Validation d'algorithmes
L3 MIAGE, UGA
2020/2021

Raphaël Charrondière
raphael.charrondiere@inria.fr
Semaine 5 - échéance S6

Feuille de TD 3: Graphes de flot de contrôle

Où l'on commence à manipuler les GFC.

```
read(i);
j ← 0;
a ← 0;
b ← 1;
tant que j < i faire
    b ← a + b;
    a ← b − a;
    i ← i + 1;
fin
retourner a;
```

Algorithme 1 : Fibonacci

```
productID ← readInput();
dureeVie ← getExpiration(productID)-clock();
dureeConservation ← getDureeConservation(productID);
stock ← getStock(productID);
reduction← 0;
si dureeVie<3 jours alors reduction ← 10;
si dureeVie<2 jours alors reduction ← 40;
si dureeVie<10 jours ET dureeConservation>30 jours alors
  reduction ← 50;
si stock<10 alors reduction ← 0;
si dureeVie<1 jour alors reduction ← 70;
retourner reduction;
```

Algorithme 2 : Réduction

Exercice 1 Fibonacci

La suite de Fibonacci se définit de la manière suivante : $\mathcal{F}_0 = 0$, $\mathcal{F}_1 = 1$ et $\forall i > 1,\ \mathcal{F}_i = \mathcal{F}_{i-1} + \mathcal{F}_{i-2}$. Les premiers termes de la suite sont $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89$. On propose le programme suivant pour calculer l'i-ème terme :

1. Vérifier que l'algorithme 1 renvoie bien ce qu'il doit renvoyer. *Vous avez vu comment faire dans le cours et les TD précédents, non ?*
2. Donner le graphe de flot de contrôle de l'algorithme.
3. Donner les chemins (sous forme algébrique) de ce graphe.
4. Proposer des données de test pour couvrir le critère « tous les nœuds ».
5. Proposer des données de test pour couvrir le critère « tous les arcs ».

Exercice 2 Les arcs sont des nœuds

Cet exercice a été fait en cours, mais il est bon de le refaire pour valider ses acquis.

1. Montrer qu'une couverture par arcs implique une couverture par nœud.
2. Montrer que l'inverse n'est pas vrai, mais donner une technique pour le rendre vrai.

Exercice 3 Enchaîner les ifs

Un magasin veut inciter les clients à acheter les produits alimentaire en fin de vie en premier pour éviter les pertes. Il baisse alors les prix suivants plusieurs critères, en considérant la date de péremption et le stock disponible. Si plusieurs critères sont applicables, le dernier critère de la liste est appliqué

C1 : Si la date de péremption est dans les trois jours, le produit reçoit une réduction de 10%

C2 : Si la date de péremption est dans les deux jours, le produit reçoit une réduction de 40%

C3 : Si la date de péremption est dans les dix jours et qu'il s'agisse d'un article à longue conservation, le produit reçoit une réduction de 50%

C4 : Si le stock est faible le produit n'est pas diminué

C5 : Si la date de péremption est le jour même, le produit reçoit une réduction de 70%

Le magasin veut coder un programme pour afficher la réduction à appliquer. Prenons la solution la plus simple, l'algorithme 2.

Raphaël Charrondière
raphael.charrondiere@inria.fr

1. Donner le graphe de flot de contrôle.
2. Donner des données de test pour couvrir le critère "tous les arcs"
3. Écrire l'expression algébrique des chemins.
4. Pour chaque chemin donner une donnée de test (on supposera que `clock()` renvoie le 1/1/2021). *Indice* : Regardez la question suivante.
5. Quel problème y a-t-il à décorréler la structure du code ?
6. Pour pouvoir exploiter un critère « tous les chemins », il faut réécrire le code. Proposez une nouvelle version du code. On demande à ce que l'instruction `retourner` soit utilisée une seule fois.
7. Donner le graphe de flot de contrôle du nouveau code, puis les données de test pour couvrir le critère « tous les chemins ».

Exercice 4 — Extension des graphes de flot de contrôle

```
suivant T faire
    cas où T = 1 faire
        instructionA;
    fin
    cas où T = 2 faire
        instructionB;
    fin
    cas où T = 3 faire
        instructionC;
    fin
    autres cas faire
        Cas par défaut
    fin
fin
```

Algorithme 3 : Switch 1

```
suivant T faire
    cas où T = 1 faire
        instructionA;
    cas où T = 2 faire
        instructionB;
    cas où T = 3 faire
        instructionC;
    fin
    cas où T = 4 faire
        instructionD;
    autres cas faire
        Cas par défaut
    fin
fin
```

Algorithme 4 : Switch 2

Vous avez certainement déjà vu l'instruction `switch` (par exemple en java) : suivant la valeur d'une variable T l'exécution du programme passe à une étiquette, ensuite l'exécution se poursuit jusqu'à la fin du `switch` sauf si l'instruction `break` est rencontrée. Dans cet exercice, nous allons ajouter un nouveau type de nœud au graphe de flot de contrôle pour gérer cette instruction.

1. Dans l'algorithme `Switch 1`, selon la valeur de T, une seule instruction est exécutée (il y a un `fin` après l'instruction). Créez le graphe correspondant, et donnez les propriétés du nouveau nœud.
2. L'algorithme `Switch 2` est une variante où une instruction peut être atteinte par plusieurs cas. Par exemple si T vaut 1 les instructions A, B et C sont exécutées. Créez le graphe correspondant.