

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):

Début:

$pas \leftarrow 1$

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

$pas \leftarrow pas \times 3 + 1$

Fin tant que.

Repete:

Pour i 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

TDD :

Var	Type
src	entier
src_1	chaîne
m	mat

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

Debut

$i \leftarrow -1$

Repete

$i \leftarrow i + 1$

$check \leftarrow m[l, i] \neq m[l, c]$

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

Retourner $check$.

Fin

TDD :

$i: entier$

$check: bool$

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-1 faire

tmp ← str[i]

str[i] ← str[n-i-1]

str[n-i-1] ← 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

Récur­sive

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

Probleme 4 :

Iterative :

Fonction isprime (n: entier): bool

Debut

Return \leftarrow fause

i \leftarrow 0

Repetier

Return \leftarrow n mod i = 0

i \leftarrow i + 1

Jusqu'a i = n div 2 ou Return

retourner non to return

Fin

Recursive:

Function isPrime(n : entier, i : entier): bool

Debut

{ initial value of i is 1, }

Si $n \text{ div } 2 = i$ alors
retourner vrai

Si non

retourner vrai et $n \bmod i \neq 0$ et isPrime($n, i+1$)

Fin Si

Fin

problem 5:

Iterative:

procedure Rev($myString$: chaîne):

Debut

$pos \leftarrow 0 - (\text{Long}(myString) - 1) \text{ div } 2$ for pos

$temp \leftarrow myString[pos]$

$myString[pos] \leftarrow myString[\text{Long}(myString) - 1 - pos]$

$myString[\text{Long}(myString) - 1 - pos] \leftarrow temp$

Fin pour

Fin

Recursive

procedure Rev(myString: chaîne):

Debut

Si Long(myString) > 1 alors

temp ← myString[i]

myString[i] ← myString[Long(myString) - 1]

myString[Long(myString) - 1] ← temp

Rev(sous-chaîne(myString, 1, Long(myString) - 2))

Fin Si

Fin

problème:

Iterative

Fonction pal(ch: chaîne): bool

Debut

i ← 0

check ← false

Repete

check ← ch[i] = ch[Long(ch) - 1 - i]

i ← i + 1

Jusqu'à (Long(ch) - 1) div 2 = i ou check

retourner check

Fin

Recursive:

Fonction pol (ch: chaîne): bool

Debut

Si $\text{len}(ch) \leq 1$ alors

retourner vrai

Sinon

retourner vrai et $ch[0] = ch[\text{len}(ch)-1]$
et pol(sous chaîne($ch, 1, \text{len}(ch)-1$))

Fin Si

Fin

problème 7:

Iterative:

Procédure Rev (@t: tab):

Debut

Pour i de 0 à $\text{Long}(t)-1$ div 2 faire

temp $\leftarrow t[i]$

$t[i] \leftarrow t[\text{Long}(t)-1-i]$

$t[\text{Long}(t)-1-i] \leftarrow \text{temp}$

Fin pour

Fin

Recursive:

procedure Rev($@t:tb$):

Debut

si $\text{Long}(t) > 1$ alors

temp $\leftarrow t[0]$

$t[0] \leftarrow t[\text{Long}(t)-1]$

$t[\text{Long}(t)-1] \leftarrow \text{temp}$

Rev($t[1, \text{Long}(t)-2]$)

Fin si

Fin

Iterative

procedure Rev($@t:tb$):

Debut

$p \leftarrow 0$

$n \leftarrow \text{Long}(t)-1$

Tant que $\text{Long}(t) \neq p$ faire

temp $\leftarrow t[p]$

$t[p] \leftarrow t[n]$

$t[n] \leftarrow \text{temp}$

$p \leftarrow p+1$

$n \leftarrow n-1$

Finit. Inpage

Fin

Recursive

procedure Rev ($t : \text{tab}, p : \text{entier}, n : \text{entier}$)

Debut

Si $\text{long}(t) \leq 2 < p$ alors

$\text{temp} \leftarrow t[p]$

$t[p] \leftarrow t[n]$

$t[n] \leftarrow \text{temp}$

Rev ($t, p+1, n-1$)

Fin Si

Fin

problème 8

Rec :

Fonction rev ($ch : \text{chaîne}; n : \text{entier}$) : chaîne

Debut

Si $n \geq 0$ alors

retourner

$ch[n] + \text{rev}(ch, n-1)$

Fin Si

Fin

Fin technique

Si check
rechner mid

Si
rechner - 1

F₂Si

F₂

P

Iterative :

Fonction rev : (ch : chaîne) : chaîne

Début

chz ← ""
pour i de long(ch) - 1 jusqu'à 0 faire

chz ← chz + ch[i]

Fin pour

retourner chz

Fin

problème 9 :

Iterative :

Fonction binary (l : lb, b : entier, e : entier, de)

Début

check ← faux

Tant que check = faux et b ≤ c faire

mid ← (b + c) div 2

si t[mid] = de

check ← vrai

Sinon

si de > t[mid]

b ← b + 1

Sinon

c ← mid - 1

Fin Si

Recursive :

Fonction binary ($t: \text{tab}$, $ele: \text{entier}$) : entier

Debut

Si $t[\text{long}(t) \text{div } 2] = ele$ alors

retourner $\text{long}(t) \text{div } 2$

Sinon Si $t[\text{long}(t) \text{div } 2] > ele$ alors

retourner $\text{binary}(t[1, (\text{long}(t) \text{div } 2) - 1], ele)$

Sinon
retourner $\text{long}(t) \text{div } 2 + \text{binary}(t[(\text{long}(t) \text{div } 2) + 1, \text{long}(t) - 1])$

Fin Si

Fin

lineaire :

Recursive :

Fonction lineaire ($t: \text{tab}$, $m: \text{entier}$, $x, i: \text{entier}$) : entier

Si $m = i$ alors

retourner -1

Sinon

Si $t[i] = x$ alors

retourner i

Sinon

returner (t, m, x, i+1)

Fin Si

Fin

Rec 2: with counts:

Function Rec(t: tab, i, x, m): entier
Debut

Si $i \geq m$ alors

returner 0

sinon

Si $t[i] = x$ alors

returner $1 + \text{Rec}(t, i+1, x, m)$

sinon

returner Rec(t, i+1, x, m)

Fin Si

Fin Si

Fin

Fonction binom (t: tab, n, p: entiers)

debut

d ← 0

f ← n - 1

p ← -1

Tant que d < f et p ≠ -1 faire

m ← (d + f) div 2

Si $\in T[m]$ alors

~~p~~ ← m

sinon

Si $\notin T[i]$ alors

f ← m - 1

sinon

d ← m + 1

fin

fin tant que

retourner m

Fin

Recursive:

Fonction Rec ($t: l \rightarrow b, d, f, p$: entiers) : entiers

Début

si $d > f$ alors

retourner -1

sinon

$m \leftarrow (d + f) \text{ div } 2$

si $t[m] > p$

alors retourner m

sinon si $t[m] < p$ alors

retourner Rec($t, d, m - 1, p$)

sinon

retourner Rec($t, d, m + 1, p$)

Fin si

Fin

Algo P.O

Debut

Size (n)

superPrime (n, "C:\file.txt")

Fin

procedure Size (@ n: entiers):

Debut

Repetes

lie (n)

Jusqua $40000 < n < 100000$

Fin

procedure SuperPrimes (n: entiers, path: chaîne)

Debut

procedură (minutes):

lie(n)

Si mai 40000(n < 100000 day

full(n)

Fin

50390

Contine 0 - > 1 fois

~~50390~~

test (00)

a c - a din no

5 din 2

1 2 | 3 4 5

$$\left(\frac{5}{1} \right) = 5 \text{ True}$$

$$\left(\frac{5}{5} \right) = 1$$

$$\frac{5}{2} \quad n=1$$

$$10 \text{ div } 2 = 5$$

$$\boxed{1} \quad 2 \quad 3 \quad 4 \quad 5 \quad \boxed{6 \quad 7 \quad 8 \quad 9 \quad 10}$$

$$\frac{10}{2} = ? \quad n=0$$

$$\underline{10 \neq \text{pair}}$$

Fonction (m:entier) : bool
 Debut

pour i de 2 à n div 2 faire

si $n \bmod i = 0$ alors
 retourner faux

Fin si

Fin pour

Retourner vrai

Fin

fonction suite(m :entier, P :entier, Up :réel):réel:
booleen

Si $m \geq 1$ alors
 $a \leftarrow Up \left(\text{power}(-2, m) \right) / - (2 \times Up)$

Si $P = m$ alors

retourner a

fin Si

retourner $\left(\text{suite}(m, P+1, a) \right)$

Fin Si

retourner 0

Fin

fonction power(a, b :entier, i :entier):entier

Debut

Si $i \leq b$ alors

retourner $a \times \text{power}(a, b, i+1)$

fin Si

retourner a

Fin

$$u_0 = "0"$$

$$u_1 = "01"$$

$$u_2$$

$$0 \rightarrow 01$$

$$1 \rightarrow 10$$

$$u_2 = "01, 10"$$

$$u_3 = "011, 010, 01"$$

fonction binary (n: entier, i: entier, res: chaîne): chaîne.

chaîne

si n > 0

si i < long(res) alors

si res[i] = "0" alors

res ← sous chaîne(res, 0, i+1) + "1" + sous chaîne(res, i+1, long(res))

sinon

res ← sous chaîne(res, 0, i+1) + "0" + sous chaîne(res, i+1, long(res))

fin si
retourne binary(n-1, i+1, res)

sinon

retourne

binary(n, 0, res)

fin si

Fin

retourne res.

Fin

\overline{F}_n

Def: 2:

ordre de récurrence: 2.

$$u_1 = 1$$

$$u_2 = 2$$

$$u_n = u_{n-1} + u_{n-2}$$

Fonction suite (m: entier, suite: chaîne, i: entier): chaîne.

Début

si $m > 0$ alors

si $i < \text{Long}(suite)$

Msg " " miniscule,

26 Alph

Alpha- \rightarrow " z c y "

"Mot sc" . "

Msg "bonjour"

Alpha- "ops VWL"

Motsc "numaoy pfita"

Motsc \leftarrow Motsc + "A" . ,

Alpha- \rightarrow

Motsc

1 m

Rev:

- Soit la suite U

$$\begin{cases} U_0 = 1 \\ U_n = \frac{(-2)^n}{2 \times U_{n-1}} \end{cases}$$

La suite U est une suite
d'ordre 1.

$$U_1 = \frac{(-2)^1}{2 \times 1} = -1 \quad / \quad U_2 = \frac{(-2)^2}{2 \times (-1)} = -2.$$

fonction Suite (n : entier): entier

Debut

Si $n \leq 0$ alors

retourner 1

sinon

retourner ($\text{puiss}(-2, n) \text{ div } (2 \times \text{Suite}(n-1))$)

Fin

fonction puiss(a, b : entier): entier

Debut

Si $b > 1$ alors

retourner $\text{puiss}(a \times a, b-1)$

Si

retourner a

Fin.

$$U \begin{cases} U_1 = 1 \\ U_2 = 1 \\ U_n = U_{n-1} + U_{n-2} \end{cases} \quad \begin{matrix} U_3 = 2. \\ U_4 = U_3 + U_2 = 2 + 1 = 3. \end{matrix}$$

$U_1, U_2, U_3, U_4, U_5, U_6, U_7, U_8, U_9, U_{10}$

fonction suite (m: entier, U1: entier, U2: entier): entier

Debut

Si m > 0 alors

retourner (m-1, U2, U1+U2)

Sinon

retourner U2.

Fin

Suite (m, 1, 1)

fonction suite (m: entier) entier

Debut

Si m = 1 ou m = 2 alors
retourner 1

Sinon
retourner suite(m-1) + suite(m-2)

Fin Si

Fin

