

Poursuite par équipes en patinage de vite



Ce projet consiste à nous apprendre les fondamentaux de la programmation du langage C. Il se prépare par groupe de deux, le but est de réaliser les différents sprints, et tester et valider le plus haut sprint qu'on a pu atteindre. Le problème posé est de programmer un interpréteur de commandes pour la gestion d'une compétition de poursuite par équipe.

TABLE DES MATIERES

Présentation de l'application	2
• Programmer un interpréteur de commande	2
• Le plus haut niveau atteint.....	3
Organisation de tests	4
Bilan de validation	5
Bilan de projet.....	6
Annexe	7

PRESENTATION DE L'APPLICATION

- Programmer un interpréteur de commande

Le but de ce projet est de programmer un interpréteur de commandes, plus précisément les neuf commandes représentent sous forme de chaînes de caractères avec une taille maximale de 30 et entrer en utilisant l'entrée standard ou par redirection d'un fichier texte sur l'entrée standard. Ces neuf commandes se repartent en cinq sprints différents. Ces cinq sprints représentant cinq incréments de fonctionnalité de l'application.

Le sprint 1 consiste à inscrire les trois patineurs d'une équipe en mettant le pays de l'équipe et les noms de trois membres afin de distribuer automatiquement les dossards à chacun d'entre eux et les afficher sur l'écran. Dans ce sprint on devrait aussi définir une commande (« exit ») qui permet de sortir de l'interpréteur.

Dans le sprint 2, on devrait définir les commandes qui consistent à mémoriser les temps de la course et afficher tous les temps de chronométrage enregistrés pour un patineur repéré par son dossard.

Le sprint 3 permet d'afficher les temps de chaque équipe inscrite d'un tour précis. Le temps d'équipe est le temps du dernier patineur de chaque équipe.

— exit	(1. Sprint#1)	Le sprint 4 consiste à définir le nombre de tours d'un parcours, puis détecter la fin d'une poursuite en vérifiant si tous les patineurs ont accompli le nombre de tours défini, et afficher le classement des équipes.
— definir_parcours	(2. Sprint#4)	
— definir_nombre_épreuves	(3. Sprint#5)	
— inscrire_equipe	(4. Sprint#1)	
— afficher_equipes	(5. Sprint#1)	
— enregistrer_temps	(6. Sprint#2)	
— afficher_temps	(7. Sprint#2)	
— afficher_temps_equipes	(8. Sprint#3)	
— detection_fin_poursuite	(9. Sprint#4)	
— detection_fin_competition	(10. Sprint#5)	

Le sprint 5 consiste à définir le nombre d'épreuves et vérifier que toutes les épreuves de poursuite sont terminées. Si la fin de compétition est détectée, le classement final de tous les équipes sera affiché grâce à un algorithme de tri par sélection.

- Le plus haut niveau atteint

Le sprint le plus haut niveau qu'on a validé la semaine 21 octobre est le sprint 5. Ce sprint est composé de tous les autres sprints (sprint 1, sprint 2, sprint 3, sprint 4 et sprints 5). Nous avons réussi à finir notre programme. C'est donc un programme complet.

Voici l'entrée et sortir de sprint 5 :

Sprint #5 – inSp5.txt	Sprint #5 – outSp5.txt
inSp5 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 53.1 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	outSp5 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

Voici le résultat de nos programmes :

64 Console de débogage Microsoft Visual Studio definir_parcours 2 definir_nombre_epreuves 2 inscrire_equipe Canada Blondin Weidemann Morrison inscription_dossard 101 inscription_dossard 102 inscription_dossard 103 inscrire_equipe Japon Takagi Sato Takagu inscription_dossard 104 inscription_dossard 105 inscription_dossard 106 inscrire_equipe France Pierron Huot Monvoisin inscription_dossard 107 inscription_dossard 108 inscription_dossard 109 inscrire_equipe Italie Lollobrigida Mascitto Valcepina inscription_dossard 110 inscription_dossard 111 inscription_dossard 112 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 detection_fin_poursuite Japon 102.3 Canada 103.1 enregistrer_temps 111 1 50.9	Canada 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 detection_fin_poursuite France 102.1 Italie 102.6 detection_fin_competition France 102.1 Japon 102.3 Italie 102.6 Canada 103.1 Sortie de C:\Users\33652\Desktop\Compe Pour fermer automatiquement la console
--	--

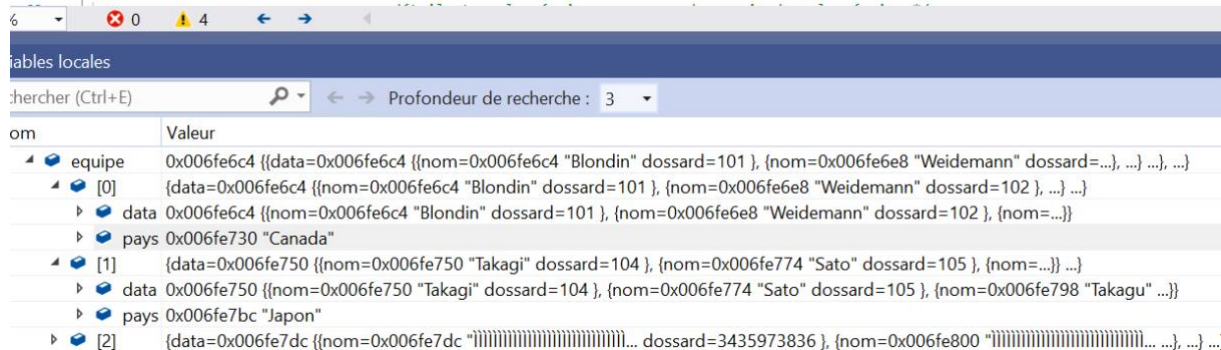
ORGANISATION DE TESTS

En effet pour créer une commande nous avons fait beaucoup d'essai, notamment des tests. Pour chaque sprint une spécification est donnée, qui nous permet de savoir le but de la commande et il nous indique, quelles sont les fonctionnalités que l'on doit associer dans les commandes de ce sprint.

En plus avec les JDT (les Jeux de Données de Test) que les professeurs nous ont donnés, on a pu comparer nos résultats et les résultats qu'on devrait obtenir afin de comprendre nos erreurs et de les corriger, car avec les mêmes entrées de l'application (fichiers inSp#n.txt), si les résultats d'exécutions de notre programme ne coïncident pas aux résultats références, c'est-à-dire les fichiers outSp#n.txt, ce qui signifie que notre programme ne fonctionne pas correctement donc il faut que nous trouvions nos erreurs et de les corriger.

Pour trouver les problèmes, on utilise souvent la fonction déboguée, pour voir quelles sont les fautes qui provoquent le dysfonctionnement de notre programme. Par exemple, lorsque le sprint 1 ne marche pas, on utilise la fonction déboguée pour voir où sont les problèmes.

```
55 void inscrire_equipe(Equipes* e) {
56
57     static unsigned int compt1 = 101;
58     scanf("%s", e->equipe[e->ins].pays);
59     for (int j = 0; j < NbPatineur; ++j) {
60         scanf("%s", e->equipe[e->ins].data[j].nom);
61         e->equipe[e->ins].data[j].dossard = compt1;
62         printf("inscription dossard %d\n", compt1);
63         ++compt1;
64     }
65     e->ins = e->ins + 1;
66 }
67
68 void afficher_equipes(const Equipes* e, int* DernierDossard) { /* cette fonction permet d'afficher
```



nom	Valeur
equipe	0x006fe6c4 {{data=0x006fe6c4 {{nom=0x006fe6c4 "Blondin" dossard=101 }, {nom=0x006fe6e8 "Weidemann" dossard=..., ...} ...}, ...}
[0]	{data=0x006fe6c4 {{nom=0x006fe6c4 "Blondin" dossard=101 }, {nom=0x006fe6e8 "Weidemann" dossard=102 }, ...} ...}
data	0x006fe6c4 {{nom=0x006fe6c4 "Blondin" dossard=101 }, {nom=0x006fe6e8 "Weidemann" dossard=102 }, {nom=...}}
pays	0x006fe730 "Canada"
[1]	{data=0x006fe750 {{nom=0x006fe750 "Takagi" dossard=104 }, {nom=0x006fe774 "Sato" dossard=105 }, {nom=...} ...}
data	0x006fe750 {{nom=0x006fe750 "Takagi" dossard=104 }, {nom=0x006fe774 "Sato" dossard=105 }, {nom=0x006fe798 "Takagu" ...}}
pays	0x006fe7bc "Japon"
[2]	{data=0x006fe7dc {{nom=0x006fe7dc "..." dossard=3435973836 }, {nom=0x006fe800 "..." dossard=..., ...} ...}

Après l'exécution de la fonction 'inscrire_equipe', on peut en déduire que tout est cohérent. Les problèmes sont probablement dans la fonction 'afficher_equipes'.


```

68 void afficher_equipements(const Equipes* e, int* DernierDossard) { /* cette fonction permet d'afficher
69 en détails tous les équipes en cours de courir dans la mémoire */
70 int EquipeEnCours = 100;
71 EquipeEnCours = *DernierDossard - EquipeEnCours;
72 EquipeEnCours = EquipeEnCours / 3;
73
74 for (int i = 0; i < 2; ++i) {
75     printf("%s ", e->equipe[i + EquipeEnCours].pays);
76     for (int j = 0; j < NbPatineur; ++j) {
77         if (j != 0)
78             printf(" ");
79         printf("%s %d", e->equipe[EquipeEnCours + i].data[j].nom, e->equipe[EquipeEnCours + i].data[j].dossard);
80     }
81     printf("\n");
82 }
83 }
84 }
85 }
86

```

Variables locales

nom	Valeur
DernierDossard	0x00afac4 {0}
e	0x00afea58 {equipe=0x00afea58 {(data=0x00afea58 {(nom=0x00afea58 "Blondin" dossard=101 }, {nom=0x00afea7c "Weidemann" ...}, ...), ...} ...}
EquipeEnCours	-33
i	0
j	1

Après l'exécution de la fonction 'afficher_equipements', on peut dire que toute la fonction va afficher le -33^{ème} équipe, ce n'est pas celle qu'on veut être affiché. On devrait donc corriger la partie orange.

Parfois on ajoute aussi le 'printf' pour voir directement sur l'écran ce qui passe dans un programme.

```

87 void enregistrer_temps(Course* crs, int* DernierDossard) {
88     /*cette fonction permet d'enregistrer le temps
89 de la patineur quand on donne son dossard et la tour*/
90     char mot[lgMot + 1];
91     scanf("%s", mot);
92     crs->data[crs->eng].dossard = atof(mot);
93     *DernierDossard = atof(mot);
94     scanf("%s", mot);
95     crs->data[crs->eng].nbtour = atof(mot);
96     scanf("%s", mot);
97     crs->data[crs->eng].temps = atof(mot);
98     crs->eng = crs->eng + 1;
99     printf("%d", crs->eng);
100 }

```

Par exemple, dans l'image ci-contre, on ajoute le 'printf' pour voir si les numéros de dossard, les nombres de tours et les temps utilisés des patineurs ont bien été enregistrer dans les différents tableaux.

BILAN DE VALIDATION

Le jour de la recette, le prof va utiliser un autre jeu de texte donné plus élaboré qui nous permet d'avancer dans le programme

Avec le nouvel ln on réalise la redirection :

```

C:\Windows\System32\cmd.exe
Microsoft Windows [version 10.0.18362.418]
(c) 2019 Microsoft Corporation. Tous droits réservés.

C:\Users\33652\Desktop\Competition\SPRINT 5 ESSAIE\SPRITN 5\Debug>"C:\Users\33652\Desktop\Competition\SPRINT 5 ESSAIE\SPRITN 5\Debug\SPRITN 5.exe"<"C:\Users\33652\Desktop\Competition\SPRINT 5 ESSAIE\SPRITN 5\Debug\inG01Sp5.txt">out5.txt

C:\Users\33652\Desktop\Competition\SPRINT 5 ESSAIE\SPRITN 5\Debug>fc "C:\Users\33652\Desktop\Competition\SPRINT 5 ESSAIE\SPRITN 5\Debug\outG01Sp5.txt"
FC : nombre insuffisant de spécifications de fichiers

C:\Users\33652\Desktop\Competition\SPRINT 5 ESSAIE\SPRITN 5\Debug>

```

Puis avec un logiciel de comparaison (cmd, diffchecker) on a pu comparer le résultat obtenu et celui que le prof nous avons donné, s'il y a des différences, on doit le corriger avec un temps limité. Si malheureusement nous n'avons pas réussi, le sprint précédent sera testé ainsi de suite.

Lors de la journée de la recette, on a essayé de passer le sprint 5, avec le Jeu de données de test que le prof nous avons donné.

Voici le résultat après la comparaison :

47	inscription dossard 147	47	inscription dossard 147
48	inscription dossard 148	48	inscription dossard 148
49	detection_fin poursuite	49	detection_fin poursuit
50	Russie 442.0	50	Russie 442.0
51	Japon 450.3	51	Japon 450.3
52	detection_fin poursuite	52	detection_fin poursuit
53	Canada 447.8	53	Canada 447.8
54	Pologne 453.6	54	Pologne 453.6
55	detection_fin poursuite	55	detection_fin poursuit
56	Hollande 455.7	56	Hollande 455.7
57	Coree 459.8	57	Coree 459.8
58	detection_fin poursuite	58	detection_fin poursuit
59	Norvege 456.7	59	Norvege 456.7
60	Italie 458.3	60	Italie 458.3
61	detection_fin poursuite	61	detection_fin poursuit
62	Australie 435.4	62	Australie 435.4
63	Chine 439.9	63	Chine 439.9
64	France Pierron 131 Huot 132 Monvoisin 133	64	France Pierron 131 Huot 132 Monvoisin 133
65	Hongrie Jaszapati 134 Bacsakai 135 Keszler 136	65	Hongrie Jaszapati 134 Bacsakai 135 Keszler 136
66	detection_fin poursuite	66	detection_fin poursuit
67	France 448.7	67	France 448.7
68	Hongrie 450.1	68	Hongrie 450.1
69	detection_fin poursuite	69	detection_fin poursuit
70	Finlande 444.3	70	Finlande 444.3
71	Autriche 450.5	71	Autriche 450.5
72	Allemagne Wolf 143 Volker 144 Rothenburger 145	72	Allemagne Wolf 143 Volker 144 Rothenburger 145
73	USA Henning 146 Young 147 Blair 148	73	USA Henning 146 Young 147 Blair 148
74	detection_fin poursuite	74	detection_fin poursuit
75	Allemagne 454.3	75	Allemagne 454.3
76	USA 455.9	76	USA 455.9

Il y a des fautes d'orthographe et certains espaces. Après la correction des fautes d'orthographe et la suppression, on peut apercevoir que tout est cohérent.

BILAN DE PROJET

Ce projet est une tâche assez compliquée pour nous, en effet, nous sommes deux à ce projet qui vient de la filière S et de la filière STI2D. L'un de nous deux sait déjà coder avant de venir en IUT et l'autre pas du tout, mais aucun de nous deux savent coder en langage C. Dans tous les cas on est très motivé pour réaliser ce projet qui est le premier projet de la programmation qui nous permet à rentrer dans le domaine informatique.

Pour nous le sprint 1 est un des plus compliqué pour réaliser, car au moment qu'on a essayé de faire ce sprint on n'avait pas encore compris les cours et ce qu'on doit faire. En plus, on est tous les deux qui ont des problèmes sur la langue française, cela fait qu'on a des difficultés pour comprendre les consignes et communiquer entre nous. On a posé les questions à toutes les personnes qui puissent nous aide pour réaliser ce premier sprint, comme les profs, les deuxièmes années et les camarades. Petit à petit nous avons compris comment fallait faire et comment on puisse avance notre programme. Mais chaque sprint reste très compliqué pour nous, nous avons utilisé au moins deux jours pour chaque sprint. Nous avons énervé entre nous, lorsqu'on n'arrive pas de résoudre les problèmes de notre programme, mais au moment qu'on réussit un sprint nous étions très fières de nous. Notamment quand on finit notre programme.

Nous avons revu tous les TD et TP, pour bien comprendre les cours et des utilisons dans notre programme, comme ce projet est juste avant le DST, cela nous permet de réviser en même temps le DST. De plus, ce projet est réalisé en groupe, ce qui permet de nous habituer de travailler en groupe et améliorer nos capacités de communication entre nous.

Bien que nous ayons réussi à atteindre le sprint 5, mais comme on le corrige en dernier moment. Il est certain que notre programme va posséder le problème sur certaine détaille, comme la conversation de 'double' en 'int' fait qu'il perde certaines données, mais on n'avait pas assez de temps pour le corriger, et on pense qu'on peut améliorer notre programme si on le refaire.

ANNEXE

Le résultat sur Diffchecker lors de la comparaison entre nos sorties et notre code du sprint 5 étant le plus haut atteint :

Pour afficher les Favoris ici, sélectionnez ☆ puis ☆, et faites glisser vers le dossier Barre des favoris. Sinon, importez-les depuis un autre navigateur. [Importer les Favoris](#)

Diffchecker Text Images PDF Folders CLI Desktop App Sign in Create an account

Split Unified Word Character Tools

Saved Diffs
You must be signed in to save diffs
Sign in

Diff history
7 minutes ago
13 minutes ago
Clear

Diff history is cleared on refresh

Students and Teachers, save up to 60% on Adobe Creative Cloud

The two files are identical

Editor

Save Diff Share

Original Text

Changed Text

79 Chine 439.9
80 Russie 442.0
81 Finlande 444.3
82 Canada 447.8
83 France 448.7
84 Hongrie 450.1
85 Japon 450.3
86 Autriche 450.5
87 Pologne 453.6
88 Allemagne 454.3
89 Hollande 455.7
90 USA 455.9
91 Mexique 456.7



```
1  /*
2      sprint 5.c
3      YE fangyuan Lisa 111
4      BASHITI Mountaser Billah 111
5      25/10/19
6  */
7
8  #include<stdio.h>
9  #include<stdlib.h>
10 #include<string.h>
11 #include<assert.h>
12 #pragma warning (disable:4996)
13 #pragma warning (disable:6031)
14
15 #define NbPatineur 3 // definie le nombre de patineur par équipe
16 #define NbEquipe 2 // definie le nombre d'équipe pqr épreuves
17 #define MaxTours 10 //definie le nombre maximum de tours
18 #define MaxEpreuves 16 // definie le nombre maximum d'épreuves
19 #define lgMot 30 // definie la taille maximale des chaînes de caractères
20
21 typedef struct {
22     char nom[lgMot + 1];
23     unsigned int dossard;
24 } Patineur;
25
26 typedef struct {
27     Patineur data[NbPatineur];
28     char pays[lgMot + 1];
29 } Equipe;
30
31 typedef struct {
32     Equipe equipe[MaxEpreuves * NbEquipe];
33     unsigned int ins;
34 } Equipes;
35
```



```

35
36 typedef struct {
37     double temps;
38     unsigned char nbtour;
39     unsigned char dossard;
40 } Mesure;
41
42 typedef struct {
43     Mesure data[MaxEpreuves * NbEquipe * NbPatineur * MaxTours];
44     int eng;
45 } Course;
46
47 typedef struct {
48     int tours;
49 } Tours;
50
51 typedef struct {
52     int epreuves;
53 } Epreuves;
54
55 typedef struct {
56     double temps[MaxEpreuves * NbEquipe];
57     int tempseq;
58 } Temps;
59
60 //Pour résumer toutes mes fonctions
61 void inscrire_equipe(Equipes* e);
62 void afficher_equipes(const Equipes* e, int* dernierd);
63 void enregistrer_temps(Course* crs, int* dernierd);
64 void afficher_temps(const Course* crs, const Equipes* e);
65 void afficher_temps_equipes(const Course* crs, const Equipes* e);
66 void definition_parcours(Tours* t);
67 void detection_fin_poursuite(Course* crs, Equipes* e, Tours* t, Temps* tp, int* nb);
68 void definir_nombre_epreuves(Epreuves* ep);
69 void detection_fin_competition(Equipes* e, Course* crs, Tours* t, Temps* tp);

```

```

69 void detection_fin_competition(Equipes* e, Course* crs, Tours* t, Temps* tp);
70
71 /*inscrire les 3 patineurs de chaque equipe et les numéros de dossard
72 commencent à 101 seront automatiquement attribués par programme dans
73 l'ordre séquentiel d'inscription.*/
74 void inscrire_equipe(Equipes* e) {
75     static unsigned int compt1 = 101;
76     scanf("%s", e->equipe[e->ins].pays);
77     for (int j = 0; j < NbPatineur; ++j) {
78         scanf("%s", e->equipe[e->ins].data[j].nom);
79         e->equipe[e->ins].data[j].dossard = compt1;
80         printf("inscription dossard %d\n", compt1);
81         ++compt1;
82     }
83     e->ins = e->ins + 1;
84 }
85

```

```

86  /* afficher les pays,les noms de membres de l'équipe qui en train de jouer
87  et de leurs numéros respectifs de dossard.*/
88  void afficher_equipes(const Equipes* e, int* dernierd) {
89      int eqencours;
90      if (((*dernierd - 101)/3) % 2 == 0){
91          eqencours = (*dernierd - 101) / 3;
92          for (int i = 0; i < 2; ++i) {
93              printf("%s ", e->equipe[i + eqencours].pays);
94              for (int j = 0; j < NbPatineur; ++j) {
95                  if (j != 0)
96                      printf(" ");
97                  printf("%s %d", e->equipe[eqencours + i].data[j].nom,
98                      e->equipe[eqencours + i].data[j].dossard);
99              }
100              printf("\n");
101          }
102      }
103      else {
104          eqencours = ((*dernierd - 101) / 3) - 1;
105          for (int i = 0; i < 2; ++i) {
106              printf("%s ", e->equipe[i + eqencours].pays);
107              for (int j = 0; j < NbPatineur; ++j) {
108                  if (j != 0)
109                      printf(" ");
110                  printf("%s %d", e->equipe[eqencours + i].data[j].nom,
111                      e->equipe[eqencours + i].data[j].dossard);
112              }
113              printf("\n");
114          }
115      }
116  }
117
118  /*cette fonction permet de memoriser les temps chronomètre de patineur
119

```

```

118
119  /*cette fonction permet de memoriser les temps chronomètre de patineur
120  lors on entre un nombre de tour et le numero de dossard */
121  void enregistrer_temps(Course* crs, int* Dernierd) {
122      char mot[lgMot + 1];
123      scanf("%s", mot);
124      crs->data[crs->eng].dossard = atof(mot);
125      //pour retourner réel à la correspondant à la chaine de mot
126      *Dernierd = atof(mot);
127      // Il permet de distinguer les equipes qui en train de jouer
128      scanf("%s", mot);
129      crs->data[crs->eng].nbtour = atof(mot);
130      scanf("%s", mot);
131      crs->data[crs->eng].temps = atof(mot);
132      crs->eng = crs->eng + 1;
133  }
134
135  // afficher tous les temps enregistrés d'un patineur repéré par son dossard
136  void afficher_temps(const Course* crs, const Equipes* e) {
137      unsigned int dossard;
138      scanf("%d", &dossard);
139      //les dossards sont commencé à 101
140      unsigned int debutant = 101;
141      unsigned int n = dossard - debutant;
142      unsigned int x = n / NbPatineur;
143      /*les patineurs d'un même equipe vont avoir un même numero (x)*/
144      unsigned int y = n % NbPatineur;
145      //mais ils n'ont pas le même (y)
146      for (int i = 0; i < crs->eng; ++i) {
147          if (crs->data[i].dossard == dossard) {
148              printf("%s %d %s %.1f\n", e->equipe[x].pays, crs->data[i].nbtour,
149                  e->equipe[x].data[y].nom, crs->data[i].temps);
150          }
151      }
152  }

```

```

153
154 /*cette fonction permet d'afficher les temps de tous les équipes
155 dans une tours spécifique*/
156 void afficher_temps_equipes(const Course* crs, const Equipes* e) {
157     int tour;
158     unsigned int debutant = 101;
159     double temps;
160
161     scanf("%d", &tour);
162     for (int eq = 0; eq < e->ins; ++eq) {
163         int k = 0;
164         for (int j = 0; j < crs->eng; ++j) {
165             if (crs->data[j].nbtour == tour &&
166                 (crs->data[j].dossard - debutant) / NbPatineur == eq) {
167                 /* enregistre le temps de la dernier patineur
168                 de cette equipe comme le temps d'equipe.*/
169                 temps = crs->data[j].temps;
170                 k++;
171             }
172         }
173         if (k == NbPatineur) {
174             /*si les 3 membres d'une équipe ont tous joué,
175             les temps d'equipes vont afficher */
176             printf("%s ", e->equipe[eq].pays);
177             printf("%.1f\n", temps);
178         }
179         else
180             // sinon "indisponible" va afficher
181             printf("indisponible");
182     }
183 }
184
185 //définir le nombre de tours d'un parcours

```

```

184
185 //définir le nombre de tours d'un parcours
186 void definition_parcours(Tours* t) {
187     char mot[lgMot + 1];
188     scanf("%s", mot);
189     t->tours = atof(mot);
190     /*verifier le nombre de tours est entre 2 et 10*/
191     assert(t->tours >= 2 && t->tours <= 10);
192 }
193
194 /*détecter la fin d'une poursuite et afficher le classement

```

```

194  /*détecter la fin d'une poursuite et afficher le classement
195  des deux derniers équipes.*/
196  void detection_fin_poursuite(Course* crs, Equipes* e, Tours* t,
197  Temps* tp, int* nb) {
198      int debutant = 101;
199      /* Les programmes intervient si tous les patineurs ont effectué
200      le nombre de tours qu'on a défini*/
201      if (crs->eng % (NbEquipe * NbPatineur * t->tours * *nb) == 0) {
202          printf("detection_fin_poursuite\n")
203          for (int eq = tp->tempseq + 1; eq < NbEquipe * *nb; ++eq) {
204              tp->tempseq++;
205              for (int j = (NbEquipe * NbPatineur * t->tours * *nb) -
206                  (NbEquipe * NbPatineur * t->tours); j < crs->eng; ++j) {
207                  if ((crs->data[j].dossard - debutant) / NbPatineur == eq) {
208                      tp->temps[tp->tempseq] = crs->data[j].temps;
209                  }
210              }
211          }
212          //afficher les equipes dans l'ordre croissant des temps finaux
213          if ((tp->tempseq - 1) % 2 == 0) {
214              if (tp->temps[tp->tempseq - 1] <= tp->temps[tp->tempseq]) {
215                  printf("%s ", e->equipe[tp->tempseq - 1].pays);
216                  printf("%.1f\n", tp->temps[tp->tempseq - 1]);
217                  printf("%s ", e->equipe[tp->tempseq].pays);
218                  printf("%.1f\n", tp->temps[tp->tempseq]);
219              }
220              else {
221                  printf("%s ", e->equipe[tp->tempseq].pays);
222                  printf("%.1f\n", tp->temps[tp->tempseq]);
223                  printf("%s ", e->equipe[tp->tempseq - 1].pays);
224                  printf("%.1f\n", tp->temps[tp->tempseq - 1]);
225              }
226          }
227          *nb = *nb + 1;
228      }
229  }

```

```

230
231  //Définir le nombre d'épreuves
232  void definir_nombre_epreuves(Epreuves* ep) {
233      char mot[lgMot + 1];
234      scanf("%s", mot);
235      ep->epreuves = atof(mot);
236      /*verifier le nombre d'epreuve est entre 1 et 16*/
237      assert(ep->epreuves >= 1 && ep->epreuves <= 16);
238  }
239

```

```

240  /*détecter la fin de la compétition toutes les épreuves de poursuite
241  sont terminées. Si oui, le classement final sera affiché*/
242  void detection_fin_competition(Equipes* e, Course* crs, Tours* t, Temps* tp) {
243      if (crs->eng == e->ins * NbPatineur * t->tours) {
244          double v;
245          int j;
246          char p[lgMot + 1];
247          printf("detection_fin_competition\n");
248          //tri par sélection
249          for (int i = 1; i < e->ins; ++i) {
250              v = tp->temps[i];
251              strcpy(p, e->equipe[i].pays);
252              j = i;
253              while (j > 0 && tp->temps[j - 1] > v) {
254                  tp->temps[j] = tp->temps[j - 1];
255                  strcpy(e->equipe[j].pays, e->equipe[j - 1].pays);
256                  j = j - 1;
257              }
258              tp->temps[j] = v;
259              strcpy(e->equipe[j].pays, p);
260          }
261          //afficher tous les equipes dans l'ordre croissant des temps finaux
262          for (int y = 0; y < e->ins; ++y) {
263              printf("%s ", e->equipe[y].pays);
264              printf("%.1f\n", tp->temps[y]);
265          }
266          exit(0); //sortie du programme
267      }
268  }
269
270  //la fonction principale sert a faire appel aux commandes entrées

```



```

269
270 //la fonction principale sert a faire appel aux commandes entrées
271 int main() {
272     Equipes e;
273     Course crs;
274     Tours t;
275     Epreuves ep;
276     Temps tp;
277     char mot[lgMot + 1];
278     mot[0] = '\0';
279     e.ins = 0;
280     crs.eng = 0;
281     tp.tempseq = 0 - 1;
282     int n = 1;
283     int dernierd = 101;
284
285     while (1) {
286         scanf("%s", mot); //scan la commande entrée
287         if (strcmp(mot, "inscrire_equipe") == 0) {
288             inscrire_equipe(&e);
289         }
290         else if (strcmp(mot, "afficher_equipes") == 0) {
291             afficher_equipes(&e, &dernierd);
292         }
293         else if (strcmp(mot, "enregistrer_temps") == 0) {
294             enregistrer_temps(&crs, &dernierd);
295             detection_fin_poursuite(&crs, &e, &t, &tp, &n);
296             detection_fin_competition(&e, &crs, &t, &tp);
297         }
298         else if (strcmp(mot, "afficher_temps") == 0) {
299             afficher_temps(&crs, &e);
300         }
301         else if (strcmp(mot, "afficher_temps_equipes") == 0) {
302             afficher_temps_equipes(&crs, &e);
303         }
304         else if (strcmp(mot, "definir_parcours") == 0) {
305             definition_parcours(&t);
306         }
307         else if (strcmp(mot, "definir_nombre_epreuves") == 0) {
308             definir_nombre_epreuves(&ep);
309         }
310         else if (strcmp(mot, "exit") == 0) {
311             exit(0); //commande de sortie du programme
312         }
313     }
314     system("pause");
315     return 0;
316 }

```