montant - 15.10.2013

```
****************************************
Calcule le nombre de billets de 20, 10, 5 euros et de pièces de 2 et 1 euros nécessaires pour
réaliser un montant donné
****************************************

1   VARIABLES
2     montant EST_DU_TYPE NOMBRE
3     reste EST_DU_TYPE NOMBRE
4     billets20 EST_DU_TYPE NOMBRE
5     billets10 EST_DU_TYPE NOMBRE
6     billets5 EST_DU_TYPE NOMBRE
7     pieces2 EST_DU_TYPE NOMBRE
8     pieces1 EST_DU_TYPE NOMBRE
9   DEBUT_ALGORITHME
10    LIRE montant
11    reste PREND_LA_VALEUR montant
12    billets20 PREND_LA_VALEUR floor(reste/20)
13    reste PREND_LA_VALEUR reste%20
14    billets10 PREND_LA_VALEUR floor(reste/10)
15    reste PREND_LA_VALEUR reste%10
16    billets5 PREND_LA_VALEUR floor(reste/5)
17    reste PREND_LA_VALEUR reste%5
18    pieces2 PREND_LA_VALEUR floor(reste/2)
19    pieces1 PREND_LA_VALEUR reste%2
20    AFFICHER "billets de 20€ : "
21    AFFICHER billets20
22    AFFICHER "billets de 10€ : "
23    AFFICHER billets10
24    AFFICHER "billets de 5€ : "
25    AFFICHER billets5
26    AFFICHER "pièces de 2€ : "
27    AFFICHER pieces2
28    AFFICHER "pièces de 1€ : "
29    AFFICHER pieces1
30  FIN_ALGORITHME
```

suite - 15.10.2013

```
****************************************
Calcule le nième terme de la suite u(n) = a.u(n-1) + b avec u(0) = c
****************************************

1   VARIABLES
2     u EST_DU_TYPE NOMBRE
3     n EST_DU_TYPE NOMBRE
4     i EST_DU_TYPE NOMBRE
5     a EST_DU_TYPE NOMBRE
6     b EST_DU_TYPE NOMBRE
7     c EST_DU_TYPE NOMBRE
8   DEBUT_ALGORITHME
9     LIRE a
10    LIRE b
11    LIRE c
12    LIRE n
13    u PREND_LA_VALEUR c
14    POUR i ALLANT_DE 1 A n
15      DEBUT_POUR
16      u PREND_LA_VALEUR a*u+b
17      FIN_POUR
18    AFFICHER "u("
19    AFFICHER n
20    AFFICHER ") = "
21    AFFICHER u
22  FIN_ALGORITHME
```

entierParfait - 15.10.2013

```
****************************************
Cet algorithme détermine si un entier n lu au clavier est parfait ou non.
(un entier est parfait s'il est égal à la somme de ses diviseurs stricts)

Premiers nombres parfaits : 6, 28, 496, 8 128, 33 550 336, 8 589 869 056
(le 7 octobre 2008, on ne connaissait que 46 nombres parfaits)
****************************************

1   VARIABLES
2     n EST_DU_TYPE NOMBRE
3     somme EST_DU_TYPE NOMBRE
4     diviseur EST_DU_TYPE NOMBRE
5   DEBUT_ALGORITHME
6     LIRE n
7     somme PREND_LA_VALEUR 0
8     POUR diviseur ALLANT_DE 1 A n/2
9       DEBUT_POUR
10      SI (n % diviseur == 0) ALORS
11        DEBUT_SI
12        somme PREND_LA_VALEUR somme+diviseur
13        FIN_SI
14      FIN_POUR
15    SI (n == somme) ALORS
16      DEBUT_SI
17      AFFICHER "Ce nombre est parfait."
18      FIN_SI
19      SINON
20        DEBUT_SINON
21        AFFICHER "Ce nombre n'est pas parfait."
22        AFFICHER "(la somme des diviseurs stricts vaut "
23        AFFICHER somme
24        AFFICHER ")"
25        FIN_SINON
26  FIN_ALGORITHME
```

listeMonotone - 15.10.2013

```
****************************************
Vérifie si une liste entrée au clavier est monotone.
****************************************

1   VARIABLES
2     n EST_DU_TYPE NOMBRE
3     l EST_DU_TYPE LISTE
4     i EST_DU_TYPE NOMBRE
5     croissant EST_DU_TYPE NOMBRE
6     trouve EST_DU_TYPE NOMBRE
7   DEBUT_ALGORITHME
8     LIRE n
9     SI (n == 0) ALORS
10      DEBUT_SI
11      AFFICHER "Liste vide"
12      FIN_SI
13      SINON
14        DEBUT_SINON
15        POUR i ALLANT_DE 0 A n-1
16          DEBUT_POUR
17          LIRE l[i]
18          FIN_POUR
19        trouve PREND_LA_VALEUR 0
20        i PREND_LA_VALEUR 0
21        //On cherche le sens de la suite (croissant ou décroissant)
22        TANT_QUE (trouve == 0 ET i < n-1) FAIRE
23          DEBUT_TANT_QUE
24          SI (l[i] < l[i+1]) ALORS
25            DEBUT_SI
26            trouve PREND_LA_VALEUR 1
27            croissant PREND_LA_VALEUR 1
28            FIN_SI
29            SINON
30              DEBUT_SINON
31              SI (l[i] > l[i+1]) ALORS
32                DEBUT_SI
33                trouve PREND_LA_VALEUR 1
34                croissant PREND_LA_VALEUR 0
35                FIN_SI
36              FIN_SINON
37          i PREND_LA_VALEUR i+1
38          FIN_TANT_QUE
39        SI (trouve == 0) ALORS
40          DEBUT_SI
41          AFFICHER "Liste constante"
42          FIN_SI
43          SINON
44            DEBUT_SINON
45            SI (croissant == 1) ALORS
46              DEBUT_SI
47              TANT_QUE (i < n-1 ET croissant == 1) FAIRE
48                DEBUT_TANT_QUE
49                SI (l[i] > l[i+1]) ALORS
50                  DEBUT_SI
51                  croissant PREND_LA_VALEUR 0
52                  FIN_SI
53                i PREND_LA_VALEUR i+1
54                FIN_TANT_QUE
55              FIN_SI
56              SINON
57                DEBUT_SINON
58                TANT_QUE (i < n-1 ET croissant == 0) FAIRE
59                  DEBUT_TANT_QUE
60                  SI (l[i] < l[i+1]) ALORS
61                    DEBUT_SI
62                    croissant PREND_LA_VALEUR 1
63                    FIN_SI
64                  i PREND_LA_VALEUR i+1
65                  FIN_TANT_QUE
66                FIN_SINON
67            FIN_SINON
68        SI (i == n-1) ALORS
69          DEBUT_SI
70          SI (croissant == 1) ALORS
71            DEBUT_SI
72            AFFICHER "liste monotone croissante"
73            FIN_SI
74            SINON
75              DEBUT_SINON
76              AFFICHER "liste monotone decroissante"
77              FIN_SINON
78          FIN_SI
79          SINON
80            DEBUT_SINON
81            AFFICHER "liste non monotone"
82            FIN_SINON
```

```
83          FIN_SINON
84
85    FIN_ALGORITHME
```

triInsertion - 15.10.2013

```
****************************************
Trie une liste entrée au clavier par insertion.
****************************************
```

```
1   VARIABLES
2     n EST_DU_TYPE NOMBRE
3     i EST_DU_TYPE NOMBRE
4     l EST_DU_TYPE LISTE
5     position EST_DU_TYPE NOMBRE
6     k EST_DU_TYPE NOMBRE
7     x EST_DU_TYPE NOMBRE
8     tailleCourante EST_DU_TYPE NOMBRE
9   DEBUT_ALGORITHME
10    LIRE n
11    POUR tailleCourante ALLANT_DE 0 A n-1
12      DEBUT_POUR
13      LIRE x
14      position PREND_LA_VALEUR tailleCourante
15      TANT_QUE (position > 0 ET l[position-1] > x) FAIRE
16        DEBUT_TANT_QUE
17        l[position] PREND_LA_VALEUR l[position-1]
18        position PREND_LA_VALEUR position - 1
19        FIN_TANT_QUE
20      l[position] PREND_LA_VALEUR x
21      FIN_POUR
22    POUR i ALLANT_DE 0 A n-1
23      DEBUT_POUR
24      AFFICHER l[i]
25      AFFICHER " "
26      FIN_POUR
27    AFFICHER " "
28  FIN_ALGORITHME
```

traceFonction - 15.10.2013

```
****************************************
Dessine la courbe de la fonction F1 dans l'intervalle [-10,10]. La précision (le pas) est
choisi par l'utilisateur.
****************************************

1   VARIABLES
2     pas EST_DU_TYPE NOMBRE
3     x EST_DU_TYPE NOMBRE
4     y EST_DU_TYPE NOMBRE
5   DEBUT_ALGORITHME
6     LIRE pas
7     x PREND_LA_VALEUR -10
8     TANT_QUE (x <= 10) FAIRE
9       DEBUT_TANT_QUE
10      TRACER_POINT (x,F1(x))
11      x PREND_LA_VALEUR x+pas
12      FIN_TANT_QUE
13  FIN_ALGORITHME

Fonction numérique utilisée :
F1(x)=(x*x)/10
```

enveloppe - 15.10.2013

```
****************************************
Dessine une enveloppe.
La hauteur et la largeur de l'enveloppe sont données par l'utilisateur
****************************************

1   VARIABLES
2     hauteur EST_DU_TYPE NOMBRE
3     largeur EST_DU_TYPE NOMBRE
4     xmin EST_DU_TYPE NOMBRE
5     ymin EST_DU_TYPE NOMBRE
6     xmax EST_DU_TYPE NOMBRE
7     ymax EST_DU_TYPE NOMBRE
8   DEBUT_ALGORITHME
9     LIRE largeur
10    LIRE hauteur
11    xmin PREND_LA_VALEUR -largeur/2
12    ymin PREND_LA_VALEUR -hauteur/2
13    xmax PREND_LA_VALEUR largeur/2
14    ymax PREND_LA_VALEUR hauteur/2
15    //Traçage du rectangle
16    TRACER_SEGMENT (xmin,ymin)->(xmax,ymin)
17    TRACER_SEGMENT (xmax,ymin)->(xmax,ymax)
18    TRACER_SEGMENT (xmax,ymax)->(xmin,ymax)
19    TRACER_SEGMENT (xmin,ymax)->(xmin,ymin)
20    //Diagonales hautes
21    TRACER_SEGMENT (xmin,ymax)->(0,-hauteur/6)
22    TRACER_SEGMENT (0,-hauteur/6)->(xmax,ymax)
23    //Diagonales basses
24    TRACER_SEGMENT (xmin,ymin)->(-largeur/5,hauteur/10)
25    TRACER_SEGMENT (xmax,ymin)->(largeur/5,hauteur/10)
26  FIN_ALGORITHME
```

polygone - 15.10.2013

```
****************************************
Cet algorithme dessine un polygone régulier à n côtés,
avec ou sans "rayons"...
****************************************

1   VARIABLES
2     angle EST_DU_TYPE NOMBRE
3     angleParcours EST_DU_TYPE NOMBRE
4     pointX EST_DU_TYPE NOMBRE
5     pointY EST_DU_TYPE NOMBRE
6     suivantX EST_DU_TYPE NOMBRE
7     rayon EST_DU_TYPE NOMBRE
8     suivantY EST_DU_TYPE NOMBRE
9     dessineRayons EST_DU_TYPE CHAINE
10    n EST_DU_TYPE NOMBRE
11    i EST_DU_TYPE NOMBRE
12  DEBUT_ALGORITHME
13    // lecture des données
14    AFFICHER "Nombre de côtés du polygone ?"
15    LIRE n
16    AFFICHER "On dessine les rayons (O/N) ?"
17    LIRE dessineRayons
18    // Initialisations
19    rayon PREND_LA_VALEUR 8
20    angle PREND_LA_VALEUR (2.0 * Math.PI)/n
21    angleParcours PREND_LA_VALEUR angle/2.0
22    suivantX PREND_LA_VALEUR rayon * cos(angleParcours)
23    suivantY PREND_LA_VALEUR - (rayon * sin(angleParcours))
24    // Dessin du polygone
25    POUR i ALLANT_DE 1 A n
26      DEBUT_POUR
27      pointX PREND_LA_VALEUR suivantX
28      pointY PREND_LA_VALEUR suivantY
29      suivantX PREND_LA_VALEUR rayon * cos(angleParcours)
30      suivantY PREND_LA_VALEUR rayon * sin(angleParcours)
31      TRACER_SEGMENT (pointX,pointY)->(suivantX,suivantY)
32      SI (dessineRayons=="O") ALORS
33        DEBUT_SI
34        TRACER_SEGMENT (0,0)->(pointX,pointY)
35        FIN_SI
36      angleParcours PREND_LA_VALEUR angleParcours + angle
37      FIN_POUR
38  FIN_ALGORITHME
```