NOM LELLOUCHE

Prénom Léo

Promo 2020

Date 13/01/2016

18,2

LELLOUCHE Léo

MATIÈRE Algorithmie

Partie A : Compétences de base

Algorithme 1 :

(B) (A) (A) (B) (B) (A) (X)

Notre algorithme va demander à l'utilisateur 2 nombres et va afficher leur produit (sans stocker la valeur du produit)

Algorithme 1 : Produit de 2 nombres X ( )
Pour dire sans parenthèses

Variables : x, nombre entier, premier nombre rentré par l'utilisateur ✓
× locales

y, nombre entier, 2ème nombre rentré par l'utilisateur. ✓

(B)

DEBUT

Afficher "Saisissez x : ↵" (retour à la ligne)

x <— valeur saisie par l'utilisateur ✓

Afficher "Saisissez y : ↵"

y <— valeur saisie par l'utilisateur ✓

Afficher 'x' + "x" + 'y' + " = " + 'x × y' + "↵"
X x y (X x y)

FIN

Sans quotes

Les simples quotes indiquent une valeur de variable alors que les doubles quotes indiquent du texte

Algorithme 2 :

Notre algorithme va regarder si x< y et retourner faux si ce n'est pas le cas. Puis va verifier que x< Z et Z< y et retourner faux si ce n'est pas le cas. Ensuite, nous allons retourner vrai car si l'algorithm a retourné faux, il s'est arrêté et ne va donc pas retourner vrai.

booléen

Algorithme 2 : Un nombre est il dans un interval (X ----) : résulta

Données copiées : x, nombre entier, borne inferieur de l'interval

✓ y, nombre entier, borne superieur de l'interval.

z, nombre entier, nombre a tester.

Retour : Vrai ou faux selon si z appartient à l'interval [x ; y]. ✓

Résultat

(A)

DEBUT

Si x > y

| retourner FAUX ✓

Si ((z < x) OU (y < z))

| Retourner FAUX

✓

Retourner VRAI

FIN.

Algorithme 3:

Notre algorithme va vérifier si $x < y$, et retourner -1 dans ce cas là. Puis va demander à l'utilisateur un nombre rz tant qu'il n'est pas dans l'interval $[x ; y]$

Algorithme 3: Saisie d'un nombre dans un interval

Données copiés: x, nombre entier positif, borne inferieur de l'interval.

y, nombre entier positif, borne superieur de l'interval.

Variable locale: rz, entier positif, nombre a tester

Retour: Nombre entier, -1 si $x > y$ sinon rz

DEBUT

Si $(x > y)$

| Retourner -1

Faire

| Afficher "Saisissez une valeur pour rz" (retour à la ligne)

| rz ← valeur rentrée par l'utilisateur

Tant que: $(rz < x)$ OU $(y < rz)$

Retourner rz

FIN

Comme tout à l'heure: simple quote = valeur de la variable

double quote = texte

Algorithme 4 :

L'algorithme va parcourir le tableau de 0 à n-1, et va additionner tous les nombres trouvés. Puis va retourner cette somme.

x Somme (t : tableau de taille n) : entier

Algorithme 4 : Somme tableau d'entiers

Données copiées : t : tableau d'entier
n : nombre entier positif, taille du tableau

Variable : s, nombre entier, somme de tous les nombres présent dans le tableau
locale
i, nombre entier, curseur permettant de parcourir le tableau

Retour : nombre entier s

DEBUT

x cas du tableau vide ?

s ← 0
i ← 0
tant que (i < n) x une boucle for est plus adaptée
    s ← s + t[i]
    i ← i + 1
FIN tant que
retourner s x

B

FIN

NOM LELLOUCHE

Prénom Leo

Promo 2020

Date 13/01/2016

MATIÈRE Algorithmie

Partie B : Analyse et conception

Suite de Syracuse

Notre algorithme va prendre en parametre un nombre puis tant que ce nombre est different de 1, on le divise par 2 si il est pair et on le multiplie par 3 et on fait +1 sinon. En parallele, un compteur tournera pour compter le nombre d'iteration et affichera le numero de l'iteration suivi du nombre à cette itération.

Algorithme 5 : Suite de Syracuse

Donnée copié : s0, nombre entier positif, premier terme de la suite

Syracuse (S : entier)

Variable : i, nombre d'iteration

B

DEBUT

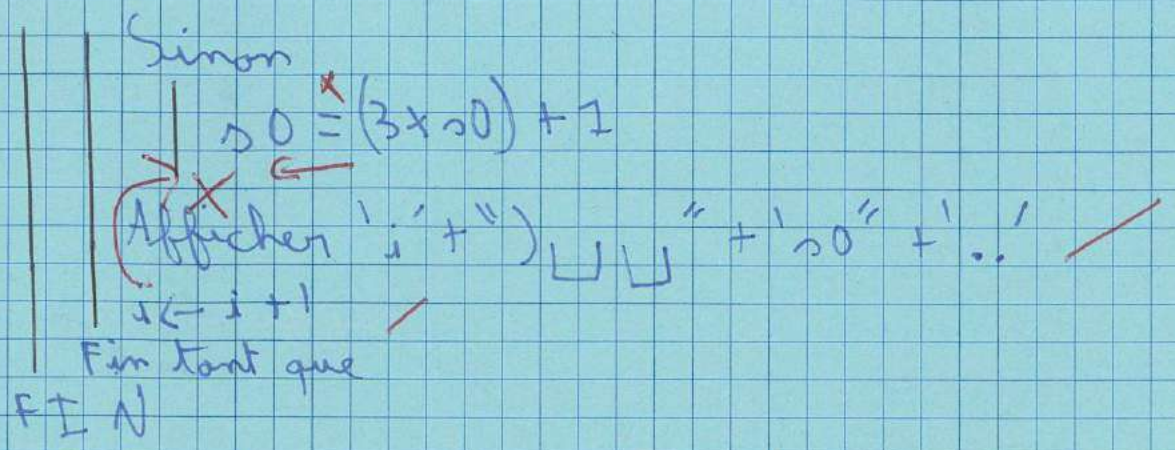
i ← 0

Afficher (i + " ␣␣ " + s0 + ' ') (caractère espace)

Tant que (s0 ≠ 1)

Si ((s0 % 2) = 0) % c'est du langage C

s0 ← s0 ÷ 2 ✓



Comme tout à l'heure, double quotes = texte
simple quote = valeur de variable

Quadrillage

Notre algorithme va écrire des choses tant que les variables d'itérations ne seront pas toutes les deux à n.
Il va écrire ## si (colonne %. 7)=0 ou (ligne %. 7)=0 sinon " ."

Algorithme 6 : Quadrillage (n, p), p ≠ 0
Données copiés : n, nombre entier positif, taille du tableau ✗ Pas besoin d'un tableau
p, nombre entier positif, taille du quadrillage
Variable : col, nombre entier positif, colonne en cours
lin, nombre entier positif, ligne en cours.

DEBUT :
Si ((n %. 7) ≠ 0) ET (7 ≠ 0)
Afficher "Erreur, impossible de faire un quadrillage parfait"
Arrêt algorithme

```
lin ← 0
Tant que (lin < n)
    col ← 0
    Tant que (col < n)
        Si (((col % 7) = 0) OU ((lin % 7) = 0))
            Afficher "#"
        Sinon
            Afficher "."
        col ← col + 1
    Fin tant que
    lin ← lin + 1
FIN Tant que
FIN
```

✓ ✓ ✓

A

← X afficher "\n"