INFO0947: Projet 2
Types Abstraits de Données

Groupe 10: Cyril RUSSE

Table des matières

1 **Introduction** 3
- 1.1 **Contexte** . . . 3
- 1.2 **Enoncé** . . . 3
  - 1.2.1 Types abstraits . . . 3
  - 1.2.2 Représentation Concrète . . . 3

2 **Signature des TAD** 3
- 2.1 **Ville** . . . 3
- 2.2 **Gaule** . . . 4

3 **Spécifications des TAD** 5
- 3.1 **Ville** . . . 5
  - 3.1.1 Structure . . . 5
  - 3.1.2 Spécification des fonctions et procédures de ville.h . . . 5
- 3.2 **Gaule** . . . 6
  - 3.2.1 Structure en Tableau . . . 6
  - 3.2.2 Structure en Liste chainée . . . 6
  - 3.2.3 Spécification des fonctions et procédures gaule.h . . . 7

1 Introduction

1.1 Contexte

En référence à la BD Astérix et Obelix, "Le tour de Gaule d'Astérix", ce travail a pour but d'implémenter ce fameux tour sous la forme d'un type abstrait de données.

1.2 Enoncé

1.2.1 Types abstraits

Dans le cadre de ce projet, nous avions pour consigne de spécifier deux types abstraits :

1. **Ville**
   Ce type abstrait doit permettre de :
   — créer une ville à partir de ses deux coordonnées X et Y et de son nom ;
   — obtenir la coordonnée X d'une escale ;
   — obtenir la coordonnée Y d'une escale ;
   — obtenir le nom de la ville ;
   — calculer la distance géographique entre deux ville ;
   — enregistrer la spécialité gastronomique de la ville ;
   — obtenir la spécialité gastronomique de la ville ;
2. **Gaule**
   Ce type abstrait doit permettre de :
   — créer un tour de Gaule sur base de deux villes. Par définition, un tour nouvellement créé ne peut pas constituer un circuit ;
   — déterminer si un tour de Gaule constitue un circuit ;
   — déterminer le nombre de villes visitées durant le tour ;
   — déterminer le nombre totale de spécialités gastronimiques dans le tour ;
   — déterminer la spécialité gastronomique d'une ville du tour ;
   — ajouter une ville à un tour ;
   — supprimer une ville d'un tour ;

1.2.2 Représentation Concrète

Ces types abstraits doivent être représentés de 2 manières :
— Un tableau
— Une liste chainée

2 Signature des TAD

2.1 Ville

Type :
— Ville
Utilise :
— $String$
— $\mathbb{R}$
Opérations :
— creer_ville : $String \times \mathbb{R} \times \mathbb{R} \rightarrow Ville$
— get_x_ville : $Ville \rightarrow \mathbb{R}$
— get_y_ville : $Ville \rightarrow \mathbb{R}$

— get_nom_ville : $Ville \rightarrow String$
— get_specialite_ville : $Ville \rightarrow String$
— set_specialite_ville : $Ville \times String \rightarrow Ville$
— distance_entre_2_villes : $Ville \times Ville \rightarrow \mathbb{R}$

Préconditions :
— $\emptyset$

Axiomes : $\forall x, y \in \mathbb{R} \wedge \forall V_1, V_2 \in Ville \wedge \forall s, t \in String$
— distance_entre_2_villes(V_1, V_2) = $\sqrt{(get\_x\_ville(V_2) - get\_x\_ville(V_1))^2 + (get\_y\_ville(V_2) - get\_y\_ville(V_1))^2}$
— distance_entre_2_villes(set_specialite_ville(V_1, s), set_specialite_ville(V_2, t)) = distance_entre_2_villes(V_1, V_2)
— get_x_ville(creer_ville(s, x, y)) = x
— get_y_ville(creer_ville(s, x, y)) = y
— get_x_ville(set_specialite_ville(V_1, s)) = get_x_ville(V_1)
— get_y_ville(set_specialite_ville(V_1, s)) = get_y_ville(V_1)
— get_nom_ville(creer_ville(s, x, y)) = s
— get_nom_ville(set_specialite_ville(V_1, s)) = get_nom_ville(V_1)
— get_specialite_ville(set_specialite_ville(V_1, s)) = s
— get_specialite_ville(creer_ville(s, x, y)) = NULL

2.2 Gaule

Type :
— Gaule

Utilise :
— $Ville$
— $Entiers$
— $String$
— $Booleen$

Opérations :
— cree_nouveau_tour : $Ville \times Ville \rightarrow Gaule$
— get_nombre_villes : $Gaule \rightarrow Entiers$
— ajoute_ville : $Gaule \times Ville \rightarrow Gaule$
— supprime_ville : $Gaule \rightarrow Gaule$
— get_est_circuit : $Gaule \rightarrow Booleen$
— get_nombre_specialite : $Gaule \rightarrow Entiers$
— get_specialite : $Gaule \times String \rightarrow String$
— ville_en_double : $Gaule \times Ville \rightarrow Booleen$

Préconditions :

Axiomes :$\forall G \in Gaule, \forall V_1, V_2, \in Ville$
— get_nombre_villes(ajoute_ville(G, V_1)) = get_nombre_villes(G)+1
— get_nombre_villes(supprime_ville(G)) = get_nombre_villes(G)-1
si get_nombre_villes(G)> 0
sinon get_nombre_villes(supprime_ville(G)) = get_nombre_villes(G)
— get_nombre_villes(cree_nouveau_tour(V_1, V_2)) = 2
— get_nombre_specialite(ajoute_ville(G, V_1)) = get_nombre_specialite(G)+1
si ville_en_double(G, V_1)=False $\wedge$ get_specialite$(G, V_1)\neq$NULL
sinon get_nombre_specialite(ajoute_ville(G, V_1)) = get_nombre_specialite(G)

— get_nombre_specialite(cree_nouveau_tour(V_1, V_2)) = 2
si get_specialite(G, V_1)≠NULL ∧ get_specialite(G, V_2)≠NULL
— get_nombre_specialite(cree_nouveau_tour(V_1, V_2)) = 1
si get_specialite(G, V_1)≠NULL ∧ get_specialite(G, V_2)=NULL
∨get_specialite(G, V_2)≠NULL ∧ get_specialite(G, V_1)=NULL
— get_nombre_specialite(cree_nouveau_tour(V_1, V_2)) = 0
si get_specialite(G, V_1)=NULL ∧ get_specialite(G, V_2)=NULL
— get_est_circuit(cree_nouveau_tour(V_1, V_2)) = False
— get_specialite(cree_nouveau_tour(V_1, V_2), V_1) = get_specialite_ville(V_1)
— get_specialite(cree_nouveau_tour(V_1, V_2), V_2) = get_specialite_ville(V_2)
— get_specialite(ajoute_ville(G, V_1), V_1) = get_specialite_ville(V_1)

3 Spécifications des TAD

3.1 Ville

3.1.1 Structure

```c
struct Ville_t{
char *nom;
float x;
float y;
char *specialite;
};
```

Extrait de Code 1 – Structure "Ville"

3.1.2 Spécification des fonctions et procédures de ville.h

```c
/*
*@pre : nom ≠ NULL ∧ x = x0 ∧ y = y0 ∧ nom = nom0
*@post : ville_init ∧ x = x0 ∧ y = y0 ∧ nom = nom0 ∧ get_x_ville(ville) = ville− > x∧
*get_y_ville(ville) = ville− > y ∧ get_nom_ville(ville) = ville− > nom
*/
Ville *creer_ville(char *nom, float x, float y);

/*
*@pre : ∅
*@post : ville = NULL
*/
void detruit_ville(Ville *ville);

/*
*@pre : ville ≠ NULL
*@post : ville = ville0 ∧ get_x_ville(ville) = ville− > x
*/
float get_x_ville(Ville *ville);

/*
*@pre : ville ≠ NULL
*@post : ville = ville0 ∧ get_y_ville(ville) = ville− > y
*/
float get_y_ville(Ville *ville);

/*
```

```
27      *@pre : ville ≠ NULL
28      *@post : ville = ville0 ∧ get_nom_ville(ville) = ville− > nom
29      */
30      char *get_nom_ville(Ville *ville);
31
32      /*
33      *@pre : ville ≠ NULL
34      *@post : ville = ville0 ∧ get_specialite_ville(ville) = ville− > specialite
35      */
36      char *get_specialite_ville(Ville *ville);
37
38      /*
39      *@pre : ville ≠ NULL ∧ specialite ≠ NULL ∧ specialite = specialite0
40      *@post : ville = ville0 ∧ specialite = specialite0 ∧ get_specialite_ville(ville) = specialite
41      */
42      void set_specialite_ville(Ville *ville, char *specialite);
43
44      /*
45      *@pre : ville1 = ville10 ≠ NULL ∧ ville2 = ville20 ≠ NULL
46      *@post : ville1 = ville10 ∧ ville2 = ville20∧
47      *distance_entre_2_villes(ville1, ville2) =
48      *√((get_x_ville(ville2) − get_x_ville(ville1))² + (get_y_ville(ville2) − get_y_ville(ville1))²
49      */
50      float distance_entre_2_villes(Ville *ville1, Ville *ville2);
51
52      /*
53      *@pre : ∅
54      *@post : retourne la taille mémoire de la struct Ville
55      */
56      int size_ville(void);
```

Extrait de Code 2 – Spécification des fonctions et procédures du header "ville.h"

3.2 Gaule

3.2.1 Structure en Tableau

```
1       struct Gaule_t{
2       Ville **tableau_ville;
3       int nombre_villes;
4       int est_circuit;
5       int nombre_specialites;
6       };
```

Extrait de Code 3 – Structure "Gaule" dans l'implémentation en tableau

3.2.2 Structure en Liste chainée

Pour la liste chainée, une deuxième structure vient s'ajouter. La première, comme pour les tableaux, garde les informations sur la liste et la deuxième sont les structures qui correspondront chacune à une des villes avec un pointeur sur l'élément suivant et précédent de la liste.

```
1       struct Gaule_t{
2       Cellule_Gaule *premiere_cellule;
3       Cellule_Gaule *derniere_cellule;
4       int nombre_villes;
5       int est_circuit;
6       int nombre_specialites;
```

```
7       };
8
9       struct Cellule_Gaule_t{
10      Cellule_Gaule *cellule_suivante;
11      Cellule_Gaule *cellule_precedente;
12      Ville *ville;
13      };
```

Extrait de Code 4 – Structure "Gaule" dans l'implémentation en tableau

3.2.3 Spécification des fonctions et procédures gaule.h