UNIVERSITÉ PARIS DESCARTES

Listing du plus haut niveau atteint

Sprint 5

Rams Léo, Lenouvel Louis, Esteves Gabriel

Groupe : 104

UNIVERSITÉ PARIS DESCARTES

```c
/*
* Projet IAP : Les fondamentaux de la programmation impérative
*
* Sprint 5
*
* Auteurs : Rams Léo, Esteves Gabriel, Lenouvel Louis
*
* Groupe : 104
*
*
* Dernière date de modification : 13/11/2021
*
* Syntaxe :
* [in] : Entre dans la fonction
* [out] : Sort de la fonction
*/

#include <stdio.h>
#include <stdlib.h>
#include<string.h>
#include<limits.h>
#pragma warning(disable:4996)
#pragma warning(disable:6031)
#pragma warning(disable:6054)
#pragma warning(disable:6262)

 enum { Joueuses = 128 }; //Nombre de joueuses//
 enum { JoueusesTournoi = 64 };  //Nombre de joueuses dans un tournoi//
 enum { lgMot = 30 }; //Longueur d'un texte//
 enum { Matchs = 127 }; //Nombre de matchs//
 enum { Tournois = 10 }; //Nombre max de Tournois//
 enum { Joueurs = 2 }; //Nombre de joueurs par match//
 enum { Carac = 2 }; //Caractéristiques d'un tournoi//
 enum { TotalJoueuses = Joueuses * Tournois }; //Nombre max de joueuses//

 //Caractéristiques d'une joueuse : Son nom et ses points//
typedef struct {
    char nom[lgMot + 1];
    unsigned int points;
} Joueuse;

 //Caractéristiques d'un match : L'index de la gagnante et l'index de la perdante//
typedef struct {
    unsigned int idxGagnante;
    unsigned int idxPerdante;
} Match;

 //Caractéristiques d'un Tournoi : Son nom, la date à laquelle il s'est déroulé et la liste de ses matchs//
typedef struct {
    char NomTournoi[lgMot];
    char Date[lgMot];
    char ListeMatchs[Matchs][Joueurs][lgMot];
}Tournoi;

 //Stockage des Tournois : Liste des tournois, avec toutes les caractéristiques qui lui ont été associées//
typedef struct {
    Tournoi ListeTournois[Tournois];
    unsigned int EnrTournois;
    unsigned int AffTournois;
}TournoisWTA;

/*Fonction qui définit le nombre de tournois, celui - ci étant indiqué sur le fichier texte
* [out] nombre
*/
int definir_nombre_tournois(TournoisWTA* w) {
    int nombre;
    scanf("%d", &nombre);
    w->EnrTournois = 0;
    w->AffTournois = 0;
    return nombre;
}

/*Enregistrement d'un Tournoi avec ses caractéristiques
* [in] Match* m qui permet de séparer dans l'enregistrement les gagnantes et les perdantes
* [in] TournoisWTA* w qui permet d'enregistrer les caractéristiques d'un tournoi
*/
void enregistrement_tournoi(Match* m, TournoisWTA* w) {
    scanf("%s", w->ListeTournois[w->EnrTournois].NomTournoi);//Stockage du nom du Tournoi//
    scanf("%s", w->ListeTournois[w->EnrTournois].Date);//Stockage de la date à laquelle se déroule le Tournoi//
    m->idxGagnante = 0;//Définition de l'index de la gagnante//
    m->idxPerdante = 1;//Définition de l'index de la perdante//
    int i;
    for (i = 0; i < Matchs; ++i) {
        //Stockage des noms des gagnantes et perdantes du Tournoi dans ListeMatchs//
        scanf("%s %s", w->ListeTournois[w->EnrTournois].ListeMatchs[i][m->idxGagnante], w->ListeTournois[w->EnrTournois].ListeMatchs[i][m->idxPerdante]);
    }
    ++w->EnrTournois;//Ajout de 1 au nombre de tournois enregistrés//
}

```

```c
/*Affichage du classement des joueuses
 * [in] TournoisWTA* w qui permet d'utiliser les données des tournois enregistrés dans la fonction précédente
 * [in] Joueuse* j qui permet de définir une joueuse du tournoi en fonction de son nom et de ses points
 * [in] Match* m qui comporte l'index de la gagnante et l'index de la perdante
 */
void afficher_classement(TournoisWTA* w, Joueuse* j, Match* m) {
    if (w->EnrTournois == 0) { //On vérifie si des tournois ont bien été enregistrés//
        printf("Pas de classement\n");
    }
    else {
        int EnrJoueuses = 0;
        int TotalTournois = w->EnrTournois;//Nombre total de tournois enregistrés//
        w->AffTournois = TotalTournois - 4; //On affiche uniquement les 4 derniers tournois//
        char apparues[TotalJoueuses][lgMot];//liste des joueuses déjà traitées//
        memset(apparues, 0, TotalJoueuses * lgMot); //La fonction memset permet de vider la liste donnée//
        int points[TotalJoueuses];//liste des points associés au nom des joueuses dans apparues//
        memset(points, 0, TotalJoueuses * sizeof(int));
        for (int y = w->AffTournois; y < TotalTournois; ++y) {//On répète l'opération en fonction du nombre de tournois//
            unsigned int compteur = 0;
            char apparuesTournoi[Joueuses][lgMot];
            memset(apparuesTournoi, 0, Joueuses * lgMot);
            for (int i = 0; i < Joueuses - 1; ++i) {// index du match//
                for (int n = 0; n < 2; ++n) {// index gagnant ou index perdant//
                    int test1 = 0, test2 = 0;
                    strcpy(j->nom, w->ListeTournois[y].ListeMatchs[i][n]);
                    for (int o = 0; o < Joueuses; ++o) {// on parcourt la liste apparuesTournoi//
                        if (strcmp(apparuesTournoi[o], j->nom) == 0) {
                            test1 = 1;//si la joueuse est apparue dans le tournoi alors le test 1 est négatif//
                        }
                    }
                    for (int l = 0; l < TotalJoueuses; ++l) {
                        if (strcmp(apparues[l], j->nom) == 0) {//On parcourt la liste apparues//
                            test2 = 1;//Si la joueuse est apparue tous tournois confondus le test 2 est négatif//
                        }
                    }
                    if (test2 == 0) {//On ajoute la nouvelle joueuse dans la liste apparues avec ses points et on ajoute 1 aux joueuses enregistrées//
                        strcpy(apparues[EnrJoueuses], j->nom);
                        points[EnrJoueuses] = 0;
                        ++EnrJoueuses;
                    }
                    if (test1 == 0) {
                        unsigned int indice;
                        for (int p = 0; p < TotalJoueuses; ++p) {
                            if (strcmp(j->nom, apparues[p]) == 0) {//On trouve l'indice de la joueuse dans la liste apparues pour ajouter les points au bon emplacement//
                                indice = p;
                            }
                        }
                        for (int k = 0; k < Matchs; ++k) { //on parcourt la liste des matchs//
                            unsigned int defaite = 128;
                            if (strcmp(j->nom, w->ListeTournois[y].ListeMatchs[k][m->idxPerdante]) == 0) {
                                defaite = k;
                            }
                            if (defaite < 64) { //A perdu en 64emes de finale//
                                points[indice] += 10;
                            }
                            else if (64 <= defaite && defaite < 96) { //A perdu en 32emes de finale//
                                points[indice] += 45;
                            }
                            else if (96 <= defaite && defaite < 112) { //A perdu en 16emes de finale//
                                points[indice] += 90;
                            }
                            else if (112 <= defaite && defaite < 120) { //A perdu en 8emes de finale//
                                points[indice] += 180;
                            }
                            else if (120 <= defaite && defaite < 124) { //A perdu en quarts de finale//
                                points[indice] += 360;
```

```
                        }
                        else if (124 <= defaite && defaite < 126) { //A perdu en demi-finale//
                            points[indice] += 720;
                        }
                        else if (defaite == 126) { //A perdu en finale//
                            points[indice] += 1200;
                        }
                    }
                    if (strcmp(j->nom, w->ListeTournois[y].ListeMatchs[126][m->idxGagnante]) == 0) { //A gagné le tournoi//
                        points[indice] += 2000;
                    }
                    strcpy(apparuesTournoi[compteur], j->nom);
                    ++compteur; //On ajoute la joueuse à celles apparues dans le tournoi et on ajoute 1 au nombre de joueuses apparues dedans//
                }
            }
        }
    }
    char Nranges[TotalJoueuses][lgMot]; //Liste des noms une fois rangés en fonction des points//
    memset(Nranges, 0, TotalJoueuses * lgMot);
    unsigned int Pranges[TotalJoueuses], position; //Liste des points une fois rangés par ordre décroissant//
    memset(Pranges, 0, TotalJoueuses * sizeof(unsigned int));
    int PlusGrand;
    for (int i = 0; i < EnrJoueuses; ++i) {
        PlusGrand = 0;
        for (int l = 0; l < EnrJoueuses; ++l) {//On trouve le score le plus haut//
            if (points[l] > PlusGrand) {
                PlusGrand = points[l];
                position = l;
            }
        }
        strcpy(Nranges[i], apparues[position]);
        Pranges[i] = PlusGrand; //On ajoute à la nouvelle liste le plus grand trouvé//
        points[position] = -1; //On retire de la liste le plus grand trouvé//
    }
    char Nfinal[TotalJoueuses][lgMot], PlusPetit[lgMot]; //Listes Nfinal et Pfinal : Listes finales qui seront affichées//
    memset(Nfinal, 0, TotalJoueuses * lgMot);
    unsigned int Pfinal[TotalJoueuses], index, compte = 0, k = 0;
    memset(Pfinal, 0, TotalJoueuses * sizeof(int));
    while (k < EnrJoueuses) {
        int nombre = 1;
        if (Pranges[k] != Pranges[k + 1]) {//Si deux joueuses ont des points différents alors on les ajoute directement à la liste//
            strcpy(Nfinal[compte], Nranges[k]);
            Pfinal[compte] = Pranges[k];
            ++compte;
            ++k;
        }
        else { //Si deux joueuses ont le même nombre de points alors on les classe par ordre alphabétique//
            int d = k;
            char lien[TotalJoueuses][lgMot]; //Liste qui permet de classer par ordre alphabétique sans modifier la liste Nranges//
            for (int u = 0; u < EnrJoueuses; ++u) {
                strcpy(lien[u], Nranges[u]);//On met le contenu de Nranges dans lien//
            }
            while (Pranges[d] == Pranges[d + 1]) {//On compte le nombre de joueuses ayant le même nombre de points//
                ++nombre;
                ++d;
            }
            int h = k;
            for (int i = 0; i < nombre; ++i) {
                strcpy(PlusPetit, lien[h]);
                int executions = k + nombre - 1;
                for (int l = k; l < executions; ++l) {//On les classe par ordre alphabétique//
                    int y = 0;
                    while (lien[l + 1][y] == PlusPetit[y]) {
                        ++y;
                    }
                    if (lien[l + 1][y] < PlusPetit[y]) {
                        strcpy(PlusPetit, lien[l + 1]);
                    }
                }
                for (int n = 0; n < EnrJoueuses; ++n) {
                    if (strcmp(PlusPetit, Nranges[n]) == 0) {
                        index = n;
                    }
                }
                strcpy(Nfinal[compte], PlusPetit);
                strcpy(lien[index], "{"); // "{" est toujours supérieur à une lettre//
                Pfinal[compte] = Pranges[index];
                ++compte;// On a ajouté aux listes finales les noms et points correspondant et on ajoute 1 au nombre de joueuses enregistrées//
            }
            k += nombre;//On ajoute à k le nombre de joueuses qu'on a classé par ordre alphabétique//
        }
    }
    for (int i = 0; i < EnrJoueuses; ++i) {//On imprime la liste finale//
        printf("%s %u\n", Nfinal[i], Pfinal[i]);
    }
}
}
```

```
                    strcpy(PlusPetit, lien[l + 1]);
                }
            }
            for (int n = 0; n < EnrJoueuses; ++n) {
                if (strcmp(PlusPetit, Nranges[n]) == 0) {
                    index = n;
                }
            }
            strcpy(Nfinal[compte], PlusPetit);
            strcpy(lien[index], "{"); // "{" est toujours supérieur à une lettre//
            Pfinal[compte] = Pranges[index];
            ++compte;// On a ajouté aux listes finales les noms et points correspondant et on ajoute 1 au nombre de joueuses enregistrées//
        }
        k += nombre;//On ajoute à k le nombre de joueuses qu'on a classé par ordre alphabétique//
    }
}
for (int i = 0; i < EnrJoueuses; ++i) {//On imprime la liste finale//
    printf("%s %u\n", Nfinal[i], Pfinal[i]);
}
}
}

int main() {
    char mot[lgMot];
    Tournoi t;
    Match m;
    Joueuse j;
    TournoisWTA w;
    int nombre_tournois;
    while (1) {//S'exécute jusqu'à ce qu'une fonction l'arrête//
        scanf("%s", &mot);//Lit tout le texte et s'arrête lorsqu'il rencontre une fonction//
        //Exécute la fonction definir_nombre_tournois s'il la rencontre dans le texte//
        if (strcmp(mot, "definir_nombre_tournois") == 0) {
            nombre_tournois = definir_nombre_tournois(&w);
        }
        //Exécute la fonction enregistrement_tournoi s'il la rencontre dans le texte//
        if (strcmp(mot, "enregistrement_tournoi") == 0) {
            enregistrement_tournoi(&m, &w);
        }
        //Exécute la fonction affichage_joueuses_tournoi s'il la rencontre dans le texte//
        if (strcmp(mot, "afficher_classement") == 0) {
            afficher_classement(&w, &j, &m);
        }
        //Arrête le programme s'il rencontre "exit" dans le texte//
        if (strcmp(mot, "exit") == 0) {
            exit(0);//Le programme s'arrête//
        }
    }
    system("pause");
    return 0;
    //Fin du programme//
}
```