Travail pratique #4: Tests Unitaires

École polytechnique de Montréal

Trimestre : Hiver 2020

Équipier1: *William Younanian*
Équipier2: *Vincent Samuel-Lafleur*

Équipe: 411

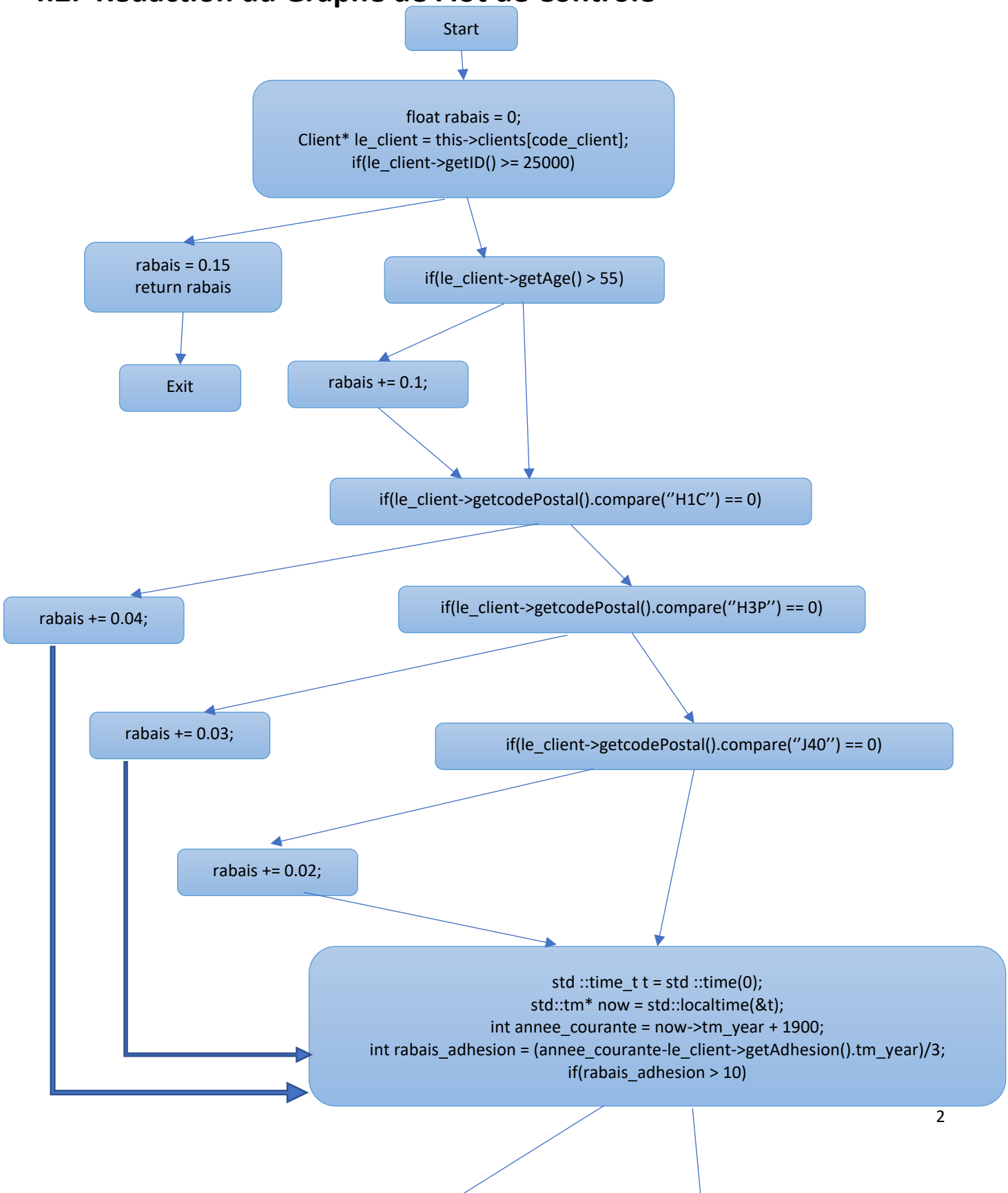
Présenté à : Annie Rochette

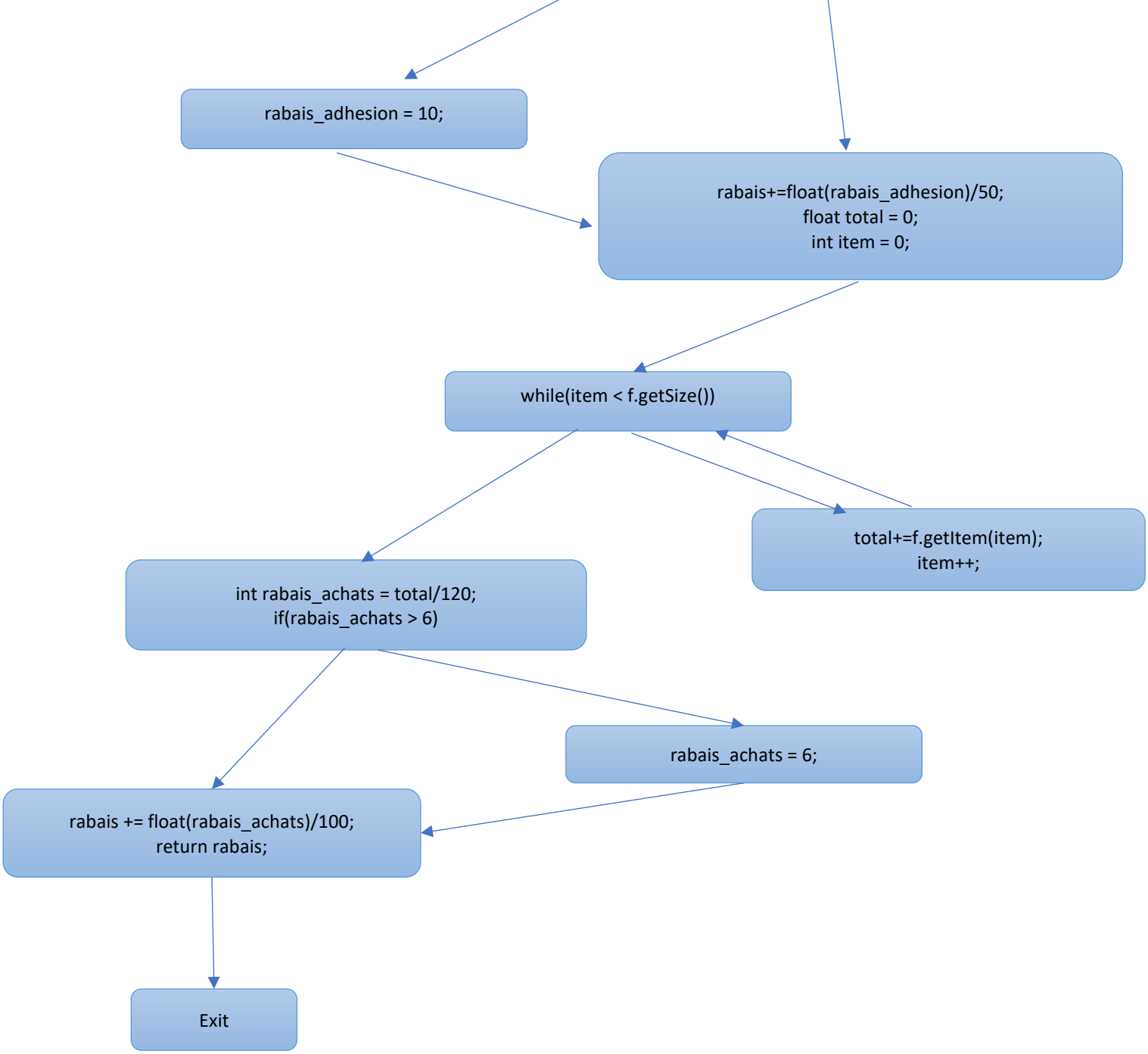
Polytechnique Montréal
Remis le 28 mars 2020

4.1. Exécution d’un test unitaire

Nous avons testé la fonctionnalité du rabais employé (d1).

4.2. Rédaction du Graphe de Flot de Contrôle




rabais_adhesion = 10;
rabais+=float(rabais_adhesion)/50;
float total = 0;
int item = 0;
while(item < f.getSize())
total+=f.getItem(item);
item++;
int rabais_achats = total/120;
if(rabais_achats > 6)
rabais_achats = 6;
rabais += float(rabais_achats)/100;
return rabais;
Exit

4.3. Rédaction des cas de tests pour la couverture des branches

Exigences :

ID	Exigence
SRS01	Les code-clients qui sont plus grands que 25 000 correspondent à des codes d'employé(e)s.
SRS02	Les employé(e)s doivent avoir accès à un rabais d'employé(e)s de 15%.
SRS03	Les employé(e)s n'ont pas accès à aucun autre rabais que le rabais d'employé(e)s.
SRS04	Les personnes de plus de 55 ans doivent avoir un rabais additionnel de 10%.
SRS05	Un rabais additionnel doit être attribué pour les zones géographiques suivantes, basées sur les trois premiers symboles du code postal. • Zone H1C : 4% additionnel, • Zone H3P : 3% additionnel, • Zone J4O : 2% additionnel.
SRS06	Un rabais additionnel doit être attribué par nombre d'année depuis la date d'adhésion de 2% par 3 années complètes.
SRS07	Le rabais additionnel basé sur la date d'adhésion ne peut pas dépasser 10%.
SRS08	Un rabais additionnel doit être attribué de 1% par tranche de 120$ sur la facture.
SRS09	Le rabais additionnel basé sur la facture ne peut pas dépasser 6%.

Cas de tests :

SRS01, SRS02, SRS03:
#25001, employé de 60 ans (code client > 25 000), devrait avoir son rabais d'employé de 15% uniquement même si il est âgé de plus que 55 ans
d1 = <{ Facture = [total = 0, Client = [25001, "Tremblay", "Huguette", 60, "G4S", 2020]] , code_client = 25001}, {rabais = 0.15}>

SRS04, SRS05:
#10000, 56 ans et vivant dans le H1C, devrait avoir 14% de rabais.
d2 = <{Facture = [total = 0, Client = [10000, "Salinas", "Harold", 56, "H1C", 2020]], code_client = 10000}, {rabais = 0.14}>

SRS04, SRS06:
#11000, 60 ans et membre depuis 4 ans, devrait avoir 12% de rabais.
d3 = <{Facture = [total = 0, Client = [11000, “Denver”, “John”, 60, “P1D”, 2016]], code_client = 11000}, {rabais = 0.12}>

SRS05:
#12000, habitant du H3P, devrait avoir 3% de rabais.
d4 = <{Facture = [total = 0, Client = [12000, “Arruda”, “Horacio”, 43, “H3P”, 2020]], code_client = 12000}, {rabais = 0.03}>

SRS05, SRS06:
#13000, membre depuis 12 ans et habitant du J4O, devrait avoir 10% de rabais.
d5 = <{Facture = [total = 0, Client = [13000, “Santos”, “Baltazar”, 30, “J4O”, 2008]], code_client = 13000}, {rabais = 0.10}>

SRS07:
#14000, membre depuis 20 ans, devrait avoir 10% de rabais.
d6 = <{Facture = [total = 0, Client = [14000, “Wayne”, “Bruce”, 14, “S3X”, 2000]], code_client = 14000}, {rabais = 0.10}>

→ *L’erreure vient de ce test. On doit avoir 10%, mais on a 12%.*

SRS08:
Pour une facture de 250$ et sans aucun autre rabais, le rabais devrait être de 2%.
d7 = <{Facture = [total = 250, Client = [15000, “Fleck”, “Arthur”, 10, “P0P”, 2020]], code_client = 15000}, {rabais = 0.02}>

SRS09:
Pour une facture de 1000$ et sans aucun autre rabais, le rabais devrait être de 6%.
d8 = <{Facture = [total = 0, Client = [16000, “Kjelberg”, “Felix”, 9, “T2S”, 2020]], rabais = 16000}, {code_client = 0.06}>

SRS04, SRS09:
Pour une facture de 1000$ et #17000 âgé de 60 ans, le rabais devrait être de 16%.
d9 = <{Facture = [total = 1000, Client = [17000, “Apple”, “Tim”, 60, “P2Q”, 2020]], code_client = 17000}, {rabais = 0.16}>

Les cas de test décrits ci-dessus couvrent bien toutes les branches de la fonction getRabais et permettent donc d’assurer que celle-ci répond aux exigences. Certaines exigences ont été regroupées puisqu’il était logique et facile de produire un seul test les couvrant toutes (comme c’est le cas pour le test d1). De plus, nous avons jugé pertinent d’élaborer certains tests faisant interagir les différents paramètres de la fonction. Par exemple, le test d9 s’assure que le rabais est bien calculé lorsqu’il inclut à la fois un rabais client et un rabais facture.

4.4. Code des cas de tests dans le projet CppUnit

Voir le code

4.5. Recherche d'un défaut dans le code source

Défaut :

L'exigence SRS07 n'est pas respectée. En effet, pour un membre depuis 20 ans, le rabais additionnel doit être de 10%. Par contre, ce qu'on voit est un rabais de 12%. Correction suggérée:
Aux lignes 109 et 111 de rabais.cpp, il faudrait changer:
109: int rabais_adhesion = (annee_courante - le_client->getAdhesion().tm_year)/3;
111: rabais += float(rabais_adhesion) / 50;

Correction potentielle :

109: int rabais_adhesion = 2*(annee_courante - le_client->getAdhesion().tm_year)/3;
111: rabais += float(rabais_adhesion) / 100;

5. Rétroaction :

On a passé environ 12 heures/personnes sur ce travail. La moitié de ce temps a été passée sur le fonctionnement du Makefile.