
# Travail pratique # 4 : Tests Unitaires

École polytechnique de Montréal

**Trimestre :** automne 2017

Équipier1 : *Julien, Legault matricule 1847125*

Équipier2 : *Duc-Thien, Nguyen matricule 1878502*

**Equipe numéro : 25**

**Présenté à :** Khalil Bennani

Polytechnique Montréal
Date de remise (20-11-2017)

# E1) Diagramme de flot de contrôle [/30]

**1.**

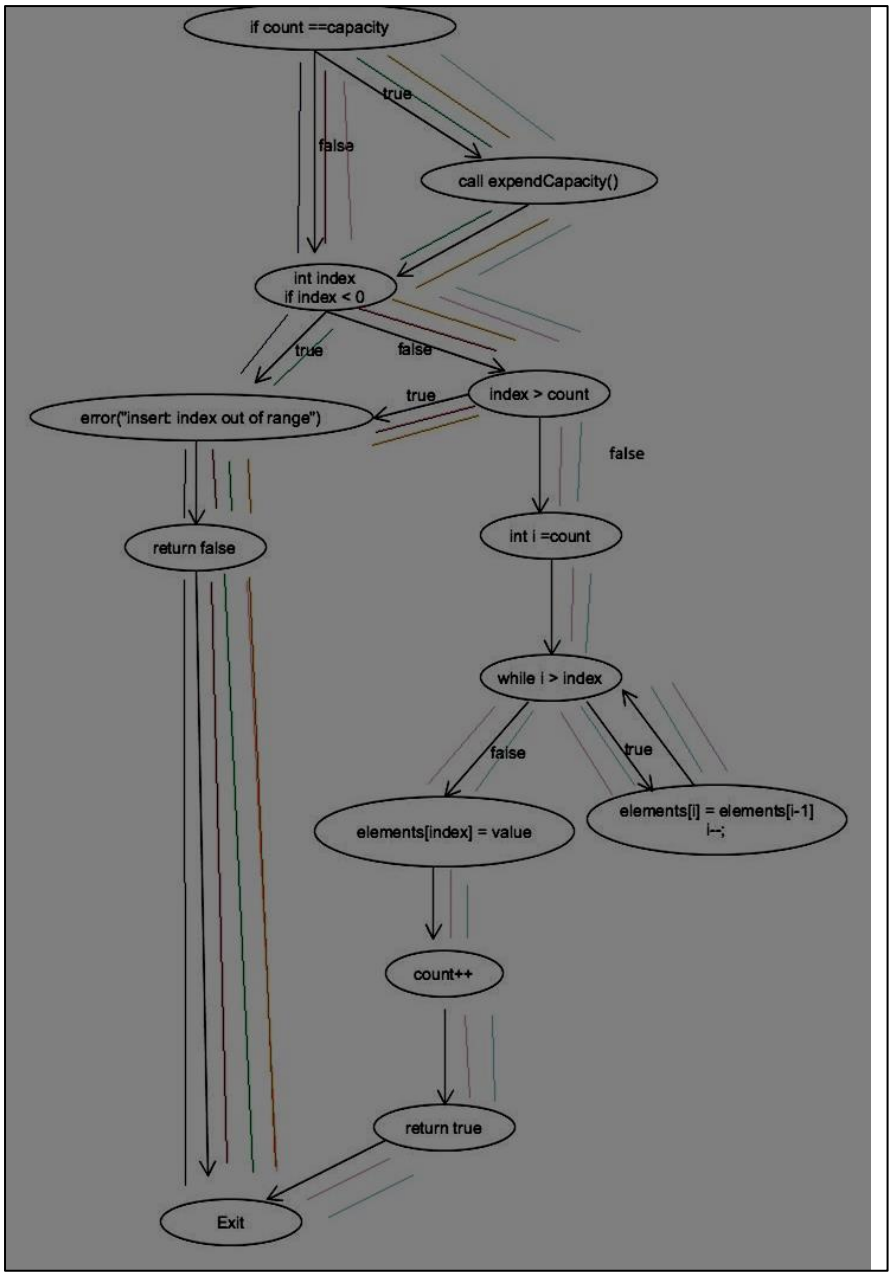


Figure1 : diagramme de flot de contrôle de la fonction Insert

**2.**

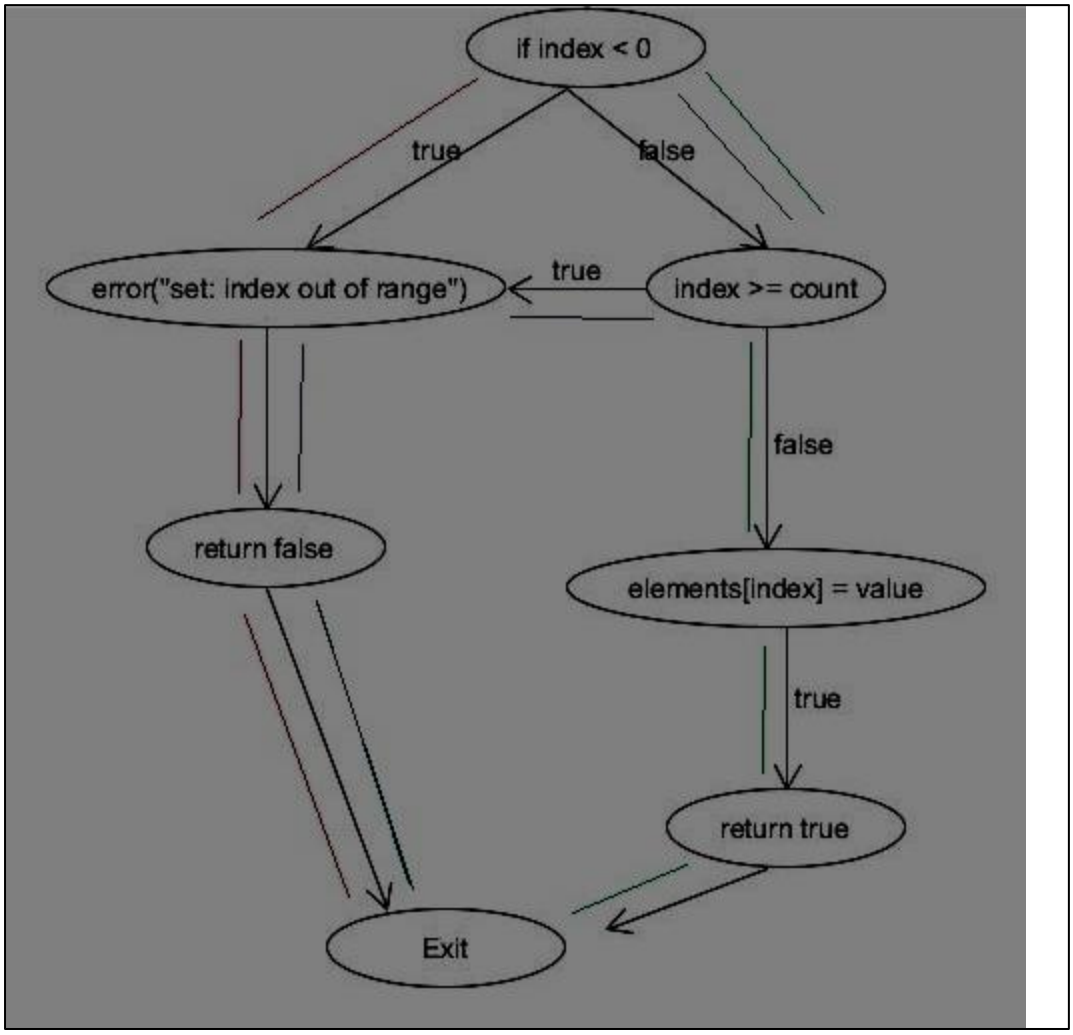


Figure 2 : diagramme de flot de contrôle de la fonction Set

**3.**

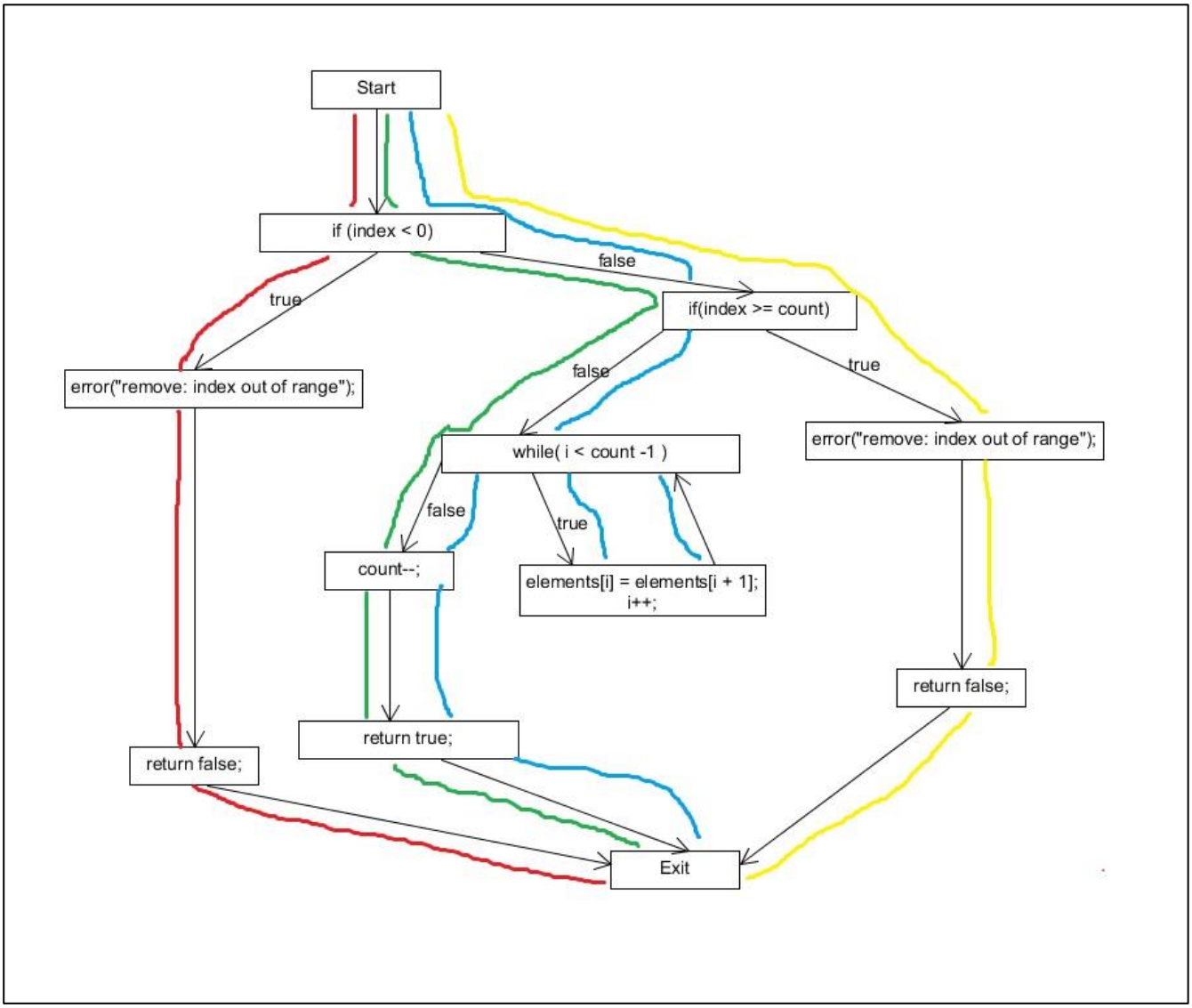


Figure 3 : diagramme de flot de contrôle de la fonction Remove

**4.**

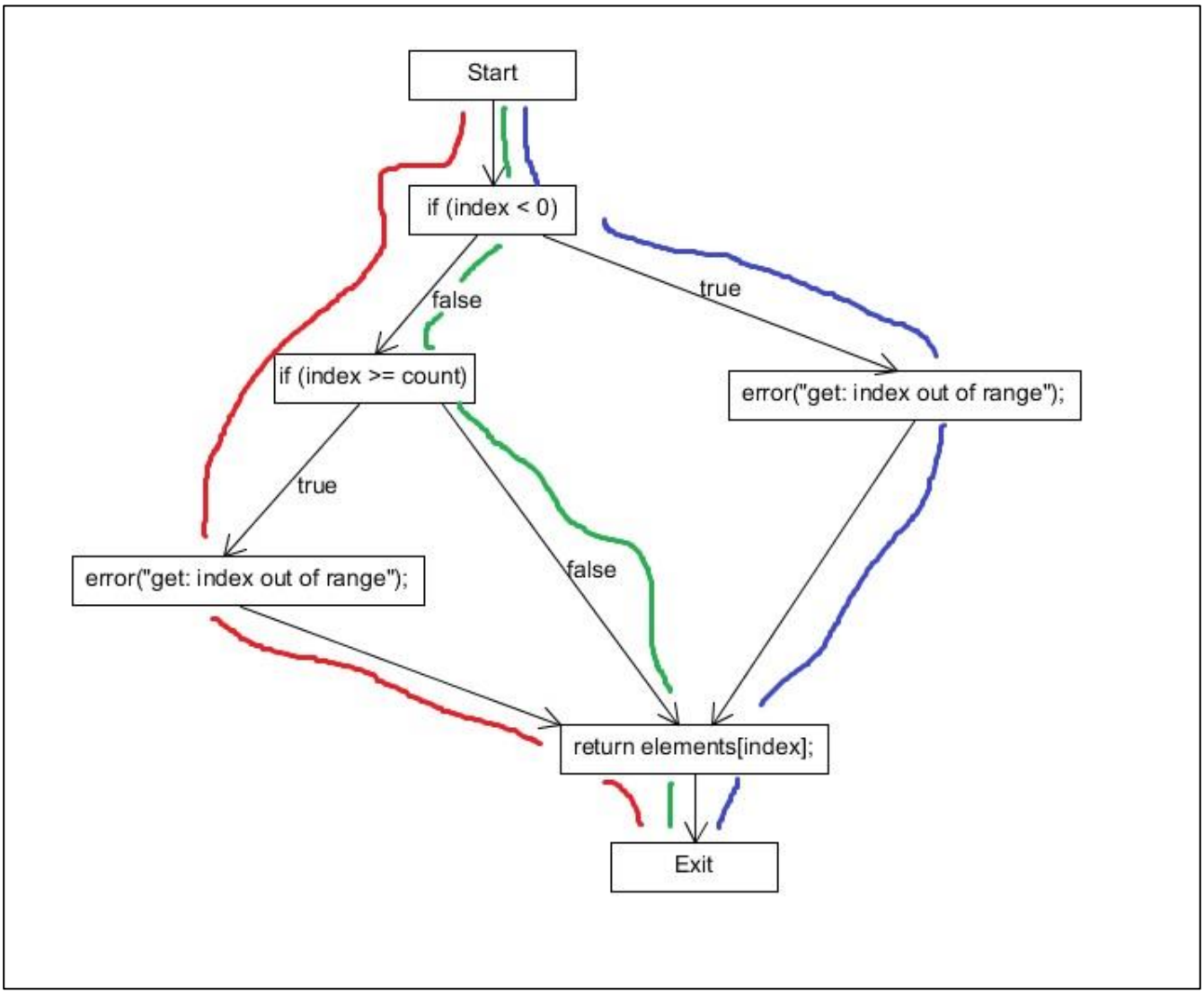


Figure 4 : diagramme de flot de contrôle de la fonction Get

Tableau 1 : Informations importantes au calcul de la complexité cyclomatique

| fonction / Carac. du graph | Insert | Remove | Get | Set |
|---|---|---|---|---|
| Arcs | 16 | 14 | 8 | 8 |
| Nœuds | 13 | 12 | 7 | 7 |
| Conditions | 4 | 3 | 2 | 2 |

**Formules**

Formule du calcul de complexité cyclomatique standard :

- M = nb arcs – nb nœud + 2

Formule du calcul de complexité cyclomatique avec points de décisions :

- M = nb conditions +1

Tableau 2 : Résultats des calculs de complexité cyclomatique selon les différentes méthodes

| Compléxité cyclomatique | insert | remove | get | set |
|---|---|---|---|---|
| Méthode 1 | 5 | 4 | 3 | 3 |
| Méthode 2 | 5 | 4 | 3 | 3 |

# E2) Cas de tests et jeu de données [/10]

Tableau 3 : Résultat de la sortie de la fonction Insert en fonction des entrées

| | Entrées | Sortie |
|---|---|---|
| Possibilité 1 (Bleu) | int index = -1<br>String value = “Test” | error("remove: index out of range");<br>return false; |
| Possibilité 2 (Vert) | int index = -1<br>String value = “Test” | error("remove: index out of range");<br>return false; |
| Possibilité 3 (Jaune) | int index = 2<br>String value = “Test” | error("remove: index out of range");<br>return false; |
| Possibilité 4 (Rouge) | int index = 2<br>String value = “Test” | error("remove: index out of range");<br>return false; |
| Possibilité 5 (Rose) | int index = 0<br>String value= “Test” | Return true; |
| Possibilité 6 (Cyan) | int index = 0<br>String value= “Test” | Return true; |

Tableau 4 : Résultat de la sortie de la fonction Set en fonction des entrées

| | Entrées | Sortie |
|---|---|---|
| Possibilité 1 (Rouge) | int index = -1<br>String value = “Test” | error("remove: index out of range");<br>return false; |
| Possibilité 2 (Bleu) | Int index = 2<br>String value = “Test” | error("remove: index out of range");<br>return false; |
| Possibilité 3 (Vert) | int index = 0<br>String value = “Test” | Return true; |

Tableau 5 : Résultat de la sortie de la fonction Remove en fonction des entrées

| | Entrées | Sortie |
|---|---|---|
| Possibilité 1 (Rouge) | count = 2<br>index = -1 | error("remove: index out of range");<br>return false; |
| Possibilité 2 (Jaune) | count = 2<br>Index = 3 | error("remove: index out of range");<br>return false; |
| Possibilité 3 (Vert) | count = 2<br>Index = 1 | Return true; |
| Possibilité 4 (Bleu) | count = 2<br>Index = 0 | Return true; |

Tableau 6 : Résultat de la sortie de la fonction Get en fonction des entrées

| | Entrées | Sortie |
|---|---|---|
| Possibilité 1 (Bleu) | Count = 1<br>Index = -1 | error("get: index out of range");<br>return elements[-1]; |
| Possibilité 2 (Vert) | Count = 2<br>Index = 1 | return elements[0]; |

| | | |
|---|---|---|
| Possibilité 3<br>(Rouge) | Count = 1 | error("get: index out of range");<br>return elements[2]; |
| | Index = 2 | |

## E3) Implémentation et exécution des tests unitaires [/40]

[juleg3@l4712-11 tests]$ '/usagers/juleg3/Ecole/A-17/log1000-25/tp4/tests/testsVector'
insertTest::test1insert: index out of range
: OK
insertTest::test2insert: index out of range
: OK
insertTest::test3 : OK

removeTest::test1 : OK
removeTest::test2remove: index out of range
: OK
removeTest::test3remove: index out of range
: OK

getTest::test1 : OK

setTest::test1set: index out of range
: OK
setTest::test2set: index out of range
: OK
setTest::test3 : OK
OK (10)

Précision : Nous avons décidé de ne pas tester les chemins où l'index est au-dessus et au-dessous des bornes pour la fonction Get afin d'éviter une faute de segmentation puisqu'il est impossible de retourner une telle chaine. De plus, certains tests ont été regroupé, car il était impossible de comparer la taille de la capacité pour la fonction insert ainsi que de comparer la valeur de i par rapport a count pour la fonction remove.

**1. Citez quatre utilités des tests unitaires en justifiant vos réponses? [/2]**

a) Les tests unitaires aident au moment de « debugging ». Si un test échoue, seules les dernières modifications apportées au code doivent être déboguées.
b) Les tests unitaires aident à trouver des bogues plus tôt avant son intégration. Les codes sont ré-solus plus tôt sans avoir impact à d'autres parties de code.
c) Puisque les tests unitaires aident à trouver des erreurs dans les codes plus tôt, ça réduit le coût pour la correction des codes ultérieur.
d) Les tests unitaires aident à l'optimisation du code. Les révisions du code donnent des fois des remarques qui peuvent améliorer l'apparence du code ainsi que faire des modifications pour une meilleur adaptabilité ou performance.

**2. Est ce qu'il sera meilleur de faire les tests unitaires à la fin de développement d'un projet? Pourquoi? [/2]**

Il est meilleur de faire les tests unitaires durant le développement d'un projet, car ça aide au développement puisqu'on sait ce qu'une partie du code fait et devrait faire. Ça aidera à trouver les bogues dans le code plus rapidement que faire les tests à la fin aussi. Aussi, lorsqu'on trouve des bogues plus tôt, ça diminue le coût pour la révision des codes.

**3. Quand considérons-nous que la couverture de test est optimale? [/2]**

Il est en effet impossible de faire la couverture des tests à 100%. Pour la couverture optimale, c'est la couverture des conditions au complet. Comme dans une méthode d'une classe, s'il a plusieurs « if », on veut décrire toute les conditions possibles qui peuvent entrer dans les « «if » ou non.

**4. Quel est la différence entre les tests de boîte blanche et les tests de boîte noire? [/2]**

Le test de la boîte noire est la méthode de test du logiciel qui est utilisée pour tester le logiciel sans connaître la structure interne du code ou du programme. Le test de boîte blanche est la méthode de test de logiciel dans laquelle la structure interne est connue du testeur qui va tester le logiciel.

**5. C'est quoi la différence entre la défaillance et erreur ? [/2]**

Une erreur est lorsqu'il y a une mauvaise interprétation de la demande pour des codes. Celui-ci fonctionnera bien, mais ce n'est pas ce qui était demandé. Une défaillance est lorsqu'il y a une bonne interprétation du code, mais il y a une incapacité d'exécution qui retourne la spécification demandée.

# Contribution au projet Ring

### 1. À partir du code source, dessinez les diagrammes de flot de contrôle pour la méthode choisie. [/2]

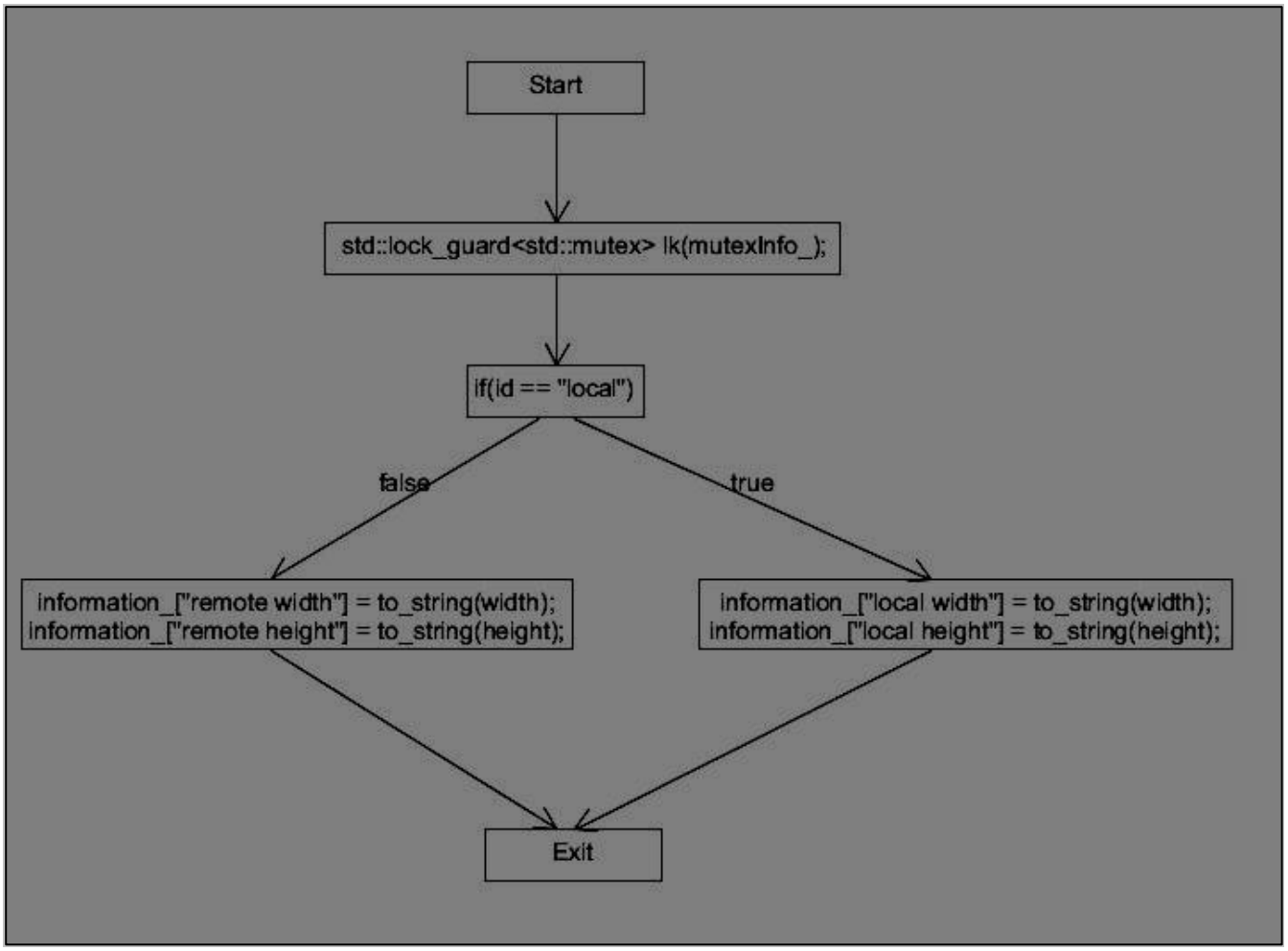

Figure 5 : diagrammes de flot de contrôle de la méthode setResolution

### 2. Calculez la complexité cyclomatique avec les deux approches vues en classe pour contrôler vos résultats. [/2]

Tableau 7 : Calcul de la compléxité cyclomatique de la méthode setResolution

| **Compléxité cyclomatique** | **setResolution ( const std::string& id, int width, int height)** |
|---|---|
| M = nb arcs – nb nœud + 2 | Arc: 6, Noeud = 6<br>Compléxité = 2 |
| M = nb conditions +1 | Condition : 1<br>Compléxité = 2 |

**3. Tracez les chemins nécessaires à parcourir pour couvrir toutes les conditions. [/4]**

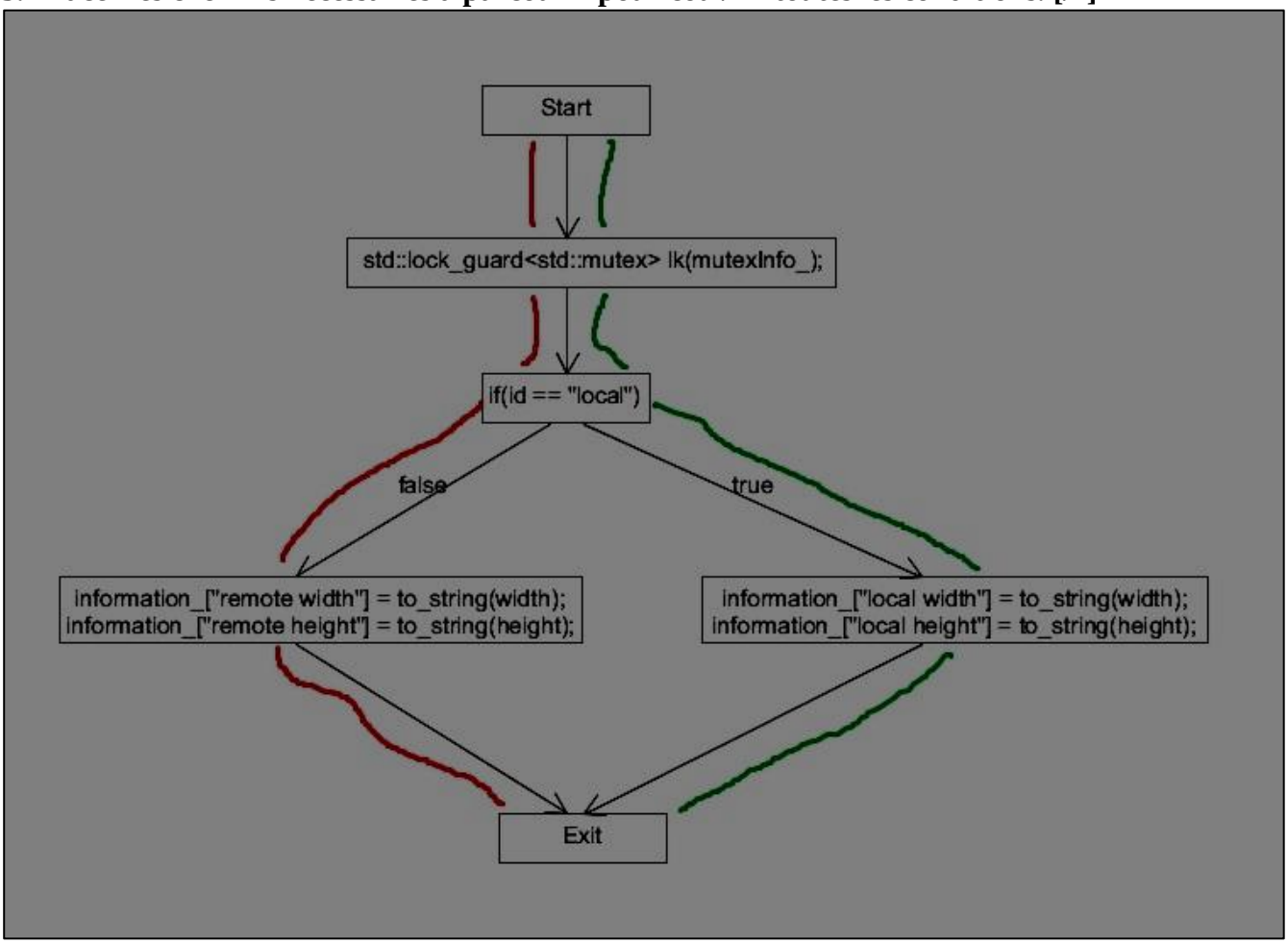


Figure 6 : diagrammes de flot de contrôle de la méthode setResolution avec chemins possibles

**4. Est-ce que les tests qui étaient conçues couvrent tous les chemins que vous avez trouvez? Sinon, quels sont-ils les cas des tests manquants? [/2]**

Les tests pour la méthode (Smartools::setResolution(const std::string& id, int width, int height)) couvrent tous les chemins possibles. Les conditions qui donnent plusieurs possibilités est le paramètre (id). La comparaison ne compare que si le paramètre est équivalent à « local ». Donc il n'y avait que deux chemins possibles, soit la condition est vraie ou la condition est fausse.