Programmation Algorithmique

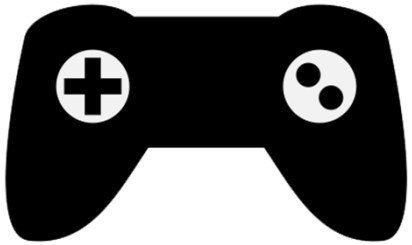
Sommaire

Les Bases de la Programmation

- La programmation, c'est quoi ?
- Faire son premier programme
- Partagez votre programme
- Histoire des langages informatiques
- Notion d'algorithmie
- Exercices

La programmation, c'est quoi ?

- https://fr.wikipedia.org/wiki/Programmation_informatique
- Activités permettant d'écrire des programmes informatiques
- Permet de développer des logiciels
- Utilise un langage de programmation
- Programmation dits impératifs
- Les programmes sont partout !



Faire son premier programme

- Créer votre recette de cuisine pour obtenir un gâteau au chocolat

```
Debut Program

nombre nbOeufs = 3
nombre qtFarine = 50

nombre resultat = nbOeufs + qtFarine

afficher resultat

Fin Program
```

Le gâteau au Chocolat

200 g de chocolat à pâtisser noir

100 g de beurre + une noix pour le moule

3 oeufs

50 g de farine

100 g de sucre en poudre

Recette :

Préchauffez votre four à 180°C (thermostat 6).

Dans une casserole, faites fondre le chocolat et le beurre coupé en morceaux à feu très doux.

Dans un saladier, ajoutez le sucre, les oeufs, la farine. Mélangez.

Ajoutez le mélange chocolat/beurre. Mélangez bien.

Beurrez et farinez votre moule puis y versez la pâte à gâteau.

Faites cuire au four environ 20 minutes.

A la sortie du four le gâteau ne paraît pas assez cuit. C'est normal, laissez-le refroidir puis démoulez- le.

Faire son premier programme

```
Debut Program

nombre nbOeufs = 3
nombre qtFarine = 50

nombre resultat = nbOeufs + qtFarine

afficher resultat

Fin Program
```

Plusieurs ingrédients

Le gâteau au Chocolat

Faire son premier programme

```
Debut Program

nombre nbOeufs = 3
nombre qtFarine = 50

nombre resultat = nbOeufs + qtFarine

afficher resultat

Fin Program
```

Suite d'instructions

Recette :

Préchauffez votre four à 180°C (thermostat 6).

Dans une casserole, faites fondre le chocolat et le beurre coupé en morceaux à feu très doux.

Dans un saladier, ajoutez le sucre, les oeufs, la farine. Mélangez.

Ajoutez le mélange chocolat/beurre. Mélangez bien.

Beurrez et farinez votre moule puis y versez la pâte à gâteau.

Faites cuire au four environ 20 minutes.

A la sortie du four le gâteau ne paraît pas assez cuit. C'est normal, laissez-le refroidir puis démoulez- le.

Partager un Programme

- Programme compiler / interpréter
- Programme Multiplateforme
- Dev / Prod
- Installer un programme
- Héberger un site internet / une application Web

Histoire des langages informatiques

- 1957 : Fortran, Lisp, COBOL
- 1964 : Basic
- 1970 : Pascal
- 1972 : C
- 1983 : C++, Objectiv C
- 1987 : Perl
- 1991 : Python
- 1993 : Ruby
- 1995 : Java, PHP, JavaScript
- 2000 ; C#

Notion d'algorithmie

- Pseudo-code I : Définition

Permet de décrire facilement un algorithme à l'aide d'un vocabulaire simple, en dehors de tous langages de programmation

Notion d'algorithmie

- Pseudo-code II : Syntaxe

Les mots clés (en majuscule)

- DEBUT, FIN, SI, ALORS, SINON, POUR, TANT QUE, JUSQU'À,

  ENTIER, CARACTERE, CHAINE, NON, OU, ET …

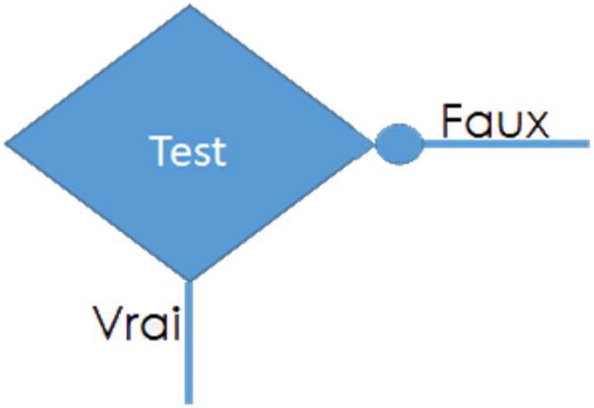
Les symboles :

- ← : l'affectation
- =, <>, <=, >=, = : égal, différent, inférieur…
- +, -, *, / : addition, soustraction, multiplication, division

Notion d'algorithmie

- L'algorigramme I : Symboles

Délimite le schéma

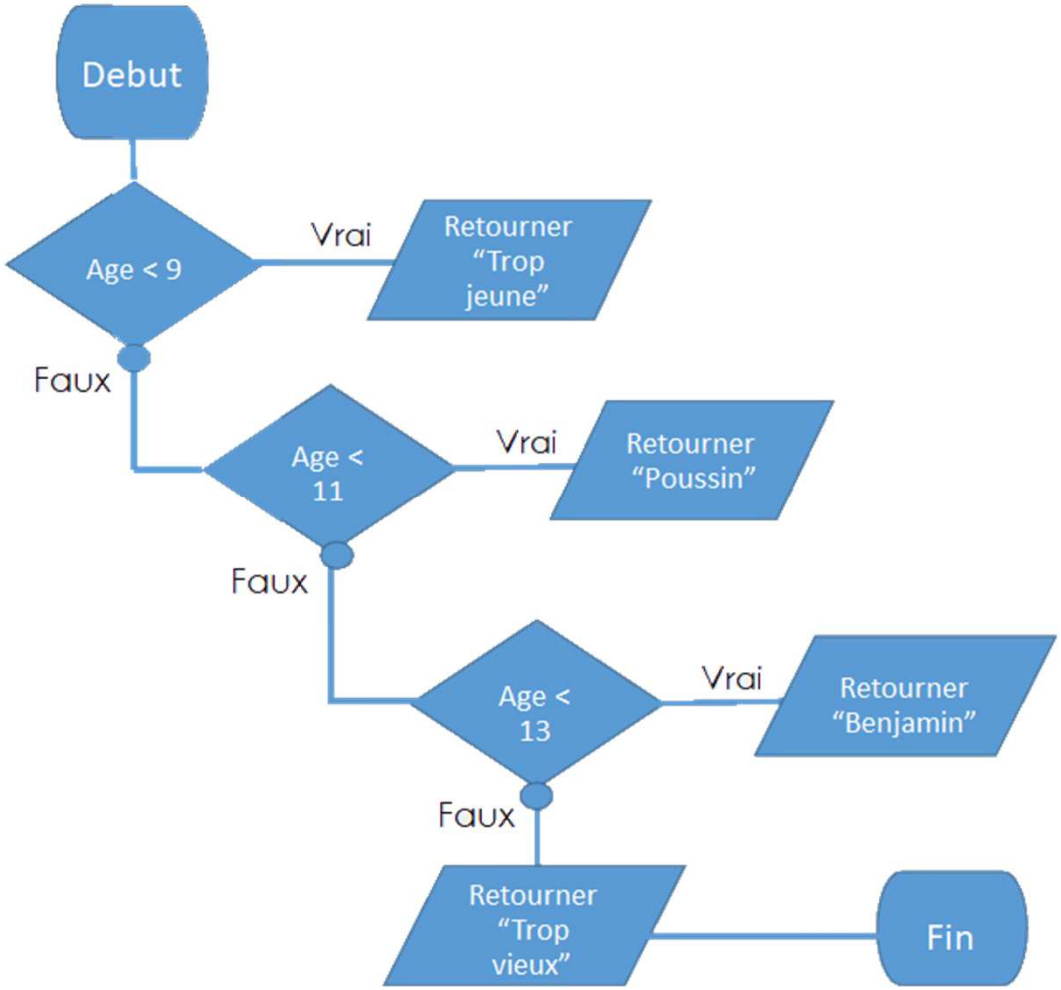


Les branchements conditionnels

Traitement de la donnée

Notion d'algorithmie

- L'algorigramme II : Exemple



Notion d'algorithmie

- Les variables

- Deux types :

- ***variable*** : sa valeur peut changer : données élémentaires

ex : nom_client

- ***constante*** : sa valeur ne changera jamais : paramètre

ex : taux_TVA

- Le nom des données doit être

- simple
- parlant, efficace
- sans accent, sans espace ni caractères spéciaux
- CamelCase

Notion d'algorithmie

Les variables

- Les types de données

- Entier (ex : 12 ; 100 ; 27 ; 658 ...)
- Réel (ex : 6,543 ; 0,007 ; 65,5874 ...)
- Chaîne - de caractères - (ex : Bonjour)
- Booléen (OUI / NON ; VRAI / FAUX ...)
- Date …

- La déclaration des variables et des constantes

Exemple :

var nomClient : Chaîne

var ageClient : Entier

const TAUX_TVA <- 0.196

Notion d'algorithmie

Les structures de programmes

- Programme et séquence d'instructions

```
ALGO NomDeAlgo
VAR
        .......
CONST
        ......
DEBUT
        Initialisation des variables
        Corps du programme
        ........
FIN
```

Notion d'algorithmie

Les structures de programmes

- Affectation

Consiste à donner une valeur à une variable Syntaxe :

nomVariable <- Expression

nomVariable <- Valeur

nomVariable <- nomVariable

Exemple :

nom <- "Dupont"

prixHT <- 10

prixHT <- prixHT + 10

Notion d'algorithmie

Les structures de programmes

- Les opérations d'entrées / sorties (clavier et écran)

Saisir (entrée)

Syntaxe :

Saisir (nomVariable)

Exemple :

Saisir (nom)

Cela se traduit par l'affichage d'une boite de dialogue dans laquelle on saisira le nom qui sera affecté à la variable nom

Notion d'algorithmie

- Les structures de programmes

Afficher (sortie)

Syntaxe :

Afficher ("Texte")

Afficher (nomVariable)

Exemple :

Afficher ("Bonjour, il est 9h30") cela affichera : Bonjour, il est 9h30

Afficher (prixHT) cela affichera la valeur de prixHT

Afficher ("Bonjour " & nomEleve) cela affichera : Bonjour suivi de la valeur de nomEleve

Notion d'algorithmie

Les structures de programmes

- Les structures conditionnelles

Le SI

Son intérêt est de permettre de réaliser des choix dans un programme. Faire telle chose SI telle condition est réalisée, SINON faire telle autre chose etc...

Syntaxe :

```
SI condition ALORS
        Suite d'opérations
SINONSI condition2 ALORS // facultatif
        Suite d'opérations
…
SINONSI condition3 ALORS // facultatif
        Suite d'opérations
SINON // facultatif
        Suite d'opérations
FINSI
```

Notion d'algorithmie

Les structures de programmes

- Les structures conditionnelles

Le SI

Syntaxe de la condition :

<NomVariable> <Opérateur_de_comparaison> <Expression>

Exemple de condition

b < 36

Liste des opérateurs = ; < ; > ; <= ; >= ; <>

Remarque : plusieurs tests peuvent être réalisés simultanément en utilisant le ET ou le OU.

Exemple :

(b > 10) ET (b < 36)

Notion d'algorithmie

Les structures de programmes

- Les structures conditionnelles

Le SELON CAS

Le SI, c'est bien mais quand il y a beaucoup de conditions, le SELON CAS c'est mieux. En effet, il permet de ne pas imbriquer de nombreux SI et d'indiquer qu'une seule fois le nom de la variable testée.

Syntaxe :

```
SELON CAS nomVariable
        Cas valeur1 : instructions // à la ligne si plusieurs lignes d'instructions
        Cas valeur2 : instructions
        …
        Cas valeurN : instructions
        AUTRES CAS // facultatif
                Instructions
FIN SELON CAS
```

Notion d'algorithmie

Les structures de programmes

- Les structures conditionnelles

Le SELON CAS

Exemple :

```
SELON CAS paysISO
        Cas "FR"
                Afficher ("France")
        Cas "ES"
                Afficher ("Espagne")
        Cas "DE"
                Afficher ("Allemgne")
        AUTRES CAS
                Afficher ("Pays Inconnu")
FIN SELON CAS
```

Notion d'algorithmie

Les structures de programmes

- Les structures itératives

Certains traitements nécessitent la répétition d'opérations. Une itération est un ensemble d'actions qui sont répétées, dans un ordre bien défini, un certain nombre de fois précisé à l'avance (ou avec un mécanisme permettant de l'arrêter). On parle plus couramment de BOUCLES.

La boucle **REPETER ... JUSQU'A**

Syntaxe :

```
REPETER
        Suite d'instructions
        ....
JUSQU'A Condition
```

Notion d'algorithmie

Les structures de programmes

- Les structures itératives

La boucle **REPETER ... JUSQU'A**

Exemple :

```
REPETER
        Afficher ("Voulez-vous continuer ? (O / N )")
        Saisir (reponse)
JUSQU'A reponse = "N"
```

Notion d'algorithmie

- Les structures de programmes

- Les structures itératives

La boucle **TANTQUE ... FINTQ**

Syntaxe :

```
TANTQUE condition FAIRE
        Suite d'instructions
        ....
FINTQ
```

Exemple :

```
i <- 1
TANTQUE i <= 10 FAIRE
        Afficher (i)
        i <- i+1
FINTQ
```

Notion d'algorithmie

Les structures de programmes

- Les structures itératives

La boucle **POUR ... FINPOUR**

Cette instruction s'utilise dans les cas où l'on connaît le nombre de fois que l'on veut répéter les instructions.

Syntaxe :

```
POUR nomVariable <--- BorneInf à BorneSup < PAS = valeurPas>
        Suite d'instructions ....
FINPOUR
```

Exemple :

```
POUR i <- 1 à 5 // si aucune valeur du PAS n'est inscrite alors PAS=1
        Afficher (i*i)
FINPOUR
```

Notion d'algorithmie

Les données structurées

- Tableau à une dimension

Supposons que l'on ait besoin dans un programme de conserver les notes de vingt élèves. Jusqu'à présent, nous avons utilisé une seule variable qui prenait successivement la valeur des différentes notes. Si on voulait avoir toutes les notes en même temps, il aurait fallu autant de variables que l'on a de notes (ex : note1, note2, note3...). Cette solution a des inconvénients car il faut trouver autant de noms différents que l'on a de valeurs de même type à conserver (ex : 20 notes, 20 noms...). Autre problème : il n'existe aucun lien entre ces différentes variables, or dans certains cas, on est amené à appliquer le même traitement à l'ensemble de ces variables (ex : saisie des notes, RAZ...).

Solution :

Utiliser un tableau, cela consiste à :

- attribuer un seul nom à l'ensemble de nos 20 variables (ex : notes)

- repérer chaque note par ce nom de variable suivi entre crochet d'un numéro d'indice compris entre 1 et 20.

Notion d'algorithmie

Les données structurées

- Tableau à une dimension

Notes :

12	Notes [1]
15	Notes [2]
10	Notes [3]
...	Notes [i]
9	Notes [19]
11	Notes [20]

Vocabulaire :

- notes[i] : désigne un élément du tableau notes, soit une valeur

- ce qui se trouve entre crochets se nomme un indice (Attention, dans la plupart des langage le premier élément de la liste est à l'index 0)

- chaque élément du tableau est assimilable à une variable, donc on peut les utiliser exactement comme n'importe quelle variable (affectation, condition, ...) à condition que la valeur de l'indice ne dépasse pas la taille du tableau.

Notion d'algorithmie

Les données structurées

- Tableau à une dimension

Déclaration :

La déclaration d'un tableau est très simple :

nomTableau[i] : Type

Ou

nomTableau[id..if] : Type

i : nombre de "cases"

id : indice de départ

if : indice de fin

Notion d'algorithmie

Les données structurées

- Tableau à une dimension

Affectation de valeurs :

Comme pour une variable classique.

Exemple : notes[2] <- 12

Tester la valeur d'une case de tableau :

Exemple : SI notes[2]=12 ALORS

…

Inconvénients :

Dans plusieurs langages, la taille d'un tableau est fixe. Autrement dit, vous ne pouvez pas ajouter ou supprimer les données d'un tableau. Il vous faut en faire une copie intégrant vos ajouts ou vos suppressions. Ce n'est pas très pratique si l'on doit changer fréquemment la taille de ce tableau. Dans ce cas il faudra préférer d'autres structures de données telles que les listes chaînées.

Notion d'algorithmie

Les données structurées

- Tableau à une deux dimensions (ou plus)

Imaginons maintenant que l'on souhaite garder les notes de tout un trimestre, pour tous les élèves de la classe. On suppose que l'on a 5 notes pour ce dernier trimestre.

En colonne : les 5 notes

En ligne : les différents élèves

Notes :

1	2	3	4	5	
1	10	9	15	12	54
2	8	14	2	5	18
3	6	7	15	12	10
...	15	5	5	12	14
19	16	6	3	14	5
20	2	5	12	20	20

Prenons la note n° 5 de l'élève n°1 (12/20). Pour la retrouver dans le tableau il faudrait taper notes[1,5]. En premier, le numéro de ligne et en second le numéro de colonne.

Notion d'algorithmie

Les données structurées

- Tableau à une deux dimensions (ou plus)

Déclaration :

Comme pour les tableaux à une seule dimension, mais là on ajoute le nombre de ligne. Par convention, toujours en premier le nombre de lignes (i) et en second le nombre de colonnes (j).

nomTableau [i,j] : Type

Ou

nomTableau [id..if,jd..jf] : Type

i : nombre de "lignes"
j : nombre de "colonnes"

id : indice de départ des lignes
if : indice de fin des lignes

jd : indice de départ des colonnes
jf : indice de fin des colonnes

Notion d'algorithmie

Les données structurées

- Tableau à une deux dimensions (ou plus)

Affectation de valeurs :

Comme pour une variable classique.

Exemple : note[2,3] <- 12

Tester la valeur d'une case de tableau :

Exemple : SI note[2,3]=12 alors

…

Notion d'algorithmie

- Les procédures

Utilisation :

Exemple : dans une application commerciale on souhaite calculer le montant d'une facture à différents endroits. On va donc faire appel au même calcul à plusieurs reprises. Il peut donc être intéressant, pour alléger le code, d'écrire qu'une seule fois cette procédure de calcul dans un petit programme à part et de pouvoir y accéder (l'appeler) à partir de n'importe quel endroit.

Cela permet également d'éclaircir l'algorithme en le découpant en plusieurs petits algorithmes plutôt que d'avoir un gros pavé. Ainsi c'est plus lisible et les erreurs sont plus faciles à retrouver.

Remarque : une procédure, comme une fonction, peut utiliser des paramètres déclarés comme des variables lors de la déclaration de la procédure (ou de la fonction). La valeur de ces paramètres sera passée par le programme appelant lors de l'appel de la procédure (ou de la fonction).

Notion d'algorithmie

- Les procédures

Ex. Déclaration :

Procedure trier(val1 : Entier, val2 : Entier)

Ou

Fonction perimetre(rayon : Reel) : Reel

Ex. Appel :

Appeler trier(150, nb_saisi)

Ou

Resultat <-- perimetre(val_saisi)

Comment savoir si on doit utiliser une procédure ou une fonction ?

- la procédure ne renvoie pas de valeur, elle réalise simplement des instructions
- la fonction renvoie une valeur (et une seule) au programme qui l'a appelée

Exercices

Calculatrice

Créer un programme se comportant comme une calculatrice, c'est-à-dire exécutant une boucle sur :

- Lecture d'une ligne supposée contenant un entier a, un entier b et un opérateur (ex : 1 + 5). Les opérateurs sont +, -, *, \.
- Boucler tant que l'utilisateur ne saisit pas le bon opérateur
- Calculer l'opération
- Afficher le résultat à l'écran
- Boucler tant que l'utilisateur ne saisit pas q à la place de l'opérateur.