Introduction à l'algorithmique

-

La pensée logique du développement

Guillaume Graciet 2019

Aujourd'hui au programme…

Chapitre 1: Logique conditionnelle

- ✓ Les variables et les constantes
- ✓ Les conditions: principe général
- ✓ Opérateurs conditionnels
- ✓ Structures conditionnelles
- ❖ Atelier

Chapitre 2: Logique itérative

- ✓ Itération et répétions: principe général
- ✓ Les boucles
- ❖ Atelier

Chapitre 3: Fonctions et récursivité

- ✓ Les fonctions: l'art de l'algorithmique générique
- ❖ Atelier
- ✓ La récursivité
- ✓ Les tableaux
- ❖ Atelier

Chapitre 1:

Logique conditionnelle

Guillaume Graciet 2019

Les variables et les constantes

Une **variable** est un nom (ou suite de caractères) auquel est associé une **valeur**. Son contenu est modifiable après sa déclaration.

Une **constante** est identique à une variable, mais sa valeur est **non-modifiable** après sa déclaration.

```
Variable age = 30
Afficher age
age = 31
Afficher age
```

```
> 30
> 31
```

Exemple de variable

```
Constante taille = 180
Afficher taille
taille = 185
Afficher taille
```

```
> 180
> 180
```

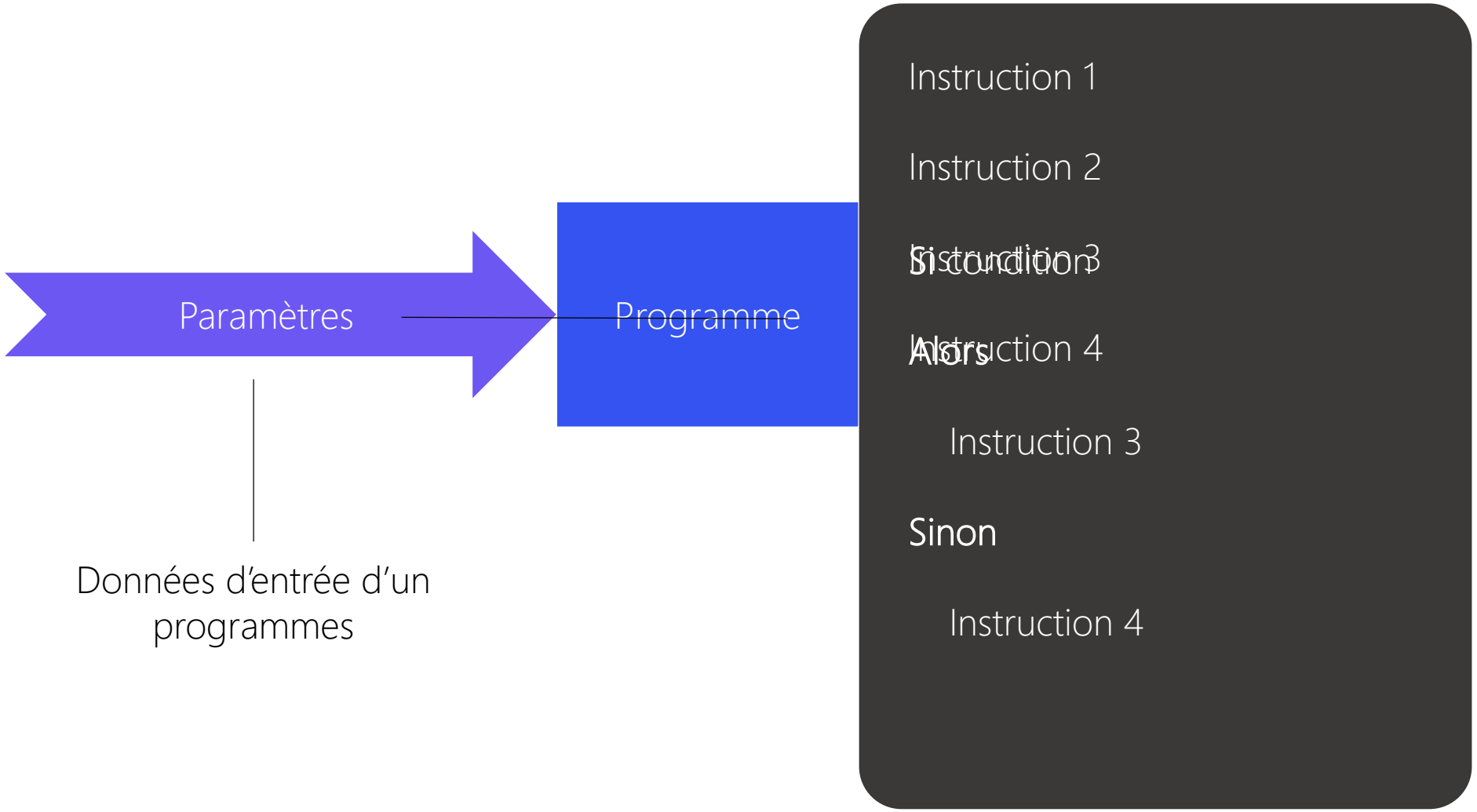
Exemple de constante

Les conditions: principe général

Données d'entrée d'un programmes

Affichage, données, fichiers exportés, etc…

Les conditions: principe général



Opérateurs conditionnels

Vérifier une condition revient à poser une question de comparaison dont la réponse est positive ou négative.

Inférieur à	Supérieur à	Inférieur ou égal à	Supérieur ou égal à	Egal à	Différent de
<	>	<=	>=	==	!=

Opérateurs de comparaison communs et leurs notations communes

```
Si  temperature <= 0
```

Exemple de condition

Opérateurs conditionnels

Il est fréquent d'associer plusieurs comparaisons au sein d'une même condition afin d'obtenir un résultat général.

Et	Ou	Priorité	Négation
&& and	|| or	()	! not

Opérateurs logiques communs et leurs notations communes

```
Si  temperature <= 0 && humidite > 80
```

Exemple de condition

Structures conditionnelles

Les différents opérateurs sont des outils permettant de d'écrire des structures correspondant à de nombreuses situations.

```
Si  temperature <= 0

Alors
    Afficher « Winter is coming »
```

Structure conditionnelle simple

```
Si  temperature <= 0

Alors
    Afficher « Winter is coming »

Sinon
    Afficher « Tout va bien »
```

Structure conditionnelle avec une alternative par défaut

Structures conditionnelles

```
Si  temperature <= 0

Alors
    Afficher « Winter is coming »

Sinon si temperature > 50
    Afficher « Ca devient vraiment
chaud par ici. »

Sinon
    Afficher « Tout va bien »
```

Structure conditionnelle avec plusieurs alternatives

```
Si  temperature <= 0

Alors
    Si manteau == « manteau chaud »

    Alors
        Afficher « Tout va bien »

    Sinon
        Afficher « On se gèle ici. »

Sinon
    Afficher « Tout va bien »
```

Structure conditionnelle avec plusieurs alternatives

Atelier 1/2

Ecrivez **deux algorithmes** de régulation pour un Casino en fonction des variables et constantes. Les clients peuvent: Entrer, Jouer et Consommer au bar.

Les valeurs à saisir par l'utilisateur sont notées **[L]**, les autres seront à initialiser arbitrairement au fil des essais.

Constantes

- Année de naissance du client **[L]**
- Interdit de jeu (« vrai » ou « faux »)

Variables

- Solde du compte client
- Action du client (« Entrer », « Jouer », « Consommer ») **[L]**
- Choix de la boisson (« Alcoolisée », « Soft ») **[L]**

Atelier 2/2

Règles pour entrer (Algorithme 1)

- Aucune limite d'âge
- Il ne faut pas être interdit de jeu pour entrer
- Le solde du compte client doit être supérieur à – 8000$

Règles dans le Casino (Algorithme 2)

- Aucune limite d'âge pour jouer
- Il faut être majeur pour gagner (+18 ans)
- Il faut avoir plus de 21 ans pour commander une boisson alcoolisée
- Il faut avoir un solde supérieur à – 8000$ pour jouer
- Il est interdit de consommer à crédit au bar
- Une boisson alcoolisée coûte 8$
- boisson « soft » coûte 5$

Actions possibles (Fonctions)

- Autoriser_entree
- Refuser_entree
- Autoriser_jeu
- Refuser_jeu

- Servir_boisson
- Refuser_boisson
- Afficher_au_client ''*Texte*"
- Refuser_jeu

- Lire_entree_utilisateur

Chapitre 2:

Logique itérative

Guillaume Graciet 2019

Les boucles: principe général

Une **boucle** permet la **répétition** d'une même série d'instructions selon une **condition**, on parle d'**itération** pour décrire cette répétition.

```
Variable age = 30

Afficher age
age = 31
Afficher age
age = 32
Afficher age
age = 33
Afficher age
age = 34
Afficher age
```

```
> 30
> 31
> 32
> 33
> 34
```

Exemple sans boucle

```
Variable age = 30

Tant que age < 35
    Afficher age
    age = age + 1
Fin tant que
```

```
> 30
> 31
> 32
> 33
> 34
```

Exemple avec boucle

Les boucles

Boucle « Tant que »	Boucle « Pour »
Tant que CONDITION … Fin tant que	Pour VARIABLE = VALEUR tant que CONDITION effectuer OPERATION … Fin pour

Boucles communes

```
Variable age = 30

Tant que age < 35
    Afficher age
    age = age + 1
Fin tant que
```

Exemple de boucle « Tant que »

```
Pour age = 30 tant que age < 35 effectuer age =age + 1
    Afficher age
Fin pour
```

Exemple de boucle « Pour »

Atelier

Sur la base des deux algorithmes précédents, implémenter des boucles pour créer un seul grand algorithme permettant au client d'effectuer les actions précédentes à volonté, tout en effectuant toujours les vérifications adéquates.

Pour terminer l'algorithme, l'utilisateur (le client) doit pourvoir saisir un nouveau choix d'action: « Sortir ».

Chapitre 3:

Fonctions et récursivité

Guillaume Graciet 2019

Les fonctions: l'art de l'algorithmique générique

Une **fonction**, également appelée **sous-programme** en algorithmique, est un **groupe d'instructions** acceptant des **paramètres** et donnant un **résultat**. C'est une sorte de programme dans le programme.

Une **fonction** permet de réutiliser un morceau de code sans avoir à le réécrire.

```
Fonction verifier_age (age)
    Si age >= 18
    Alors
        Retourner 'adulte'
    Sinon
        Retourner 'enfant'
Fin fonction
```

Exemple de fonction

Atelier

Réorganisez l'algorithme précédent en fonctions pour obtenir une résultat plus organisé.

La récursivité

On appelle **récursive** une fonction s'appelant elle-même. Un tel procédé permet la résolution de situation nécessitant l'exécution successive d'une même tâche plusieurs fois, il s'agit parfois d'une alternative l'itération.

On l'utilise notamment pour effectuer des opération comme le tri.

```
Fonction factorielle (n)
    Si n == 0
    Alors
       Retourner 1
    Sinon
      Retourner n * factorielle(n-1)
Fin fonction
```

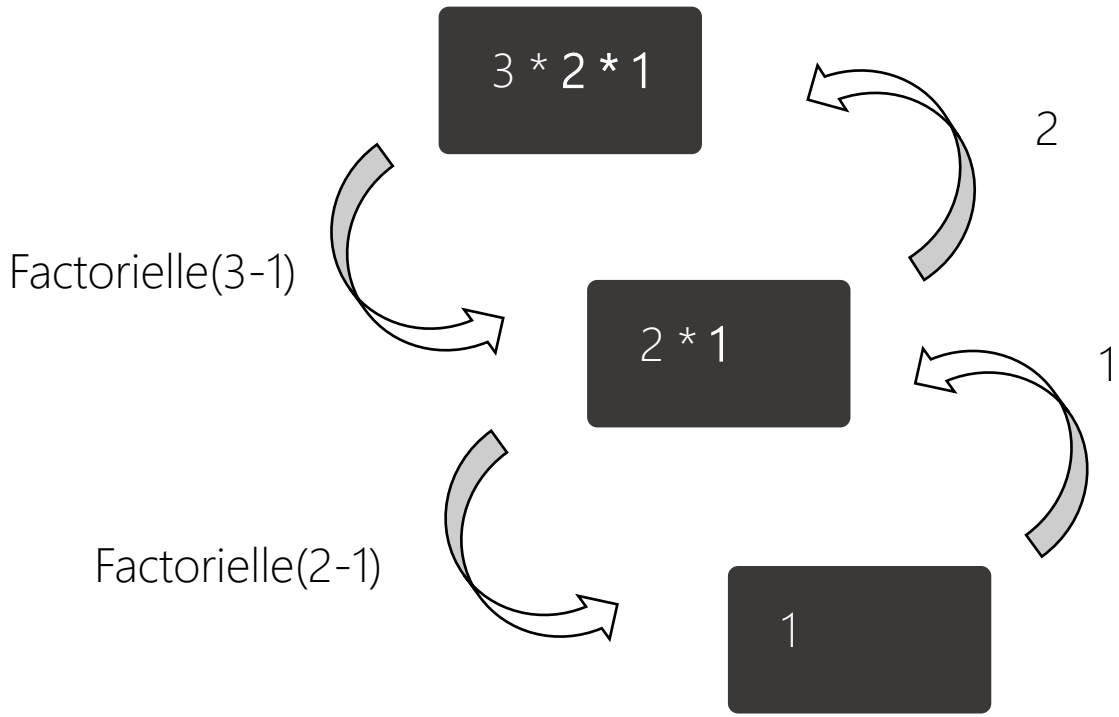
Exemple de récursivité

La récursivité

```
Fonction factorielle (n)
    Si n == 1 || n == 0
    Alors
        Retourner 1
    Sinon
        Retourner n * factorielle(n-1)
Fin fonction


factorielle(3)
```

>6



Exemple de récursivité

Exécution

Les tableaux

Une **tableau** ou une **liste** est une variable contenant une **série de plusieurs valeurs**.

```
Variable liste_des_courses = [‘pain’, ‘eau’, ‘vin’, ‘riz’]

Variable numéro_du_loto = [2, 42, 38, 25]
```

Exemple de tableaux

Atelier

Imaginez deux algorithmes permettant de trier la série de nombre ci-après.

Le premier algorithme doit utiliser une structure itérative, basée sur une boucle.

Le second algorithme doit utiliser une structure récursive, basée sur une fonction s'appelant elle-même.

```
Variable liste = [2, 42, 38, 25, 3, 7, 20, 32, 45, 4]
```

Liste à trier

Fin de la partie, à suivre…