

Tri Selection:

procedure Selection(n : entier, @ t : tab):

debut

pour i de 1 à $n-1$ fois

$j \leftarrow i$

$\text{tmp} \leftarrow t[i]$

tant que $j > 1$ et $\text{tmp} < t[j-1]$ fois

$t[j] \leftarrow t[j-1]$

$j \leftarrow j-1$

fin tant que

$t[j] \leftarrow \text{tmp}$

Fin pour

Fin

Var	type
i	entier
j	entier
tmp	entier ou réel!

Tri a bulle:

procedure bubble(n : entier, @ t : tab):

Debut

pour i de 0 à $n-2$ fois

si $t[i] > t[i+1]$ alors

$\text{tmp} \leftarrow t[i]$

$t[i] \leftarrow t[i+1]$

$t[i+1] \leftarrow \text{tmp}$

bubble(n, t)

Fin si

Fin pour

Fin

Var	Type
i	entier
tmp	entier ou réel
bubble	procedure

Tri Shell:

procedure Shell(n : entier, $@$) t : tab):

Debut:

$pas \leftarrow 1$

tant que $pas \times 3 + 1 \leq n$ faire

$pas \leftarrow pas \times 3 + 1$

Fin tant que.

pour

de pas à $n-1$ faire

$tmp \leftarrow t[i]$

$j \leftarrow i$

tant que $j \geq 1$ et $t[j-pas] < tmp$ faire

$t[j] \leftarrow t[j-pas]$

$j \leftarrow j - pas$

Fin tant que

$t[j] \leftarrow tmp$

Fin pour

$pas \leftarrow pas \div 3$

Jusqu'à ce que $(pas \leq 1)$

tmp, j, i	entier
pas	entier

Fin

project: mathcomplex.

Collisions:

Def func col: $(2-2i)^2$

$0.2+2.2a+0.1b+0.1c$

function col (req: String): String:

begin

// separate,

13/12/21:

Correction devoir Sygm n°1:

P.P

Algorithme point-equilibre

Debut

saissir(l, c)

fillmat(l, c, m)

$src \leftarrow \text{"equilibre.dat"}$

remplir(m, src, l, c)

$src_1 \leftarrow \text{"motsSomme.txt"}$

remplir2(src, src_1, m)

Fin

TDO:

Var	Type
src	entier
src_1	chaîne
m	mat

Fonction($m: mat, l: entier, c: entier$): booléen:

Debut

$i \leftarrow -1$

Repete

$i \leftarrow i + 1$

$check \leftarrow \text{an}[l, i] \neq \text{an}[8c]$

Jusqu'à $check = \text{faux}$, ou $i = c - 1$.

Retourner $check$.

Fin

TDO:

$i: entier$

$check: bool$

Recursive:

fonction fact (n : entier) : entier

Debut
Si $n = 0$ alors $f \leftarrow 1$

Si non

$f \leftarrow 1$
pour i de n à 1 faire
 $f \leftarrow f \times i$

Fin pour
retourner f

Fin

fonction dec (n : entier) : entier

Debut

Si $n = 0$ alors
retourner 1

Si non

retourner $n \times \text{dec}(n-1)$

Fin Si

Fin

Exemple 2:

encadre La somme de n premier entier positive de 2 nombres;

fonction Sum (n : entier) : entier

Debut

$S \leftarrow 0$

pour i de 0 à n faire

$S \leftarrow S + i$

Fin pour

retourner S

Fin

fonction Sum (n : entier) : entier

Debut

Si $n = 0$ alors

retourner 0

Si non

retourner $n + \text{Sum}(n-1)$

Fin Si

Fin

Act = écrire une fonction récursive permettant de effectuer la multiplication de 2 entiers > 0 notés A et B , en utilisant uniquement l'addition entière.

$$\text{en effet } A \times B = A + A + \dots + B$$

Fonction Rec(a: entier, b: entier): entier

Debut

Si $b = 1$
Pour i de 1 à b faire

Si $b = 1$

Fin pour

Retourner S

Fin

Fonction Rec(a: entier, b: entier): entier

Debut

Si $b = 0$ alors

Retourner 0

Sinon

Retourner $a + \text{Rec}(a, (b-1))$

Fin Si

Fin

Act 2: Soit une chaîne de caractères une \tilde{f} qui nous renvoie l'inverse de cette chaîne

Fonction Rev(str: chaîne): chaîne

Debut

Str ← str 1
Pour i de 0 à n div 2 faire

$\text{tmp} \leftarrow \text{str}[i]$

$\text{str}[i] \leftarrow \text{str}[n-i-1]$

$\text{str}[n-i-1] \leftarrow \text{tmp}$

Fin pour

Retourner str

Fin

fonction pal (ch: chaîne, n = 0) : chaîne

Debut

si ch = "" alors

retourner ""

sinon

retourner (ch[0]) + pal (sans chaine(ch, 1, long(ch))

Fin si

Fin

palindrome Algo:

Iterative:

Fonction est palindrome (ch: chaîne): boolean

Debut

i ← 0

check ← vrai

tant que i < long(ch) div 2 et check faire

check ← ch[i] = ch[long(ch)-1-i-1]

i ← i + 1

Fin tant que

retourner check

Fin

ou :

Fonction polyin (ch: chaîne) boolean

Debut

pour i de 0 à long(ch)-1 div 2 faire

si ch[i] ≠ ch[long(ch)-1-i] alors

retourner faux

```

    Fin Si
  Fin Pour
  retourner Vrai
Fin

```

Recursive

```

Fonction recString (myString: chaîne, l: entier, b: entier, e: entier): boolean
  Debut
    Si b = e ou b + 1 = e alors
      retourner Vrai
    Sinon
      Si myString[b] = myString[e] alors
        retourner Vrai et recString(myString, l, b + 1, e - 1)
      Sinon
        retourner faux
    Fin Si
  Fin Si
Fin

```

Exercises:

problème 1:

Iterative:

```

Fonction prod (p: entier, q: entier): entier
  Debut
    S ← 0
    Pour i de 1 à q faire
      S ← S + p
    Fin Pour
  Fin

```


retourner S
Fin

Rec:

Fonction prodRec(p : entier, q : entier): entier

Debut

Si $q = 0$ alors
retourner 0

Sinon
retourner $+ \text{prodRec}(p, q-1)$

Fin Si

Fin

problème 2:

Itérative:

Fonction sum(n : entier): entier

Debut

$S \leftarrow 0$

Pour i de 1 à n faire

$S \leftarrow S + i$

Fin pour

retourner S

Fin

Fonction sum (n : entier) : entier

Debut

Si $n = 0$ alors
retourner 0

Sinon

retourner $n + \text{sum}(n-1)$

Fin Si

Fin

problème 3

Méthode d'Euclide :

Fonction pgcd (a : entier, b : entier) : entier

Debut

tant que $b \neq 0$ faire

$r \leftarrow a \bmod b$

$a \leftarrow b$

$b \leftarrow r$

Fin tant que

retourner a

Fin

Récurssive

Fonction pgcd (a : entier, b : entier) : entier

Debut

- Si $b = 0$ alors
retourner a

Si non
retourner $(b, a \bmod b)$

Fin Si

Fin

Méthode de diff

Iterative :

Fonction pgcd(a : entier, b : entier): entier

Début

Tant que $a \neq b$

Si $a > b$ alors

$a \leftarrow a - b$

Si non

$b \leftarrow b - a$

Fin Si

Fin Tant que

retourner a

Fin

Recursive :

Fonction pgcd (a: entier, b: entier): entier

Debut

Si $a = b$ alors

retourner a

sinon

Si $a > b$ alors

retourner pgcd (a - b, b)

sinon

retourner pgcd (b, b - a)

Fin Si

Fin Si

Fin

Fonction Rev (myString): chaîne

Debut

Si myString = "" alors

retourner ""

sinon

procedure raplin lb (e)



S_1, S_2, \dots, S_n

col 1 8

fonction (m: nat, l: entiers, p1: entiers, p2: entiers): entiers

Debut

$S \leftarrow 0$

Pour i de p1 à p2 faire

$S \leftarrow S + m[l, i]$

Fin pour

retourner S

Fin

TPO

i: entiers

S: entiers