Langage algorithmique qui peuvent tomber aux rattrapages

Petit rappel :
Le mot clé variable est l'inverse d'un const autrement dit il servira pour une variable qui sera modifiée dans le programme lorsqu'on ne met pas le mot clé alors la variable ne sera pas modifiée

Remplissage (tombera à 99.99%)

On part avec comme consigne :

```
CONSTANTE
NBMAX = 100 ;
TYPE
TYPTAB = TABLEAU [ 1 .. NBMAX ] DE CARACTERE ;
VARIABLE
TAB : TYPTAB ;
```

Voici donc notre procédure

```
PROCEDURE Remplissage ( NBMAX : ENTIER ;
VARIABLE TAB : TYPTAB ;
VARIABLE DIM : ENTIER ) ;
VARIABLE
I : ENTIER ;
DEBUT
POUR I VARIANT DE 1 A NBMAX FAIRE
TAB [ I ] <- ‘ ’ ;
FAIT ;

REPETER
ECRIRE ('ENTREZ LA VALEUR DE DIM : ') ;
LIRE (DIM) ;
JUSQU'A ( ( DIM > 0 ) ET ( DIM <= NBMAX ) ) ;

POUR I VARIANT DE 1 A DIM FAIRE
ECRIRE ('TAB [', I, ‘] = ’ ) ;
LIRE (TAB[ I ]);
FAIT ;
FIN ;
```

Procédure Affichage

```
PROCEDURE Affichage ( TAB : TYPTAB ;
DIM : ENTIER ) ;
VARIABLE
I : ENTIER ;
DEBUT
POUR I VARIANT DE 1 A DIM FAIRE
ECRIRE (TAB [ I ], ' ') ;
FAIT ;
FIN ;
```

Procédure ComptageMots

```
PROCEDURE ComptageMots ( TAB : TYPTAB ;
DIM : ENTIER ;
VARIABLE NBMOTS : ENTIER ) ;
VARIABLE
I : ENTIER ;
DEBUT
NBMOTS <- 0 ;
POUr I VARIANT DE 1 A DIM FAIRE
SI ( TAB [ I ] = ‘ ’ ) ;
ALORS
NBMOTS <- NBMOTS + 1 ;
FIN SI ;
FAIT ;
NBMOTS <- NBMOTS + 1 ;
FIN ;
```

Procédure Inversion

```
PROCEDURE Inversion (VARIABLE TAB : TYPTAB ;
DIM : ENTIER ) ;
VARIABLES
I : ENTIER ;
TAMPON : CARACTERE ;
DEBUT
POUR I VARIANT DE 1 A ( DIM DIV 2 ) FAIRE
TAMPON <- TAB [ I ] ;
TAB [ I ] <- TAB [ DIM - I + 1 ] ;
TAB [ DIM - I + 1 ] <- TAMPON ;
FAIT ;
FIN ;
```

Procédure Comptage Occurences

```
PROCEDURE Comptage ( TAB : TYPTAB ;
DIM : ENTIER ;
VAL : CARACTERE ;
VARIABLE NB : ENTIER ) ;
VARIABLE
I : ENTIER ;
DEBUT
NB <- 0 ;
POUR I VARIANT DE 1 A DIM FAIRE
SI ( TAB [ I ] = VAL )
ALORS
NB <- NB + 1 ;
FIN SI ;
FAIT ;
FIN ;
```

Fonction Bigramme

```
FONCTION Bigramme ( TAB : TYPTAB ;
DIM : ENTIER ;
VARIABLE NB : ENTIER ) : BOOLEEN ;
VARIABLES
BIG : TABLEAU [ 1 .. 2 ] DE CARACTERE ;
I : ENTIER ;
DEBUT
ECRIRE (' Entrez le premier caractere du bigramme : ' ) ;
LIRE(BIG [ 1 ] ) ;
ECRIRE('Entrez le second caractere du bigramme : ' ) ;
LIRE(BIG [ 2 ] );
NB <- 0 ;
POUR I VARIANT DE 2 A DIM FAIRE
SI ( ( TAB [ I - 1 ] = BIG [ 1 ] ) ET ( TAB [ I ] = BIG [ 2 ] ) )
ALORS
NB <- NB + 1 ;
FIN SI ;
FAIT ;
RENVOYER ( NB ) ;
FIN ;
```

Fonction factorielle

```
FONCTION Factorielle ( VARIABLE N : ENTIER ) : ENTIER ;
VARIALBES
FACTORIELLE : ENTIER ;
I : ENTIER ;
DEBUT
SI ( N = 0 )
ALORS
POUR I ALLANT DE 1 A N FAIRE
FACTORIELLE <- FACTORIELLE * I ;
FAIT ;
FIN SI ;
RENVOYER FACTORIELLE
FIN ;
```

Fonction Somme avec de la récursivité

```
FONCTION SOMME (N : ENTIER ) : ENTIER ;
DEBUT
SI ( N = 0 )
ALORS
RENVOYER 0 ;
SINON
RENVOYER N + SOMME ( N - 1 ) ;
FIN SI ;
FIN ;
```

Fonction Somme avec boucle

```
FONCTION Somme(N : ENTIER) : ENTIER ;
VARIABLES
S : ENTIER ;
I : ENTIER ;
DEBUT
SI ( N = 0 )
ALORS
POUR I ALLANT DE 1 A N FAIRE
S <- S + S ;
FAIT ;
FIN SI ;
RENVOYER S ;
FIN ;
```

Procédure Est Bissextile

```
PROCEDURE EstBissextile ( A : ENTIER ) ;
DEBUT
SI ( ( A MOD 4 = 0 ) ET ( ( A MOD 100 > 0 ) OU ( A MOD 400 = 0 ) )
ALORS
ECRIRE ('Lannee', A,' est bissextile') ;
SINON
ECRIRE ('Lannee', A,' nest pas bissextile') ;
FIN SI;
FIN ;
```

Procédure de swap

```
PROCEDURE Echange(VARIABLES X , Y : ENTIER ) ;
VARIABLE
AUX : ENTIER ;
DEBUT {Echange}
AUX <- X ;
X <- Y ;
Y <- AUX ;
ECRIRE ('X = ‘, X, ‘ Y = ’, Y) ;
FIN ; {Echange}
```

Procédure nombre premier

```
PROCEDURE EstPremier(N : ENTIER ) ;
VARIABLES
P, I : ENTIER
DEBUT
P <- 0 ;
POUR I ALLANT DE 1 A N FAIRE
SI ( N MOD 1 = 0 )
```

```
ALORS
P <- P + 1 ;
FIN SI ;
FAIT ;
SI ( P = 2)
ALORS
ECRIRE (P, ‘ est un nombre premier’) ;
SINON
ECRIRE (P, ‘ est pas un nombre premier' ) ;
FIN SI ;
FIN ;
```

Decimal à binaire

```
PROGRAMME ConversionBinaire ;
CONSTANTE
MAX = 20 ;
TYPE
TYPTAB = TABLEAU [ 1 .. MAX ] DE ENTIER ;
VARIABLES
N : ENTIER ;
OCT : TYPTAB ;
I, J : ENTIER ;
AUX : ENTIER ;
DEBUT
ECRIRE ('Entrez un nombre en base 10 : ' ) ;
LIRE ( N ) ;
POUR I VARIANT DE 1 A MAX FAIRE
OCT [ I ] <- 0 ;
FAIT ;
I <- 1 ;
TANT QUE ( N >= 2 ) FAIRE
OCT [ I ] <- N MOD 2 ;
N <- N DIV 2 ;
I <- I + 1 ;
FAIT ;
OCT [ I ] <- N ;
POUR J VARIANT DE 1 A ( I DIV 2 ) FAIRE
AUX <- OCT [ J ] ;
OCT [ J ] <- OCT [ I - J + 1 ] ;
OCT [ I - J + 1 ] <- AUX ;
FAIT ;
ECRIRE ('Nombre en base 2');
POUR J VARIANT DE 1 A I FAIRE
ECRIRE (OCT [ J ] , ‘ ’ ) ;
FAIT ;
FIN.
```

Binaire à décimal

```
PROGRAMME BinaireToEntier;
```

```
CONSTANTE
MAX = 20 ;
TYPE
TYPTAB = TABLEAU [ 1 .. MAX ] DE ENTIER ;
VARIABLES
OCT : TYPTAB ;
I : ENTIER ;
NB : ENTIER ;
N : ENTIER ;
EXP : ENTIER ;
DEBUT
POUR I VARIANT DE 1 A MAX FAIRE
OCT [ I ] <- 0 ;
FAIT ;
ECRIRE ( ‘ENTREZ UN NOMBRE EN BASE 2 : ’ ) ;
REPETER
ECRIRE ( ‘ENTREZ LE NOMBRE D’ELEMENTS A SAISIR : ’ ) ;
LIRE ( NB )
JUSQU'A ( NB > 0 ) ;
POUR I VARIANT DE 1 A NB FAIRE
REPETER
ECRIRE ( ‘OCT [’, I, ‘] = ’ ) ;
LIRE ( OCT [ I ] )
JUSQU'A ( ( OCT [ I ] ≥ 0 ) ET ( OCT [ I ] < NB) ;
FAIT ;
N <- 0 ;
EXP <- 1 ;
POUR I VARIANT DE NB A 1 PAS -1 FAIRE
N <- N + OCT [ I ] * EXP ;
EXP <- EXP * 2 ;
FAIT ;
ECRIRE ( ‘NOMBRE EN BASE 10 : ’, N ) ;
FIN.
```

Fibonnaci en récursif

```
FONCTION Fibonacci ( INDEX : ENTIER ) ;
DEBUT
SI ( INDEX < 0 )
ALORS
RENVOYER - 1 ;
FIN SI ;
SI ( INDEX = 0 )
ALORS
RENVOYER 0 ;
FIN SI ;
SI ( INDEX = 1 )
ALORS
RENVOYER 1 ;
FIN SI ;
RENVOYER Fibonacci ( INDEX - 1 ) + Fibonacci ( INDEX - 2 ) ;
FIN ;
```

Tri par selection

Le principe du tri par sélection est que pour classer N valeurs, il faut rechercher la plus petite valuer et la placer en 1 ère position, puis la
plus petite valeur dans les valeurs restantes et la placer en seconde position et ainsi de suite ...

```
PROGRAMME TriSelection ;
CONSTANTE
MAX = 10 ;
TYPE
TYPTAB = TABLEAU [ 1 .. MAX ] DE ENTIER ;
VARIABLES
TAB : TYPTAB ;
I, J : ENTIER ;
MINI, POS : ENTIER ;
DEBUT
POUR I VARIANT DE 1 A MAX FAIRE
ECRIRE ('TAB [', I, '] = ');
LIRE (TAB[I]);
FAIT ;

POUR I VARIANT DE 1 A ( MAX - 1 ) FAIRE
MINI <- TAB [ I ] ;
POS <- I ;
POUR J VARIANT DE ( I + 1 ) A MAX FAIRE
SI ( TAB [ J ] < MINI )
ALORS
MINI <- TAB [ J ] ;
POS <- J ;
FIN SI ;
FAIT ;
TAB [ POS ] <- TAB [ I ] ;
TAB [ I ] <- MINI ;
FAIT ;

POUR I VARIANT DE 1 A MAX FAIRE
ECRIRE (TAB [ I ], ' ' ) ;
FAIT ;
FIN .
```

Tri insertion

Le tri par insertion consiste à piocher une à une les valeurs du tableau et à les insérer au bon endroit, dans le tableau trié constitué des valeurs précédemment piochées et triées.

```
PROGRAMME TriInsertion ;
CONSTANTE
MAX = 10 ;
```

```
TYPE
TYPTAB = TABLEAU [ 1 .. MAX ] DE ENTIER ;
VARIABLES
TAB : TYPTAB ;
I , J : ENTIER ;
MEM : ENTIER ;
DEBUT
POUR I VARIANT DE 1 A MAX FAIRE
ECRIRE ('TAB [' , I , ‘ ] = ’ ) ;
LIRE ( TAB [ I ] ) ;
FAIT ;

POUR I VARIANT DE 2 A MAX FAIRE
MEM <- TAB [ I ] ;
J <- i ;
TANT QUE ( ( J > 1 ) ET ( TAB [ J - 1 ] > MEM ) ) FAIRE
TAB [ J ] <- TAB [ J - 1 ] ;
J <- J - 1 ;
FAIT ;
TAB [ J ] <- MEM ;
FAIT ;

POUR I VARIANT DE 1 A MAX FAIRE
ECRIRE ( TAB [ I ], ‘ ’ );
FAIT ;
FIN .
```

Tri à bulles

L'algorithme du tri à bulles consiste à regarder les différentes valeurs adjacentes d'un tableau de taille N, et à les permuter si le premier
des deux éléments est supérieur au second.

```
PROGRAMME Tri Bulles
CONSTANTE
MAX = 10 ;
TYPE
TYPTAB = TABLEAU [ 1 .. MAX ] DE ENTIER ;
VARIABLES
TAB : TYPTAB ;
I, J : ENTIER ;
MEM : ENTIER ;
DEBUT
POUR I VARIANT DE 1 A MAX FAIRE
ECRIRE ('Tab [, I, ‘] = ’) ;
LIRE ( TAB [ I ] ) ;
FAIT ;

POUR I VARIANT DE N A 1 FAIRE
POUR J VARIANT DE 2 A I FAIRE
SI ( TAB [ J - 1 ] > TAB [ J ] )
```

```
ALORS
MEM <- TAB [ J - 1 ] ;
TAB [ J - 1 ] <- TAB [ J ];
TAB [ J ] <- MEM ;
FIN SI ;
FAIT ;
FAIT ;

POUR I VARIANT DE 1 A MAX FAIRE
ECRIRE ( TAB [ I ] , ‘ ’ ) ;
FAIT ;
FIN .
```