Master Langue et Informatique
Université Paris-Sorbonne

MASTER LANGUE ET INFORMATIQUE

Travaux Pratiques n° 4
Structures de contrôle

Les structures de contrôle modifient l'ordre séquentiel d'exécution des instructions d'un programme. Elles permettent de traduire un algorithme mathématique en un programme informatique.

1. BRANCHEMENT CONDITIONNEL

Le branchement conditionnel permet d'exécuter des traitements selon certaines conditions (alternatives de traitements). Sa syntaxe est: if (<condition>) <blocV> [else <blocF>] avec (<condition>) une expression logique.

Soit le programme (disponible sur Git)

```
public static void main(String[] args) {
        int p1 = 10, p2 = 9, p3 = -1, p4 = 4, p5 = 8;
        int p4;
        …
        System.out.println(p4) ;
    }
```

Exercice 1 : Ajouter les instructions structurées par des branchements conditionnels permettant d'obtenir le maximum des variables p1, p2, p3 dans la variable p4.

Exercice 2 : Ajouter les instructions structurées par des branchements conditionnels permettant d'obtenir le minimum des variables p1, p2, p3 dans la variable p4.

Exercice 3 : Ajouter les instructions structurées par des branchements conditionnels permettant d'obtenir la médiane des variables p1, p2, p3 dans la variable p4.

Soit le programme (disponible sur Git)

```
public static void main(String[] args) {
        int p1 = 10, p2 = 9, p3 = -1, p4 = 4, p5 = 8;
        int p6;
        …
        System.out.println(p6) ;
    }
```

Exercice 4 : Ajouter les instructions structurées par des branchements conditionnels permettant d'obtenir la médiane des variables p1, p2, p3, p4, p5 dans la variable p6.

2. BOUCLE CONDITIONNELLE

La boucle conditionnelle permet, avec un contrôle a posteriori ou a priori et par une condition de continuation, de répéter un traitement. Sa syntaxe est :

Boucle a posteriori : do <blocB> while (<condition>);

Boucle a priori : while (<condition>) <blocB>;

Soit le programme (disponible sur Git)

```
public static void main(String[] args) {
        String s1 = "azerty", s2 = "ytreza";
        int i1 = 0, i2 = s2.length()-1;
        char c1, c2; boolean flag;
        …
        c1 = s1.charAt(i1); c2 = s2.charAt(i2);
        … }
```

Exercice 1 : Ajouter les instructions structurées par une boucle conditionnelle a posteriori permettant d'obtenir true dans la variable flag si s1 et s2 sont deux mots miroirs, false sinon

Exercice 2 : Ajouter les instructions structurées par une boucle conditionnelle a priori permettant d'obtenir true dans la variable flag si s1 et s2 sont deux mots miroirs, false sinon

3. ITERATION

L'itération permet un traitement itératif contrôlée par un mécanisme de variables d'itération et de critère d'arrêt. Sa syntaxe est : for (<initialisation_s>; <condition>; <mise_s_à _jour>) <blocB>

Soit le programme (disponible sur Git)

```
public static void main(String[] args) {
        String[] prenoms = {"Christian", "Karen", "Victoria", "Agatha", "Philippe", "Djamé"};
        int imin;
        prenoms[0].compareTo(prenoms[1]);
}
```

Exercice 1 : Ajouter les instructions structurées par une itération permettant d'obtenir dans la variable imin l'indice du premier prénom dans l'ordre lexicographique.