CORRECTION SESSION NORMALE D'INITIATION A L'ALGORITHMIQUE INF 111 (2021-2022)

Proposez par : GROUPE GENIUS REPETITION

Par : Joël_yk

Exercice 01 : 04 pts

1) **Donner le Tableau de Trace :**

Variables	x	i	s
Instructions			
01	2		2
02		4	
03			5
04		3	
05			11
06		2	
07			22
08		1	
09			52
10		0	
11			107
12			

2) **Expression du résultat :**

$T(X) = \sum_{K=0}^{n} a_k X^k$ Ou a est un élément de poids k de notre Tableau T .

Cette fonction calcul efficacement l'image T(x) d'un nombre réel x par une fonction polynomiale associée à un polynôme T de degré n (Algorithme de Horner)

$T(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x^1 + a_0 x^0 = s$

Nb: Cette méthode permet aussi d'effectuer une conversion rapide d'un nombre écrit en base x_0 en écriture en base 10.

Propriété :

Soit a € k, Lorsqu'on cherche à évaluer un polynôme $T(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_0$, il n'est clairement pas optimal de calculer chaque a_kx^k, puis de les ajouter. En effet, pour calculer a_nx^n, il est intéressant de se souvenir que l'on a déjà calculé $a_{n-1}x^{n-1}$, et donc x^{n-1}.L'écriture du polynôme suggère de procéder itérativement en sommant chacun des monômes évalués en x, ce qui implique de calculer à chaque fois une exponentielle rapide. La méthode dite de la factorisation de Horner (qui n'est pas réellement une factorisation au sens usuel) consiste à écrire le polynôme T(X) sous la forme $T_1(X).X+a_0$ et à répéter cette opération pour le polynôme $T_1(X)$ et ainsi de suite jusqu'au polynôme constant $T_n(X)=a_n$ qui achève le processus :

$$T(X)=(a_nX^{n-1}+a_{n-1}X^{n-2}+\cdots+a_2X+a_1)X+a_0$$

$$=((a_nX^{n-2}+a_{n-1}X^{n-3}+\cdots+a_2)X+a1)X+a0$$

$$\vdots$$

$$=((\ldots(a_nX+a_{n-1})X+a_{n-2})X+\cdots+a_2)X+a_1)X+a_0$$

On peut construire cette suite de polynômes $T_k(X)$ par la relation de récurrence suivante, où le premier terme $T_0(X)$ de la suite est défini par le polynôme T(X) :

$$T_k(X):=\begin{cases} T(X) & \text{si } k=0 \\ \frac{1}{x}\,(T_{k-1}(X)-a_{k-1}) & \text{si } 1\leq k\leq n \end{cases}$$

Preuve : Soit **T** notre tableau contenant les valeurs suivantes :

3	8	0	1	1	2

avec x = 2 & n = 5

On a $T(2) = 2*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 8*2^1 + 3*2^0$

$= 64+16+8+0+16+3=$**107**

La méthode consiste donc à multiplier le premier coefficient par x_n et à lui ajouter le deuxième coefficient. On multiplie alors le nombre obtenu par x_n et on lui ajoute le troisième coefficient ainsi de suite.

L'instant Genius 💡: William George Horner (Naissance (Bristol en Angleterre 9 juin 1786) – Décès(Bath en Angleterre, 22 septembre 1837)) est un mathématicien britannique. Horner est connu pour sa méthode qui permet d'évaluer rapidement un polynôme en un point.

3) **Fonction récursive :**

```
Fonction  GeniusCalcul ( T : tableau , x : entier , n : entier ) : réel ;
 Debut
        Si ( n = 0 ) alors
            GeniusCalcul ← T[0]
        Sinon
            GeniusCalcul ← x ^ n - 1 * T[n-1]+ GeniusCalcul(T,x,n-1) ;
       Fsi
Fin ;
```

Preuve : Soit T notre tableau contenant les valeurs suivantes :

4	2	3

avec x = 5 & n = 3

On a $T(5) = 3*5^2 + 2*5^1 + 4*5^0$

$= 75+10+4=\mathbf{89}$

Il faut calculer chacune des puissances de x_0, multiplier celle-ci par son coefficient a_k puis faire la somme de ce que l'on a trouvé. Si on calcule les puissances de x_0 en multipliant successivement x_0 par lui-même, le nombre nécessaire de produits est alors de $n + (n - 1) + \dots + 2 + 1 = n(n+1)/2$, quantité qui croît comme le carré du degré du polynôme. Le Cas de base de notre récursivité est un tableau d'un seul élément qui retourne $T(x)= a_0x^0$ et le cas de récursif : $T(x)= a_nx^n + a_{n-1}x^{n-1}+\dots + a_2x^2 + a_1x^1+ a_0x^0$

Exercice 02 : 04 pts

1) **Structure de donnée : 0.5pts**

```
Type Point   = Enregistrement
                    Abs :  réel ;
                    Ord : réel ;
                  Fin;
        tabPoint = tableau [1..50] de Point ;
```

2) **Fonction : Distance 0.5pts**

fonction distance (P1 : Point , P2 : Point) : réel ;
Début
$\mathbf{distance} \leftarrow \sqrt{((\mathbf{P1.abs} - \mathbf{P2.abs}) * (\mathbf{P1.abs} - \mathbf{P2.abs}) + (\mathbf{P1.ord} - \mathbf{P2.ord}) * (\mathbf{P1.ord} - \mathbf{P2.ord}))};$
Fin ;

3) **Procédure : create_save_range_Point 1pts**

```
Procedure  create_save_range_Point ( var  t_Point : tabPoint );
 Var P : Point ; i : entier ;
 Debut
        Pour i  de 1 à 50 faire
            Ecrire ( ``Entrer les coordonnées de votre Point  ``) ;
            Ecrire ( ``Entrer l'abscisse de votre Point  ``) ; Lire (P.abs) ;
            Ecrire ( ``Entrer l'ordonne de votre Point  ``) ; Lire (P.ord) ;
            t_Point[i] ← P ;
        Fpr
Fin ;
```

4) **Fonction : sum_Distance_Point 2pts**

```
fonction  sum_Distance_Point ( t_Point : tabPoint ) : réel ;
  Var Sum  : reel ; i : entier ;
 Debut
      Sum ← 0 ;
      Pour i  de 1 à 49 faire
        Sum ← Sum + distance ( t_Point[i] , t_Point[i+1] )  ;
      Fpr
  sum_Distance_Point ← sum ;
Fin ;
```

Exercice 03 : 04 pts

On désire calculer l'exponentiel de x par un développement limite :

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!}$$

1) **Ecriture de l'algorithme : 2pts**

```
Algorithme Exponentiel ;
 Var i : entier ; x , facteur , expo : reel ;
  Debut
    Ecrire ( " Entrer la valeur de x `` ) ; Lire ( x ) ;
   facteur ← 1 ; expo ← 1 ;
   Pour i de 1 a 100 faire
      facteur ← facteur * ( x / i );
      expo ← expo +facteur ;
  fpr
  Ecrire (" Exponentiel de x ``, x , `` est :  `, expo );
Fin.
```

2) Ecriture de l'algorithme interactif : 2pts

```
Algorithme Exponentiel_version2 ;
Var i : entier ; x , facteur , expo : reel ; reponse : booleen ;
  Debut
   Repeter
     Ecrire ( " Entrer la valeur de x `` ) ; Lire ( x ) ;
     facteur ← 1 ; expo ← 1 ;
   Pour i de 1 a 100 faire
      facteur ← facteur * ( x / i );
      expo ← expo +facteur ;
   fpr
  Ecrire (" Exponentiel de x `` , x , `` est : `` , expo );
  Ecrire ( " Voulez-vous déterminer l'exponentiel e^x d'un autre nombre ? répondre
par vrai pour OUI  ou faux pour NON `` );
  Lire ( reponse ) ;
 Jusqu'à ( reponse < > vrai ) ;

Fin.
```

Exercice 04 : 04 pts

1) Algorithme : 4pts

```
Algorithme NombreParfait_Limite;
var cpt,i,j,k: entier; lim : reel ;
    parfait : tableau[1..100] de entier ;
Début
  répéter
    écrire ( `` Entrez la limite `` );
    lire(lim);
  jusqu'à(lim>6);  co Pour avoir au moins un nombre Parfait à afficher fco
  cpt← 0; k←1 ;
  pour i de 1 à lim-1 faire
      pour j de 1 à (i div 2)faire
         si( i mod j  = 0)alors
                cpt← cpt+j ;
         fsi
```

```
    fpr
    si(cpt=i)alors
       parfait[k] ← i ;  k←k+1 ;
    fsi
   fpour
       pour i de 1 à k faire
         Ecrire ( `` Les nombres parfaits inferieurs a :`` , lim , `` : `` , parfaits[i]) ;
     fpour

Fin.
```

Exercice 05 : 04 pts

1) **Procédure Carre : 1.5pts**

```
     Procedure Carre( X :entier ; var Cp : booléen; var RC : entier ) ;
     Var I:entier ;
     Debut
       Cp← Faux ; I ← 0 ;
      Tantque (I<= X div 2)et( Cp < > vrai) Faire
        Si (X=I*I) Alors
            Cp← Vrai ;
            rc ← I ;
         sinon
               I← I + 1 ;
        Fsi ;
      ftque ;
Fin ;
     co www.pandacodeur.com fco
```

2) **Algorithme : 2.5 pts**

```
co www.pandacodeur.com fco

Algorithme  MonBeauCarreParfait;
Var I,N,S,P,X,Rac :entier ; CParfiat:booleen;

Debut
  Ecrire("Donner le nombre d'éléments N") ;
Repeter
  Lire(N)
Jusqu'à N>0 ;
S← 0 ; P ← 1 ;
Pour I de 1 à N Faire
     Lire(X) ;
carre(X,Cparfait,Rac) ;
Si ( CParfait = vrai ) Alors
   S← S+Rac ;
   P← P*Rac ;
Fsi ;
Fpr
co www.pandacodeur.com fco
carre(S,Cparfait,Rac) ;
Si ( CParfait = vrai ) Alors
   Ecrire("La somme S=",S," est un carré parfait")
  Fsi ;
carre(P,Cparfait,Rac) ;
co www.pandacodeur.com fco
Si ( CParfait = vrai ) Alors
   Ecrire("Le produit=",P," est un carré parfait")
Fsi ;
Fin.
```


" La persévérance, c'est ce qui rend l'impossible possible, le possible probable et le probable réalisé. "

Bonne chance pour le rattrapage les amies.