Pour remettre votre travail :

* Créez votre document de remise qui servira à la correction à partir de celui-ci. Changez son nom pour

**<votre nom de famille>\_<votre DA>\_TP1.docx)**

**Exemple :**

**Tremblay\_1324567\_TP1.docx**

* Placez le document Word ET le répertoire du projet Visual Studio dans un répertoire que vous nommerez **TP1** (sans votre nom ni votre DA puisqu’il il sera identifié automatiquement par Léa avec votre nom et votre DA
* À partir de ce répertoire, créez un document archive (.zip) qui se nommera **TP1.zip**
* Envoyez le document ainsi créé à votre prof **par Léa**

# Barème de correction

Ne pas modifier cette section.

Voir les commentaires en marge du texte pour plus d’explications.

## Résultat final : /20

## Structure de données (remise le 2 octobre) /2

## Programmation /5

### Structure du programme, découpage des procédures et fonctions, bonne utilisation des pointeurs, respect des normes de programmation

## Fonctionnement du programme /13

### Création des listes : /5

### Options du menu : /8

Écrasez ceci avec votre code C++

Important : le format de la police doit être : Courier New, Taille : 8 pt et en interligne simple

Dans Visual Studio, sélectionner tout votre code et faites un copier-coller

#include <iostream>

#include <fstream>

#include <locale>

#include <stdio.h>

#include <string.h>

using namespace std;

#include "TP1.h"

void menu();

void fichierActivites(Activite \*&listeActivites);

void insereAct(Activite \*&listeActivites, char titre[LONGUEUR\_LIGNE], char cod[LONGUEUR\_LIGNE], int nb);

void afficherAct(Activite \*listeActivites, Membre \*&listeMembresNom);

void fichierDemande(Membre \*&listeMembresNom, Membre \*&listeMembresNumero, Activite \*&listeActivites);

void insereTabActiv(Membre \*&listeMembresNom, Membre \*&listeMembresNumero, Activite \*&listeActivites, char numero[LONGUEUR\_LIGNE], char prenom[LONGUEUR\_LIGNE], char nom[LONGUEUR\_LIGNE], char code[LONGUEUR\_LIGNE]);

void insereNomNumero(Membre \*&listeMembresNom, Membre \*&listeMembresNumero, Activite \*&listeActivites, char numero[LONGUEUR\_LIGNE], char prenom[LONGUEUR\_LIGNE], char nom[LONGUEUR\_LIGNE], char code[MAX\_ACTIVITES]);

void afficherNom(Membre\* listeMembresNom);

void affichageInscription(Activite \*&listeActivites);

void afficherNumero(Membre\* listeMembresNumero);

void afficherMember(Membre \*&listeMembresNom);

void inscriptionAttent(Activite \*&listeActivites, Membre \*&listeMembresNom, Demande \*&demandesEnAttente);

void afficherAttente(Demande\* demandesEnAttente);

void demandeInscription(Activite \*&listeActivites, Demande \*&demandesEnAttente);

void insereDemend(Demande \*&demandesEnAttente, char numero[LONGUEUR\_LIGNE], char prenom[LONGUEUR\_LIGNE], char nom[LONGUEUR\_LIGNE], char code[MAX\_ACTIVITES]);

void annulerInscription(Demande \*&demandesEnAttente);

void annulerDemend(Demande \*&demandesEnAttente, char code[LONGUEUR\_LIGNE], char numero[LONGUEUR\_LIGNE]);

void main() {

setlocale(LC\_ALL, "");

Activite\* listeActivites = NULL;

Membre\* listeMembresNumero = NULL;

Membre\* listeMembresNom = NULL;

Demande\* demandesEnAttente = NULL;

int valeur;

fichierActivites(listeActivites);

fichierDemande(listeMembresNom, listeMembresNumero, listeActivites);

//inscriptionAttent(listeActivites, listeMembresNom, demandesEnAttente);

menu();

cin >> valeur;

while (valeur != 0) {

switch (valeur - 1) {

case 0:

afficherAct(listeActivites, listeMembresNom);

break;

case 1:

afficherNom(listeMembresNom);

break;

case 2:

afficherNumero(listeMembresNumero);

break;

case 3:

affichageInscription(listeActivites);

break;

case 4:

afficherMember(listeMembresNom);

break;

case 5:

afficherAttente(demandesEnAttente);

break;

case 6:

demandeInscription(listeActivites, demandesEnAttente);

break;

case 7:

annulerInscription(demandesEnAttente);

break;

;

}

menu();//aficher menu

cin >> valeur;//lire le choix sur clavier

}

system("pause");

}

//methode qui afiche la menu

void menu(){

cout << "------------------------" << endl;

cout << " MENU " << endl;

cout << "------------------------" << endl;

cout << "1 - Produire la liste des activités" << endl;

cout << "2 - Produire la liste des membres par ordre de nom" << endl;

cout << "3 - Produire la liste des membres par ordre de numéro" << endl;

cout << "4 - Produire la liste des inscriptions par activité" << endl;

cout << "5 - Produire la liste des activités auxquelles est inscrit un participant" << endl;

cout << "6 - Produire la liste des demandes d'inscription en attente" << endl;

cout << "7 - Ajouter des demandes d'inscription" << endl;

cout << "8 - Annuler des inscriptions" << endl;

cout << endl;

cout << "0 - Quitter l'application" << endl;

cout << endl;

cout << endl;

cout << "Faites votre choix : ";

}

//methode qui lite le fichier Activités.txt

void fichierActivites(Activite \*&listeActivites) {

char titre\_act[LONGUEUR\_LIGNE]; //la titre d'activité

char cod\_act[LONGUEUR\_LIGNE]; //la code d'activité

char nb\_char[LONGUEUR\_LIGNE]; //la nombre d'activité

int nb; //pour int de nombre d'activité

ifstream lectureActiv;

lectureActiv.open(FICHIER\_ACTIVITES, ios::in);

if (lectureActiv.fail()) {

cout << "Erreur d'ouverture du fichier input" << endl;

exit(1);

}

lectureActiv.getline(titre\_act, LONGUEUR\_LIGNE, '/');

while (!lectureActiv.eof()) {

lectureActiv.getline(cod\_act, LONGUEUR\_LIGNE, '/');

lectureActiv.getline(nb\_char, LONGUEUR\_LIGNE);

nb = atol(nb\_char);

insereAct(listeActivites, titre\_act, cod\_act, nb);

lectureActiv.getline(titre\_act, LONGUEUR\_LIGNE, '/');

}

lectureActiv.close();

}

//methode qui insere l'information dans la liste d'activités

void insereAct(Activite \*&listeActivites, char titre[LONGUEUR\_LIGNE], char cod[LONGUEUR\_LIGNE], int nb) {

Activite \*p = new Activite;

if (p == NULL){

exit(EXIT\_FAILURE);

}

else {

Activite \*p\_tmp = NULL;

Activite \*p\_tete = listeActivites;

p->titre = new char[LONGUEUR\_LIGNE];

strcpy\_s(p->titre, LONGUEUR\_LIGNE, titre);

p->code = new char[LONGUEUR\_LIGNE];

strcpy\_s(p->code, LONGUEUR\_LIGNE, cod);

p->nbMaxParticipants = nb;

p->tabInscrits = new Membre\*[nb];

for (int i = 0; i < nb; i++){

p->tabInscrits[i] = NULL;

}

p->actSuivante = NULL;

while (p\_tete != NULL && (int)p\_tete->code[0] <= (int)cod[0])

{

p\_tmp = p\_tete;

p\_tete = p\_tete->actSuivante;

}

p->actSuivante = p\_tete;

if (p\_tmp != NULL)

{

p\_tmp->actSuivante = p;

}

else

{

listeActivites = p;

}

}

}

//afficher la liste d'activités dans l'option 1

void afficherAct(Activite \*listeActivites, Membre \*&listeMembresNom){

int comp = 0;//compteur

int place;//les places disponibles

cout << endl;

cout << endl;

cout << "Liste des activités" << endl;

cout << "--------------------" << endl;

cout << endl;

if (listeActivites == NULL)

{

exit(EXIT\_FAILURE);

}

Activite \*p = listeActivites;

while (p != NULL)

{

cout << p->code << " : " << p->titre << endl;

cout << " Nombre maximale de participants : " << p->nbMaxParticipants << endl;

comp = 0;

Membre \*m = listeMembresNom;

while (m != NULL)

{

for (int i = 0; i < MAX\_ACTIVITES && m->tabActivites[i] != NULL; i++){

if (p->code == m->tabActivites[i]->code){

++comp;

}

}

m = m->nomSuivant;

}

if (p->nbMaxParticipants - comp < 0){

place = 0;

}

else{

place = p->nbMaxParticipants - comp;

}

cout << " Nombre de place(s) disponible(s) : " << place << endl;

cout << endl;

p = p->actSuivante;

}

cout << endl;

}

//lire fichier de Demend.txt

void fichierDemande(Membre \*&listeMembresNom, Membre \*&listeMembresNumero, Activite \*&listeActivites) {

char prenom[LONGUEUR\_LIGNE]; //prenom du membre

char nom[LONGUEUR\_LIGNE]; //nom du membre

char numero[LONGUEUR\_LIGNE]; //numero du membre

char code[LONGUEUR\_LIGNE]; //code du membre

ifstream lectureDemande;

lectureDemande.open(FICHIER\_INSCRIPTIONS, ios::in);

if (lectureDemande.fail()) {

cout << "Erreur d'ouverture du fichier input" << endl;

exit(1);

}

lectureDemande.getline(numero, LONGUEUR\_LIGNE, '/');

while (!lectureDemande.eof()) {

lectureDemande.getline(prenom, LONGUEUR\_LIGNE, '/');

lectureDemande.getline(nom, LONGUEUR\_LIGNE, '/');

lectureDemande.getline(code, LONGUEUR\_LIGNE);

insereTabActiv(listeMembresNom, listeMembresNumero, listeActivites, numero, prenom, nom, code);

lectureDemande.getline(numero, LONGUEUR\_LIGNE, '/');

}

lectureDemande.close();

}

//methode qui insere la liste de membre du nom et la liste de membre de numero

void insereTabActiv(Membre \*&listeMembresNom, Membre \*&listeMembresNumero, Activite \*&listeActivites, char numero[LONGUEUR\_LIGNE], char prenom[LONGUEUR\_LIGNE],

char nom[LONGUEUR\_LIGNE], char code[MAX\_ACTIVITES]) {

bool areter = false;//pour areter le boucle qui ramplire tabActiviter

bool ver\_nom = false;//pour savoir si le meembre existe dans la liste

Membre \*p\_mem = NULL;

Membre\*p\_tete = listeMembresNom;

if (p\_tete != NULL){

while (p\_tete != NULL) {

if (strcmp(p\_tete->noMembre, numero) == 0 && !ver\_nom){

ver\_nom = true;

p\_mem = p\_tete;

}

p\_tete = p\_tete->nomSuivant;

}

if (ver\_nom == true){

Activite\* a\_tete = listeActivites;

while (a\_tete != NULL) {

if (strcmp(a\_tete->code, code) == 0){

for (int i = 0; i < MAX\_ACTIVITES && !areter; i++){

if (p\_mem->tabActivites[i] == NULL){

p\_mem->tabActivites[i] = new Activite

p\_mem->tabActivites[i] = a\_tete;

areter = true;

}

}

}

a\_tete = a\_tete->actSuivante;

}

}

else {

insereNomNumero(listeMembresNom, listeMembresNumero, listeActivites, numero, prenom, nom, code);

}

}

else {

insereNomNumero(listeMembresNom, listeMembresNumero, listeActivites, numero, prenom, nom, code);

}

}

//method qui faire 2-em partie de insertion de membre

void insereNomNumero(Membre \*&listeMembresNom, Membre \*&listeMembresNumero, Activite \*&listeActivites, char numero[LONGUEUR\_LIGNE], char prenom[LONGUEUR\_LIGNE],

char nom[LONGUEUR\_LIGNE], char code[MAX\_ACTIVITES]){

Membre \*p\_tmp = NULL;

Membre\*p\_tete = listeMembresNom;

Membre\*num\_tete = listeMembresNumero;

Membre\* p = new Membre;

for (int i = 0; i < MAX\_ACTIVITES; i++){

p->tabActivites[i] = NULL;

}

p->noMembre = new char[LONGUEUR\_LIGNE];

strcpy\_s(p->noMembre, LONGUEUR\_LIGNE, numero);

p->prenom = new char[LONGUEUR\_LIGNE];

strcpy\_s(p->prenom, LONGUEUR\_LIGNE, prenom);

p->nomFamille = new char[LONGUEUR\_LIGNE];

strcpy\_s(p->nomFamille, LONGUEUR\_LIGNE, nom);

Activite\* a\_tete = listeActivites;

while (a\_tete != NULL){

if (strcmp(a\_tete->code, code) == 0){

p->tabActivites[0] = a\_tete;

}

a\_tete = a\_tete->actSuivante;

}

////////////insere la liste de Nom////////////////////

p->nomSuivant = NULL;

while (p\_tete != NULL && (int)p\_tete->nomFamille[0] <= (int)nom[0])

//while (p\_tete != NULL && strcmp(p\_tete->nomFamille, nom) < 0)

{

p\_tmp = p\_tete;

p\_tete = p\_tete->nomSuivant;

}

p->nomSuivant = p\_tete;

if (p\_tmp != NULL)

{

p\_tmp->nomSuivant = p;

}

else

{

listeMembresNom = p;

}

///////////insere la liste de Numero////////////////

while (num\_tete != NULL && (int)num\_tete->noMembre[2] <= (int)numero[2])

//while (num\_tete != NULL && strcmp(num\_tete->noMembre, numero) < 0)///

{

p\_tmp = num\_tete;

num\_tete = num\_tete->numeroSuivant;

}

p->numeroSuivant = num\_tete;

if (p\_tmp != NULL)

{

p\_tmp->numeroSuivant = p;

}

else

{

listeMembresNumero = p;

}

}

// methode qui affiche la liste de membres par nom, option 2

void afficherNom(Membre\* listeMembresNom){

int comp = 0; //compteur

cout << endl;

cout << endl;

cout << "Liste des membres par nom" << endl;

cout << "------------------------------" << endl;

cout << endl;

if (listeMembresNom == NULL)

{

exit(EXIT\_FAILURE);

}

Membre \*p = listeMembresNom;

while (p != NULL)

{

cout << p->nomFamille << " " << p->prenom << " - " << p->noMembre << endl;

p = p->nomSuivant;

++comp;

}

cout << endl;

cout << "Nombre de memberes = " << comp << endl;

cout << endl;

cout << "--------------------------" << endl;

cout << endl;

}

// methode qui affiche la liste des membres par numero, option 3

void afficherNumero(Membre\* listeMembresNumero){

int comp = 0; //compteur

cout << endl;

cout << endl;

cout << "Liste des membres par numero" << endl;

cout << "------------------------------" << endl;

cout << endl;

if (listeMembresNumero == NULL)

{

exit(EXIT\_FAILURE);

}

Membre \*p = listeMembresNumero;

while (p != NULL)

{

cout << p->noMembre << " - " << p->prenom << " " << p->nomFamille << endl;

p = p->numeroSuivant;

++comp;

}

cout << endl;

cout << "Nombre de memberes = " << comp << endl;

cout << endl;

cout << "--------------------------" << endl;

cout << endl;

}

//methode qui affiche la liste des inscription à toutes les activités, option 4

void affichageInscription(Activite \*&listeActivites){

int comp = 0; //compteur

int place = 0; //nombre place encore disponible

cout << endl;

cout << "Liste des inscriptions par activités" << endl;

cout << "------------------------------------" << endl;

cout << endl;

if (listeActivites == NULL)

{

exit(EXIT\_FAILURE);

}

Activite \*p = listeActivites;

while (p != NULL)

{

comp = 0;

cout << p->code << " : " << p->titre << endl;

for (int i = 0; i < p->nbMaxParticipants && p->tabInscrits[i] != NULL; ++i){

cout << " " << p->tabInscrits[i]->prenom << " " << p->tabInscrits[i]->nomFamille << " - "<< p->tabInscrits[i]->noMembre<< endl;

++comp;

}

place = p->nbMaxParticipants - comp;

if (comp > p->nbMaxParticipants){

place = 0;

}

cout << endl;

cout << "Nombre d'inscrits à l'activité = " << comp << endl;

cout << "Nombre de place(s) encore disponible(s) = " << place << endl;

cout << endl;

p = p->actSuivante;

}

cout << endl;

}

//methode qui affiche les activites du membre, option 5

void afficherMember(Membre \*&listeMembresNom){

int len; //pour le longeur de chaine numero

bool aret = false; //pour areter la boucle

char numero[LONGUEUR\_LIGNE]; //le numero du membre

int comp = 0; //compteur

cout << "Entrez le numero du membre ";

cin.clear(); // Nettoyer le flux

cin.ignore(cin.rdbuf()->in\_avail()); // Vider le buffer

cin.getline(numero, LONGUEUR\_LIGNE);

len = strlen(numero);

if (len == 0){

cout << endl;

cout << "Liste des inscription de tous les membres" << endl;

cout << "-----------------------------------------" << endl;

cout << endl;

}

Membre \*p = listeMembresNom;

while (p != NULL && !aret)

{

if (len == 0){

cout << endl;

cout << "Membre " << p->noMembre << " " << p->prenom << " " << p->nomFamille << endl;

cout << endl;

comp = 0;

for (int i = 0; i < MAX\_ACTIVITES && p->tabActivites[i] != NULL; i++){

cout << p->tabActivites[i]->code << " " << p->tabActivites[i]->titre << endl;

++comp;

}

cout << endl;

cout << "Nombre d'activite auxquelles est inscrit le Membre = " << comp << endl;

cout << endl;

}

else{

if (strcmp(p->noMembre, numero) == 0){

cout << endl;

cout << "Liste des activites" << endl;

cout << "Membre " << p->noMembre << " " << p->prenom << " " << p->nomFamille << endl;

cout << endl;

comp = 0;

for (int i = 0; i < MAX\_ACTIVITES && p->tabActivites[i] != NULL; i++){

cout << p->tabActivites[i]->code << " " << p->tabActivites[i]->titre << endl;

++comp;

}

cout << endl;

cout << "Nombre d'activite auxquelles est inscrit le Membre = " << comp << endl;

cout << endl;

aret = true;

}

}

p = p->nomSuivant;

}

if (aret == false && len != 0){

cout << endl;

cout << "Numero de membre inexistant" << endl;

cout << endl;

cout << endl;

}

}

// methode qui insere la liste de demende et les members inscrite pour chaque activites

void inscriptionAttent(Activite \*&listeActivites, Membre \*&listeMembresNom, Demande \*&demandesEnAttente){

Activite \*p\_tete = listeActivites;

Activite \*p = new Activite;

while (p\_tete != NULL)

{

int comp = 0;

Membre \*m = listeMembresNom;

while (m != NULL)

{

for (int i = 0; i < MAX\_ACTIVITES && m->tabActivites[i]!= NULL; i++){

if (p\_tete->code == m->tabActivites[i]->code){

++comp;

bool areter = false;

for (int i = 0; i < p\_tete->nbMaxParticipants && areter == false; i++){

if (p\_tete->tabInscrits[i] == NULL){

p\_tete->tabInscrits[i] = m;

areter = true;

}

}

if (comp > p\_tete->nbMaxParticipants){

Demande \*p = new Demande;

if (p == NULL){

exit(EXIT\_FAILURE);

}

else {

Demande \*p\_tmp = NULL;

Demande \*p\_tete = demandesEnAttente;

p->demandeSuivante = NULL;

p->noMembre = m->noMembre;

p->nomFamille = m->nomFamille;

p->prenom = m->prenom;

p->code = m->tabActivites[i]->code;

p->demandeSuivante = NULL;

if (demandesEnAttente == NULL)

{

demandesEnAttente = p;

}

else

{

Demande \*temp = demandesEnAttente;

while (temp->demandeSuivante != NULL)

{

temp = temp->demandeSuivante;

}

temp->demandeSuivante = p;

}

}

}

}

}

m = m->nomSuivant;

}

p\_tete = p\_tete->actSuivante;

}

}

// methode qui affiche les membres en attente

void afficherAttente(Demande\* demandesEnAttente){

int comp = 0;//compteur

Demande \*p = demandesEnAttente;

cout << endl;

cout << endl;

cout << "Liste des demandes en attente" << endl;

cout << "------------------------------" << endl;

cout << endl;

if (demandesEnAttente == NULL)

{

cout << "Liste des demandes en attente est vide" << endl;

cout << endl;

}

else{

while (p != NULL)

{

cout << p->prenom << " " << p->nomFamille << " - " << p->noMembre << endl;

p = p->demandeSuivante;

++comp;

}

cout << endl;

cout << "Nombre de demandes = " << comp << endl;

cout << endl;

cout << "--------------------------" << endl;

cout << endl;

}

}

// methode qui faire inscription un person pour l'activite

void demandeInscription(Activite \*&listeActivites, Demande \*&demandesEnAttente){

char prenom[LONGUEUR\_LIGNE]; //prenom de la personne

char nom[LONGUEUR\_LIGNE]; //nom de la personne

char numero[LONGUEUR\_LIGNE]; //numero de la personne

char code[LONGUEUR\_LIGNE]; //code de la personne

bool activite = false; // pour arter la boucle de l'activite

bool exist = false; // pour areter la boucle de membre

cout << endl;

cout << "--------------"<<endl;

cout << "Entrez votre code de l'activite ";

cin >> code;

cout << endl;

cout << "Entrez votre numero de membre ";

cin >> numero;

cout << endl;

cout << "Entrez votre prenom ";

cin >> prenom;

cout << endl;

cout << "Entrez votre nom ";

cin >> nom;

cout << endl;

Activite \*p = listeActivites;

while (p != NULL)

{

if (strcmp(p->code, code) == 0){

activite = true;

}

p = p->actSuivante;

}

if (activite == false){

cout << "Activite que vous voulez etre inscrit n'existe pas" << endl;

}

else{

Demande \*p\_tete = demandesEnAttente;

while (p\_tete != NULL)

{

if (strcmp(p\_tete->code, code) == 0 && strcmp(p\_tete->noMembre, numero) == 0){

exist = true;

}

p\_tete = p\_tete->demandeSuivante;

}

if (exist == true){

cout << "Vous ne'pouvez pas demander l'inscription parce que " << prenom << " " << nom << " deja demander inscription pour cette activité" << endl;

}

else{

insereDemend(demandesEnAttente, numero, prenom, nom, code);

cout << endl;

cout << prenom << " " << nom << " - Vous etes dans la liste attende!" << endl;

}

}

}

// methode qui ajoute un memebre dans la liste attente

void insereDemend(Demande \*&demandesEnAttente, char numero[LONGUEUR\_LIGNE], char prenom[LONGUEUR\_LIGNE], char nom[LONGUEUR\_LIGNE], char code[MAX\_ACTIVITES]){

Demande \*p = new Demande;

if (p == NULL){

exit(EXIT\_FAILURE);

}

else {

Demande \*p\_tmp = NULL;

Demande \*p\_tete = demandesEnAttente;

p->noMembre = new char[LONGUEUR\_LIGNE];

strcpy\_s(p->noMembre, LONGUEUR\_LIGNE, numero);

p->code = new char[LONGUEUR\_LIGNE];

strcpy\_s(p->code, LONGUEUR\_LIGNE, code);

p->prenom = new char[LONGUEUR\_LIGNE];

strcpy\_s(p->prenom, LONGUEUR\_LIGNE, prenom);

p->nomFamille = new char[LONGUEUR\_LIGNE];

strcpy\_s(p->nomFamille, LONGUEUR\_LIGNE, nom);

p->demandeSuivante = NULL;

if (demandesEnAttente == NULL)

{

demandesEnAttente = p;

}

else

{

Demande \*temp = demandesEnAttente;

while (temp->demandeSuivante != NULL)

{

temp = temp->demandeSuivante;

}

temp->demandeSuivante = p;

}

}

}

// methode qui demende les donnes de la membre qui veut annuler inscription

void annulerInscription(Demande \*&demandesEnAttente){

char numero[LONGUEUR\_LIGNE]; //numero de la membre

char code[LONGUEUR\_LIGNE]; //code de la membre

bool activite = false; //pour areter la boucle de l'activite

bool annuler = false; //pour verifier annulation

cout << endl;

cout << "Annulation d'inscription" << endl;

cout << "------------------------" << endl;

cout << endl;

cout << "Entrez votre code de l'activite ";

cin >> code;

cout << endl;

cout << "Entrez votre numero de membre ";

cin >> numero;

cout << endl;

Demande \*p = demandesEnAttente;

while (p != NULL)

{

if (strcmp(p->code, code) == 0 && strcmp(p->noMembre, numero) == 0){

activite = true;

}

p = p->demandeSuivante;

}

if (activite == false){

cout << "Inscription que vous voulez annuler n'existe pas" << endl;

}

else{

annulerDemend(demandesEnAttente, code, numero);

while (p != NULL)

{

if (strcmp(p->code, code) == 0 && strcmp(p->noMembre, numero) == 0){

annuler = true;

}

p = p->demandeSuivante;

}

if (annuler == true){

cout << "Inscription que vous voulez annuler échoué !" << endl;

}

else{

cout << endl;

cout << " L'inscription pour le membre -" << numero << "ets annuler !" << endl;

}

}

}

// methode qui enleve le membre dans la liste de demande l'inscription

void annulerDemend(Demande \*&demandesEnAttente, char code[LONGUEUR\_LIGNE], char numero[LONGUEUR\_LIGNE]){

Demande \*p\_tmp = NULL;

Demande \*p\_tete = demandesEnAttente;

//p\_tete->demandeSuivante = NULL;

if (demandesEnAttente == NULL)

{

cout << "Le fichier d'inscription est vide" << endl;

}

else

{

if (strcmp(p\_tete->code, code) == 0 && strcmp(p\_tete->noMembre, numero) == 0){

p\_tete = p\_tete->demandeSuivante;

free(p\_tete);

demandesEnAttente = p\_tete;

}

while (p\_tete != NULL)

{

if (strcmp(p\_tete->demandeSuivante->code, code) == 0 && strcmp(p\_tete->demandeSuivante->noMembre, numero) == 0){

p\_tmp = p\_tete->demandeSuivante;

p\_tete->demandeSuivante = p\_tete->demandeSuivante->demandeSuivante;

free(p\_tmp);

//break;

}

p\_tete = p\_tete->demandeSuivante;

}

}

}