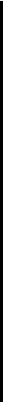
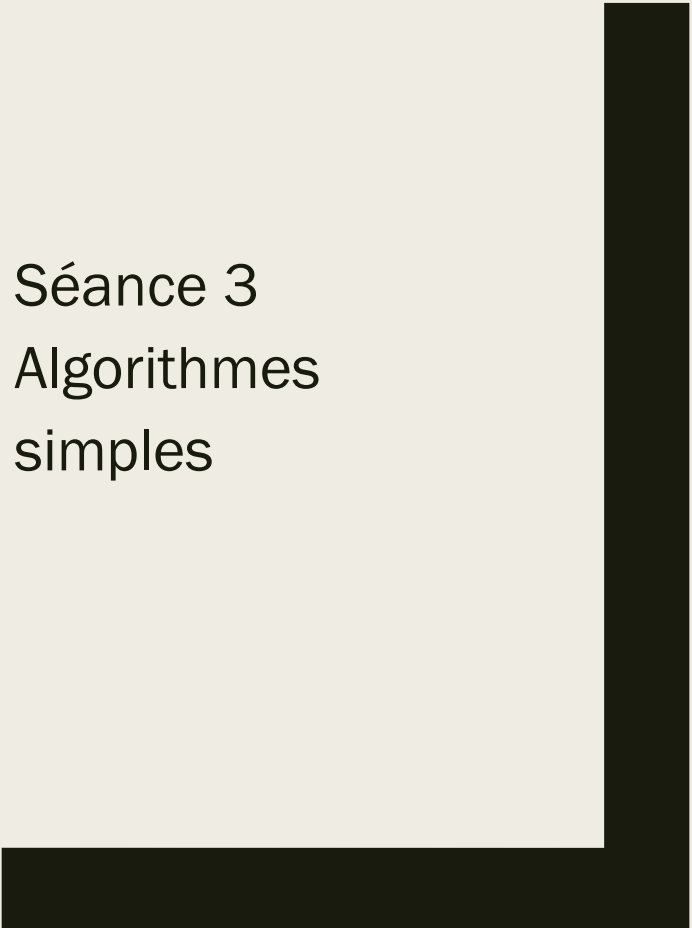




ALGORITHMIQUE



Séance 3
Algorithmes
simples



Camille SIMON – La Manu Le Havre

Bonnes pratiques

- Réfléchir au nombre de variables avant d'écrire
- Nombre d'action lire = nombre de données à saisir
- Eviter les imbrications de « Si »



Bonnes pratiques

■ Boucle « Pour »

- Ne pas initialiser la variable de compteur
- Ne pas incrémenter la variable de compteur
- Ne pas modifier la valeur des variables gérant la boucle
- Il n'est pas possible de sortir de la boucle outre la condition d'arrêt
- Il est possible d'utiliser la valeur du compteur dans la boucle
- La condition de la boucle ne peut pas dépendre du compteur

indice \leftarrow 5

Pour indice de indice à 10

Bonnes pratiques

■ Boucle « Pour »

- Ne pas initialiser la variable de compteur
- Ne pas incrémenter la variable de compteur
- Ne pas modifier la valeur des variables gérant la boucle
- Il n'est pas possible de sortir de la boucle outre la condition d'arrêt
- Il est possible d'utiliser la valeur du compteur dans la boucle
- La condition de la boucle ne peut pas dépendre du compteur

~~indice ← 5~~

~~Pour indice de indice à 10~~

Bonnes pratiques

- Boucle « Tant que »
 - Les variables de la condition doivent être initialisées
 - Les instructions doivent modifier les variables de la condition sinon la boucle est infinie
- Pour les boucles « Pour » et « Tant que »
 - Toute boucle « Pour » peut être écrite comme une boucle « Tant que »
 - Plus long et moins lisible
 - Lorsque les valeurs sont clairement définies, privilégier la boucle « Pour »

Exercices

Exercice 1

Ecrire un algorithme calculant factoriel N avec $N \leq 100$ et $N \geq 0$.

NB : La fonction « factoriel » est souvent écrite !, « factoriel N » s'écrit $N!$

Exercice 2

Ecrire un algorithme calculant la moyenne de notes. L'utilisateur devra saisir le nombre de notes ainsi que leurs valeurs.

Exercice 3

Ecrire un algorithme calculant la moyenne de notes. L'utilisateur saisie une suite de notes valides, la valeur 30 marquera la fin de la saisie.

Génération de valeurs aléatoires

- La fonction `AleatoireReel()` permet d'obtenir un nombre réel aléatoire compris entre 0 et 1 exclus ($[0;1[$).
- La fonction `AleatoireReel(<variable>)` donne un réel aléatoire compris entre 0 et `<variable> - 1` exclut.
- Exemple
 `AleatoireReel(5)` donne une valeur entre 0 et 4 avec 4 exclus.
- De même, `AleatoireEntier(<variable>)` donne une valeur aléatoire entière compris entre 0 et `<variable> - 1`.
- `AleatoireBooleen()` renvoie 0 ou 1.

Exercices - Aléatoire

Exercice 5

Ecrire un algorithme simulant le lancer d'un dé à 6 face.

Exercice 6

Modifier l'exercice précédent pour simuler le lancer d'un dé dont le nombre de faces est donné par l'utilisateur.

Exercice 7

Ecrire un algorithme qui simule un archer tirant sur une cible. On donne à l'algorithme un nombre d'essais et il doit dire pour chaque essaie si la flèche à atteint ou non la cible sachant que la probabilité de viser juste est de 92,8%.

Exemple :

Donner un nombre d'essais :

2

Cible touchée

Cible ratée

Exercices - Aléatoire

Exercice 8*

Simuler le comportement d'une personne qui a trop bu (le vrai terme est « marcheur aléatoire »). A chaque pas la personne choisit au hasard une direction (est, ouest, nord ou sud) et se déplace d'un pas dans cette direction. Autrement dit, si la personne est au point (x, y) , elle choisit parmi les points $(x-1, y)$, $(x+1, y)$, $(x, y-1)$ et $(x, y+1)$.

Simulez 1 000 pas et mesurez la distance entre le point de départ et le point final.

Exercice 9*

Modifiez le programme de l'exercice précédent de la façon suivante : demandez à l'utilisateur une distance d et l'algorithme s'arrête lorsque la distance parcourue entre le point de départ et d'arrivée du marcheur est plus grande que d .

Affichez le nombre de pas parcouru.