**E1) Diagramme de flot de contrôle**

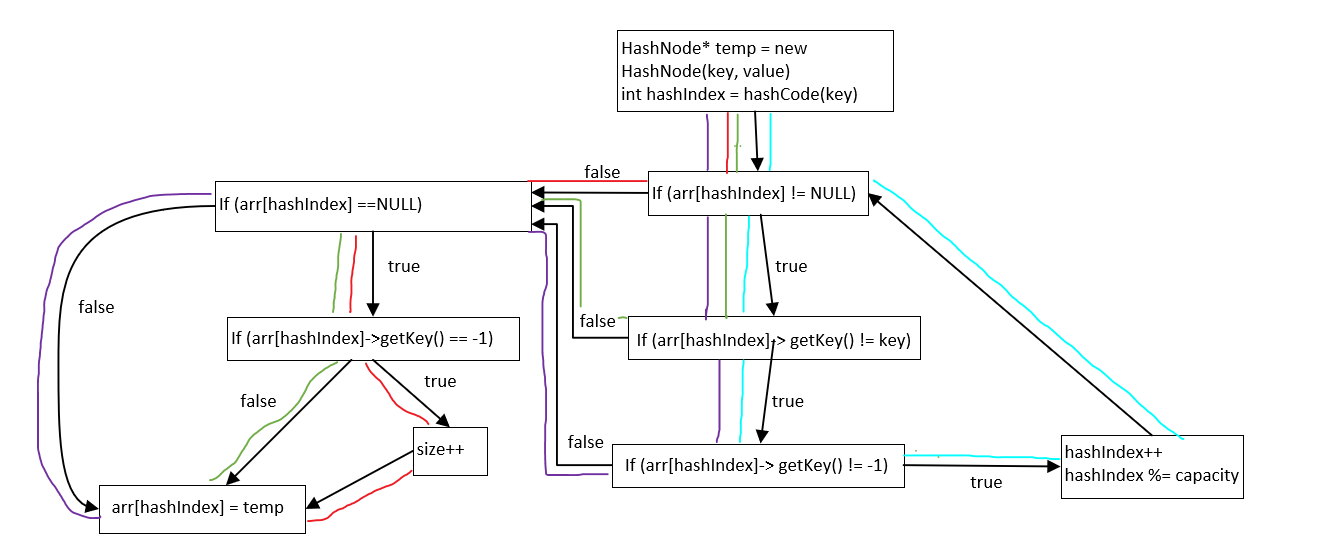


Figure 1 Diagramme de flot de la méthode void HashMap::insertNode(int key, int value)

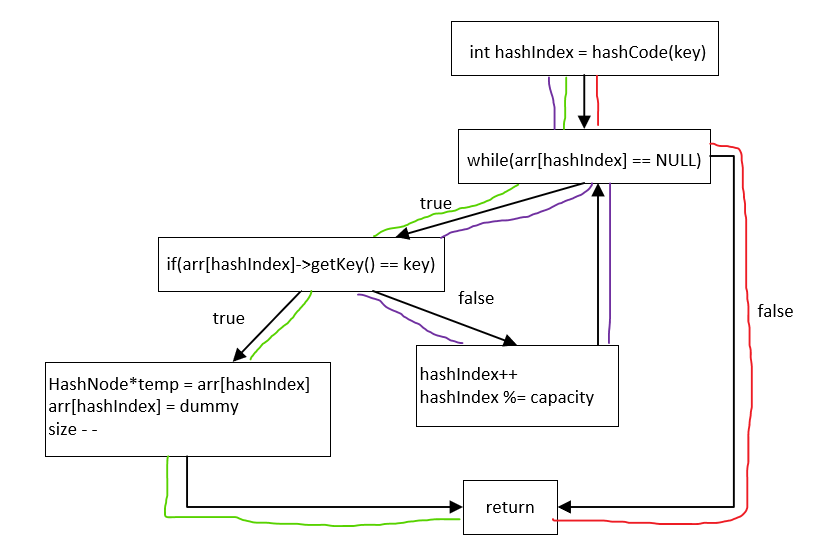


Figure 2 Diagramme de flot de la méthode int HashMap::deleteNode(int key)

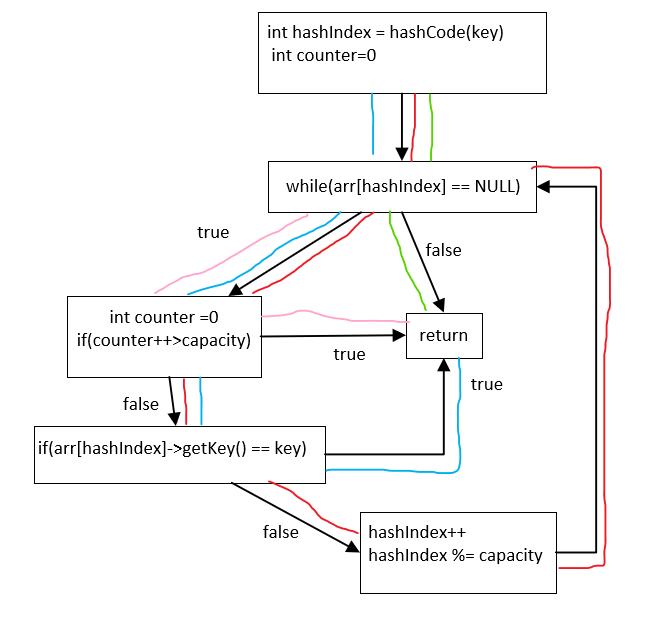


Figure 3 Diagramme de flot de la méthode int HashMap::get(int key)

1. La complexité cyclomatique est calculer avec les deux approches suivantes :
2. M = NombrePointDeDecisions + 1
3. M = NombreAretes – NombreNoeuds + 2
   1. Pour la méthode insertNode :
4. M = 5 + 1 = 6
5. M = 13 – 9 + 2 = 6
   1. Pour la méthode deleteNode :
6. M = 2 + 1 = 3
7. M = 7 – 6 + 2 = 3
   1. Pour la méthode get :
8. M = 3 + 1 = 4
9. M = 8 – 6 + 2 = 4

**E2) Cas de tests et jeu de données**

Méthode insertNode :

Comme cette méthode n’a pas de variable de retour, les variables pris en entrée ne vont pas affecter la sortie. Cette méthode permet seulement d’insérer une valeur et une clé qui lui est assigner dans un tableau.

|  |  |
| --- | --- |
| **Entrée** | **Sortie** |
| key = 5, value = 2 | Pas de sortie |
| key = 8, value = 0 | Pas de sortie |

Méthode deleteNode :

Soit on prend en considération le tableau suivant :

|  |  |  |  |
| --- | --- | --- | --- |
| key = 5, value = 2 | key = 10, value = 1 | key = 8, value = 0 | key = 11, value = 5 |

|  |  |
| --- | --- |
| **Entrée** | **Sortie** |
| key = 5 | Retour = 2  En effet la méthode va enlever la première case du tableau et retournera la valeur correspondant au key pris en paramètre |
| key = 15 | Retour = 0  En effet, c’est a cause que le key = 15 ne se retrouve pas dans le tableau donc il est contre-productif d’essai de l’enlevé du tableau |

Méthode get :

Soit on reprend le tableau précédent pour cette méthode.

|  |  |
| --- | --- |
| **Entrée** | **Sortie** |
| key = 5 | Retour value = 2  La méthode va retourner la valeur correspondant au key pris en paramètre |
| key = 9 | Retour = 0  Car le tableau ne contient pas un key qui est égale à 9 (donc il n’y pas un valeur a retourner) |

**E3) Implémentation et exécution des tests unitaires**

**E4) Débogage**

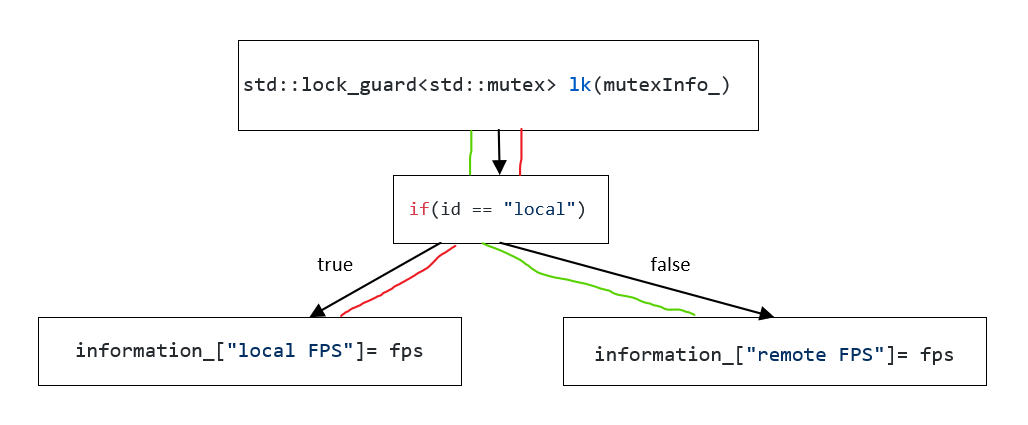
**E5) Contribution au projet Ring**

Figure 4 Diagramme de flot de la méthode Void Smartools::setFrameRate(const std::string& id, const std::string& fps)

La complexité cyclomatique est calculer avec les deux approches suivantes :

1. M = NombrePointDeDecisions + 1 = 1 + 1 = 2
2. M = NombreAretes – NombreNoeuds + 2 = 3 - 4 + 2 = 1