Université de Mila
Institut des Sciences et de la Technologie
Département de Mathématique et Informatique
1ère Année LMD MI

SÉRIE DE TD N°2

Algorithmique et structures de données 2 Année universitaire : 2019 / 2020

SOLUTION

Remarque : il faut faire le découpage modulaire avec tous les exercices

Exercice 1

a) Décrire la partie déclaration.

Elle contient la liste de tous les objets (Constante, variables, fonction et procédure….) utilisés et manipulés dans le corps de l'algorithme. Ici nous avons déclaré variables S et N, et une fonction Somme.

b) Décrire la parie corps.

Représente la suite des instructions à exécuter (ensemble d'opérations à exécuter sur les données i.e variables).ici nous avant 3 instructions (programme principale).la deuxième contient un appel à la fonction somme.

c) Quelles sont les variables ***d'entrées*** et ***sorties***

	Entrées	Sorties
Programme principale	N	S
Fonction somme	Nbr	R

d) Quelles sont les paramètres ***formels***et***effectifs***.

Les paramètres ***formels*** **:** Nbr

Lesparamètres***effectifs*** : N

e) Quelles sont les variables ***globales*** et les variables***locales***

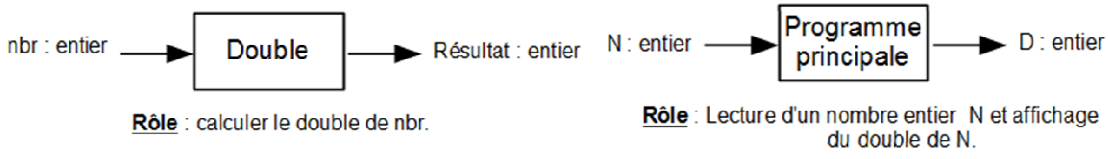
Les variables ***globales :*** ***N,*** S

Les variables ***locales :*** Nbr,R, i

Exercice 2

1) Ecrire un algorithme qui affiche le double d'un nombre entier N.

- **Le découpage modulaire :**



- **L'algorithme :**

```
Algorithme Exo2-1
   N, D : entiers ;
```

```
Fonction Double (nbr : entier) : entier ;
Resultat : entier ;
Début
Resultat ← nbr * nbr ;
Retourner (Resultat) ;
Fin ;
Début /* Programme principale */
Ecrire (''Tapez un nombre entier SVP :'') ;
Lire(N) ;
D ← Double (N) ;
Ecrire (''Le double de'', N, '' est '', D) ;
Fin.
```

2) Ecrire un algorithme qui affiche le double et le triple d'un nombre entier N.

Deux sorties donc utilisation d'une procédure

```
Algorithme Exo2-2
N, D, T : entiers ;
Procédure Double-Triple (nbr: entier; Var Resultat1, Resultat2 : entiers);
Début
Resultat1 ← nbr * nbr ;
Resultat2 ← nbr * nbr * nbr ;
Fin ;
Début /* Programme principale */
Ecrire (''Tapez un nombre entier SVP :'') ;
Lire(N) ;
Double-Triple (N, D, T) ;
Ecrire (''Le double de'', N, '' est '', D) ;
Ecrire (''Le Triple de'', N, '' est '', T) ;
Fin.
```

Remarque importante : Les modules (fonctions, procédures) sont indépendants, on peut donc utiliser les mêmes noms de variables pour définir les paramètres et les variables locales.

Avec **VAR** => C'est la variable originale qui est passée à la fonction ou à la procédure (passage de l'adresse de la variable) => pas de protection des variables

Sans **VAR** => une copie de la valeur de la variable est passe à la fonction ou a la procédure(passage d'une copie de la valeur) => protection des variables

3) Écrire un algorithme qui lie un nombre entier positif N et affiche si il est premier ou non.

```
Algorithme Exo2_3
    N : entiers;
Fonction premier (Nbr : entier) : booléen ;
       R : booléen ;
       i : entiers ;
Début
       i ← 2 ;    R ← vrai ;
      Tant que  (i<= Nbr div 2 et R=vrai)faire
        Si (Nbr div i = 0) alors
              R ← Faux ;
Finsi
     i←i+1
     Fin tantque
     Retourner (R) ;
Fin.
Début
Lire (N);
```

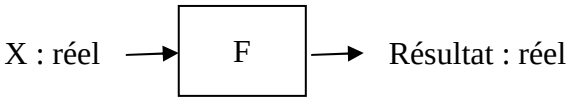
```
  Si (N<=0) alors
Ecrire (‘’ erreur… ‘’) ;
  Sinon
 Si (premier (N) = vrai) alors
            Ecrire (''Nombre premier'') ;
     Sinon
           Ecrire (''Nombre non premier'') ;
Finsi
Finsi
Fin.
```

Exercice 3

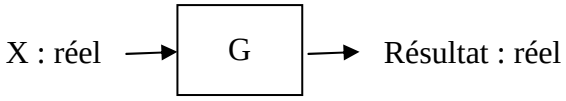
Soit les fonctions suivantes : f(x) = 2 x2 + 1, g(x) = 3 x / (x-1), h(x) = f(x) + g(x)

- Écrire un algorithme qui lie un nombre réel z et affiche : f(z), g(z) et h(z).

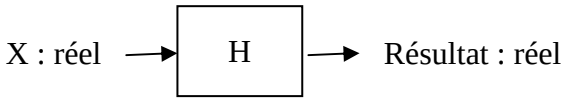
➢ **Le découpage modulaire :**



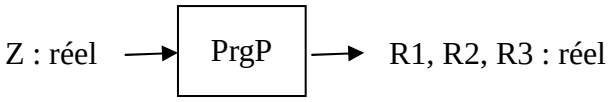
Rôle : Calcule de la fonction F(X)



Calcule de la fonction G(X)



Calcule de la fonction H(X)



Rôle : Lecture d’un nombre Z puis Calcule et affiche le résultat de F(X), G(X) et H(X)

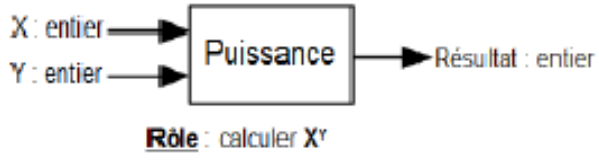
```
Algorithme Exo2
   Z, R1, R2, R3: Réels;
   Fonction F (X : réel) : réel ;
Résultat : réel ;
Début
Résultat ← (2*X*X) +1;
Retourner (Résultat) ;
Fin ;
Fonction G (X : réel) : réel ;
Résultat : réel ;
Début
Résultat ← (3*X) /(X-1);
Retourner (Résultat) ;
Fin ;
   Fonction H (X : réel) : réel ;
Résultat : réel ;
Début
Résultat ← F(X) + G(X); /* On peut écrire : Résultat ← ((2*X*X) +1) +
((3*X) /(X-1))*/
Retourner (Résultat) ;
Fin ;
```

```
Début /* Programme principale */
Ecrire (''Tapez un nombre réel SVP :'') ;
Lire(Z) ;
R1 ← F(Z) ;
R2 ← G(Z) ;
R3 ← H(Z) ;
Ecrire (R1, R2, R3) ;
Fin.
Algorithme Exo2
  Z, R1, R2, R3: Réels ;
Fonction F (X: réel) : réel ;
//
Fonction G (X: réel): réel;
//
Fonction H (X : réel) : réel ;
//
ProcedureCalcul (X: réel; var Resultat1, Resultat2, Resultat3: reels);
  Début
    Resultat1 ← F(X);
    Resultat2 ← G(X);
Resultat3 ← H(X);
Fin ;
Début /* Programme principale */
Ecrire (''Tapez un nombre réel SVP :'') ;
Lire(Z) ;
Calcul(Z, R1, R2, R3) ;
Ecrire (R1, R2, R3) ;
Fin.
```

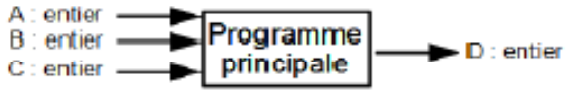
Exercice 4

- Écrire un algorithme qui lie trois nombres positifs non nuls A, B, et C et calcule et affiche lasomme suivante : $((A! + (B)^C)!)^B$

➢ **Le découpage modulaire :**



Rôle : calculer X^Y



Rôle : Lecture de trois nombres entiers A,B, et C et affichage de la somme D.

Rôle : calculer la factorielle de nbr

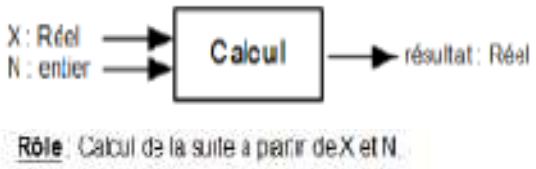
➢ **L'algorithme :**

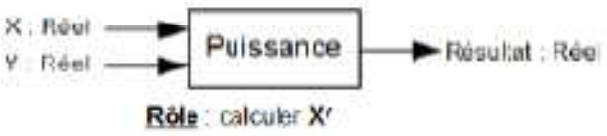
```
Algorithme Exo4
Fonction Factorielle (Nbr : entier) : entiers ;
Résultat, fact, i : entiers
Début
fact ← 1;
pouriallant de N a 1 pas -1 faire
fact ← fact * i ;
Fin pour ;
Résultat ← fact ;
```

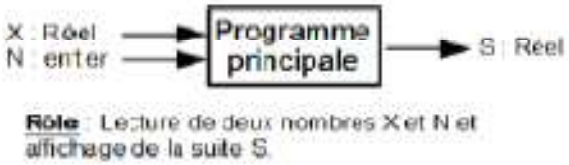
```
Retourner(Résultat) ; /*On peut éventuellement supprimer la variable Résultat et faire
DirectementRetournez(fact)*/
Fin ;
Fonction Puissance (X, Y : entier) : entier ;
Résultat, p, i : entiers
Début
p ← 1;
Pour iallant de 1 à Y faire
p← p * X ;
Fin pour ;
Résultat ← p ;
Retourner(Résultat) ;
  Fin ;
Début /* Programme principale */
Lire (A, B, C) ;
D ← Puissance (Factorielle (Factorielle(A) + Puissance (B, C)), B) ;
Ecrire (D) ;
Fin.
```

Écrire un algorithme qui lie deux nombres positifs **n** (entier) et **x** (réel) puis calcule et affiche la somme suivante : $x - x^2/2! + x^3/3! - \ldots x^n/n!$

- **Le découpage modulaire :**







- **L'algorithme :**

```
Algorithme Exo3-2
   X, S: Réels;
N : entier ;
Fonction Factorielle (Nbr : entier) : entiers ;
//
//
Fonction Puissance (X : réel, Y : entier) : entiers ;
//
//
Fonction Calcul (X : réel, N : entier) : entiers ;
Résultat, somme : réel ; i : entier ;
Début
somme ← 0 ;
Pour iallant de 1 aNfaire
Si (i mod 2 = 0)
somme ← somme – (puissance(X, i)/factorielle(i)) ;
Sinon
somme ← somme + (puissance(X,i)/factorielle(i)) ;
finsi
Fin pour ;
Résultat← somme ;
Retourner(Résultat) ;
Fin ;
```

```
Début /* Programme principale */
Lire(X, N) ;
S ← Calcul(X, N) ; // On peut écrire directement : Écrire (Calcul(X,N)) ;
Ecrire (S) ;
Fin.
```

Exercice 5:

- Écrire un algorithme permettant de lire deux nombres quelconque X, Y et de dire si ses deux nombres sont amicaux ou pas.

```
Algorithme Exo5
X, Y : Réels ;
Fonction Diviseur (Nbr, N : entier ;) : booléen ; /*Remarque : On peut se passer de cette fonction*/
Résultat : booléen ;
Début
Si (Nbrmod N) = 0 alors
Résultat← vrai;
Sinon
Résultat← faux;
Finsi
Retourner(Résultat) ;
Fin ;
Fonction Somme Diviseurs (Nbr : entier ;) : entier ;
Résultat, somme : entiers;
Début
Somme ← 0 ;
Pour iallant de 1 à (Nbr div 2) faire
Si (Diviseur (Nbr,i) = vrai) alors
somme ← somme + i;
finsi
    Fin pour
Résultat← somme ;
Retourner(Résultat) ;
Fin ;
Fonction Amicaux (X, Y : entier ;) : booléen ;
Resultat: booléen; DX, DY : entiers ;
Début
DX ← Somme_Diviseurs(X) ;
DY ← Somme_Diviseurs(Y) ;
Si ((DX = DY) et (X+Y = DX)) alors
Résultat← vrai;
Sinon
Résultat← faux;
Finsi
Retourner(Résultat) ;
Fin ;
Début /* Programme principale */
Lire(X, Y) ;
Si (Amicaux(X, Y) = vrai) alors
Ecrire (''Les deux nombres sont amicaux'');
Sinon
Ecrire (''Les deux nombres ne sont pas amicaux'');
Fin.
```