Présentation de la réalisation

1. **Choix de la structure de données dans la STL**

Nous voulions créer une structure de données de type arbre qui permettrait une recherche rapide d'élément à partir d'un autre élément : une clé, nous nous sommes donc tournés vers la structure *map* qui permet de lier un élément clé unique au sein de la structure à un autre élément. Dans la classe References, nous avons introduit une variable de type map ayant comme clés des identificateurs pointant vers un objet de type AssocRefFichier. Ce dernier comporte aussi une variable de type map ayant comme clés des noms de fichiers où sont présents les identificateurs qui pointent à leur tour sur des ensembles d'entiers. Pour représenter cette information, nous avons choisi une structure de données de type *vector* car nous n’avons pas besoin de trier les numéros de lignes. L’ajout des lignes a lieu en même temps que la lecture du fichier se fait.

1. **Interface et réalisation de la classe**

* Interface

// Rôle de la classe <References>

//References est une classe générale qui permet de connaitre les identificateurs

//d'un ou plusieurs fichiers. Cela est fonction des mots-clés demandés par

//un client ou par défaut, et fonction de l'utilisateur à vouloir connaitre

//les identificateurs présents dans les mots-clés ou qui ne le sont pas.

//

//------------------------------------------------------------------------

**class** References

{

//----------------------------------------------------------------- PUBLIC

**public**:

//----------------------------------------------------- Méthodes publiques

**void** **TraiterFichiers** ( **bool** exclureMotsCles, **char** \* nomFichierMotsCles,

set<string> &nomsFichiers );

// Mode d'emploi : Calcule les identificateurs souhaités. Le calcul dépend

//du paramètre exclureMotsCles qui précise si on veut connaitre

//les idenctificateurs présents dans les mots-clés ou les autres.

//nomFichierMotsCles est l'adresse du nom de fichier où sont présnts les

//mots clés. En l'absence de ce paramètre, le programme prend une liste de

//mots clés par défaut. nomsFichiers est un objet de type Set qui comporte

//les noms de fichiers à analyser.

//

// Contrat : aucun

//

string **AfficherResultat** ( );

// Mode d'emploi : Affiche Les identificateurs avec les fichiers et les

//numéros de lignes qui leurs sont liés

//

// Contrat : aucun

//

//-------------------------------------------- Constructeurs - destructeur

**References** ( **const** References & unReferences );

// Mode d'emploi (constructeur de copie) :

//

// Contrat :

//

**References** ( );

// Mode d'emploi :

//

// Contrat :

//

**virtual** **~References** ( );

// Mode d'emploi :

//

// Contrat :

//

//------------------------------------------------------------------ PRIVE

**protected**:

//----------------------------------------------------- Méthodes protégées

//Lit le fichier nomFichier et retourne les mots-clés présents dans le

//ficher

vector<string> \* **lireFichierMotsCles** ( **char** \* nomFichier );

//Lit un fichier quelconque et en discerne les identificateurs pour pouvoir

//les traiter

**void** **lireFichier** ( **const** **char** \* nomFichier );

//Ajoute le mot en fonction de la variable exclureMotsCles

**void** **traiterMot** ( string &mot, **const** **char** \* nomFichier, **int** numLigne );

//ajoute le mot comme identificateur si il n'a pas déjç était ajouté et

//le nom de fichier et le numéro de ligne.

**void** **ajouterReference** ( string &mot, **const** **char** \* nomFichier,

**int** numLigne );

//----------------------------------------------------- Attributs protégés

//references possède comme clés les identificateurs retenus et pointent vers

//un de la classe AssocRefFichier

map<string, AssocRefFichier> references;

//motsCles est un tableau dynamique de mots-clés

vector<string> \* motsCles;

//exclureMotsCles est un booléen qui précise si on veut connaitre

//les idenctificateurs présents dans les mots-clés ou les autres.

**bool** exclureMotsCles;

};

**class** AssocRefFichier

{

//----------------------------------------------------------------- PUBLIC

**public**:

//----------------------------------------------------- Méthodes publiques

**void** **TraiterFichier** ( string nomFichier, **int** numLigne );

// Mode d'emploi :Lie le nom nomFichier à un tableau d'entier

//en ajoutant l'entier numLigne à cet objet.

//

// Contrat :

//

string **AfficherFichiers** ( );

// Mode d'emploi : Retourne une chaine de caractère représentant

//les fichiers avec les numéros de lignes qui leurs sont liés

//

// Contrat :

//

string **AfficherLignes** (vector<**int**> lignes);

// Mode d'emploi : Retourne une chaine de caractère représentant des

//numéros de lignes

//

// Contrat :

//

//-------------------------------------------- Constructeurs - destructeur

**AssocRefFichier** ( **const** AssocRefFichier & unAssocRefFichier );

// Mode d'emploi (constructeur de copie) :

//

// Contrat :

//

**AssocRefFichier** ( );

// Mode d'emploi :

//

// Contrat :

//

**virtual** **~AssocRefFichier** ( );

// Mode d'emploi :

//

// Contrat :

//

//------------------------------------------------------------------ PRIVE

**protected**:

//----------------------------------------------------- Méthodes protégées

//----------------------------------------------------- Attributs protégés

//fichiers possède comme clés des noms de fichiers et pointent vers un

//un tableau d'entier

map<string, vector<**int**> > fichiers;

};

* Réalisation

1. **Réalisation du plan de tests fonctionnels**

* Fichiers nécessaires pour les besoins des tests avec leur contenu :

**keywords.txt:**

int

world

template

**keywords2.txt** est vide.

**keywords3.txt:**

int main();int entier;

**keywords4.txt :**

//commentaire

int a;

/\*voici

un autre commentaire\*/

char b = 'a';

string c = "Hello";

**fileTest1.txt:**

// affiche le message "Hello world"

int main() {

cout<<"Hello world"<<endl;

cout<<endl;

return 0;

}

**fileTest2.txt:**

int main();

**fileTest3.txt:**

int main();int entier;

**fileTest4.txt:**

//commentaire

int a;

/\*voici

un autre commentaire\*/

char b = 'a';

string c = "Hello";

* Tests réalisés en concordance avec le plan de test dans le document de spécification et conception :

**Test1.1**:TP\_STL.exe -k keywords.txt fileTest1.txt -e

**Test1.2:**TP\_STL.exe -k keywords.txt -e fileTest1.txt

**Test1.3**:TP\_STL.exe -k -e keywords.txt fileTest1.txt

**Test1.4**:TP\_STL.exe -e keywords.txt -k fileTest1.txt

**Test1.5**:TP\_STL.exe

**Test1.6**:TP\_STL.exe -e -k keywords.txt

**Test1.7**:TP\_STL.exe -f -k keywords.txt

**Test2**:TP\_STL.exe fileTest2.txt fileTest2.txt

**Test3**:TP\_STL.exe fileTest3.txt

**Test4**:TP\_STL.exe -k keywords2.txt fileTest1.txt

**Test5**:TP\_STL.exe fileTest4.txt

**Test6**:TP\_STL.exe -e -k keywords3.txt fileTest1.txt

**Test7**:TP\_STL.exe -e -k key.txt fileTest1.txt

**Test8**:TP\_STL.exe -e fileTest1.txt fileTest2.txt fileTest3.txt toto.txt

**Test9**:TP\_STL.exe -e -k keywords.txt fileTest1.txt fileTest2.txt

**Test10**:TP\_STL.exe -e fileTest1.txt fileTest2.txt

**Test11**:TP\_STL.exe -k keywords.txt fileTest1.txt fileTest2.txt

**Test12**:TP\_STL.exe fileTest1.txt fileTest2.txt

* Résultat des Tests

**Test1.1**: Position de l'argument -e invalide

**Test1.2:** Position de l'argument -e invalide

**Test1.3**: FichierMotCles attendu apres l'option -k

**Test1.4**: Position de l'argument -k invalide

**Test1.5**: Nombre d'arguments insuffisant

**Test1.6**: Nombre d'arguments insuffisant

**Test1.7**: Option inconnue

**Test2**: int fileTest2.txt 1

**Test3**: int fileTest3.txt 1 1

**Test4**:

**Test5**: char fileTest4.txt 5

int fileTest4.txt 2

**Test6**: cout fileTest1.txt 3 4

endl fileTest1.txt 3 4

int fileTest1.txt 2

main fileTest1.txt 2

return fileTest1.txt 5

**Test7**: Erreur

**Test8**: Erreur

**Test9**: cout fileTest1.txt 3 4

endl fileTest1.txt 3 4

main fileTest1.txt 2 fileTest2.txt 1

return fileTest1.txt 5

**Test10**: cout fileTest1.txt 3 4

endl fileTest1.txt 3 4

main fileTest1.txt 2 fileTest2.txt 1

**Test11**: int fileTest1.txt 2 fileTest2.txt 1

**Test12**: int fileTest1.txt 2 fileTest2.txt 1

return fileTest1.txt 5

* Validation/commentaires éventuels

Tous les tests ont été validés.