IUT de Paris - Rives de Seine
Université de Paris

Rapport IAP: Classement WTA des joueuses de tennis

IUT de Paris - Département Informatique

Allan Herillus, Elias Ait-Khelifa - Groupe 110

IUT de Paris - Rives de Seine
Université de Paris

Tables des matières

Allan Herillus, Elias Ait-Khelifa

IUT de Paris - Rives de Seine
Université de Paris

Introduction

La création d'un programme est un processus compliqué qui nécessite souvent une réflexion au préalable sur toute sa structure. Grace au TD/TP et au cours magistraux, nous avons pu réaliser ce programme.
Le langage que nous avons utilisé pour créer ce programme, est le C.

Problématique

Le projet consiste en la création d'un Interpréteur de commandes permettant de définir le nombre de tournois, d'enregistrer pour chaque tournoi les résultats de chacun des 127 matchs, d'extraire des résultats les 128 joueuses qualifiées du tournoi, de calculer le nombre de points gagnés par les joueuses dans un tournoi donné, de donner le résultat des matchs d'un tournoi donné pour une joueuse donnée, de tenir à jour une table des joueuses classées avec le cumul des points gagnés dans les 4 derniers tournois au plus et enfin de donner le classement.

Développement

Conception

Dans le cadre de ce projet, nous devions mettre en exécution nos connaissances acquises au cours des TD/TP et CM.
Pour ce faire nous avons utilisé quelques bibliothèques comme par exemple : "stdio.h", sdlib.h, "pragma warning(disable :4996)" etc...

Programmation

Pour ce projet, l'utilisation du langage C a été choisie : ainsi nous devions nous familiariser avec ce dernier. Nous devions ainsi comprendre son fonctionnement général :

- Les déclarations de variables
- Les boucles for, while
- Les différents types
- Les fonctions

Ce que le programme permet

Ce programme permet d'enregistrer différentes joueuses, en prenant en option un nombre de tournois définissable, d'enregistrer des tournois avec leur nom et leur date en année pour les différencier ainsi que les matchs avec leur gagnante et perdantes, de faire un classement par tournoi ou un total de plusieurs tournois et de pouvoir les afficher.

Allan Herillus, Elias Ait-Khelifa

IUT de Paris - Rives de Seine
Université de Paris

Bilan du projet

Difficultés rencontrées

Nous avons rencontré plusieurs difficultés dans la réalisation de ce projet. Parmi toutes ces difficultés, il y a notamment le développement à terme de celui-ci tout en ayant des projets en parallèle (e-portfolio, Saé, Semaine de DST).

Les difficultés liées au projet lui-même sont multiples, nous avons rencontré un très grand problème dès le début. Nous devions réfléchir à la façon d'enregistrer les joueuses pour les stocker ensuite.

La plus grande difficulté que nous avons rencontrée est l'enregistrement des joueuses ou nous avons passé au moins 2 semaines à réfléchir sur comment faire cela. Car au début, nous n'avions pas encore réalisé qu'il fallait utiliser des fonctions annexe pour grandement se faciliter la tâche ("idxfind/index_tournoi").

Apport du projet

Tout au long de ce projet, nous avons réalisé l'importance de l'organisation dans le travail de groupe.
Nous avons par exemple utilisé Discord pour planifier des réunions/sessions de travail (à l'IUT ou chez nous).
Les appels à distance se faisaient aussi par Discord bien sûr et chaque lundi après les cours magistraux, nous établissons les objectifs de la semaine et les sessions de travail qui auront lieu selon notre emploi du temps.

De même au niveau des fichiers sources, mais surtout au niveau des commentaires : il fallait tout structurer correctement pour pouvoir comprendre et surtout ne pas oublier le pourquoi du comment nous avions fait ça. Bien entendu, nous avons grandement consolidé nos connaissances dans la structuration du code et acquis des connaissances en C.

Pistes d'améliorations

En termes d'améliorations, nous aurions pu faire une meilleure mise en place des structures, améliorer les fonctions nécessaires au bon fonctionnement du programme (surtout celles qui n'étaient pas demandées).
Nous avons aussi pensé que nos commentaires n'étaient pas assez détaillé en voulant les faire trop simple à comprendre, leur placement n'est pas aussi optimal.

Sprint le plus haut

Notre sprint le plus haut est le 4.

Allan Herillus, Elias Ait-Khelifa

```c
#include "stdio.h"

#include "stdlib.h"

#include "string.h"

#pragma warning(disable: 4996)

// Définitions des constantes necessaires

/*

Position des joueuses dans l'index

*/

#define POS_64  64

#define POS_32  96

#define POS_16 112

#define POS_8  120

#define POS_4  124

#define POS_2  126



#define MAXTOURNOI 10           // nombre maximum de tournois

#define NBMATCH 127   // nombre de matchs par tournoi

#define NBJOUEUSES 128 // nombre de joueuses par tournoi

#define TAILLECHAR 30                   // taille maximale d'une chaine de caractères
saisie
```

```
#define MAXTOURNOI_CLASSEMENT 4      // Nombre de tournois nécessaire pour créer un classement WTA



/*

Nombre de points

*/

#define NBPOINTS_64EME 10

#define NBPOINTS_32EME 45

#define NBPOINTS_16EME 90

#define NBPOINTS_8EME 180

#define NBPOINTS_QUARTS 360

#define NBPOINTS_DEMI 720

#define NBPOINTS_FINALE_PERDANT 1200

#define NBPOINTS_FINALE_GAGNANT 2000



// Définition des structures nécessaires

typedef struct {                // structure Joueuse

    char nom[TAILLECHAR + 1];

    unsigned int points;

} Joueuse;
```

```
typedef struct {                    // structure Match

    unsigned int idxGagnante;    // indexe de la joueuse  gagnante d'un match de type entier non signé

    unsigned int idxPerdante;    // indexe de la joueuse perdante d'un match de type entier non signé

} Match;

typedef struct {                 // structure Tournoi

    char nom[TAILLECHAR + 1];

    unsigned int date;

    Match dataMatch[NBMATCH];   // les matchs d'un tournoi de type match

} Tournoi;

typedef struct {              // structure TournoiWTA

    unsigned int nbTournois,

          idxT,                      // le nombre de tournois, l'indice d'un tournoi, l'indice d'une joueuse de type entier non signé

        idxJ;

    Tournoi dataTournois[MAXTOURNOI];   // tableau de tournois de type tournoi

        Joueuse dataJoueuses[NBJOUEUSES * MAXTOURNOI];    // tableau des joueuses participante à un tournoi

} TournoiWTA;
```

```
typedef struct {          // structure ClassementWTA

    char orderBy[TAILLECHAR + 5 + 1];   // chaine de caractère qui identifiera le nom d'une joueuse

    Joueuse joueuse;            // joueuse participante à un tournoi de type joueuse

} ClassementWTA;



//Définition des fonctions



/*

 @brief Rend l'indice du tableau tournoiWTA->dataTournois[i] en fonction du nom et date en paramètre

 @param[in] tournoiWTA le pointeur sur TournoiWTA

 @param[in] nom le nom du tournoi recherché

 @param[in] date la date du tournoi recherché

 @return Si le tournoi est trouvé Alors son indice Sinon MAXTOURNOI + 1

 */

unsigned int index_tournoi(const TournoiWTA* tournoiWTA, char nom[TAILLECHAR + 1], unsigned int date) { //trouve l'indice d'un tournoi en fonction de son nom et sa date

    unsigned int indice = MAXTOURNOI + 1;



    for (unsigned int i = 0; i < tournoiWTA->idxT; i++) {
```

```
                    if   (strcmp(tournoiWTA->dataTournois[i].nom,   nom)   ==   0   &&
tournoiWTA->dataTournois[i].date == date) {

            indice = i;

            break;

        }

    }

     return indice;  //si le tournoi est trouvé, alors on retourne son indice, sinon
MAXTOURNOI + 1

}



/*

 @brief Rend l'indice du tableau tournoiWTA->dataJoueuses[] d'une joueuse

 @param[in] tournoiWTA le pointeur sur TournoiWTA

 @param[in] joueuse le nom de la joueuse recherchée

 @return l'indice de la joueuse recherchée sinon NBJOUEUSES * MAXTOURNOI + 1

 */

unsigned int idxfind(const TournoiWTA* tournoiWTA, const char joueuse[TAILLECHAR +
1]) {     //fonction pour dénicher les index des joueuses en fonction du nom de la
joueuse

    unsigned int position = NBJOUEUSES * MAXTOURNOI + 1;

    for (unsigned int i = 0; i < tournoiWTA->idxJ; i++) {

        if (strcmp(tournoiWTA->dataJoueuses[i].nom, joueuse) == 0) {
```

```
            position = i;
            break;
        }
    }
    return position;     // si la joueuse est trouvé on retourne son indice, sinon NBJOUEUSES * MAXTOURNOI + 1
}

/*
  @brief Initialise le nombre de points à 0 de toutes les joueuses dans tournoiWTA->dataJoueuses[]
 @param[in] tournoiWTA le pointeur sur TournoiWTA
*/
void creer_classement(TournoiWTA* tournoiWTA) {    //initialise tous les points des joueuses à 0
    for (unsigned int i = 0; i < tournoiWTA->idxJ; i++) {
        tournoiWTA->dataJoueuses[i].points = 0;
    }
}

/*
```

```
 @brief Ajoute des points aux points des joueuses participante à un tournoi selon leur position

 @param[in] tournoiWTA  le pointeur sur TournoiWTA

 @param[in] tournoi le tournoi qui donnera des points aux joueuses

*/

void ajouter_points_joueuse(TournoiWTA* tournoiWTA, Tournoi tournoi) {  //ajoute des points aux points des joueuses participante à un tournoi selon leur position

    // Joueuses perdantes du tournoi

    for (unsigned int i = 0; i < NBMATCH; i++) {

                tournoiWTA->dataJoueuses[tournoi.dataMatch[i].idxPerdante].points  = tournoiWTA->dataJoueuses[tournoi.dataMatch[i].idxPerdante].points                    + calculer_points_tournoi(i);

    }



    // Joueuse gagnante du tournoi

      tournoiWTA->dataJoueuses[tournoi.dataMatch[NBMATCH  -  1].idxGagnante].points  = tournoiWTA->dataJoueuses[tournoi.dataMatch[NBMATCH    -    1].idxGagnante].points    + NBPOINTS_FINALE_GAGNANT;

}



/*

 @brief Calcule le nombre de points d'une joueuse selon le nombre de points obtenu a une position et retourne son nombre de points en fonction de cette position

 @param[in] i la position du match perdu
```

```
 @return Les points obtenu par la joueuse

*/

unsigned int calculer_points_tournoi(const int i) { // Calcule le nombre de points d'une joueuse selon le nombre de points obtenu a une position et retourne son nombre de points en fonction de cette position

    if (i < POS_64)

        return NBPOINTS_64EME;

    else if (i < POS_32)

        return NBPOINTS_32EME;

    else if (i < POS_16)

        return NBPOINTS_16EME;

    else if (i < POS_8)

        return NBPOINTS_8EME;

    else if (i < POS_4)

        return NBPOINTS_QUARTS;

    else if (i < POS_2)

        return NBPOINTS_DEMI;

    else

        return NBPOINTS_FINALE_PERDANT;

}
```

```
/*
 @brief trie le tableau classementWTA[] par ordre croissant
 @param[in] tournoiWTA le pointeur sur TournoiWTA (en lecture)
 @param[in] le tableau classementWTA[] contenant la liste des joueuses à trier
*/
void tri(const TournoiWTA* tournoiWTA, ClassementWTA classementWTA[]) {    //tri le tableau classementWTA par ordre croissant
    for (unsigned int i = 1; i <= tournoiWTA->idxJ; i++) {
        ClassementWTA classementWTARef = classementWTA[i];
        unsigned int j = i;

                    while (j > 0 && strcmp(classementWTA[j - 1].orderBy, classementWTARef.orderBy) > 0) {
            classementWTA[j] = classementWTA[j - 1];
            j--;
        }

        classementWTA[j] = classementWTARef;
    }
}
```

```
/*
 @brief Affiche le nom des joueuses et leur points du classementWTA[]
 @param[in] tournoiWTA le pointeur sur TournoiWTA
 @param[in] classementWTA[] le tableau des joueuses
*/
void afficher_classementWTA(const TournoiWTA* tournoiWTA, const ClassementWTA classementWTA[]) {    //affiche le nom des joueuses et leur points du classementWTA[]
    for (unsigned int i = 0; i < tournoiWTA->idxJ; i++) {
        if (classementWTA[i].joueuse.points > 0) {
                                    printf("%s   %d\n",   classementWTA[i].joueuse.nom, classementWTA[i].joueuse.points);
        }
    }
}

/*
 @brief Enregistre dans tournoiWTA le nombre de tournois total
 @param[in] tournoiWTA le pointeur sur TournoiWTA
*/
void definir_nombre_tournois(TournoiWTA* tournoiWTA) {   //enregistre dans tournoiWTA le nombre de tournois total
```

```
    unsigned int nb = 0;
    scanf("%d", &nb);
    tournoiWTA->nbTournois = nb;
}

/*
 @brief Enregistre dans ins->dataTournois[] le tournoi
 @param[in] tournoiWTA le pointeur sur TournoiWTA
*/
void enregistrement_tournoi(TournoiWTA* ins) {  //enregistre dans ins->dataTournois[] le tournoi
    unsigned int date;

    // Lecture du nom et de la date du tournoi
    scanf("%s %d", &ins->dataTournois[ins->idxT].nom, &date);
    ins->dataTournois[ins->idxT].date = date;

     for (int i = 0; i < NBMATCH; i++) { // On boucle sur tous les matchs pour les enregistrer
        char gagnante[TAILLECHAR + 1], perdante[TAILLECHAR + 1];
        int idxGagnante, idxPerdante;
```

```
            scanf("%s %s", &gagnante, &perdante);   // Lecture du nom de la joueuse gagnante et du nom de la joueuse perdante




        idxGagnante = idxfind(ins, gagnante);
        if (idxGagnante < NBJOUEUSES * MAXTOURNOI + 1) {
            // si la gagnante est connue
            ins->dataTournois[ins->idxT].dataMatch[i].idxGagnante = idxGagnante;
        }
        else {
            // sinon on l'ajoute
            strcpy(ins->dataJoueuses[ins->idxJ].nom, gagnante);
            ins->dataTournois[ins->idxT].dataMatch[i].idxGagnante = ins->idxJ;
            ins->idxJ = ins->idxJ + 1;
        }



        idxPerdante = idxfind(ins, perdante);
        if (idxPerdante < NBJOUEUSES * MAXTOURNOI + 1) {
            // si la perdante est connue
```

```
            ins->dataTournois[ins->idxT].dataMatch[i].idxPerdante = idxPerdante;
        }
        else {
            // sinon on l'ajoute
            strcpy(ins->dataJoueuses[ins->idxJ].nom, perdante);
            ins->dataTournois[ins->idxT].dataMatch[i].idxPerdante = ins->idxJ;
            ins->idxJ = ins->idxJ + 1;
        }
    }
    ins->idxT = ins->idxT + 1;
}

/*
 @brief Affiche la liste des matchs d'un tournoi
 @param[in] tournoiWTA le pointeur sur TournoiWTA
 //Si le tournoi n'est pas trouvé affiche "tournoi inconnu"
 //Sinon on affiche la liste des matchs
*/
void affichage_matchs_tournoi(const TournoiWTA* tournoiWTA) {   //Afiche la liste des matchs d'un tournoi
```

```
    char nomTournoi[TAILLECHAR + 1];
    unsigned int dateTournoi = 0;
    unsigned int idxT;

    scanf("%s %d", nomTournoi, &dateTournoi);
    idxT = index_tournoi(tournoiWTA, nomTournoi, dateTournoi);
    if (idxT < MAXTOURNOI + 1) {
                            printf("%s    %d\n",    tournoiWTA->dataTournois[idxT].nom,
tournoiWTA->dataTournois[idxT].date);
        for (unsigned int i = 0; i < NBMATCH; i++) {
            if (i == 0)
                printf("64emes de finale\n");
            if (i == POS_64)
                printf("32emes de finale\n");
            if (i == POS_32)
                printf("16emes de finale\n");
            if (i == POS_16)
                printf("8emes de finale\n");
            if (i == POS_8)
                printf("quarts de finale\n");
```

```
            if (i == POS_4)
                printf("demi-finales\n");
            if (i == POS_2)
                printf("finale\n");

            printf("%s %s\n",
tournoiWTA->dataJoueuses[tournoiWTA->dataTournois[idxT].dataMatch[i].idxGagnante].nom
,
tournoiWTA->dataJoueuses[tournoiWTA->dataTournois[idxT].dataMatch[i].idxPerdante].nom
             );   //On affiche le nom de la joueuse qui a pour indice de joueuse i du
match i du tournoi idxT 2 fois pour la gagnante et la perdante
        }
    }
    else {  // si le tournoi n'est pas trouvé
        printf("tournoi inconnu\n");
    }
}

/*
 @brief Affiche tous les matchs d'une joueuse d'un tournoi
```

```
 @param[in] tournoiWTA le pointeur sur TournoiWTA

 //Si le tournoi n'est pas trouvé affiche "tournoi inconnu"

 //Si la joueuse n'est pas trouvée affiche "joueuse inconnue"

 //Sinon on affiche la liste des matches auxquels la joueuse a participé

*/

void afficher_matchs_joueuse(TournoiWTA* tournoiWTA) {    //Affiche tous les matchs d'une joueuse d'un tournoi

    char  nomDuTournoiRecherche[TAILLECHAR + 1], joueuseRecherchee[TAILLECHAR + 1];

    unsigned int dateTournoie;



    scanf("%s %d %s", nomDuTournoiRecherche, &dateTournoie, joueuseRecherchee);



          unsigned  int  idxT  =  index_tournoi(tournoiWTA,  nomDuTournoiRecherche, dateTournoie);

    if (idxT < MAXTOURNOI + 1) {

        unsigned int idxJ = idxfind(tournoiWTA, joueuseRecherchee);



        if (idxJ < NBJOUEUSES * MAXTOURNOI + 1) {

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

                if (tournoiWTA->dataTournois[idxT].dataMatch[i].idxGagnante == idxJ ||
```

```
                    tournoiWTA->dataTournois[idxT].dataMatch[i].idxPerdante == idxJ)
{
                printf(
                    "%s %s\n",
tournoiWTA->dataJoueuses[tournoiWTA->dataTournois[idxT].dataMatch[i].idxGagnante].nom
,
tournoiWTA->dataJoueuses[tournoiWTA->dataTournois[idxT].dataMatch[i].idxPerdante].nom
                        );  //affiche la liste des matches auxquels la joueuse a
participé
            }
        }
    }
    else {      //si la joueuse n'est pas trouvé
        printf("joueuse inconnue\n");
    }
}
else {      //si le tournoi n'est pas trouvé
    printf("tournoi inconnu\n");
}
}
```

```
/*

 @brief Affiche le classement des joueuses participante à un tournoi par ordre alphabétique

 @param[in] tournoiWTA le pointeur sur TournoiWTA

 //Si le tournoi n'est pas trouvé affiche "tournoi inconnu"

 //Sinon on affiche la liste triée des joueuses avec leurs points

*/

void affichage_joueuses_tournoi(TournoiWTA* tournoiWTA) {    //Affiche le classement des joueuse participante à un tournoi par ordre alphabétique

    char  nomDuTournoiRecherche[TAILLECHAR + 1];

    unsigned int dateTournoie;



    scanf("%s %d", nomDuTournoiRecherche, &dateTournoie);



          unsigned int idxT = index_tournoi(tournoiWTA, nomDuTournoiRecherche, dateTournoie);

    if (idxT < MAXTOURNOI + 1) {

        // Le tournoi selectionné

        Tournoi tournoi = tournoiWTA->dataTournois[idxT];
```

```
        creer_classement(tournoiWTA);

        ajouter_points_joueuse(tournoiWTA, tournoi);



        ClassementWTA classementWTA[NBJOUEUSES * MAXTOURNOI];

        for (unsigned int i = 0; i < tournoiWTA->idxJ; i++) {

            classementWTA[i].joueuse = tournoiWTA->dataJoueuses[i];

             strcpy(classementWTA[i].orderBy, tournoiWTA->dataJoueuses[i].nom); //tri sur le nom des joueuses participantes

        }



        tri(tournoiWTA, classementWTA);

        printf("%s %d\n", nomDuTournoiRecherche, dateTournoie);

           afficher_classementWTA(tournoiWTA, classementWTA);   //Affiche le nom des joueuses et leur nombre de points

    }

    else {  //si le tournoi n'est pas trouvé

        printf("tournoi inconnu\n");

    }

}



/*
```

```
 Affiche le classement des joueuses participantes à des tournois au maximun les 4 derniers tournois

 @param[in] tournoiWTA le pointeur sur TournoiWTA

*/

void afficher_classement(TournoiWTA* tournoiWTA) {    //Affiche le classement des joueuses participantes à des tournois

    creer_classement(tournoiWTA);



    unsigned int i_min_tournoi = 0;

     if (tournoiWTA->idxT > MAXTOURNOI_CLASSEMENT) i_min_tournoi = tournoiWTA->idxT - MAXTOURNOI_CLASSEMENT;



     for (unsigned int i = i_min_tournoi; i < tournoiWTA->idxT; i++) {    // On ajoute les points des joueuses participantes

         ajouter_points_joueuse(tournoiWTA, tournoiWTA->dataTournois[i]);

    }



    ClassementWTA classementWTA[NBJOUEUSES * MAXTOURNOI];

    for (unsigned int i = 0; i < tournoiWTA->idxJ; i++) {



        char orderBy[TAILLECHAR + 5 + 1];
```

```
        // on met à orderBy ses points inversé et le nom de la joueuse

                sprintf(orderBy,  "%05d",  NBPOINTS_FINALE_GAGNANT  *  MAXTOURNOI  -
tournoiWTA->dataJoueuses[i].points);

        strcat(orderBy, tournoiWTA->dataJoueuses[i].nom);



        classementWTA[i].joueuse = tournoiWTA->dataJoueuses[i];

        strcpy(classementWTA[i].orderBy, orderBy);

    }



    tri(tournoiWTA, classementWTA);

    afficher_classementWTA(tournoiWTA, classementWTA);

}



// Le main



int main()

{

    TournoiWTA tournoiWTA;

    tournoiWTA.nbTournois = 0;

    tournoiWTA.idxT = 0;
```

```
tournoiWTA.idxJ = 0;


while (1) {
    char mot[TAILLECHAR + 1];

    // Lecture de la commande (mot)
    scanf("%s", mot);

    // si la commande est "exit"
    if (strcmp(mot, "exit") == 0) {
        exit(0); // sortie du programme principal
    }

    // si la commande est "definir_nombre_tournois"
    else if (strcmp(mot, "definir_nombre_tournois") == 0) {
        definir_nombre_tournois(&tournoiWTA);
    }

    // si la commande est "enregistrement_tournoi"
```

```
else if (strcmp(mot, "enregistrement_tournoi") == 0) {
    enregistrement_tournoi(&tournoiWTA);
}

// si la commande est "affichage_matchs_tournoi"
else if (strcmp(mot, "affichage_matchs_tournoi") == 0) {
    affichage_matchs_tournoi(&tournoiWTA);
}

// si la commande est "afficher_matchs_joueuse"
else if (strcmp(mot, "afficher_matchs_joueuse") == 0) {
    afficher_matchs_joueuse(&tournoiWTA);
}

// si la commande est "affichage_joueuses_tournoi"
else if (strcmp(mot, "affichage_joueuses_tournoi") == 0) {
    affichage_joueuses_tournoi(&tournoiWTA);
}
```

```
        // si la commande est "afficher_classement"
        else if (strcmp(mot, "afficher_classement") == 0) {
            afficher_classement(&tournoiWTA);
        }

        else {
            printf("!!! Commande inconnue : *%s*\n", mot);
        }
    }

    system("pause");
    return 0;
}
```

Annexe

Les fonctions en annexe sont des fonctions qui nous ont aidé à faire le projet, mais n'ont pas été explicitement demandées dans le document de cours de ce projet.

Fonction index_tournoi

Permet de trouver un tournoi selon son nom et sa date

```
unsigned int index_tournoi(const TournoiWTA* tournoiWTA, char nom[TAILLECHAR + 1],
unsigned int date) { //trouve l'indice d'un tournoi en fonction de son nom et sa date
```

```
    unsigned int indice = MAXTOURNOI + 1;

    for (unsigned int i = 0; i < tournoiWTA->idxT; i++) {
                    if   (strcmp(tournoiWTA->dataTournois[i].nom,   nom)   ==   0   &&
tournoiWTA->dataTournois[i].date == date) {
            indice = i;
            break;
        }
    }
     return indice;  //si le tournoi est trouvé, alors on retourne son indice, sinon
MAXTOURNOI + 1
 }
```

Fonction idxfind

Permet de trouver l'index d'une joueuse selon son nom

```
unsigned int idxfind(const TournoiWTA* tournoiWTA, const char joueuse[TAILLECHAR +
1]) {     //fonction pour dénicher les index des joueuses en fonction du nom de la
joueuse
    unsigned int position = NBJOUEUSES * MAXTOURNOI + 1;
    for (unsigned int i = 0; i < tournoiWTA->idxJ; i++) {
        if (strcmp(tournoiWTA->dataJoueuses[i].nom, joueuse) == 0) {
            position = i;
            break;
        }
    }
     return position;     // si la joueuse est trouvé on retourne son indice, sinon
NBJOUEUSES * MAXTOURNOI + 1  }
```

Fonction creer_classement

Permet de creer un classement en initialisant tous les points des joueuse à 0 points

```
void creer_classement(TournoiWTA* tournoiWTA) {    //initialise tous les points des
joueuses à 0
    for (unsigned int i = 0; i < tournoiWTA->idxJ; i++) {
        tournoiWTA->dataJoueuses[i].points = 0;
    }
 }
```

Fonction ajouter_points_joueuse

ajoute des points aux points des joueuses participante à un tournoi selon leur position

```
void ajouter_points_joueuse(TournoiWTA* tournoiWTA, Tournoi tournoi) {  //ajoute des points aux points des joueuses participante à un tournoi selon leur position
     // Joueuses perdantes du tournoi
     for (unsigned int i = 0; i < NBMATCH; i++) {
                 tournoiWTA->dataJoueuses[tournoi.dataMatch[i].idxPerdante].points  = tournoiWTA->dataJoueuses[tournoi.dataMatch[i].idxPerdante].points                    + calculer_points_tournoi(i);
     }

     // Joueuse gagnante du tournoi
       tournoiWTA->dataJoueuses[tournoi.dataMatch[NBMATCH  -  1].idxGagnante].points  = tournoiWTA->dataJoueuses[tournoi.dataMatch[NBMATCH    -    1].idxGagnante].points    + NBPOINTS_FINALE_GAGNANT;
 }
```

Fonction calculer_points_tournoi

Calcule le nombre de points d'une joueuse selon un rang et retourne son nombre de points en fonction de cette position

```
unsigned int calculer_points_tournoi(const int i) { // Calcule le nombre de points d'une joueuse selon un rang et retourne son nombre de points en fonction de cette position
     if (i < POS_64)
         return NBPOINTS_64EME;
     else if (i < POS_32)
         return NBPOINTS_32EME;
     else if (i < POS_16)
         return NBPOINTS_16EME;
     else if (i < POS_8)
         return NBPOINTS_8EME;
```

```
    else if (i < POS_4)
        return NBPOINTS_QUARTS;
    else if (i < POS_2)
        return NBPOINTS_DEMI;
    else
        return NBPOINTS_FINALE_PERDANT;
}
```

Fonction tri

Permet de trier le tableau classementWTA par ordre croissant

```
void tri(const TournoiWTA* tournoiWTA, ClassementWTA classementWTA[]) {    //tri le tableau classementWTA par ordre croissant
    for (unsigned int i = 1; i <= tournoiWTA->idxJ; i++) {
        ClassementWTA classementWTARef = classementWTA[i];
        unsigned int j = i;

                    while (j > 0 && strcmp(classementWTA[j - 1].orderBy, classementWTARef.orderBy) > 0) {
            classementWTA[j] = classementWTA[j - 1];
            j--;
        }

        classementWTA[j] = classementWTARef;
    }
}
```

Fonction afficher_classementWTA

Permet d'afficher le nom des joueuses et leur points du classementWTA

```
void afficher_classementWTA(const TournoiWTA* tournoiWTA, const ClassementWTA classementWTA[]) {    //affiche le nom des joueuse et leur points du classementWTA[]
    for (unsigned int i = 0; i < tournoiWTA->idxJ; i++) {
        if (classementWTA[i].joueuse.points > 0) {
            printf("%s %d\n", classementWTA[i].joueuse.nom, classementWTA[i].joueuse.points);
        }
    }
}
```

Trace d'exécution

Le test du sprint que nous avons atteint est le 4.

Voici le out de notre fichier source pour le inSp4 nommé testOutSp4.txt et le out du inSp4 donné en début de projet pour comparer nommé outSp4.txt:

Allan Herillus, Elias Ait-Khelifa

testOutSp4.txt - Bloc-notes

Fichier Modifier Format Affichage Aide

```
Krejcikova 2180
Barty 2045
Pavlyuchenkova 1290
KaPliskova 1245
Sabalenka 810
Sakkari 765
Kerber 730
Zidansek 730
Badosa 540
Gauff 540
Jabeur 540
Rybakina 540
Swiatek 540
Muchova 450
Tomljanovic 405
Golubic 370
Cirstea 270
Keys 270
Stephens 270
Azarenka 225
Kenin 225
Kostyuk 225
Vondrousova 225
SWilliams 190
Linette 180
Mertens 180
Raducanu 180
Samsonova 180
Siniakova 180
Brengle 135
Collins 135
Kasatkina 135
Martincova 135
Pegula 135
Sasnovich 135
Svitolina 135
Vesnina 135
Begu 100
Bogdan 100
Gracheva 100
Hercog 100
Juvan 100
```

outSp4.txt - Bloc-notes

Fichier Modifier Format Affichage Aide

```
Krejcikova 2180
Barty 2045
Pavlyuchenkova 1290
KaPliskova 1245
Sabalenka 810
Sakkari 765
Kerber 730
Zidansek 730
Badosa 540
Gauff 540
Jabeur 540
Rybakina 540
Swiatek 540
Muchova 450
Tomljanovic 405
Golubic 370
Cirstea 270
Keys 270
Stephens 270
Azarenka 225
Kenin 225
Kostyuk 225
Vondrousova 225
SWilliams 190
Linette 180
Mertens 180
Raducanu 180
Samsonova 180
Siniakova 180
Brengle 135
Collins 135
Kasatkina 135
Martincova 135
Pegula 135
Sasnovich 135
Svitolina 135
Vesnina 135
Begu 100
Bogdan 100
Gracheva 100
Hercog 100
Juvan 100
```