

RAPPORT DE PROJET

Projet IAP

Mondial de slalom a ski

CARRIERE Florian – VIGNARAJAH Daran | IAP | 05/11/2018

REMERCIEMENTS

Nous adressons nos remerciements à nos professeurs d’IAP qui ont su durant les TP et les aides aux projet nous conseiller et nous corriger. Nous tenons aussi à remercier l’ensemble de notre groupe (103) au sein duquel règne un véritable esprit d’entre-aide.

Table des matières

I.Présentation de l’application ………………………...…………………………….. 1

Objectifs ………………………………………………………………………………….. 1

Contraintes ………………………………………………………………………………. 1

II.Organisation des tests………………………………………………………….………. 2

III.Bilan de validation des tests de développement ………….…………….. 4

IV.Bilan de projet ……………………………………………………………………..………5

Annexe …………………………………………………………………………………………….6

Sprint 1 ………………………………………………………………………………………………... 6

Sprint 2 ………………………………………………………………………………………………… 9

Sprint 3 ………………………………………………………………………………………………. 14

Sprint 4 ………………………………………………………………………………………………. 19

InSp4 ………………………………………………………………………………………………….. 26

OutSp4 ………………………………………………………………………………………………. 26

**I. Présentation de l’application**

**Objectifs :**

Précisons avant de parler de l’objectif, le contexte de ce projet :

Le sujet nous place dans un mondial de slalom à ski. Chaque skieur doit réaliser le parcours proposé le plus rapidement possible. Ce dernier doit passer à travers plusieurs portes durant le parcours et se voit ensuite attribué le temps qu’il a effectué à la porte indiquée. S’il ne passe pas une porte, il peut être disqualifié.

L’objectif du projet, comme indiqué dans l’énoncé, est de « programmer un interpréteur de commandes permettant de définir le nombre de portes du parcours, d’inscrire les skieurs, d’enregistrer les temps chronométrés aux différentes portes et à l’arrivée, d’enregistrer les disqualifications et de calculer les classements aux points intermédiaires et le classement final. »

**Contraintes :**

Ce projet fut soumis à de nombreuses contraintes. La première fut le temps : en effet, nous devons réaliser ce projet, en binôme, en un mois (du 25/09 au 26/10, rédaction du projet non-compris).

(Le logiciel que nous avons utilisé afin de coder est Visual Studio 2017, un logiciel d’outil de développement permettant de générer des applications web ou bien de bureautique. Il permet de coder en plusieurs langages mais celui qui nous intéresse est le langage C. En effet, on nous demande de réaliser ce projet en langage C.)

**II. Organisation des tests**

Le projet a été développé de manière « agile ». « *Le développement Agile désigne une méthodologie qui vise à apporter une flexibilité et un pragmatisme accrus aux produits finis. Il privilégie la simplification du code, des tests fréquents et la livraison des fonctionnalités de l'application dès qu'elles sont prêts.* » Le projet était divisé en cinq sprints.

Chacun des sprints a été codé selon le cycle de développement logiciel : prise de connaissance du sujet (**spécification**), conception préliminaire (**pseudo-code**) puis détaillée du code en insérant les fonctions que nous avons utilisé. Enfin, la dernière étape fut le test.  
 Afin de les tester, avait été mis à notre disposition un jeu de données de test (JDT). Ce dernier se composait de cinq fichier inSpn.txt et outSpn.txt. Si les sorties obtenues par notre programme correspondaient aux fichier outSpn.txt, le sprint était validé.

A la fin du projet, l’enseignant responsable du TP IAP 3 a testé notre programme afin de vérifier le sprint de niveau le plus élevé atteint. Si le test fonctionnait, le sprint était considéré comme valide. En revanche s’il y avait une erreur, cinq minutes nous étaient accordées afin de remédier au problème. Dans le cas échéant, le professeur devait tester le sprint de niveau inférieur à celui qui a été montré.

Il nous a été demandé de programmer huit commandes :

***1.  Commande de sortie du programme***

Servant à mettre fin à l’exécution du programme. La chaine de caractère est : "exit"

***2. Commande de définition du parcours***

Elle sert définir le nombre de portes du parcours. La chaine de caractère est :

"definir\_course" suivie du nombre de portes du parcours.

***3. Commande d’inscription d’un skieur***

L’utilité de cette commande est d’enregistrer les temps chronométrés aux différentes portes et à l’arrivée. La chaine de caractère est : "inscrire\_skieur" suivie du nom du skieur ainsi que de son pays. La commande attribue automatiquement un dossard au skieur, en sachant que l’attribution commence à 101 et peut aller jusqu’à un maximum de 151 (50 skieurs maximum)

***4. Commande d’affichage des skieurs***

Elle sert à afficher l’ensemble des skieurs inscrits à une course. Un skieur par ligne sera affiché avec pour chacun leur nom, leur pays et leur numéro de dossard. La chaine de caractère est : "afficher\_skieurs"

***5. Commande d’enregistrement d’un temps chronométré à une porte***

Servant à enregistrer l’ensembles des temps enregistrés pour un skieur à une porte donnée via son dossard. La chaine de caractère est :  "enregistrer\_temps" suivie du temps chronométré (un réel), de la porte considérée et du numéro de dossard.

***6. Commande de disqualification d’un skieur***

Cette commande sert à enregistrer la disqualification d’un skieur à une porte donnée. La chaine de caractère est :  "disqualification\_skieur" suivie du numéro de la porte non passée et du numéro de dossard du skieur.

***7. Commande d’affichage des temps chronométrés d’un skieur***

Elle sert à afficher l’ensembles des temps enregistrés pour un skieur via son dossard. La commande donnera les temps chronométrés disponibles dans les enregistrements du skieur (départ, points intermédiaires dont l’arrivée, disqualification. Elle affichera sur la sortie standard, une ligne pour chaque temps chronométré dans l’ordre chronologique. Chaque ligne aura le format suivant : numéro de la porte, numéro de dossard, et temps chronométré sous forme d’un réel avec une précision de deux chiffres après la virgule. La chaine de caractère est : "afficher\_temps" suivie du numéro de dossard du skieur.

***8. Commande d’affichage du classement à une porte***

La dernière commande sert à afficher le classement des skieurs à une porte donné. La commande calcule pour chaque skieur chronométré à la porte p, le temps de course effectif depuis le départ. Elle affichera, par ligne et dans l’ordre croissant des temps de course, le numéro de la porte, le numéro de dossard, le nom et le pays du skieur, et le temps de course sous forme d’un réel avec une précision de deux chiffres après la virgule. La chaine de caractère est :  "afficher\_classement" suivie du numéro de la porte considérée p (entier supérieur à 0 et inférieur ou égal au nombre de portes du parcours).

**III. Bilan de validation des tests de développement**

**Sprint 1**

Nous devions coder les fonctions inscription\_skieur, affichage\_inscrits, qui avaient, pour la première fonction le rôle d’inscrire le nom et pays du skieur (il se verra ensuite attribué automatiquement un numéro de dossard), et pour la seconde d’afficher tous les skieurs que nous avions inscrits.

Bilan : Après avoir testé les entrées avec le fichier inSp1.txt et après avoir comparé les sorties avec le fichier outSp1.txt, on observe que les sorties coïncident : Le Sprint 1 est donc validé.

Nous avons ainsi pu implémenter les commandes exit, inscrire\_skieur et enfin afficher\_skieurs

**Sprint 2**

Nous devions coder les fonctions definition\_course, enregistrement\_temps et affichage\_temps afin de déclarer le nombre de portes, enregistrer le ou les temps des skieurs à chaque porte et enfin afficher le temps du skieur à la porte indiquée.

Bilan : Après avoir testé les entrées avec le fichier inSp2.txt et après avoir comparé les sorties avec le fichier outSp2.txt, on observe que les sorties coïncident : Le Sprint 2 est donc validé.

Nous avons ainsi pu implémenter les commandes definir\_course, enregistrer\_temps et afficher\_temps.

**Sprint 3**

Nous devions coder la fonction disqualification qui vise à, comme son nom l’indique, disqualifier un skieur à une porte indiquée.

Bilan : Après avoir testé les entrées avec le fichier inSp3.txt et après avoir comparé les sorties avec le fichier outSp3.txt, on observe que les sorties coïncident : Le Sprint 3 est donc validé.

Nous avons ainsi pu implémenter la commande disqualification.

**Sprint 4**

Nous devions coder la fonction affichage\_classement qui doit afficher le classement des skieurs à une porte indiquée.

Bilan : Après avoir testé les entrées avec le fichier inSp4.txt et après avoir comparé les sorties avec le fichier outSp4.txt, on observe que les sorties coïncident : Le Sprint 4 est donc validé.

Nous avons ainsi pu implémenter la commande affichage\_classement.

**Sprint 5**

Nous devions coder la fonction detection\_fin\_course mais nous n’avions pas pu la coder car le temps était manquant et notre compréhension pauvre.

Bilan : Nous n’avons donc pas réalisé le Sprint 5.

**IV. Bilan de projet**

Ce projet nous a permis d’acquérir de nouvelles compétences notamment dans le langage C mais également dans le travail d’équipe.

Bien que nous n’ayons pas totalement terminé l’interpréteur de commande, nous avions réalisé plus de la moitié des commandes (8/12) et nous nous sommes arrêtés au Sprint 4 (avant dernier sprint). Nous sommes quand même satisfaits de notre travail et de ce projet qui a été une première nous concernant.

Ce projet peut être vu comme une initiation au monde du travail. Il nous montre un exemple de ce que pourrait être un projet dans le domaine professionnel : avoir une demande spécifique du client (créer un interpréteur de commande) et répondre à son besoin dans un délai déterminé. En parlant du temps, celui-ci a sans doute été notre plus grande difficulté. Nous aurions probablement pu terminer le sprint 5 si nous avions mieux gérer notre temps. Nous retenons donc que le temps est un facteur très important dans un projet et qu’il faut savoir le gérer.

A travers ce mois de projet, nous avons pu appliquer nos compétences en C que nous avons étudier depuis le début de l’année. Nous avons également rencontré de nombreuses difficultés dans le codage des Sprints, le plus compliqué qui a été le Sprint 4, avec l’affichage du classement : en effet, au début nous arrivions qu’à afficher sans classer les skieurs. Puis avec l’aide de nos professeurs d’IAP qui nous ont expliqué le fonctionnement de tri, nous avons réussi à bien classer les skieurs.

Au début du projet, nous avions eu du mal à maîtriser les pointeurs, qui est une notion très importante du langage C. Mais avec les aides des Travaux Pratiques, Travaux Dirigés et aussi des camarades de notre groupe, nous arrivons maintenant à utiliser les pointeurs bien qu’il nous arrive quelques fois de nous tromper.

Un autre problème qui nous a fait face dès le début du projet était l’utilisation du « scanf ».

En effet ce dernier ne fonctionnait pas. Nous avons donc recherché sur internet afin de remédier à ce problème qui a été d’insérer « #pragma warning(disable:4996) » à l’entête du code.

En conclusion, ce projet n’a été que bénéfique pour nous. Il nous a permis de nous exercer dans le langage C. Nous avons également remarqué que nous faisions plusieurs erreurs d’inattention dans le codage mais cela se corrigera peut-être avec la pratique

**Annexe**

SPRINT 1

/\*

\* sprint\_4.c

\* Auteur :VIGNARAJAH Daran & CARRIERE Florian

\* Groupe : 103

\*/

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<assert.h>

#define lgMot 30 //30 caractères maximum lors de l'entrée

#define maxSkieurs 50 //50 skieurs au maximum dans la course

#define minPortes 2

#define maxPortes 20

#pragma warning(disable:4996)

// Type des skieur

typedef struct {

   //+1 pour stocker le caractère '\0' l'indicateur de fin de chaîne

   char nom[lgMot + 1]; //Nom du skieur

   char pays[lgMot + 1]; //Pays du skieur

   unsigned int dossard; //Dossard du skieur

}Skieur;

// Type des skieurs inscrits

typedef struct {

   Skieur liste[maxSkieurs]; //Tableau ou sont enregistrés les skieurs inscrits

   unsigned int nbInscrits; //Nombre d'inscrits

}Inscrits;

/\*

\*    Fonction qui permet d'inscrire un skieur (nom, pays, dossard) et de le stocker dans un tableau

\*    [in-out] pointeur \*ins vers la structure Insrcits dans laquelle sont définis en détail les skieurs

\*/

void inscrire\_skieur(Inscrits \*ins);

/\*

\*    Fonction qui permet d'afficher les skieurs inscrits

\*    [in-out] pointeur \*ins vers la structure Inscrits dans laquelle sont définis en détail les skieurs

\*/

void afficher\_skieurs(const Inscrits \*ins);

//FONCTION PRINCIPALE début

int main() {

char mot[lgMot+1]; //on stocke ici ce que va remplir l'utilisateur

Inscrits ins;

ins.nbInscrits = 0;

   while (1) //boucle infinie permettant via les 8 commandes l'appel des fonctions du programme

   {

      scanf("%s", &mot); //envoie de l'entrée faite par l'utilisateur à mot

      if (strcmp(mot, "inscrire\_skieur") == 0)

      {

          inscrire\_skieur(&ins);

      }

      else if (strcmp(mot, "afficher\_skieurs") == 0)

      {

          afficher\_skieurs(&ins);

      }

      else if (strcmp(mot, "exit") == 0)

      {

          exit(1); //la boucle infinie est rompue et le programme s'arrête si "exit" vaut 1

      }

   }

   system("pause");

   return 0;

}

//FONCTION PRINCIPALE fin

void inscrire\_skieur(Inscrits \*ins) {  
    char mot[lgMot+1];  
    int dossard = 101;  
  
    assert(ins->nbInscrits < 50);  
  
    scanf("%s", mot);  
    strcpy(ins->liste[ins->nbInscrits].nom, mot);  
    scanf("%s", mot);  
    strcpy(ins->liste[ins->nbInscrits].pays, mot);  
    ins->liste[ins->nbInscrits].dossard = 101 + ins->nbInscrits;  
    ins->nbInscrits++; //incrémentation du nombre d'inscrits après chaque nouvelle inscription  
  
    Skieur s;  
    s.dossard = (100 + ins->nbInscrits); //numéro de dossard attribué en fonction du nombre d'inscrits  
    printf("inscription dossard %d \n", s.dossard);  
}  
  
void afficher\_skieurs(const Inscrits \*ins) {  
  
    for (int i = 0; i < ins->nbInscrits; ++i) //boucle for parcourant l'ensemble du tableau "liste" pour afficher les skieurs  
        printf("%s %s %d \n", ins->liste[i].nom, ins->liste[i].pays, ins->liste[i].dossard);  
}

SPRINT 2

/\*

\* sprint\_4.c

\* Auteur :VIGNARAJAH Daran & CARRIERE Florian

\* Groupe : 103

\*/

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<assert.h>

#define lgMot 30 //30 caractères maximum lors de l'entrée

#define maxSkieurs 50 //50 skieurs au maximum dans la course

#define minPortes 2

#define maxPortes 20

#pragma warning(disable:4996)

// Type des skieur

typedef struct {

   //+1 pour stocker le caractère '\0' l'indicateur de fin de chaîne

   char nom[lgMot + 1]; //Nom du skieur

   char pays[lgMot + 1]; //Pays du skieur

   unsigned int dossard; //Dossard du skieur

}Skieur;

// Type des skieurs inscrits

typedef struct {

   Skieur liste[maxSkieurs]; //Tableau ou sont enregistrés les skieurs inscrits

   unsigned int nbInscrits; //Nombre d'inscrits

}Inscrits;

//Type des mesures propres à chaque skieur

typedef struct {

   double temps; //Temps du skieur lors de la mesure

   unsigned int portes; //Porte passée par le skieur lors de la mesure

   unsigned int dossard; //Dossard du skieur

}Mesure;

//Type des informations données pour une course

typedef struct {

   unsigned int nb\_portes; //Nombre de portes dans la course

   Mesure liste[maxSkieurs\*maxPortes]; //Tableau ou sont enregistrés les mesures de chaque skieur

   int num\_mesure; //Nombre de mesures

}Course;

/\*

\*    Fonction qui permet d'inscrire un skieur (nom, pays, dossard) et de le stocker dans un tableau

\*    [in-out] pointeur \*ins vers la structure Inscrits dans laquelle sont définis en détail les skieurs

\*/

void inscrire\_skieur(Inscrits \*ins);

/\*

\*    Fonction qui permet d'afficher les skieurs inscrits

\*    [in-out] pointeur \*ins vers la structure Inscrits dans laquelle sont définis en détail les skieurs

\*/

void afficher\_skieurs(const Inscrits \*ins);

/\*

\*    Fonction qui permet de définir un nombre de portes pour la course

\*    [in-out] pointeur \*crs vers la structure Course dans laquelle on définit le nombre de porte de la course

\*/

void definition\_course(Course \*crs);

/\*

\*    Fonction qui permet d'enregistrer un temps pour un skieur à une porte précise grâce à son dossard

\*    [in-out] pointeurs \*ins et \*crs vers les structures Inscrits et Course

\*/

void enregistrer\_temps(Inscrits \*ins, Course \*crs);

/\*

\*    Fonction qui permet : d'afficher le temps d'un skieur à une porte donnée

\*                       : d'afficher l'éventuelle disqualification d'un skieur à une porte donnée

\*    [in-out] pointeur \*crs vers la structure Course

\*/

void afficher\_temps(Course \*crs);

//FONCTION PRINCIPALE début

int main() {

char mot[lgMot+1]; //on stocke ici ce que va remplir l'utilisateur

Inscrits ins;

Course crs;

ins.nbInscrits = 0;

crs.num\_mesure = 0;

   while (1) //boucle infine permettant via les 8 commandes l'appel des fonctions du programme

   {

      scanf("%s", &mot); //envoie de l'entrée faite par l'utilisateur à mot

      if (strcmp(mot, "inscrire\_skieur") == 0)

      {

          inscrire\_skieur(&ins);

      }

      else if (strcmp(mot, "afficher\_skieurs") == 0)

      {

          afficher\_skieurs(&ins);

      }

      else if (strcmp(mot, "exit") == 0)

      {

          exit(1); //la boucle infinie est rompue et le programme s'arrête si "exit" vaut 1

      }

      else if (strcmp(mot, "definir\_course") == 0)

      {

          definition\_course(&crs);

      }

      else if (strcmp(mot, "enregistrer\_temps") == 0)

      {

          enregistrer\_temps(&ins,&crs);

      }

      else if (strcmp(mot, "afficher\_temps") == 0)

      {

          afficher\_temps(&crs);

      }

   }

   system("pause");

   return 0;

}

//FONCTION PRINCIPALE fin

void inscrire\_skieur(Inscrits \*ins) {

   char mot[lgMot+1];

   int dossard = 101;

   assert(ins->nbInscrits < 50);

   scanf("%s", mot);

   strcpy(ins->liste[ins->nbInscrits].nom, mot);

   scanf("%s", mot);

   strcpy(ins->liste[ins->nbInscrits].pays, mot);

   ins->liste[ins->nbInscrits].dossard = 101 + ins->nbInscrits;

   ins->nbInscrits++; //incrémentation du nombre d'inscrits après chaque nouvelle inscription

   Skieur s;

   s.dossard = (100 + ins->nbInscrits); //numéro de dossard attribué en fonction du nombre d'inscrits

   printf("inscription dossard %d \n", s.dossard);

}

void afficher\_skieurs(const Inscrits \*ins) {

   for (int i = 0; i < ins->nbInscrits; ++i) //boucle for parcourant l'ensemble du tableau "liste" pour afficher les skieurs

      printf("%s %s %d \n", ins->liste[i].nom, ins->liste[i].pays, ins->liste[i].dossard);

}

void definition\_course(Course \*crs) {

   int portes;

   scanf("%d", &portes);

   assert(portes >= minPortes || portes <= maxPortes);

}

void enregistrer\_temps(Inscrits \*ins, Course \*crs) {

   float temps;

   unsigned int portes, dossard;

   Mesure mesure;

   scanf("%f", &temps);

   mesure.temps = temps;

   scanf("%u", &portes);

   mesure.portes = portes;

   scanf("%u", &dossard);

   mesure.dossard = dossard;

   crs->liste[crs->num\_mesure] = mesure; //stockage des composantes de la variable "mesure" dans le tableau "liste" à l'indice "num\_mesure"

   crs->num\_mesure++;

}

void afficher\_temps(Course \*crs) {

   unsigned int dossard;

   Mesure mesure;

   scanf("%u", &dossard);

   for (int i = 0; i < crs->num\_mesure; i++) {

      if (dossard == crs->liste[i].dossard)

          printf("%u %u %.2f \n", crs->liste[i].portes, crs->liste[i].dossard, crs->liste[i].temps);

}

SPRINT 3

/\*

\* sprint\_4.c

\* Auteur :VIGNARAJAH Daran & CARRIERE Florian

\* Groupe : 103

\* Date de création : 18/10/2018

\*/

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<assert.h>

#define lgMot 30 //30 caractères maximum lors de l'entrée

#define maxSkieurs 50 //50 skieurs au maximum dans la course

#define minPortes 2

#define maxPortes 20

#pragma warning(disable:4996)

// Type des skieur

typedef struct {

   //+1 pour stocker le caractère '\0' l'indicateur de fin de chaîne

   char nom[lgMot + 1]; //Nom du skieur

   char pays[lgMot + 1]; //Pays du skieur

   unsigned int dossard; //Dossard du skieur

}Skieur;

// Type des skieurs inscrits

typedef struct {

   Skieur liste[maxSkieurs]; //Tableau ou sont enregistrés les skieurs inscrits

   unsigned int nbInscrits; //Nombre d'inscrits

}Inscrits;

//Type des mesures propres à chaque skieur

typedef struct {

   double temps; //Temps du skieur lors de la mesure

   unsigned int portes; //Porte passée par le skieur lors de la mesure

   unsigned int dossard; //Dossard du skieur

}Mesure;

//Type des informations données pour une course

typedef struct {

   unsigned int nb\_portes; //Nombre de portes dans la course

   Mesure liste[maxSkieurs\*maxPortes]; //Tableau ou sont enregistrés les mesures de chaque skieur

   int num\_mesure; //Nombre de mesures

}Course;

/\*

\*    Fonction qui permet d'inscrire un skieur (nom, pays, dossard) et de le stocker dans un tableau

\*    [in-out] pointeur \*ins vers la structure Inscrits dans laquelle sont définis en détail les skieurs

\*/

void inscrire\_skieur(Inscrits \*ins);

/\*

\*    Fonction qui permet d'afficher les skieurs inscrits

\*    [in-out] pointeur \*ins vers la structure Inscrits dans laquelle sont définis en détail les skieurs

\*/

void afficher\_skieurs(const Inscrits \*ins);

/\*

\*    Fonction qui permet de définir un nombre de portes pour la course

\*    [in-out] pointeur \*crs vers la structure Course dans laquelle on définit le nombre de porte de la course

\*/

void definition\_course(Course \*crs);

/\*

\*    Fonction qui permet d'enregistrer un temps pour un skieur à une porte précise grâce à son dossard

\*    [in-out] pointeurs \*ins et \*crs vers les structures Inscrits et Course

\*/

void enregistrer\_temps(Inscrits \*ins, Course \*crs);

/\*

\*    Fonction qui permet : d'afficher le temps d'un skieur à une porte donnée

\*                       : d'afficher l'éventuelle disqualification d'un skieur à une porte donnée

\*    [in-out] pointeur \*crs vers la structure Course

\*/

void afficher\_temps(Course \*crs);

/\*

\*    Fonction qui permet : de disqualifier un skieur à une porte donnée grâce à son dossard

\*    [in-out] pointeur \*crs vers la structure Course

\*/

void disqualification\_skieur(Course \*crs);

//FONCTION PRINCIPALE début

int main() {

char mot[lgMot+1]; //on stocke ici ce que va remplir l'utilisateur

Inscrits ins;

Course crs;

ins.nbInscrits = 0;

crs.num\_mesure = 0;

   while (1) //boucle infinie permettant via les 8 commandes l'appel des fonctions du programme

   {

      scanf("%s", &mot); //envoie de l'entrée faite par l'utilisateur à mot

      if (strcmp(mot, "inscrire\_skieur") == 0)

      {

          inscrire\_skieur(&ins);

      }

      else if (strcmp(mot, "afficher\_skieurs") == 0)

      {

          afficher\_skieurs(&ins);

      }

      else if (strcmp(mot, "exit") == 0)

      {

          exit(1); //la boucle infinie est rompue et le programme s'arrête si "exit" vaut 1

      }

      else if (strcmp(mot, "definir\_course") == 0)

      {

          definition\_course(&crs);

      }

      else if (strcmp(mot, "enregistrer\_temps") == 0)

      {

          enregistrer\_temps(&ins,&crs);

      }

      else if (strcmp(mot, "afficher\_temps") == 0)

      {

          afficher\_temps(&crs);

      }

      else if (strcmp(mot, "disqualification") == 0)

      {

          disqualification\_skieur(&crs);

      }

   }

   system("pause");

   return 0;

}

//FONCTION PRINCIPALE fin

void inscrire\_skieur(Inscrits \*ins) {

   char mot[lgMot+1];

   int dossard = 101;

   assert(ins->nbInscrits < 50);

   scanf("%s", mot);

   strcpy(ins->liste[ins->nbInscrits].nom, mot);

   scanf("%s", mot);

   strcpy(ins->liste[ins->nbInscrits].pays, mot);

   ins->liste[ins->nbInscrits].dossard = 101 + ins->nbInscrits;

   ins->nbInscrits++; //incrémentation du nombre d'inscrits après chaque nouvelle inscription

   Skieur s;

   s.dossard = (100 + ins->nbInscrits); //numéro de dossard attribué en fonction du nombre d'inscrits

   printf("inscription dossard %d \n", s.dossard);

}

void afficher\_skieurs(const Inscrits \*ins) {

   for (int i = 0; i < ins->nbInscrits; ++i) //boucle for parcourant l'ensemble du tableau "liste" pour afficher les skieurs

      printf("%s %s %d \n", ins->liste[i].nom, ins->liste[i].pays, ins->liste[i].dossard);

}

void definition\_course(Course \*crs) {

   int portes;

   scanf("%d", &portes);

   assert(portes >= minPortes || portes <= maxPortes);

}

void enregistrer\_temps(Inscrits \*ins, Course \*crs) {

   float temps;

   unsigned int portes, dossard;

   Mesure mesure;

   scanf("%f", &temps);

   mesure.temps = temps;

   scanf("%u", &portes);

   mesure.portes = portes;

   scanf("%u", &dossard);

   mesure.dossard = dossard;

   crs->liste[crs->num\_mesure] = mesure; //stockage des composantes de la variable "mesure" dans le tableau "liste" à l'indice "num\_mesure"

   crs->num\_mesure++;

}

void afficher\_temps(Course \*crs) {

   unsigned int dossard;

   Mesure mesure;

   scanf("%u", &dossard);

   for (int i = 0; i < crs->num\_mesure; i++) {

      if (dossard == crs->liste[i].dossard) {

          if (crs->liste[i].temps == -1) { //si skieur détecté comme disqualifié, affichage disqualification à x portes, x dossard

              printf("%u %u disqualification  \n", crs->liste[i].portes, crs->liste[i].dossard);

          }

          else

          {

              printf("%u %u %.2f \n", crs->liste[i].portes, crs->liste[i].dossard, crs->liste[i].temps);

          }

      }

   }

}

void disqualification\_skieur(Course \*crs) {

   unsigned int portes;

   unsigned int dossards;

   Mesure mesure;

   scanf("%u", &portes);

   mesure.portes = portes;

   scanf("%u", &dossards);

   mesure.dossard = dossards;

   mesure.temps = -1.; //La valeur -1 définit un skieur comme étant disqualifié

   crs->liste[crs->num\_mesure] = mesure;

   crs->num\_mesure++;

}

SPRINT 4

/\*

\* sprint\_4.c

\* Auteur :VIGNARAJAH Daran & CARRIERE Florian

\* Groupe : 103

\*/

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<assert.h>

#define lgMot 30 //30 caractères maximum lors de l'entrée

#define maxSkieurs 50 //50 skieurs au maximum dans la course

#define minPortes 2

#define maxPortes 20

#pragma warning(disable:4996)

// Type des skieur

typedef struct {

   //+1 pour stocker le caractère '\0' l'indicateur de fin de chaîne

   char nom[lgMot + 1]; //Nom du skieur

   char pays[lgMot + 1]; //Pays du skieur

   unsigned int dossard; //Dossard du skieur

}Skieur;

// Type des skieurs inscrits

typedef struct {

   Skieur liste[maxSkieurs]; //Tableau ou sont enregistrés les skieurs inscrits

   unsigned int nbInscrits; //Nombre d'inscrits

}Inscrits;

//Type des mesures propres à chaque skieur

typedef struct {

   double temps; //Temps du skieur lors de la mesure

   unsigned int portes; //Porte passée par le skieur lors de la mesure

   unsigned int dossard; //Dossard du skieur

}Mesure;

//Type des informations données pour une course

typedef struct {

   unsigned int nb\_portes; //Nombre de portes dans la course

   Mesure liste[maxSkieurs\*maxPortes]; //Tableau ou sont enregistrés les mesures de chaque skieur

   int num\_mesure; //Nombre de mesures

}Course;

/\*

\*    Fonction qui permet d'inscrire un skieur (nom, pays, dossard) et de le stocker       \* dans un tableau

\*    [in-out] pointeur \*ins vers la structure Inscrits dans laquelle sont définis en détail   \* les skieurs

\*/

void inscrire\_skieur(Inscrits \*ins);

/\*

\*    Fonction qui permet d'afficher les skieurs inscrits

\*    [in-out] pointeur \*ins vers la structure Inscrits dans laquelle sont définis en détail  \* les skieurs

\*/

void afficher\_skieurs(const Inscrits \*ins);

/\*

\*    Fonction qui permet de définir un nombre de portes pour la course

\*    [in-out] pointeur \*crs vers la structure Course dans laquelle on définit le nombre     \* de porte de la course

\*/

void definition\_course(Course \*crs);

/\*

\*    Fonction qui permet d'enregistrer un temps pour un skieur à une porte précise      \* grâce à son dossard

\*    [in-out] pointeurs \*ins et \*crs vers les structures Inscrits et Course

\*/

void enregistrer\_temps(Inscrits \*ins, Course \*crs);

/\*

\*    Fonction qui permet : d'afficher le temps d'un skieur à une porte donnée

\*                    : d'afficher l'éventuelle disqualification d'un skieur à une         \*                    porte donnée

\*    [in-out] pointeur \*crs vers la structure Course

\*/

void afficher\_temps(Course \*crs);

/\*

\*    Fonction qui permet : de disqualifier un skieur à une porte donnée grâce à son     \*                    dossard

\*    [in-out] pointeur \*crs vers la structure Course

\*/

void disqualification\_skieur(Course \*crs);

/\*

\*    Fonction qui permet d'afficher le classement (dans l'ordre croissant) d'une            \* course à une porte donné (sous la forme porte, dossard, nom, pays, temps)

\*    [in-out] pointeurs \*ins et \*crs vers les structures Inscrits et Course

\*/

void affichage\_classement(Inscrits \*ins, Course \*crs);

//FONCTION PRINCIPALE début

int main() {

char mot[lgMot+1]; //on stocke ici ce que va remplir l'utilisateur

Inscrits ins;

Course crs;

ins.nbInscrits = 0;

crs.num\_mesure = 0;

   while (1) //boucle infinie permettant via les 8 commandes l'appel des fonctions du programme

   {

      scanf("%s", &mot); //envoie de l'entrée faite par l'utilisateur à mot

      if (strcmp(mot, "inscrire\_skieur") == 0)

      {

          inscrire\_skieur(&ins);

      }

      else if (strcmp(mot, "afficher\_skieurs") == 0)

      {

          afficher\_skieurs(&ins);

      }

      else if (strcmp(mot, "definir\_course") == 0)

      {

          definition\_course(&crs);

      }

      else if (strcmp(mot, "enregistrer\_temps") == 0)

      {

          enregistrer\_temps(&ins,&crs);

      }

      else if (strcmp(mot, "afficher\_temps") == 0)

      {

          afficher\_temps(&crs);

      }

      else if (strcmp(mot, "disqualification") == 0)

      {

          disqualification\_skieur(&crs);

      }

      else if (strcmp(mot, "afficher\_classement") == 0)

      {

          affichage\_classement(&ins, &crs);

      }

      else if (strcmp(mot, "exit") == 0)

      {

          exit(1); //la boucle infinie est rompue et le programme s'arrête si "exit               vaut 1

      }

   }

   system("pause");

   return 0;

}

//FONCTION PRINCIPALE fin

void inscrire\_skieur(Inscrits \*ins) {

   char mot[lgMot+1];

   int dossard = 101;

   assert(ins->nbInscrits < 50);

   scanf("%s", mot);

   strcpy(ins->liste[ins->nbInscrits].nom, mot);

   scanf("%s", mot);

   strcpy(ins->liste[ins->nbInscrits].pays, mot);

   ins->liste[ins->nbInscrits].dossard = 101 + ins->nbInscrits;

   ins->nbInscrits++; //incrémentation du nombre d'inscrits après chaque nouvelle inscription

   Skieur s;

   s.dossard = (100 + ins->nbInscrits); //numéro de dossard attribué en fonction du nombre d'inscrits

   printf("inscription dossard %d \n", s.dossard);

}

void afficher\_skieurs(const Inscrits \*ins) {

   for (int i = 0; i < ins->nbInscrits; ++i) //boucle for parcourant l'ensemble du tableau "liste" pour afficher les skieurs

      printf("%s %s %d \n", ins->liste[i].nom, ins->liste[i].pays, ins->liste[i].dossard);

}

void definition\_course(Course \*crs) {

   int portes;

   scanf("%d", &portes);

   assert(portes >= minPortes || portes <= maxPortes);

}

void enregistrer\_temps(Inscrits \*ins, Course \*crs) {

   float temps;

   unsigned int portes, dossard;

   Mesure mesure;

   scanf("%f", &temps);

   mesure.temps = temps;

   scanf("%u", &portes);

   mesure.portes = portes;

   scanf("%u", &dossard);

   mesure.dossard = dossard;

   crs->liste[crs->num\_mesure] = mesure; //stockage des composantes de la variable "mesure" dans le tableau "liste" à l'indice "num\_mesure"

   crs->num\_mesure++;

}

void afficher\_temps(Course \*crs) {

   unsigned int dossard;

   Mesure mesure;

   scanf("%u", &dossard);

   for (int i = 0; i < crs->num\_mesure; i++) {

      if (dossard == crs->liste[i].dossard) {

          if (crs->liste[i].temps == -1) { //si skieur détecté comme disqualifié, affichage disqualification à x portes, x dossard

              printf("%u %u disqualification  \n", crs->liste[i].portes, crs->liste[i].dossard);

          }

          else

          {

              printf("%u %u %.2f \n", crs->liste[i].portes, crs->liste[i].dossard, crs->liste[i].temps);

          }

      }

   }

}

void disqualification\_skieur(Course \*crs) {

   unsigned int portes;

   unsigned int dossards;

   Mesure mesure;

   scanf("%u", &portes);

   mesure.portes = portes;

   scanf("%u", &dossards);

   mesure.dossard = dossards;

   mesure.temps = -1.; //La valeur -1 définit un skieur comme étant disqualifié

   crs->liste[crs->num\_mesure] = mesure;

   crs->num\_mesure++;

}

void affichage\_classement(Inscrits \*ins, Course \*crs) {

   int i = 0, j, dossard, porte;

   int tDos[maxSkieurs], nb\_mesure = 0;

   double tTps[maxSkieurs], v = 0;

   scanf("%d", &porte);

   /\* On parcourt le tableau de mesure. Si la porte entrée par l'utilisateur correspond

à une porte enregistrée dans une mesure pour un skieur non disqualifié, on stocke son dossard dans  "dossard". On parcourt encore le tableau pour trouver la mesure du skieur à la première porte passée (0). On sort ensuite de la boucle, et on stocke le dossard du skieur dans tDos[] et son temps dans tTPs qui à une porte donnée correspond à la soustraction de ce temps et de son temps à la première porte passée. \*/

   while (i < crs->num\_mesure) {

      ++i;

      if ((crs->liste[i].portes == porte) && (crs->liste[i].temps != -1)) {

          dossard = crs->liste[i].dossard;

          for (j = 0; j < crs->num\_mesure; ++j) {

              if ((crs->liste[j].dossard == dossard) && (crs->liste[j].portes == 0))

                  break;

          }

          tDos[nb\_mesure] = dossard;

          tTps[nb\_mesure] = crs->liste[i].temps - crs->liste[j].temps;

          nb\_mesure++;

      }

   }

//Algorithme de tri donné dans le cours projet

   for (int n = 0; n < nb\_mesure; ++n) {

      v = tTps[n];

      j = n;

      int tmp = tDos[n];

      while ((j > 0) && (tTps[j - 1] > v))

      {

          tTps[j] = tTps[j - 1];

          tDos[j] = tDos[j - 1];

          j = j - 1;

      }

      tTps[j] = v;

      tDos[j] = tmp;

   }

   //affichage du classement

   for (int n = 0; n < nb\_mesure; ++n) {

      for (i = 0; i < ins->nbInscrits; ++i) {

          if (tDos[n] == ins->liste[i].dossard)

              printf("%d %d %s %s %.2f \n", porte, tDos[n], ins->liste[i].nom, ins->liste[i].pays, tTps[n]);

      }

   }

}

\*

**InSp4 OutSp4**

