Gaspar Faure 1903623

Paul Clas 1846912

LOG1000

Ingénierie Logicielle

TP 4

8 avril 2018

E1:

1. Digramme de flot de contrôle [ /50]

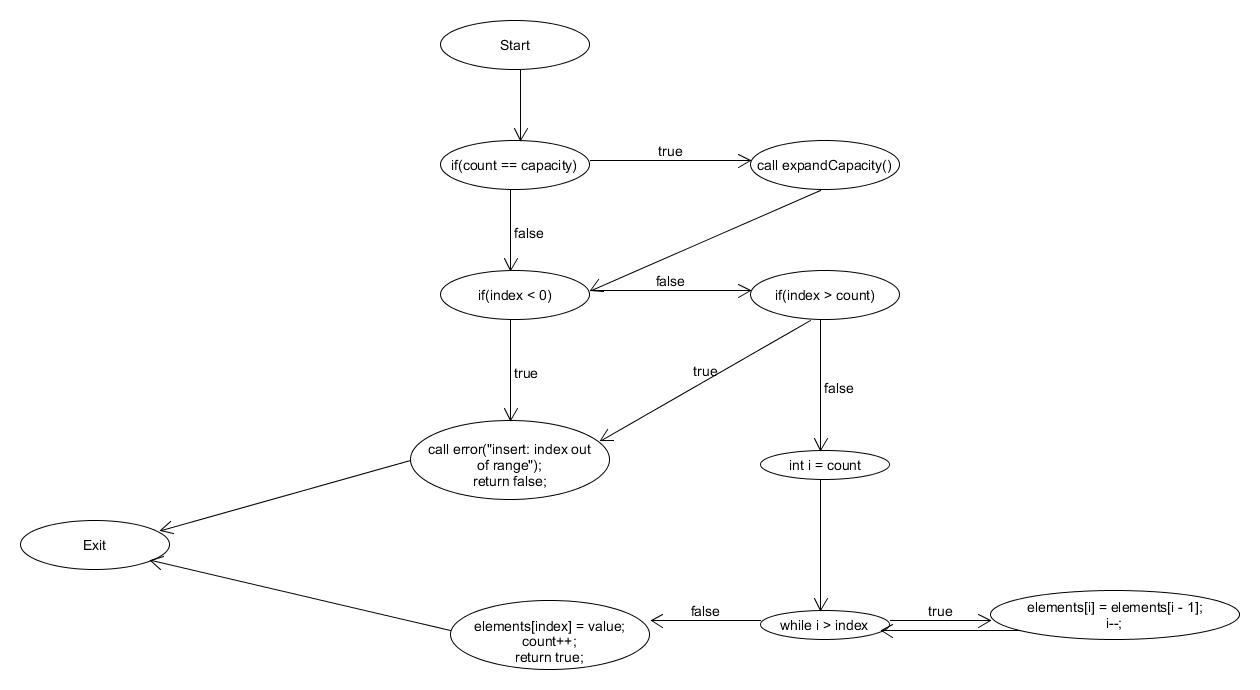
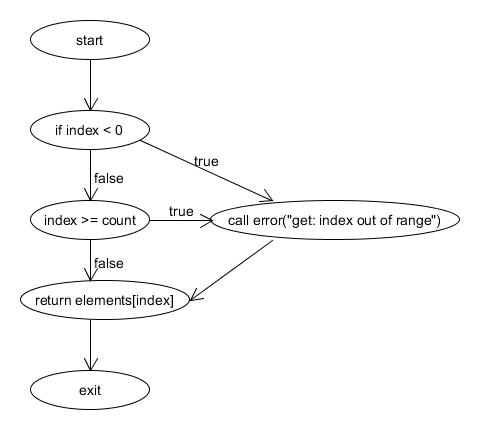




Figure 1 : Diagramme de flot de vector ::insert



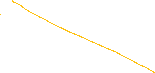
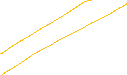
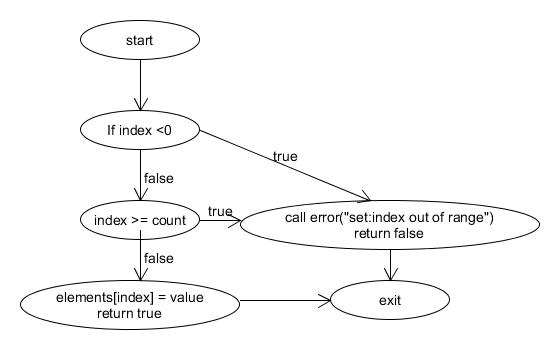


Figure 2 : Diagramme de flot de vector ::get



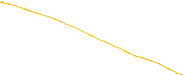


Figure 3 : Diagramme de flot de vector ::set

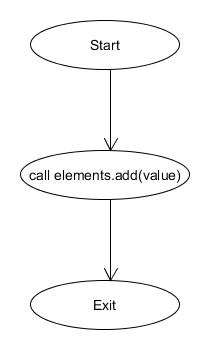




Figure 4 : Diagramme de flot de stack ::push

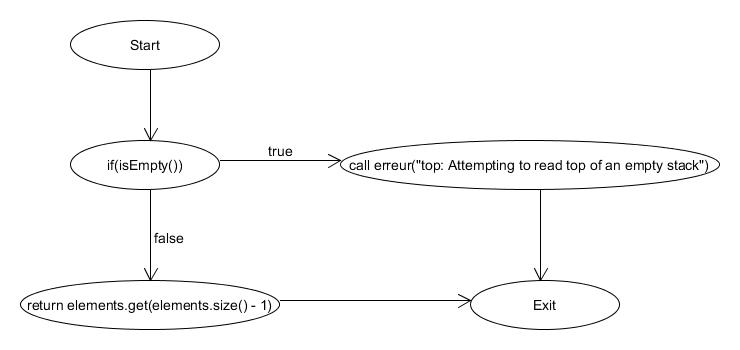




Figure 5 :Diagramme de flot de stack ::top

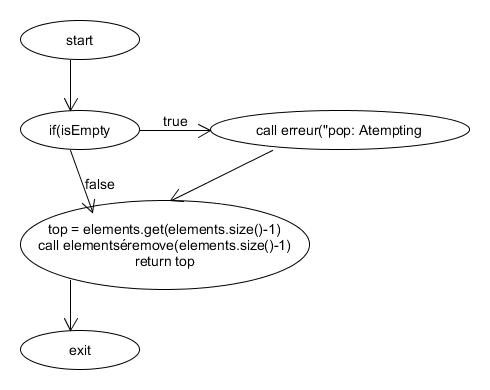




Figure 6 : Diagramme de flot de stack ::pop

1. Complexité Cyclomatique

|  |  |  |  |
| --- | --- | --- | --- |
| Fonctions | Vector ::insert | Vector ::get | Vector ::set |
| Arcs | 14 | 7 | 7 |
| Noeuds | 11 | 6 | 6 |
| Conditions | 4 | 2 | 2 |

Tableau 1 : Decription des arcs, nœuds et conditions de chaque fonctions

|  |  |  |  |
| --- | --- | --- | --- |
| Fonctions | Stack ::pop | Stack :: top | Stack :: push |
| Arcs | 5 | 5 | 2 |
| Noeuds | 5 | 5 | 3 |
| Conditions | 1 | 1 | 0 |

Tableau 2 : Decription des arcs, nœuds et conditions de chaque fonctions

**Méthode 1 de calcul de la complexité cyclomatique :**

Formule de la méthode 1 du calcul de la complexité cyclomatique :

Complexité\_cyclomatique = nombre\_d’arcs – nombre\_noeuds +2

|  |  |  |  |
| --- | --- | --- | --- |
| Fonctions | Vector :: insert | Vector ::get | Vector :: set |
| Complexité cyclomatique | 5 | 3 | 3 |

Tableau 3 : Méthode 1 de calcul de la complexité cyclomatique pour chaque fonction

|  |  |  |  |
| --- | --- | --- | --- |
| Fonctions | Stack :: pop | Stack ::top | Stack :: push |
| Complexité cyclomatique | 2 | 2 | 1 |

Tableau 4 : Méthode 1 de calcul de la complexité cyclomatique pour chaque fonction

**Méthode 2 de calcul de la complexité cyclomatique :**

Complexité\_cyclomatique = conditions+1

|  |  |  |  |
| --- | --- | --- | --- |
| Fonctions | Vector :: insert | Vector ::get | Vector :: set |
| Complexité cyclomatique | 5 | 3 | 3 |

Tableau 5 : Méthode 1 de calcul de la complexité cyclomatique pour chaque fonction

|  |  |  |  |
| --- | --- | --- | --- |
| Fonctions | Stack :: pop | Stack ::top | Stack :: push |
| Complexité cyclomatique | 2 | 2 | 1 |

Tableau 6 : Méthode 1 de calcul de la complexité cyclomatique pour chaque fonction

E2

Vector::insert

|  |  |  |
| --- | --- | --- |
| Chemin | Entrées | Sortie |
| Jaune | index = 1  index > count | true |
| Orange | Index = 1  Index < count | false |
| Rose | Index = 0  Index > count | false |
| Rouge | Index = 0  Index < count  (i>index) | true |
| Violet | Index = 0  Index < count  (i < index) | true |

Vector ::get

|  |  |  |
| --- | --- | --- |
| Chemin | Entrées | Sortie |
| Jaune | Index = -1  Index <= count | Elements[index] |
| Orange | Index = 0  Index <= count | Elements[index] |
| Rose | Index = 0  Index >= count | Elements[index] |

Vector ::set

|  |  |  |
| --- | --- | --- |
| Chemin | Entrées | Sortie |
| Jaune | Index = -1  Index <= count | false |
| Orange | Index = 0  Index >= count | false |
| Rose | Index = 0  Index <= count | true |

Stack ::pop

|  |  |  |
| --- | --- | --- |
| Chemin | Entrées | Sortie |
| Jaune | isEmpty = true | top |
| Orange | isEmpty = false | top |

Stack ::top

|  |  |  |
| --- | --- | --- |
| Chemin | Entrées | Sortie |
| Jaune | isEmpty = true | Elements.get(elements.size()-1) |
| Orange | isEmpty = false | Elements.get(elements.size()-1) |

Stack ::push

|  |  |  |
| --- | --- | --- |
| Chemin | Entrées | Sortie |
| Jaune | Value=test | Elements.add(value) |

E3

À compléter

E4

Nous avons choisis le fichier utf8\_utils.cpp et la méthode utf8\_make\_valid.

1. A screenshot of a cell phone

   Description generated with very high confidenceVoici à droite le diagramme de flot de contrôle de la méthode utf8\_make\_valid
2. Voici le calcul de la complexité cyclomatique avec les deux approches vues en classe

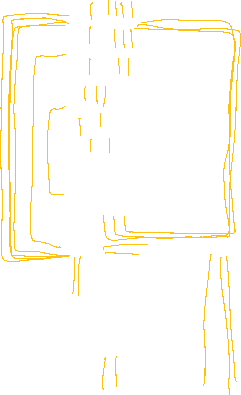
Méthode 1 : Complexité\_cyclomatique = nombre\_d’arcs – nombre\_noeuds +2

Méthode 1 : 13 -10 +2 =5

Méthode 2 : Complexité cyclomatique = conditions+1

Méthode 2 : 4+1 =5

1. Voir à gauche



1. Non, tous les tests ne sont pas implémentés. Nous avons déterminé que cette fonction possédait une complexité cyclomatique de 5, ce qui signifie que cette fonction nécessite au moins 5 tests unitaires afin de vérifier adéquatement chacune des situations dans laquelle la fonction peut se retrouver. Les deux tests implémentés correspondent à un test d’une chaine de caractères « valide » pour le premier test, et une chaine de caractères « non-valide » pour le deuxième test. Il aurait fallu au moins des tests ayant comme entrée une chaine de caractères de différentes longueurs et possédant des caractères différents pour vérifier si chaque cas permet de rendre valide une chaine de caractères en utf8.