**Architecture logiciel**

**NettoyageEcran.h**

**Statique.h**

**Historique.h**

**Pile.h**

**Avion.h**

**Ciel.h**

**Dynamisme.h**

**PBM.h**

**Stats.h**

**Clavier.h**

**Variable\_ENV.h**

**Bibliothèque**

**PBM.h**

Struct PBM ;

Tableau dynamique ;

PBM chargerImageDansTableau (char \*nomfichier) ;

PBM lectureImage (\*tableau) ;

PBM afficherImageInversee (\*tableau) ;

Void choixRandomLimage (char \*nomfichier) ;

**Statique.h**

Void afficherImage (\*nomFichier) ;

**Pile.h**

Struct Element ;

Struct Pile ;

Pile \*initialiser () ;

Void empiler (Pile \*pile, int nombre) ;

Int depiler (Pile \*pile) ;

**Stats.h**

Void creerParametre () ;

Void ?

**Clavier.h**

Void detecterLaTouche () ;

**Dynamisme.h**

Void afficherPhrase () ;

Int afficherPointsPhrase () ;

Void afficherHorloge (\*nomFichier) ;

Void actualiserPage (n) ;

**Historique.h**

Struct Temps ;

Struct Historique ;

Struct Taille ;

Temps enregistrerDate (int, int) ;

Temps enregistrerHeure (int, int, int) ;

Historique enregistrerNumeroEcran (int) ;

Historique enregistrerTailleChiffre (int, int) ;

**Avion.h**

Struct Avion ;

Void initialiserAvion () ;

Void afficherAvion () ;

**Variable\_ENV.h**

Void lireVariableEnvironnement ();

?

**NettoyageEcran.h**

Void nettoyageEcran() ;

**Ciel.h**

Struct Ciel ;

Void initialiserCiel () ;

**Description des fonctions**

**Stats.h :** - on créer une fonction qui permettra de créer le paramètre -stats pour la console.

**VariableEnvironnement.h :** - cette fonction nous permettra de lire toutes les variables d’environnement afin de les utiliser et de les stocker pour ensuite faire notre historique par exemple.

**Avion.h :** - cette fonction servira à initialiser l’avion, c’est-à-dire à le placer aléatoirement dans la console.

- cette fonction nous permettra d’afficher l’avion.

**Historique.h :** - la fonction enregistrement de la date va nous permettre de récupérer les données de la date courante dans des variables.

- la fonction enregistrement de l’heure va nous permettre de récupérer les données de l’heure courante dans des variables.

- cette fonction va stocker le numéro de l’écran lancé dans une variable.

- cette fonction va stocker la taille des caractères de l’horloge dans des variables si l’écran dynamique est lancé.

**Ciel.h :** - la fonction va initialiser le ciel, c’est-à-dire qu’on va définir la taille du ciel (la taille de la console en gros).

**PBM.h :** Ce header répertorie les structures et les fonctions utilisées pour la manipulation des PBM.

Pour cela, on définit une structure PBM qui sera un tableau dynamique.

Ensuite, on appelle la fonction chargerImageDansTableau afin de transcrire une image PBM en informations rentrées dans un tableau à deux dimensions.

La fonction lectureImage permet la lecture de l'image à travers la lecture du tableau.

Puis, afficherImageInversee permet de lire mais cette fois-ci à l'envers le tableau afin d'obtenir une image inversée.

Enfin choixRandomImage permet d'obtenir une image PBM tirée aléatoirement parmi plusieurs afin de n'en utiliser qu'une seule et unique dans le cas de l’écran statique.

**Pile.h:** Structure Pile et structure élément (vue dans les anciens prosits) pour manipuler des listes chaînées.

Initialisation de la liste chaînée par Pile \*initialiser.

Fonctions d'empilage et de dépilage à notre guise de la liste chaînée.

**Dynamisme.h:** Header spécifique à l'écran dynamique.

AfficherPhrase va permettre d'afficher la phrase en dessous de l'heure.

AfficherPointPhrase nous permet d'afficher les points après la phrase afin de voir quand est-ce que le rafraichissement de l'horloge va se faire avec un paramètre n.

AfficherHorloge affiche l'horloge à travers la lecture de chaîne de caractères (avec ASCII).

ActualiserPage correspond au refresh de l'heure sur l'horloge.

**Clavier.h:** DetecterLaTouche aura pour but de capter la touche sur laquelle l'utilisateur va appuyer pour envoyer l'information correspondante sur le programme en question (par exemple -> pour aller vers la droite).

**NettoyageEcran.h**: Cette fonction va nous permettre de vider la console avec l’instruction system(« clear »).

Struct Pile

{

Element \*premier ;

}

**STRUCTURES**

Struct PBM

{

Int largeurDeLimage ;

Int longueurDeLimage ;

Char nomFichier [] ;

}

Struct Temps

{

Int Tm\_mday ;

Int Tm\_mon ;

Int Tm\_hour ;

Int Tm\_min ;

Int Tm\_sec ;

}

}

Struct Element

{

int nombre;

Element \*suivant ;

}

Struct Avion

{

Int x ;

Int y ;

Struct ciel ;

}

Struct Ciel

{

Int largeur ;

Int longueur ;

Struct Avion ;

}

Struct Taille ;

{

Int colonne ;

Int ligne ;

}

Struct Historique

{

Struct Taille ;

Struct Temps ;

Int numeroFichier ;

}

**+ Tableau dynamique à deux dimensions pour les différentes tailles des caractères.**