INFO0947: TAD & Récursivité

Groupe 27: Alexandru Dobre, Sami Ouazouz

1 Production

1.1 Introduction

Dans ce projet, notre objectif est de créer un programme visant à modéliser une course de cycliste à plusieurs étapes. Cette course est structurée, divisée en une séquence d'escales dans plusieurs villes où l'on fourni également des coordonnées géographiques et, éventuellement, le meilleur temps entegistré. Une course sera une séquance d'escales avec un point de départ et une arrivée.

Notre but est de fournir un programme implémentant ces structures en type abstrait de données (TAD). Nous allons les définir de manière abstraite puis les implémenter en C.

Nous avons 2 TAD à concevoir, un premier sachant gérer les escales et le second les courses. De plus, le second devra pouvoir être récursif si possible.

1.2 Définitions

1.2.1 TAD : Escale

Type :
Escale
Utilise :
Float, String
Opérations :
create_escale : Float × Float × String → Escale — Constructeur
set_best_time : Escale × Float → Escale — Transformateur
calculate_distance : Escale × Escale → Float
get_x : Escale → Float — Observateur
get_y : Escale → Float
get_name : Escale → String
get_best_time : Escale → Float
Préconditions :
$\forall x \in Float, \forall y \in Float, \forall name \in String$:
create_escale(x, y, name) != NULL
Axiomes :
$\forall e_1, e_2 \in$ Escale, $\forall x_1, x_2, y_1, y_2 \in$ Float $\forall name_1, name_2 \in$ String :
$get\_x(create\_escale(x, y, name_1)) = x$
$get\_y(create\_escale(x, y, name_1)) = y$
$get\_name(create\_escale(x, y, name_1)) = name_1$
$get\_best\_time(create\_escale(x, y, name_1)) = 0$
$get\_best\_time(set\_best\_time(e_1, t)) = t$
$calculate\_distance(create\_escale(x_1, y_1, name_1), create\_escale(x_2, y_2, name_2)) = x_2 - x_1 + y_2 - y_1$

1.2.2 TAD : Course

Type :
Course
Utilise :
Escale, Float, String, Boolean, Integer
Opérations :
create_course : Escale × Escale → Course — Constructeur
add_escale : Course × Escale → Course — Transformateur
del_escale : Course → Course
is_circuit : Course → Boolean — Observateur
escale_nb : Course → Integer
etape_nb : Course → Integer
get_best_time_course : Course → Float
get_best_time_escale : Course × Escale → Float
Préconditions :
$\forall e_1, e_2 \in$ Escale :
$create\_course(e_1, e_2)\ != NULL$
$\forall e \in$ Escale, $\forall c \in$ Course :
$get\_best\_time\_escale(c, e)\ != NULL$
$add\_escale(c, e)\ != NULL$
$del\_escale(c)\ != NULL$
Axiomes :
$\forall e_1, e_2, e_3 \in$ Escale, $\forall c \in$ Course :
$is\_circuit(create\_course(e_1, e_2)) = False$
$get\_best\_time\_escale(create\_course(e_1, e_2)) = 0$
$escale\_nb(create\_course(e1, e2)) = 2$
$etape\_nb(create\_course(e1, e2)) = 1$
$escale\_nb(add\_escale(c, e_1)) = escale\_nb + 1$
$etape\_nb(add\_escale(c, e_1)) = etape\_nb + 1$
$escale\_nb(del\_escale(c, e_3)) = escale\_nb - 1$
$etape\_nb(del\_escale(c, e_3)) = etape\_nb - 1$
$get\_best\_time\_escale(add\_escale(c, e_3), get\_nom(e_3)) = get\_best\_time\_(e_3)$

2 Question(s)