INFO0947: Polylignes, Milestone 1

Groupe 06: Maxime Deravet, Luca Matagne

Table des matières

1 Remarques

Une section complète étant dédiée aux opérations nécessitant un invariant, celles-ci ne se retrouveront donc pas dans la section "structure de données"

2 Structures de données

2.1 Point2D : fonctions et structure

```
struct Point2D{
    float x;
    float y;
};
```

```

/*
 * @pre: /
 * @post: (get_x(create_Point2D) = x
 * ∧
 * get_y(create_Point2D) = y)
 */
Point2D* CreatePoint2D(float x, float y);
/*
 * @pre: A != NULL ∧ B != NULL
 * @post: $A = Translate_{(A,B)} \wedge B = B_0$
 */
void TranslatePoint2D(Point2D* A, Point2D* B);
/*
 * @pre: A != NULL ∧ B != NULL
 * @post: $A = Rotate_{(A,B),x} \wedge B = B_0$
 */
void RotatePoint2D(Point2D* A, Point2D* B, float x);
/*
 * @pre: A != NULL
 * @post: $A = A_0 \wedge get_x = x$
 */
float get_x(Point2D* A);
/*
 * @pre: A != NULL
 * @post: $A = A_0 \wedge get_y = y$
 */
float get_y(Point2D* A);
/*
 * @pre: A != NULL ∧ B != NULL
 * @post: $A = A_0 \wedge B = B_0 \wedge EuclDist = \sqrt{(X_a - X_b) + (Y_a - Y_b)}$
 */
unsigned float EuclDist(Point2D* A, Point2D* B);
```

2.2 Polyligne : fonctions et structure

```
struct Polyline{
    boolean open;
    unsigned nbpoint;
    unsigned float length;
    unsigned arraySize;
    Point2D** pointArray;
};
```

```
/*
 * @pre: A != NULL ∧ B != NULL ∧
 * @post: $A = A_0$ ∧ $B = B_0$ ∧ open=open ∧ create_Polyligne = P ∧
 * nbpoint(P) = NbrPoint(P) ∧ length(P) = Length(P)
 */
Polyline* CreatePolyline(Point2D* A, Point2D* B, boolean open);
/*
 * @pre: P != NULL
 * @post: $P = P_0$ ∧ open = False ∧ $nbpoint = nbpoint_0$
 */
void Open(Polyline* P);
/*
 * @pre: P != NULL
 * @post: $P = P_0$ ∧ open(P) = True ∧ $nbpoint = nbpoint_0$
 */
void Close(Polyline* P);
/*
 * @pre: P != NULL
 * @post: $P = P_0$ ∧
 */
void IsOpen(Polyline* P);
Poser la question close et open
/*
 * @pre: P != NULL
 * @post: P=$P_0$ ∧ nbpoint = NbrPoint(P)
 */
unsigned NbrPoint(Polyline* P);
/*
 * @pre: $P! = NULL\wedge$ numero < nbpoint
 * @post: $P = P_0 \wedge GetPoint = A_{numero}$
 */
Point2D GetPoint(Polyline* P, unsigned numero);
/*
 * @pre: P != NULL ∧ A != NULL
 * @post: $A = A_0$ ∧ $open = open_0$ ∧ $nbpoint = nbpoint_0 + 1$
 */
Poser la question distance en post condition, recalculer?
void AddPoint2D(Polyline* P, Point2D* A);
/*
 * @pre: P != NULL ∧ A != NULL
 * @post: $A = A_0$ ∧ $open = open_0$ ∧ $nbpoint = nbpoint_0 - 1$
 */
 void SuppPoint2D(Polyline* P);
```

3 Invariants

A)ajouter les tableaux d'invariant et les invariants formels vendredi après le boulot
B)finir rotate et translate

3.1 length

```
/*
 * @pre: P != NULL
 * @post: P=$P_0$ ∧ length = Length(P)
 */
unsigned float length(Polyline* P);
```

3.2 Rotate

3.3 Translate