candidat: FLEURE WAMBA           -Nombre de voies: 0   -Pourcentage: 0.00
-Candidat: 1 PRESIDENT            -Nombre de voies: 0   -Pourcentage: 0.00

Press any key to continue . . .
```

- **ORGANISATION DU TRAVAILLE.**

- **Recherche et organisation des d'idées**
- Vu que notre programme va gérer deux catégories de personne (Candidat aux élections et électeurs) et deux options supplémentaires sur ces deux catégories de personne nous avons jugé nécessaire de faire un menu (et

comme nous a même recommande le cahier de charge) pour notre programme qui va comprendre 4 grandes parties (**Gestion des candidats ; Gestion des électeurs ; Voter ; Afficher les statistiques du vote**) et une option pour quitter le programme. Ainsi sans sans toucher le clavier on a défini quatre fonctions essentielles du programme (sans oublier la fonction main) de prototypes :

- `Void GestionCandidat (void) ;`
- `Void GestionElecteur (void) ;`
- `Int Voter (void) ;`
- `Void AfficheStatistique (void) ;`
- De plus étant donné que chaque électeur et chaque candidat sera grossièrement identifier par : son nom, son prénom son sexe, son numéro de CNI alors l'usage des structures comme représentant d'un électeur ou d'un candidat a été la meilleure option pour nous. Ainsi on a donc créé deux structures :
 - `typedef struct Candidat {`
`int identifiant ;`
`char nom [TAILLE_MAX] ;`
`char prenom [TAILLE_MAX] ;`
`char CNI [10] ;`
`char sexe ;`
`int nbre_de_voie ;`
`} Candidat ;`
 - `typedef struct Electeur{`
`int numero ;`
`char nom [TAILLE_MAX] ;`

```

    char prenom [TAILLE_MAX] ;

    char CNI [10] ;

    char sexe ;

    int voie ;

} Electeur ;

```

Ou ici **TAILLE_MAX** est une constante.

- Enregistrer un candidat ou un électeur pour nous c'est enregistrer ces informations dans un fichier ouvert au préalable en mode ajout en fin de fichier (pour compléter la liste). Donc nous avons créé deux fichiers texte ou l'un va contenir tous nos électeurs que l'on va appeler **electeur.txt** et l'autre nos candidats que l'on va appeler **candidat.txt**.
- D'autre part selon le cahier de charge l'utilisateur doit être capable de d'ajouter un nouveau candidat ; modifier un candidat ; d'afficher la liste des candidats et des électeurs ; d'ajouter un électeur ; de modifier un électeur ; de supprimer un électeur. On a donc créé **7** nouvelles fonctions de prototype :

```

int AfficherListeCandidats (FILE* fichier);
void AjouteCandidat(FILE* enregFile, FILE* sourceFile, int option, int nbre_de_voie);
int ModifierCandidat (void );
int AfficherListeElecteurs (FILE* fichier);
void AjouteElecteur (FILE* enregFile , FILE* sourceFile , int option);
int ModifierElecteur (void);
int SupprimerElecteur (void);

```

- Etant donné que le cahier de charge nous impose qu'il ne pas avoir de doublon de candidat et de personnage nous avons ajoutons à ces fonctions déjà définies deux autres fonctions de prototypes :

```

char CandidatIsDouble(FILE* fichier, Candidat candidat , int option_modification);
char ElecteurIsDouble(FILE* fichier, Electeur electeur , int option_modification);

```

Vu cette longue suite de fonction nous avons séparé le coder dans trois fichiers : un fichier main que tous les projet ont évidemment, un fichier **election.c** qui contient tous les implémentations de tous nos fonctions et un fichier **election.h**

qui contient lui le prototype de tous nos fonctions les définitions de nos structure et de nos constantes/variables.

(Nous détaillerons le rôle de chacune de ces variables et de ces constantes plus bas)

- **Description des fonctions principales du programme et de leurs différents modes de fonctionnement**

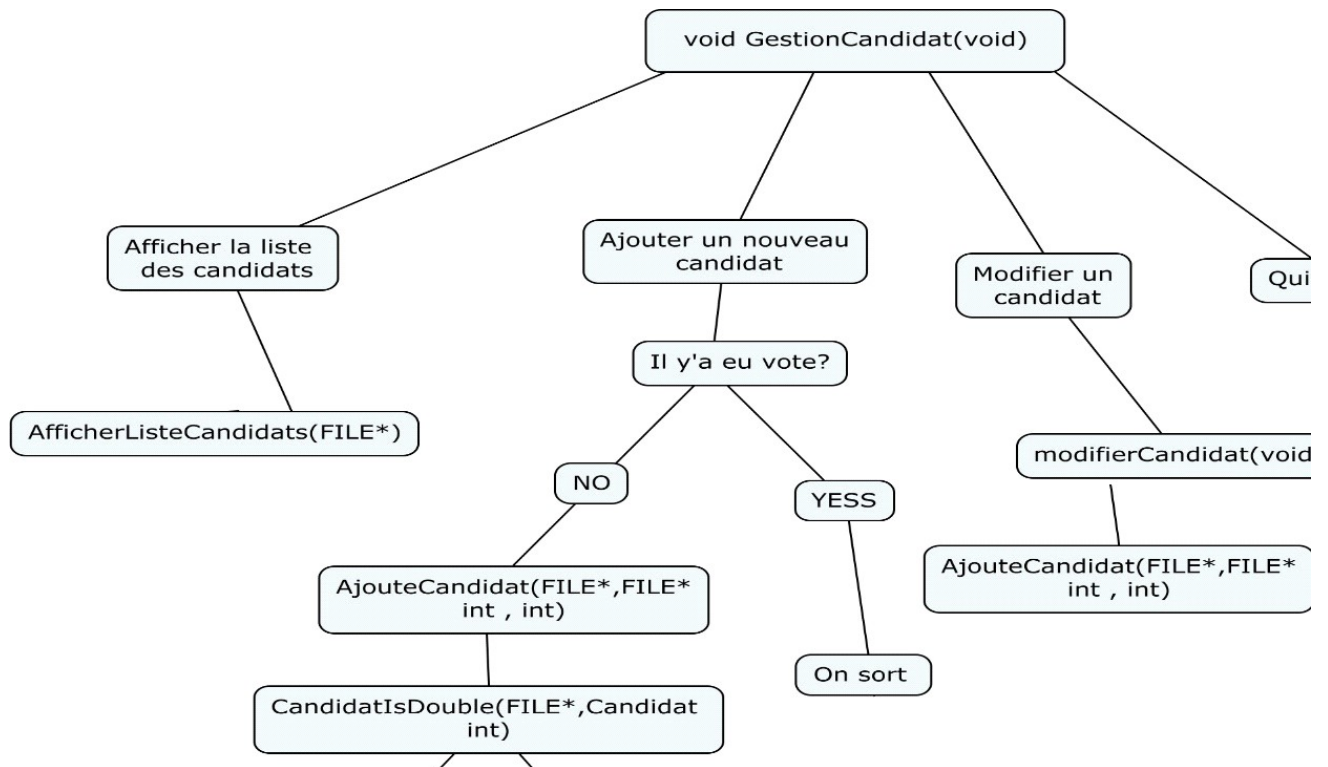
- La fonction `int main (int argc, char* argv [])`.

Elle a pour rôle principale d'afficher le menu du programme ; de récupérer le choix de l'utilisateur et de le renvoyer vers la fenêtre qui correspond à son choix en appelant la bonne fonction on fait donc un switch sur la valeur entrer par l'utilisateur :si c'est **1** alors on appelle **GestionCandidat ()** ; Si c'est **2** on appelle **GestionElecteur ()** ; Si c'est **3** on appelle la fonction **Voter ()**

- La fonction `void GestionCandidat(void)`

C'est elle qui gère tout ce qui concerne les candidats : L'ajout d'un candidat ; la modification d'un candidat et l'affichage de la liste de tous les candidats.

Schéma résumant le principe de fonctionnement de GestionCandidat ()



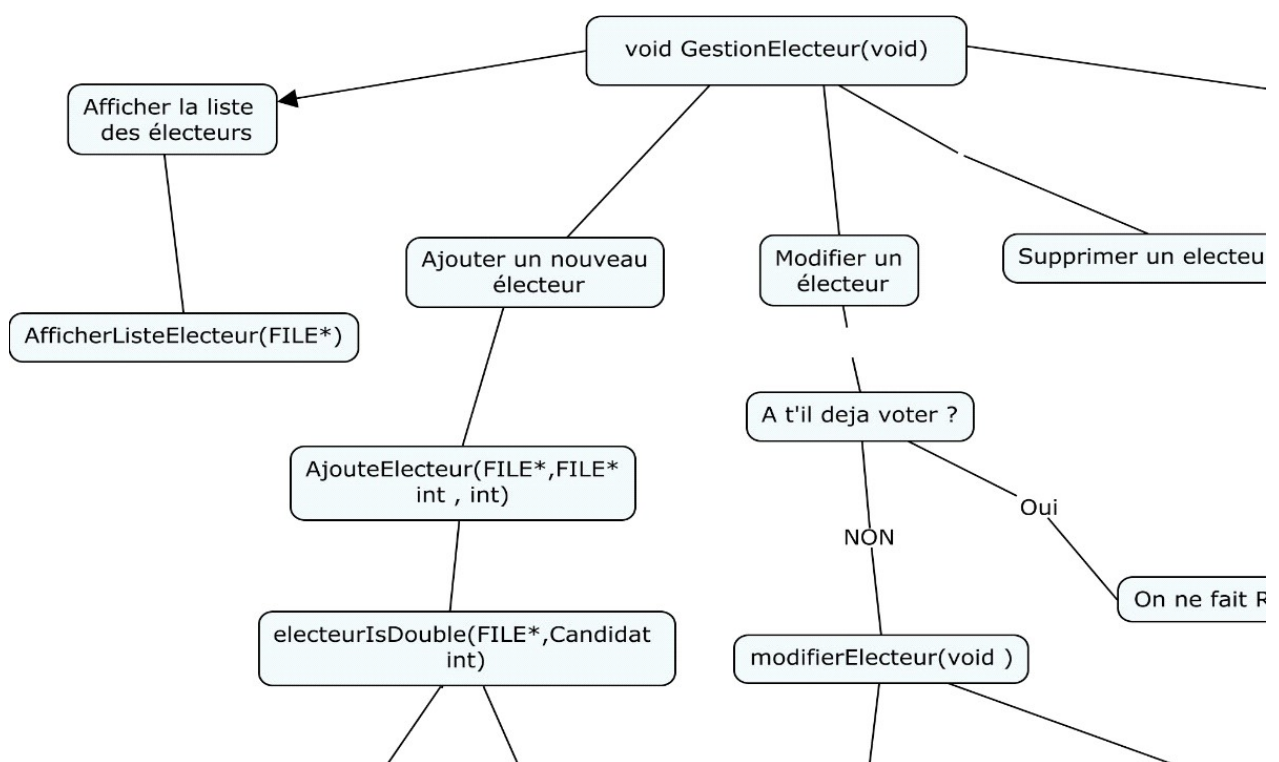
- Ici pour ajouter un candidat ,on ouvre le fichier candidat.txt en mode ajout en fin de fichier puis on demande à l'utilisateur d'entrer ses informations qu'on enregistre dans une structure du type Candidat et ensuite on teste s'il y'a pas déjà eu un candidat qui est déjà enregistré avec soit le même nom ;prénom ou numero de CNI enregistré dans la structure (car ces information sont unique a une personne) et ceci a l'aide de la fonction **candiatsDouble (FILE* fichier , Candidat candidat , int option)** ou ici **fichier** correspond au fichier contenant nos candidats ; **candidat** est le candidat que nous souhaitons vérifier s'il n'est pas un doublon et **option** est une variable un peu spéciale elle permet d'autoriser les doublons. Elle est activée que lorsque on modifie un candidat. Si le test étant réussi on enregistre donc la structure dans le fichier **candidat.txt** sinon on annule le processus d'ajout d'un nouveau candidat. (Dans le respect du cahier on n'ajoute que les candidats quand il y'a pas encore eu vote)
- Modifier un candidat consiste à créer un nouveau fichier texte temporaire ; de l'ouvrir en mode ajout en fin de fichier dans lequel on mettra dans une première partie les informations de tous les candidats du fichier **candidat.txt** qui se trouve avant lui ensuite le nouveau candidat modifier et enfin le reste des candidats qui se trouvait après lui dans **candidat.txt**. Après

pour finir on supprime le fichier original **candidat.txt** et on renomme le fichier temporaire créé en **candidat.txt**.

- La fonction [void GestionElecteur\(void\)](#).

C'est elle qui gère tout ce qui concerne les électeurs : L'ajout d'un électeur ; la modification d'un électeur et l'affichage de la liste de tous les électeurs, la suppression d'un électeur.

Schéma résumant le principe de fonctionnement de GestionElecteur ()



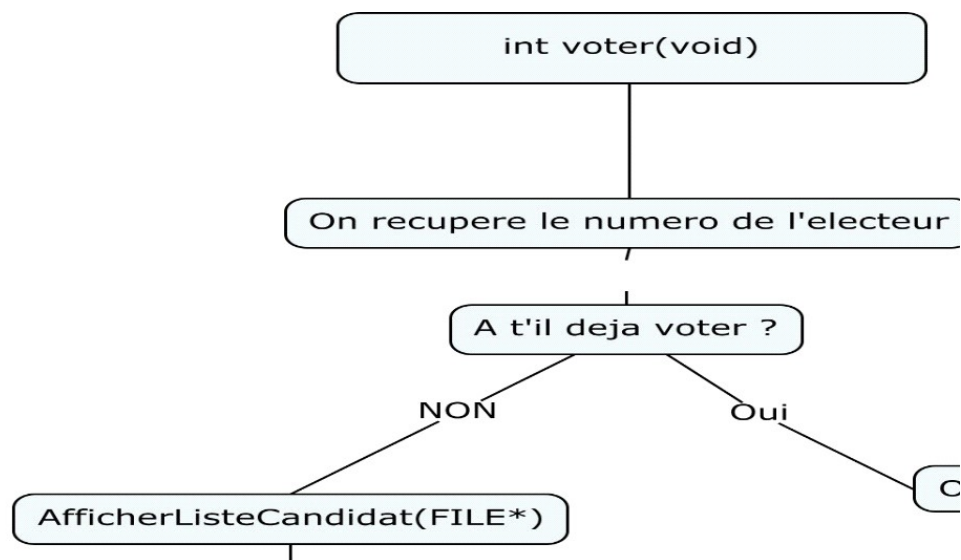
- Ici pour ajouter un électeur ,on ouvre le fichier **electeur.txt** en mode ajout en fin de fichier puis on demande à l'utilisateur d'entrer ses informations qu'on enregistre dans une structure du type `Electeur` et ensuite on teste s'il y'a pas déjà eu un électeur qui est déjà enregistré avec soit le même nom ;prénom ou numero de CNI enregistrer dans la structure (car ces information sont unique a une personne) et ceci a l'aide de la fonction **electeursIsDouble (FILE* fichier , Electeur electeur, int option)** ou ici **fichier** correspond au fichier contenant nos candidats ; **electeur** est le candidat que souhaitons vérifier s'il n'est pas un doublon et **option** est une variable un peu spéciale elle permet d'autoriser les doublons. Elle est activée que

lorsque on modifie un électeur. Si le test étant réussi on enregistre donc la structure dans le fichier **electeur.txt** sinon on annule le processus d'ajout d'un nouvel électeur. (Dans le respect du cahier on n'ajoute que les électeurs quand il y'a pas encore eu vote)

- Modifier un électeur consiste a créé un nouveau fichier texte temporaire ; de l'ouvrir en mode ajout en fin de fichier dans lequel on mettra dans un première partie les informations de tous les électeurs du fichier **electeur.txt** qui se trouve avant lui ensuite le nouvel électeur modifier et enfin le reste des électeurs qui se trouvait après lui dans **electeur.txt**. Après pour finir on supprime le fichier original **electeur.txt** et on renomme le fichier temporaire créé en **electeur.txt**.
- Supprimer un electeur se fait comme dans le cas de la modification d'un electeur a la seule différence qu'on n'appelle plus ici la fonction **AjouteElecteur(...)** ;

- La fonction `int Voter(void)`

C'est elle qui gère la partie du vote son principe de fonctionnement est généralisé par le schéma suivant :



- La fonction `void AfficheStatistiques(void)`

Elle a pour rôle d'afficher tous les candidats avec leur pourcentage de gain.

III Récapitulatif fonctions et leurs rôles respectifs

// fonction pour la partie candidat

void GestionCandidat(**void**); (son fonctionnement a déjà été expliquer).

int AfficherListeCandidats(**FILE*** fichier); (affiche la liste de tous les candidats).

void AjouteCandidat(**FILE*** enregFile , **FILE*** sourceFile , **int** option , **int** nbre_de_voie); (ajoute un nouveau candidat a liste de tous les candidats).

int ModifierCandidat(**void**);(modifie un candidat).

int NombreDeCandidat(**FILE*** fichier);(compte le nombre de candidat présent dans le fichier qu'on lui passe en paramètre).

char CandidatIsDouble(**FILE*** fichier,**Candidat** candidat , **int** option_modification);
(vérifie si un candidat n'est pas un doublon).

// fonction pour la partie electeur

void GestionElecteur(**void**); (son fonctionnement a déjà été expliquer).

int AfficherListeElecteurs(**FILE*** fichier); (affiche la liste de tous les candidats).

void AjouteElecteur(**FILE*** enregFile , **FILE*** sourceFile , **int** option); (ajoute un nouveau candidat a liste de tous les candidats).

int ModifierElecteur(**void**); (modifie un candidat).

int SupprimerElecteur(**void**);(supprime un electeur)

int NombreDelecteur(**FILE*** fichier); (compte le nombre d'électeurs présent dans le fichier qu'on lui passe en paramètre).

char ElecteurIsDouble(**FILE*** fichier,**Electeur** electeur , **int** option_modification);

// fonctions pour la partie voter

int Voter(); (Gere tous la partie qui concerne le vote des candidats part les électeurs).

// gestion des statistiques du vote

void affiHeStatistique();(affiche les statistique du vote).

// Extra

`void color(int t,int f);`(Pour la gestion des couleurs)

`void traiteChaine(char* chaine);`(Pour traiter les chaines de caractère : enlever par exemple le '\n' etc).

`int saisirNombre(void);`

- Quelques problèmes rencontrés
 - **L'affichage calibré** : ce programme a été réaliser sous windows 10 et vu que le fonctionnement des consoles des systèmes d'exploitation diffère, l'affichage ne sera donc pas la même selon les systèmes d'exploitation donc affichage calibré sous la console de windows 10 ne sera pas nécessairement calibré dans la console de windows 7. Ainsi notre programme n'aura pas dans certains systèmes d'exploitation un affichage calibré.

CONCLUSION

Tout au long de notre projet, le travail consistait à programmer un mini système électoral en langage C. Cette tâche a été réalisé grâce à l'utilisation des fonctions, fichiers, tableaux, pointeurs, structures conditionnelles (if et switch), boucles (while, do ... while) et les structures (électeur et candidat).