Nom : Nkumbe Aurelien

Classe : Master 1 info

TD/TP Algorithmique

Contents

Exercice 1

Donner la valeur des expressions suivantes :

- 2 + 3 * 4 = 14
- 8 / 2 + 1 = 5
- 8/ (2 + 1) = 2.67
- 12 < 5 = Faux
- « cours » + « algo » = « cours algo »
- 54 / ln (1) = Impossible
- Vrai ET Faux OU Vrai = VRAI
- (2 < 3) ET (4 > 5) = FAUX
- (2 + 6 = 10) OU (8 / 2 > 3) = VRAI

Exercice 2

1- X > 3 ET Y > 3
2- X = Y ET Y =Z
3- Y <= X ET X <= Z
4- X = Y OU Y = Z OU X = Z
5- (X = Y ET Y <> Z) OU (X = Z ET Y <> Z)

Exercice 3

Remaques

- La variable pof n'est pas nécessaire
- Certains mots clés ne sont pas dans le bon formalisme (en Majuscule) :

Rôle : Calculer la circonférence du cercle 2 Pi R

```
ALGORITHME mystere
CONSTANTES (PIF : Reel) <-- 3.14
VARIABLES paf , puf : Réels
DEBUT
  SAISIR paf
  puf <-- 2 * PIF * paf
  AFFICHER puf
FIN
```

Exercice 4

```
ALGORITHME permutation
VARIABLES a, b : Nombres
DEBUT
  SAISIR a
  SAISIR b
  a <-- a + b
  b <-- b - a
  a <-- b - a
  AFFICHER a
  AFFICHER b
FIN
```

Exercice 5

1er Cas :

L'algorithme boucle à l'infini (jusqu'à interruption volontaire de l'utilisateur ou du système d'exploitation pour éviter un stack overflow) car la variable x reste toujours inchangé a la valeur 1

2e Cas :

Le programme affiche la valeur de x égale

```
3628800
```

Preuve

K	1	2	3	4	5	6	7	8	9	10
X	1	1	2	6	24	120	720	5040	40320	362880
X * K	1	2	6	24	120	720	5040	40320	362880	**3628800**

Exercice 6

Analyse

L'algorithme initiale échouera à partir de l'analyse syntaxique, comme amélioration on peut

- On peut déclarer la variable Z
- Afficher les résultats de C et Z en plus

```
FONCTIONS_UTILISEES
VARIABLES
  A EST_DU_TYPE NOMBRE
  B EST_DU_TYPE NOMBRE
  C EST_DU_TYPE NOMBRE
  // new ligne
  Z EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  LIRE A
  LIRE B
  C PREND_LA_VALEUR 2 * A + B
  SI (C = 10) ALORS
    DEBUT_SI
    A PREND_LA_VALEUR C
    // new ligne
    AFFICHER C
    FIN_SI
    SINON
      DEBUT_SINON
      Z PREND_LA_VALEUR 2 * C
      // new ligne
      AFFICHER Z
      FIN_SINON
FIN_ALGORITHME
```

Exercice 7

```
def exercice7 () :


    LIMIT = 10
    list_nombres_paires = []

    print("Trouver les  nombres  paires consecutifs")

    nb = int(input("Entrer un nombre entier <--  "))

    while (len(list_nombres_paires) < LIMIT) :
        if (nb % 2 == 0) :
            list_nombres_paires.append(nb)
        nb += 1

    print("Les nombres  paires consecutifs trouvés", list_nombres_paires)
```

```
Trouver les  nombres  paires consecutifs
Entrer un nombre entier <--  10
Les nombres  paires consecutifs trouvés [10, 12, 14, 16, 18, 20, 22, 24, 26, 28]
```

Exercice 8

```
def exercice8 () :

    produit = 1

    print("Trouver le produit des entiers")

    nb = int(input("Entrer un nombre  <--  "))

    for x in range(1, nb + 1) :
        produit *= x

    print("Le produit factoriel ", nb ,"! est", produit)
```

```
Trouver le produit des entiers
Entrer un nombre  <--  3
Le produit factoriel de 3 est 6
```

Exercice 9

1) Fonction d'Armstrong

```
def exercice9_test_nombre_armstrong(nombre):

    charaters = list(str(nombre))
    result = 0
    for item in charaters :
        nb = int(item)
        result = result + pow(nb,3)
    return int(result) == int(nombre)
```

2) Programme

```
def exercice9 () :
    # Nombre d'Armstrong
    print("Nombre d'Armstrong")
    nb = input("Entrer le nombre a vérifié <--  ")
    nb = int(nb)
    for nb_rank in range(1,nb+1):
        if (exercice9_test_nombre_armstrong(nb_rank)) :
            print(nb_rank)
```

Exercise 10

1) Afficher le max

```
def exercice10_1 () :

    LIMIT = 10
    max = 0
    for n in range (LIMIT) :
        nb = input("Entrer le ("+ str(n + 1) +") nombre <--") or 0
        nb = int(nb)
        if (nb > max) :
            max = nb
    print("Le max est", max)
```

2) Conserver les nombres entrés

```
def exercice10_2 () :

    LIMIT = 10
    liste_nombres = [] #++++
    max = 1
    for n in range (LIMIT) :
        nb = input("Entrer le ("+ str(n + 1) +") nombre <--") or 0
        nb = int(nb)
        liste_nombres.insert(n, nb) #++++
        if (nb > max) :
            max = nb
    print("Le max est", max)
```

3) Enregistrer et afficher l'index du max

```
def exercice10_3 () :

    LIMIT = 10
    liste_nombres = []
    max = 0
    max_idx = None  #++++
    for n in range (LIMIT) :
        nb = input("Entrer le ("+ str(n + 1) +") nombre <--") or 0
        nb = int(nb)
        liste_nombres.insert(n, nb)
        if (nb > max) :
            max = nb
            max_idx = n + 1 #++++
    print("Le max est", max, " et d'indice " , max_idx) #++++
```

Exercise 11

1) Programme

```
import random

def exercice11 ():
    LIMIT = 100
    list_1_to_100 = []
    somme = 0
    for x in range(1, LIMIT + 1):
        x = round(random.random() * LIMIT)
        list_1_to_100.append(x)
        somme += x
    # print(list_1_to_100)
    print("La sommme donne", somme)
```

2) Découpage en fonction

```


import random

def exercice11_random_integer_list(limit = 100):
    list_1_to_100 = []
    for x in range(1, limit + 1):
        x = round(random.random() * limit)
        list_1_to_100.append(x)
    return list_1_to_100

def exercice11_somme(liste):
    s = 0
    for x in liste:
        s += x
    return s


def exercice11_main():
    list_1_to_100 = exercice11_random_integer_list(100)
    somme = exercice11_somme(list_1_to_100)
    print("La sommme donne", somme)
```

Exercice 12

```python
import math

def exercice12 ():

    UNIT_M3 = 8

    print("Calcul du nombre de radiateur pour chauffer un piece")

    hauteur = input("Entrer la hauteur <--") or 0
    largeur = input("Entrer la largeur <--") or 0
    longeur = input("Entrer la longeur <--") or 0

    aire = float(longeur) * float(largeur) * float(hauteur)

    nb_radiateurs = aire / UNIT_M3

    print("Le nombre de radiateurs necessaires est", math.ceil(nb_radiateurs))
```

```
Calcul du  nombre de radiateur pour chauffer un piece par 8 m3
Entrer la hauteur <--10
Entrer la largeur <--10
Entrer la longeur <--8
Le nombre de radiateurs necessaires est 100
```

Excerice 13

1) Completer l'algo

```
//  Exercice 13 - 1  ================================
FONCTIONS_UTILISEES
VARIABLES
  nb EST_DU_TYPE NOMBRE
  code EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  code PREND_LA_VALEUR round(random()*10)
  AFFICHER code
  AFFICHER "\nDeviner  le code caché"
  LIRE nb
  TANT_QUE (code != nb) FAIRE
    DEBUT_TANT_QUE
    AFFICHER "reessayé"
    LIRE nb
    FIN_TANT_QUE
  AFFICHER "\nBien joué"
FIN_ALGORITHME
```

2) Permettre de choisir entre 0 et 100

```
//  Exercice 13 - 2  ================================
```

```
FONCTIONS_UTILISEES
VARIABLES
  nb EST_DU_TYPE NOMBRE
  tailleCode EST_DU_TYPE NOMBRE
  code EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  AFFICHER "\nChoisir la taille du code entre 0 et 100"
  LIRE tailleCode
  TANT_QUE (tailleCode < 0 OU tailleCode > 100) FAIRE
    DEBUT_TANT_QUE
    AFFICHER "\nReessayé, Choisir la taille du code entre 0 et 100"
    LIRE tailleCode
    FIN_TANT_QUE
  code PREND_LA_VALEUR round(random()*tailleCode)
  AFFICHER code
  AFFICHER "\nDeviner  le code caché"
  LIRE nb
  TANT_QUE (code != nb) FAIRE
    DEBUT_TANT_QUE
    AFFICHER "reessayé"
    LIRE nb
    FIN_TANT_QUE
  AFFICHER "\nBien joué"
FIN_ALGORITHME
```

3) Ajouter les conditions

```
FONCTIONS_UTILISEES
VARIABLES
  nb EST_DU_TYPE NOMBRE
  tailleCode EST_DU_TYPE NOMBRE
  code EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  AFFICHER "\nChoisir la taille du code entre 0 et 100"
  LIRE tailleCode
  TANT_QUE (tailleCode < 0 OU tailleCode > 100) FAIRE
    DEBUT_TANT_QUE
    AFFICHER "\nReessayé, Choisir la taille du code entre 0 et 100"
    LIRE tailleCode
    FIN_TANT_QUE
  code PREND_LA_VALEUR round(random()*tailleCode)
  AFFICHER code
  AFFICHER "\nDeviner  le code caché"
  LIRE nb
  TANT_QUE (code != nb) FAIRE
    DEBUT_TANT_QUE
     SI (nb < code) ALORS
      DEBUT_SI
      AFFICHER "\nDonner une valeur superieur a "
      AFFICHER nb
      FIN_SI
```

```
      SINON
        DEBUT_SINON
        AFFICHER "\nDonner une valeur inferieur a "
        AFFICHER nb
        FIN_SINON
    LIRE nb
    FIN_TANT_QUE
  AFFICHER "\nBien joué"
FIN_ALGORITHME
```

4) Limiter le nombre de tentative

```
//  Exercice 13 - 4  ================================
FONCTIONS_UTILISEES
VARIABLES
  nb_tentative EST_DU_TYPE NOMBRE
  nb EST_DU_TYPE NOMBRE
  tailleCode EST_DU_TYPE NOMBRE
  code EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  nb_tentative PREND_LA_VALEUR 3
  AFFICHER "\nChoisir la taille du code entre 0 et 100"
  LIRE tailleCode
  TANT_QUE (tailleCode < 0 OU tailleCode > 100) FAIRE
    DEBUT_TANT_QUE
    AFFICHER "\nReessayé, Choisir la taille du code entre 0 et 100"
    LIRE tailleCode
    FIN_TANT_QUE
  code PREND_LA_VALEUR round(random()*tailleCode)
  AFFICHER code
  AFFICHER "\nDeviner  le code caché"
  LIRE nb
  TANT_QUE (nb_tentative > 1 ET code != nb) FAIRE
    DEBUT_TANT_QUE
    nb_tentative PREND_LA_VALEUR nb_tentative - 1
    AFFICHER "\nVous avez "
    AFFICHER nb_tentative
    AFFICHER " tentatives restantes"
    SI (nb < code) ALORS
      DEBUT_SI
      AFFICHER "\nDonner une valeur superieur a "
      AFFICHER nb
      FIN_SI
      SINON
        DEBUT_SINON
        AFFICHER "\nDonner une valeur inferieur a "
        AFFICHER nb
        FIN_SINON
    LIRE nb
    FIN_TANT_QUE
  SI (nb == code) ALORS
```

```
      DEBUT_SI
      AFFICHER "\nBien joué"
      FIN_SI
      SINON
        DEBUT_SINON
        AFFICHER "\nDésolé, vous avez perdu, "
        AFFICHER "le code a deviné etait "
        AFFICHER code
        FIN_SINON
FIN_ALGORITHME
```

```
Choisir la taille du code entre 0 et 100
Entrer tailleCode : 125

Reessayé, Choisir la taille du code entre 0 et 100
Entrer tailleCode : 50
46
Deviner  le code caché
Entrer nb : 23

Vous avez 2 tentatives restantes
Donner une valeur superieur a 23
Entrer nb : 50

Vous avez 1 tentatives restantes
Donner une valeur inferieur a 50
Entrer nb : 46

Bien joué
***Algorithme terminé***
```

Exercice 14

```
def exercice14 ():

    UNIT_LKM = 100

    cout = None

    print("Calcul cout du carburant pour le voyage")


    distance_km = input("Entrer la distance du voyage <--") or 0
    prix_litre = input("Entrer le prix du litre de carburant <--") or 0
    consommation_moy_l_km = input("Entrer la consommation moyenne (en L/"+ str(UNIT_LKM)
+"km) <--") or 0

    # Raisonnement par la regle de 3
    # consommation_moy_l_km -----> UNIT_LKM
    # nb_litre ?????            -----> distance_km

    nb_litre = float(distance_km)  * float(consommation_moy_l_km) / UNIT_LKM
    cout = nb_litre * float(prix_litre)

    print("Le cout du carburant pour le voyage est", math.ceil(cout))
```

```
Calcul cout du carburant pour le voyage
Entrer la distance du voyage <--1000
Entrer le prix du litre de carburant <--10
Entrer la consommation moyenne (en L/ 100km) <--100
Le cout du carburant pour le voyage est 10000
```