RQ: ce cours est un très bref résumer du cours sur l'algorithmique. Il donne le stricte nécessaire à connaitre sur l'algorithmique.

Bref, c'est très important de retenir les grands points sur quoi on a fait allusion. Ce qui vous facilitera la tâche pour la suite

De notre étude du langage java... et même de n'importe quel autre langage de programmation. Pour faire cours, à chaque fois

Que vous voulez étudier un nouveau langage, commencer par vous rassurer que vous connaissez l'équivalent dans ce langage des éléments listé si dessous.

Algorithmique et programmation informatique

un algorithme: représente les étapes de résolution d'un problème donné

pour écrire un algorithme, on a besoin de certains outils... donc les principaux sont:

* les données: qui peuvent être

de diffèrent type:

@ Entier : -2, -1, 0, 1, 2 ...

@ Reel : 0.5, 1.4, -6, 10 ...

@ Caractere : '4', 'F', '€', '£' ...

@ Chaine de caractere : "Java", "4F€£" ...

@ Boolean : true, false

@ Des types personalisés. Exemple:

Les énumérations : ce sont des structures de données permettant de prédéfinir (délimiter) des valeurs possibles pour une variable

Exemple : féminin, masculin

Petit, moyen, grand

Les enregistrements : ce sont des structures de données permettant d'associer des données de types différents

Les objets : dans le cas de la poo

* les variables:

Qui ont pour rôle de contenir des données. Ainsi, il existe autant de type de variable que de type de données.

Une variable est donc identifiée (représenté) par un nom et un type de données.

Pour la plupart des langages de programmation et même en algorithmique, le nom d'une variable est un identificateur.

Un identificateur: est un mot former de lettres de l'alphabet anglais, des chiffres, des caractères Under score ("_")

Et de certains caractères spéciaux. Un identificateur ne commence pas par un un chiffre.

Exemple de variable en considérant la notation:

nom_variable : type_de_donnees

On aura:

nom_du_joueur : Chaine de caractere

ageDuJoueur : Entier tailleDu_joueur : Reel

* les constantes:

Ce sont des variables a contenu fixe (non modifiable) après l'initialisation.

Exemple de constante en considérant la notation:

nom_constante = valeur : type_de_donnees

On aura:

Pi = 3.14 : Reel

* les tableaux:

Un tableau est un regroupement de plusieurs données de même type. C'est un peu comme une liste de variable de même type, ayant un même nom,

Et donc les éléments sont identifiés par un indice.

Exemple de tableau en considérant la notation:

nom_tableau : TABLEAU [premier_indice...dernier_indice] : type_de_donnees

On aura:

T1 : TABLEAU [1...10] : Entier

tableau_2 : TABLEAU [0...3] : Chaine de caractere

NB: un tableau est référencier par son nom, le type données qu'il contient, et le nombre de données qu'il contient (ca taille)

. Pour accéder à un élément du tableau T1 d'indice (numéro) i, on utilise la syntaxe : T1[i]

* les commentaires:

Ils permettent de commenter le code pour le rendre plus lisible, plus compréhensible.

Les données inscrit dans le cadre commentaires ne sont pas exécuté.

syntaxe d'écriture des commentaire:

```
//je suis un commentaire mono line

/*je suis un

Commentaire

Multi line*/
```

* les instructions:

Une instruction est un ordre (élémentaire) que l'algorithme demande d'exécuter. Comme instruction élémentaire, on peut avoir:

@ l'affectation : c'est l'action de remplacer le contenu d'une variable par une autre valeur.

Cette instruction s'effectue par un operateur: "<-" ou "="

Dans la majorité des langages de programmation, on utilise: "="

Exemple:

nom_du_joueur : Chaine de caractere

nom_du_joueur = "java"

ageDuJoueur : Entier

ageDuJoueur = 17

ageDuJoueur_2 : Entier

ageDuJoueur_2 = ageDuJoueur;

RQ: on ne peut affecter qu'une donnée de même type que la variable dans celle ci

On peut affecter la valeur d'une variable dans l'autre

@ les opérations arithmétiques: //pour des exemples ici, on considère que les variables ageDuJoueur, et ageDuJoueur_2 son défini

on utilise les opérateurs arithmétiques connues:

+ : addition

ageDuJoueur = 17 + 3 //resultat: 20

- : soustraction

ageDuJoueur_2 = ageDuJoueur - 5 //resultat: 15

* : multiplication ageDuJoueur = 5 * 2 //resultat: 10

/ : division entiere

ageDuJoueur_2 = ageDuJoueur / 3 //resultat: 3

% : reste de la division entiere

ageDuJoueur = ageDuJoueur_2 % 5 //resultat: 3

@ les operations logiques: ce sont des operations donc le resultat est de type boolean (ou binaire). b1 : Boolean

b2 : Boolean

b3 : Boolean

on a entre autre:

les comparaisons : <, <=, >, >=, ==, !=

b1 = (4<2) //resultat: false

b2 = (0!=-1) //resultat: true

la conjonction : et(&&), renvois true si tout les entrées sont a true

b3 = b1 && b2 //resultat: false

la disjonction : ou(||), renvois true si au moins une entrée est a true

b3 = b1 || b2 //resultat: true

@ la concatenation: c'est une operation qui permet d'ajouter une chaine de caractère a la fin de l'autre.

exemple:

nom_du_joueur : Chaine de caractere

salutation : Chaine de caractere

nom_du_joueur = "cr7"

salutation = "bonjour " + nom_du_joueur; //resultat : bonjour cr7

* les structures conditionnelles:

ce sont des structures qui permettent d'executer un bloc de code si une condition est vérifié

exemple :

@ 1er cas:

si (condition) alors

/*bloc

de code

a executer si condition==vrai*/

finsi

@ 2eme cas:

si (condition) alors

/*bloc

de code

a executer si condition==vrai*/

sinon si(condition_2) alors

/*bloc

de code

a executer si condition==faux et condition_2==vrai*/

sinon

/*bloc

de code

a executer si condition==faux et condition_2==faux*/

finsi

* les structures iteratives:

ce sont des structures qui permettent d'executer iterativement un bloc de code si une condition est vérifié

exemple :

@ 1er cas: la boucle pour faire. Utiliser quand le nombre d'iteration est connu

pour i allant de n0 a n1 faire

/*bloc d'instruction a executer

tanque i est compris entre n0 et n1*/

finpour

@ 2eme cas: la boucle tanque faire. Utiliser quand le nombre d'iteration n'est pas connu

tanque (condition) faire

/*bloc d'instruction a executer

tanque condition==vrai*/

fintanque

@ 3eme cas: la boucle repete jusqu'a. Utiliser aussi quand le nombre d'iteration n'est pas connu,

mais que l'on souhaite que le bloc de code s'execute au moins une fois

repete

/*bloc d'instruction a executer

tanque condition==faux*/

jusqu'a(condition)

* les sous programmes

ce sont des blocs de pogramme prealablement ecrit et qui est réutilisé (ou réutilisable) dans un autre programme.

en algorithmique ou dans des languages de programmation, il y'a des sous programme deja écrit qu'on peut directement utiliser.

exemple:

la fonction ecrire(text) qui permet d'ecrire une donnée dans la console (a l'ecran), ou lire() qui permet de lire une entrée du clavier :

NB: on peut ecrire ces propre sous programme.

un programme informatique est un algorithme ou un ensemble d'algorithme, ecrit dans un language de programmation, pour etre executé automatiquement par une appareil informatique

il existe plusieurs languages de programmation:

* le language c
* le language c++
* le language java
* le language phyton
* le language php
* ...

pour la suite de ce cours, nous allons étudier et ecrire des algorithmes en language java

Pour plus entrer en détail dans l'étude de l'algorithmique