

# Projet IAP

Poursuite par équipes en  
patinage de vitesse

---

**14 DECEMBRE**

---

VIDAL Tanguy  
LOPEZ Frédéric  
TRAORE Samba

**Groupe 110**



---

# Introduction Présentation du rôle fonctionnel de l'application

L'objectif de ce projet d'IAP est de développer une application permettant de gérer les équipes et les temps de patineurs lors d'une course de patinage de vitesse. L'application doit ainsi interpréter 8 commandes, « `definir_parcours` » permettra par exemple de définir le nombre de tours que comporte l'épreuve.

Le langage de programmation utilisé pour coder l'application est le langage C qui a été étudié durant toute cette première partie du semestre. De plus nous avons utilisé l'environnement de programmation de Visual Studio pour travailler.

Le cycle de développement suit un cycle de type agile fractionné en 5 sprints qui comportent chacun un ajout de fonctionnalité. Nous avons testé chaque sprint au moyen de jeux de donnée d'entrée (in) et de sortie (out).

---

## Table des matières

Introduction Présentation du rôle fonctionnel de l'application .....	2
I. Rôle Fonctionnel de chaque Sprint et entrés/sorties de l'application .....	4
a. Sprint 1 .....	4
b. Sprint 2 .....	5
c. Sprint 3 .....	6
d. Sprint 4 .....	7
e. Sprint 5 .....	8
II. Organisation des tests .....	9
III. Bilan de Validation des Tests de Développement .....	10
IV. Bilan de Projet.....	11
Annexes .....	12
Entrées et Sorties des jeux de test de tous les sprints.....	12
Programme Complet (sprint atteint : 4) .....	16

# I. Rôle Fonctionnel de chaque Sprint et entrées/sorties de l'application

## a. Sprint 1

Le sprint 1 vise l'inscription des équipes ainsi que leur affichage

### i. Entrées

Pour la première phase du projet, le sprint 1, nous devons déclarer trois commandes :

- o `exit`
- o `inscrire_equipe`
- o `afficher_equipes`

Nom de la commande	Rôle de la commande	Fonction appelée par la commande
<code>exit</code>	Arrêter le Programme	Aucune, code directement dans la fonction <code>main()</code> .
<code>inscrire_equipe</code>	Inscrire une équipe, ses patineurs et leur affecter un dossard	<code>void inscription_equipe</code>
<code>afficher_equipes</code>	Afficher toutes les équipes de l'épreuve	<code>void afficher_equipes</code>

### ii. Sorties

Si on utilise les commandes définit dans cette première partie du programme nous aurons les retours suivants :

Nom de la commande	Sortie associée
<code>exit</code>	Aucune sortie attendue
<code>inscrire_equipe</code>	« inscription dossard ### », renvoie le dossard de chaque patineur inscrit avec un numéro allant de 101 au nombre de patineur total inscrits.
<code>afficher_equipes</code>	« Canada Blondin 101 Weidemann 102 Morrison 103 », Renvoie les données rentrées pour chaque équipe : nom de l'équipe et le nom de chaque patineur suivi de son numéro de dossard associé.

## b. Sprint 2

Le sprint 2 ajoute à l'application la possibilité d'enregistrer des temps pour chaque patineur et chaque tour et d'afficher tous les temps associés à un dossard.

### i. Entrées

Pour le sprint 2 nous devons déclarer deux commandes :

- o enregistrer\_temps <dossard> <N° du tour> <temps>
- o afficher\_temps <dossard>

<i>Nom de la commande</i>	<i>Rôle de la commande</i>	<i>Fonction appelée par la commande</i>
<i>enregistrer_temps</i>	L'utilisateur enregistre un temps pour un dossard donné et pour un N° de tour donné	Void enregistrement_temps
<i>afficher_temps</i>	Affiche le temps d'un patineur en fonction du dossard demandé	void affichage_temps

### ii. Sorties

Les retours attendus pour l'appel de ces deux fonctions sont les suivants :

<i>Nom de la commande</i>	<i>Sortie associée</i>
<i>enregistrer_temps</i>	Aucun retour attendu
<i>afficher_temps</i>	Renvoie l'équipe le nom du joueur ainsi que le/les n° du tour et son/ses temps associés

### c. Sprint 3

Pour ce sprint nous avons codé une fonction permettant d'afficher le dernier temps de chaque équipe. Concrètement cette fonction permet à l'utilisateur de connaître le pire temps d'une équipe soit finalement de comparer le temps des différentes équipes. Cette fonction est essentielle pour la phase suivante de programmation. La commande initialisée dans cette phase est la suivante :

« `afficher_temps-equipes` »

#### i. Entrées

Nom de la commande	Rôle de la commande
<code>afficher_temps-equipes</code>	Permet d'afficher le temps du dernier patineur de l'équipe à avoir fini tous les tours de l'épreuve

#### ii. Sorties

Les retours attendus pour l'appel de ces deux fonctions sont les suivants :

Nom de la commande	Sortie associée
<code>afficher_temps-equipes</code>	Renvoie le nom de chaque équipe suivie du dernier temps enregistré pour ses patineurs exemple : « Canada 54.1 Japon 53.9 »

## d. Sprint 4

Dans cette Partie nous devons ajouter une commande (« `definir_parcours` ») permettant de définir le nombre de tours de chaque épreuve. Une épreuve n'est terminée que lorsque tous les patineurs ont réalisé tous les tours demandés (Le nombre de tours est au minimum de et peut aller jusqu'à 10).

De plus on ajoute à l'application une fonctionnalité de détection de fin d'une épreuve. Cette fonction va permettre de détecter lorsque tous les patineurs ont enregistré un temps pour tous les tours demandés afin de terminer cette épreuve et afficher le temps des équipes de l'épreuve à l'aide de la fonction « `afficher_temps_equipes` »

### i. Entrées

Nom de la commande	Rôle de la commande
<b><code>definir_parcours</code></b>	Initialise le nombre de tour que doit comporter le parcours. Cette commande doit être suivie du nombre de tour choisi (entre 2 et 10).

### ii. Sorties

Les retours attendus pour l'appel de ces deux fonctions sont les suivants :

Nom de la commande	Sortie associée
<b><code>definir_parcours</code></b>	Aucun retour attendu

---

e. Sprint 5



---

## II. Organisation des tests

---

### III. Bilan de Validation des Tests de Développement

---

## IV. Bilan de Projet

# Annexes

## Entrées et Sorties des jeux de test de tous les sprints

### InSprint1.txt

```
inscrire_equipe Canada Blondin Weidemann Morrison  
inscrire_equipe Japon Takagi Sato Takagu  
afficher_equipes  
exit
```

### OutSprint1.txt

```
inscription dossard 101  
inscription dossard 102  
inscription dossard 103  
inscription dossard 104  
inscription dossard 105  
inscription dossard 106  
Canada Blondin 101 Weidemann 102 Morrison 103  
Japon Takagi 104 Sato 105 Takagu 106
```

### InSprint2.txt

```
inscrire_equipe Canada Blondin Weidemann Morrison  
inscrire_equipe Japon Takagi Sato Takagu  
enregistrer_temps 101 1 53.1  
enregistrer_temps 102 1 53.2  
enregistrer_temps 104 1 53.3  
enregistrer_temps 105 1 53.7  
enregistrer_temps 106 1 53.9  
enregistrer_temps 103 1 54.1  
enregistrer_temps 105 2 100.6  
afficher_temps 102  
afficher_temps 105  
exit
```

### OutSprint2.txt

```
inscription dossard 101  
inscription dossard 102  
inscription dossard 103  
inscription dossard 104  
inscription dossard 105  
inscription dossard 106  
Canada 1 Weidemann 53.2  
Japon 1 Sato 53.7  
Japon 2 Sato 100.6
```

### InSprint3.txt

```
inscrire_equipe Canada Blondin Weidemann Morrison  
inscrire_equipe Japon Takagi Sato Takagu  
enregistrer_temps 101 1 53.1  
enregistrer_temps 102 1 53.2  
enregistrer_temps 104 1 53.3
```

```
enregistrer_temps 105 1 53.7
enregistrer_temps 106 1 53.9
enregistrer_temps 103 1 54.1
enregistrer_temps 105 2 100.6
afficher_temps_equipes 1
exit
```

#### OutSprint3.txt

```
inscription dossard 101
inscription dossard 102
inscription dossard 103
inscription dossard 104
inscription dossard 105
inscription dossard 106
Canada 54.1
Japon 53.9
```

#### InSprint4.txt

```
definir_parcours 2
inscrire_equipe Canada Blondin Weidemann Morrison
inscrire_equipe Japon Takagi Sato Takagu
inscrire_equipe DEUTCHLAND xyszi ipshg uwfln
inscrire_equipe NEW_ZEALAND xksfq xeuus fpikr
enregistrer_temps 101 1 53.1
enregistrer_temps 102 1 53.2
enregistrer_temps 104 1 53.3
enregistrer_temps 105 1 53.7
enregistrer_temps 106 1 53.9
enregistrer_temps 103 1 54.1
enregistrer_temps 105 2 100.6
enregistrer_temps 106 2 101.7
enregistrer_temps 104 2 102.3
enregistrer_temps 101 2 102.5
enregistrer_temps 103 2 102.8
enregistrer_temps 102 2 103.1
enregistrer_temps 109 1 65.04
enregistrer_temps 107 1 65.15
enregistrer_temps 112 1 68.00
enregistrer_temps 111 1 79.06
enregistrer_temps 108 1 81.54
enregistrer_temps 110 1 93.16
enregistrer_temps 112 2 104.54
enregistrer_temps 108 2 113.38
enregistrer_temps 111 2 120.23
enregistrer_temps 109 2 126.15
enregistrer_temps 107 2 148.39
enregistrer_temps 110 2 148.80
exit
```

#### OutSprint4.txt

```
inscription dossard 101
inscription dossard 102
```

```
inscription dossard 103
inscription dossard 104
inscription dossard 105
inscription dossard 106
detection_fin_poursuite
Japon 102.3
Canada 103.1
```

#### InSprint5.txt

```
definir_parcours 2
definir_nombre_epreuves 2
inscrire_equipe Canada Blondin Weidemann Morrison
inscrire_equipe Japon Takagi Sato Takagu
inscrire_equipe France Pierron Huot Monvoisin
inscrire_equipe Italie Lollobrigida Mascitto Valcepina
enregistrer_temps 101 1 53.
enregistrer_temps 102 1 53.2
enregistrer_temps 104 1 53.3
enregistrer_temps 105 1 53.7
enregistrer_temps 106 1 53.9
enregistrer_temps 103 1 54.1
enregistrer_temps 105 2 100.6
enregistrer_temps 106 2 101.7
enregistrer_temps 104 2 102.3
enregistrer_temps 101 2 102.5
enregistrer_temps 103 2 102.8
enregistrer_temps 102 2 103.1
enregistrer_temps 111 1 50.9
enregistrer_temps 108 1 52.1
enregistrer_temps 112 1 53.2
enregistrer_temps 107 1 53.5
enregistrer_temps 109 1 53.8
enregistrer_temps 110 1 54.1
enregistrer_temps 110 2 99.1
enregistrer_temps 109 2 100.3
enregistrer_temps 107 2 101.5
enregistrer_temps 112 2 101.8
enregistrer_temps 108 2 102.1
enregistrer_temps 111 2 102.6
exit
```

#### OutSprint5.txt

```
inscription dossard 101
inscription dossard 102
inscription dossard 103
inscription dossard 104
inscription dossard 105
inscription dossard 106
inscription dossard 107
inscription dossard 108
inscription dossard 109
inscription dossard 110
```

---

```
inscription dossard 111
inscription dossard 112
detection_fin_poursuite
Japon 102.3
Canada 103.1
detection_fin_poursuite
France 102.1
Italie 102.6
detection_fin_competition
France 102.1
Japon 102.3
Italie 102.6
Canada 103.1
```

## Programme Complet (sprint atteint : 4)

```
L1  /*
L2  sprint4.c
L3  LOPEZ Frédéric | TRAORE Samba | VIDAL Tanguy
L4  Groupe 110
L5  31/10/2019
L6  */
L7
L8
L9  #include<stdio.h>
L10 #include<stdlib.h>
L11 #include <string.h>
L12 #pragma warning (disable:4996)//Désactiver Les avertissements sur la sécurité du scanf
L13 #pragma warning (disable:6031)// Désactive Les avertissement quant au retour du scanf
L14
L15
L16 enum { LgMot = 30 };// Nolbre maximal d'une chaîne de caractère pour tout Le programme
L17 enum { MaxTours = 10 }; // Nombre maximum de tours
L18 enum { maxEpreuves = 16 }; // Nombre maximal d'épreuves
L19 enum { Nb_Equipes = 2 }; // Nombre d'équipes par épreuve
L20 enum { Nb_Patineur = 3 };//Nombre de Patineurs par équipe
L21
L22 /*-----*/
L23
L24 typedef struct {
L25     double temps;
L26     unsigned int noTour;
L27     unsigned int dossard;
L28 }Mesure;
L29
L30 typedef struct {
L31     char nom[LgMot + 1];
L32     unsigned int dossard;
L33 } Patineur;
L34
L35 typedef struct {
L36     char pays[LgMot + 1];//Nationalité de L'équipe
L37     Patineur data[Nb_Patineur];//Nombre de patineurs
L38 } Equipe;
L39
L40
L41 typedef struct {
L42     //Nombre maximum de temps à enregistrer
L43     Mesure data[Nb_Equipes * maxEpreuves * Nb_Patineur * MaxTours];
L44     unsigned int nbpatineur;
L45 }Epreuve;
L46
L47 /*-----*/
L48 void inscription_equipe(Equipe equ[], int* cmp_equ, int* cmp_dossard);
L49 void afficher_equipes(const Equipe equ[], int* cmp_equ);
L50 void affichage_temps(Epreuve* epr, Equipe* equ);
L51 void afficher_temps_equipes(const Epreuve* epr, const Equipe* equ, const int noTour);
```



```

L52 void detection_fin_poursuite(Epreuve* epr, Equipe* equ, int nbTours, int noTour);
L53 void enregistrement_temps(Epreuve* epr, int nbTours, Equipe* equ, int noTour);
L54
L55 /*-----*/
L56
L57
L58
L59 int main() {
L60
L61
L62     Equipe equ[Nb_Equipes * maxEpreuves]; //Déclare la Liste d'équipes
L63     int cmp_equ = 0; //Sert à compter le nombre d'équipes renseigné
L64     int cmp_dossard = 1; //Sert à compter le nombre de temps renseigné
L65     Epreuve epr; //Déclare la Liste d'épreuves
L66     epr.nbpatineur = 0; //Initialise le compteur de patineurs
L67     int nbTours = 2; //Initialise le nombre de tours à 2 par défaut
L68     char mot[LgMot + 1]; //mot de commande de l'utilisateur
L69     int noTour = 0;
L70
L71
L72
L73
L74     do { //lit et compare la commande de l'utilisateur
L75
L76         scanf("%s", mot); // Lecture de la commande (mot)
L77
L78         if (strcmp(mot, "definir_parcours") == 0) {
L79
L80             scanf("%d", &nbTours);
L81             noTour = nbTours;
L82         }
L83
L84         if (strcmp(mot, "inscrire_equipe") == 0) {
L85             // si la commande est "inscrire_equipe"
L86             inscription_equipe(&equ[Nb_Patineur], &cmp_equ, &cmp_dossard);
L87
L88             if (strcmp(mot, "afficher_equipes") == 0) {
L89                 // si la commande est "afficher_equipes"
L90                 afficher_equipes(&equ[Nb_Patineur], &cmp_equ);
L91             }
L92
L93             if (strcmp(mot, "enregistrer_temps") == 0) {
L94                 enregistrement_temps(&epr, nbTours, &equ[Nb_Patineur], noTour);
L95             }
L96
L97             if (strcmp(mot, "afficher_temps") == 0) {
L98                 affichage_temps(&epr, &equ[Nb_Patineur]);
L99             }
L100
L101             if (strcmp(mot, "afficher_temps_equipes") == 0) {
L102                 scanf("%d", &noTour);
L103                 afficher_temps_equipes(&epr, &equ[Nb_Patineur], noTour);
L104             }

```

```

L105
L106
L107
L108         if (strcmp(mot, "exit") == 0) {
L109             exit(0); // sortie du programme principal
L110         }
L111
L112
L113     } while (1);
L114     system("pause"); return 0;
L115 }
L116
L117 /*-----*/
L118
L119
L120 /*inscription des équipes de patineurs
L121 [in] un pointeur sur une structure Equipe
L122 [in]le compteur d'équipes
L123 [in]le compteur de dossards
L124 [out] un tableau de types Patineur avec les informations sur les patineurs
L125 préconditions : Le nombre maximal d'entrée est 32
L126 Le nombre maximum pour chaque nom de pays ou de patineur est 30
L127 */
L128 void inscription_equipe(Equipe equ[], int* cmp_equ, int* cmp_dossard) {
L129
L130     char pays[LgMot + 1];
L131     scanf("%s", pays);
L132     strcpy(equ[*cmp_equ].pays, pays);
L133     for (int j = 0; j < Nb_Patineur; j++) {
L134         scanf("%s", equ[*cmp_equ].data[j].nom);
L135         equ[*cmp_equ].data[j].dossard = 100 + *cmp_dossard;
L136         printf("inscription dossard %d\n", equ[*cmp_equ].data[j].dossard);
L137         *cmp_dossard += 1;
L138     }
L139     *cmp_equ += 1;
L140
L141
L142 }
L143
L144
L145 /*Afficher les équipes inscrites et leurs patineurs
L146 [in] un pointeur sur une structure de type Equipe
L147 [in]le compteur d'équipes
L148 [out] affichage du tableau d'équipe
L149 */
L150 void afficher_equipes(const Equipe equ[], int* cmp_equ) {
L151
L152
L153     for (int i = 0; i < *cmp_equ; i++) {
L154         printf("%s ", equ[i].pays);
L155         for (int j = 0; j < Nb_Patineur; j++) {
L156             printf("%s ", equ[i].data[j].nom);
L157             printf("%d ", equ[i].data[j].dossard);
L158         }

```

```

L159         printf("\n\n");//
L160     }
L161
L162 }
L163
L164 /*affichage du temps
L165 [in] un pointeur sur une structure Epreuve
L166 [in] un pointeur sur une structure Equipe
L167 [out] affichage des temps d'un joueur au dossard donné (affiche autant
L168 de temps qu'il a fait de tours
L169 */
L170 void affichage_temps(Epreuve* epr, Equipe* equ) {
L171     unsigned int dossard;
L172     scanf("%ud", &dossard);
L173     for (unsigned int i = 0; i < epr->nbpateur; ++i) {
L174         //Chercher Le temps associé au dossard renseigné
L175         if (epr->data[i].dossard == dossard) {
L176             //Si Le dossard entré correspond, afficher son chrono
L177             for (unsigned int j = 0; j < Nb_Equipes; ++j) {
L178                 //Parcourir Les équipes à La recherche du dossard
L179                 for (unsigned int k = 0; k < Nb_Equipes * Nb_Pateur; ++k)
L180                 {//regarder dans L'équipe si Le dossard renseigné s'y trouve
L181                     if (equ[j].data[k].dossard == dossard) {
L182                         // si Le dossard se trouve dans L'équipe
L183                         printf("%s ", equ[j].pays);
L184                         //afficher son pays
L185                         printf("%d ", epr->data[i].noTour);
L186                         //Afficher Le numéro du tour
L187                         printf("%s ", equ[j].data[k].nom);
L188                         //afficher son nom
L189                         printf("%.11f\n", epr->data[i].temps);
L190                         //Afficher son temps
L191                     }
L192                 }
L193             }
L194         }
L195     }
L196 }
L197
L198 /*affichage du temps des équipes selon leurs classement
L199 [in] un pointeur sur une structure Epreuve
L200 [in] un pointeur sur la structure Equipe
L201 [in]Le nombre de tour d'une épreuve
L202 [out] Affichage des équipes et du chrono réalisé par
L203 chacune dans l'ordre du classement
L204 */
L205 void afficher_temps_equipes(const Epreuve* epr, const Equipe* equ, const int noTour) {
L206     int a;
L207     int cmp_tmp;
L208     double tab_temps[16];
L209     char tab_pays[Nb_Equipes * maxEpreuves][LgMot + 1];
L210
L211     static int z = 1;
L212     double max_temps = 0;

```

```

L207     double min_temps = 0;
L208     double tmp = 0;
L209     int j = 0;
L210
L211
L212
L213     for (int i = 1; i <= Nb_Equipes; ++i) {//Parcourir Les équipes
L214         a = 0;
L215         cmp_tmp = 0;
L216
L217         for (unsigned int k = 0; epr->nbpatineur > k; ++k) {
L218             if (epr->data[k].dossard <= 100 + z * Nb_Patineur && 100 + Nb_Patineur
L219 * (z - 1) < epr->data[k].dossard && (epr->data[k].noTour == noTour)) {
L219                 cmp_tmp += 1;
L220                 if (cmp_tmp == Nb_Patineur) {
L221                     tab_temps[i - 1] = epr->data[k].temps;
L222                     strcpy(tab_pays[i - 1], equ[z - 1].pays);
L223                     a = 1;
L224                 }
L225             }
L226         }
L227
L228
L229         if (a != 1) {
L230             printf("%s ", equ[i - 1].pays);
L231             printf("indisponible\n");
L232             tab_temps[i - 1] = 0;
L233
L234         }
L235         z += 1;
L236     }
L237
L238     if (tab_temps[0] > tab_temps[1]) {
L239         printf("%s ", tab_pays[1]);
L240         printf("%.11f\n", tab_temps[1]);
L241         printf("%s ", tab_pays[0]);
L242         printf("%.11f\n", tab_temps[0]);
L243     }
L244     if (tab_temps[1] > tab_temps[0]) {
L245         printf("%s ", tab_pays[0]);
L246         printf("%.11f\n", tab_temps[0]);
L247         printf("%s ", tab_pays[1]);
L248         printf("%.11f\n", tab_temps[1]);
L249     }
L250
L251 }
L252
L253
L254 /*Déetecte la fin d'une épreuve (lorsque tous les patineurs des équipes ont
L255 enregistrés le nombre de tours requi)
L256 [in] un pointeur sur une structure Epreuve
L257 [in] un pointeur sur la structure Equipe
L258 [in]Le nombre de tour d'une épreuve
L259 [out]Si l'épreuve est finit lance la fonction afficher_temps_equipes
L260 [out]Sinon aucun retour

```

```

L261  */
L262  void detection_fin_poursuite(Epreuve* epr, Equipe* equ, int nbTours, int noTour) {
L263
L264      static int i = 0;
L265      i += 1;
L266      if (i == (Nb_Patineur * Nb_Equipes * nbTours)) {
L267          printf("detection_fin_poursuite\n");
L268          afficher_temps_equipes(epr, &equ[0], noTour);
L269          i = 0;
L270      }
L271
L272  }
L273
L274
L275  /*enregistrement du temps
L276  [in] un pointeur sur une structure Epreuve
L277  [out] ajout du temps d'un joueur pour un tour au tableau data
L278  */
L279  void enregistrement_temps(Epreuve* epr, int nbTours, Equipe* equ, int noTour) {
L280      scanf("%ud", &epr->data[epr->nbpatineur].dossard);
L281      scanf("%ud", &epr->data[epr->nbpatineur].noTour);
L282      scanf("%lf", &epr->data[epr->nbpatineur].temps);
L283      epr->nbpatineur += 1;
L284      detection_fin_poursuite(epr, &equ[0], nbTours, noTour);
L285
L286  }

```