LES algorithmes arithmétiques

Objectifs :

- L'élève sera capable de manipuler des algorithmes permettant de résoudre des traitements de calcul :
  - Du PGCD, du PPCM,
  - De décomposition en facteurs premiers,
  - Du factoriel,
  - De conversion entre bases de numérotation

I. Le calcul du PGCD :

a) Définition :

Le PGCD (**P**lus **G**rand **C**ommun **D**iviseur) de deux entiers est le plus grand entier permettant de diviser ces deux entiers

Activité : Ecrire un programme qui permet de saisir deux entiers **A** et **B** strictement positifs, puis calculer et afficher leur PGCD.

b) Exemple :

Pour **A = 64** et **B = 38**, on procède comme suit pour calculer le PGCD :
Si A = B alors le PGCD est A ou B
Si non si A > B alors A = A – B
Si non B = B – A

Etape1: (A>B) A = 64 – 38 = 26 **(A ≠ B)**
Etape2: (A<B) B = 38 – 26 = 12 **(A ≠ B)**
Etape3: (A>B) A = 26 – 12 = 14 **(A ≠ B)**
Etape4: (A>B) A = 14 – 12 = 2 **(A ≠ B)**
Etape5: (A<B) B = 12 – 2 = 10 **(A ≠ B)**
Etape6: (A<B) B = 10 – 2 = 8 **(A ≠ B)**
Etape7: (A<B) B = 8 – 2 = 6 **(A ≠ B)**
Etape8: (A<B) B = 6 – 2 = 4 **(A ≠ B)**
Etape9: (A<B) B = 4 – 2 = 2 **(A = B) Arrêt**

c) Analyses et algorithmes du problème :

1. **Analyse du programme principale :**
   - **Résultat :** Afficher le PGCD de deux entiers
   - **Traitements :** Il faut calculer le PGCD, en utilisant une fonction **PGCD**
   - **Données :** Il faut saisir deux entiers strictement positifs A et B, en utilisant la procédure **Saisie**

2. **Algorithme du programme principal :**

0) Début Calcul_PGCD
1) Saisie (A, B)
2) Ecrire ("PGCD (", A, ", ", B, ") = ", **PGCD** (A, B))
3) Fin Calcul_PGCD

Tableau de déclaration des Objets

Objets	Type/Nature
A, B	Entier
Saisie	Procédure
PGCD	Fonction

3. **Analyse de la procédure Saisie :**
   - **Résultat :** Saisir **Af** et **Bf**
   - **Traitements :** La saisie des entiers **Af** et **Bf** doit être contrôlée pour ne pas saisir un entier négatif ou nul. Cette procédure admet deux paramètres formels qui sont **Nf** et **Bf.**

4. **Algorithme de la procédure Saisie :**

```
0) Début procédure Saisie (VAR Af, Bf : Entier)
1) Répéter
       Ecrire ("Donner le premier entier : "), Lire (Af)
   Jusqu'à (Af > 0)
2) Répéter
       Ecrire ("Donner le deuxième entier : "), Lire (Bf)
   Jusqu'à (Bf > 0)
3) Fin Saisie
```

5. **Analyse de la fonction PGCD :**
   - **Résultat :** Déterminer le **PGCD** de deux entiers **Af** et **Bf**
   - **Traitements :** Si **Af = Bf** alors le **PGCD** est **Af** ou **Bf** Si non il s'agit d'un traitement répétitif à condition d'arrêt, la boucle utiliser est TANT … QUE. Dans cette boucle on teste si **Af > Bf** alors **Af = Af – Bf si non Bf = Bf - Af**

   Les paramètres formels de cette fonction sont **Af**, **Bf**

6. **Algorithme de la fonction PGCD :**

```
0) Début fonction PGCD (Af, Bf : Entier) : Entier
1) Tant que (Af ≠ Bf) Faire
       Si (Af > Bf) Alors
              Af ← Af – Bf
       Sinon
              Bf ← Bf – Af
       FinSi
   Fin Tant que
2) PGCD ← Af
3) Fin PGCD
```

II. Le calcul du PPCM :

a) Définition :

Le PPCM (**P**lus **P**etit **C**ommun **M**ultiple) de deux entiers est le plus petit entier multiple à la fois de ces deux entiers

Activité : Ecrire un programme qui permet de saisir deux entiers **A** et **B** strictement positifs, puis calculer et afficher leur PPCM.

b) Exemple :

Pour **A = 64** et **B = 3**, on procède comme suit pour calculer le PPCM :

On cherche tout d'abord le minimum **(Min)** et le maximum **(Max)** entre **A** et **B**, puis on calcule le reste de la division de **Max** par **Min**, s'il est égale à zéro alors **Max** est le PPCM, sinon on cherche le multiple successive de **Max** et **Min** qui est égale à **Max + A + B – Min** est ainsi de suite jusqu'à ce qu'on trouve comme reste de division de **Max** par **Min** égale à zéro.

Etape1: (A > B) Max = A et Min = B

Etape2: Max **MOD** Min = 64 **MOD** 3 = 1 ≠ **0 Alors** Max = 64 + 64 + 3 – 3 = 128

Etape3: Max **MOD** Min = 128 **MOD** 3 = 2 ≠ **0 Alors** Max = 128 + 64 + 3 – 3 = 192

Etape4: Max **MOD** Min = 192 **MOD** 3 = 0 **Alors Arrêt** avec **PPCM = Max = 192**

c) Analyses et algorithmes du problème :

1. **Analyse du programme principale :**
   - **Résultat :** Afficher le PPCM de deux entiers
   - **Traitements :** Il faut calculer le PPCM, en utilisant une fonction **PPCM**

- **Données :** Il faut saisir deux entiers strictement positifs A et B, en utilisant la procédure **Saisie**

2. **Algorithme du programme principal :**

0) **Début** Calcul_PPCM
1) **Saisie** (A, B)
2) **Ecrire** ("PPCM (", A, ", ", B, ") = ", **PPCM** (A, B))
3) **Fin** Calcul_PPCM

Tableau de déclaration des Objets

Objets	Type/Nature
A, B	Entier
Saisie	Procédure
PPCM	Fonction

3. **Analyse de la fonction PPCM :**

- **Résultat :** Déterminer le **PPCM** de deux entiers **Af** et **Bf**
- **Traitements :** Il s'agit de cherche tout d'abord le minimum **(Min)** et le maximum **(Max)** entre **Af** et **Bf**, puis on calcule le reste de la division de **Max** par **Min**, s'il est égale à zéro alors **Max** est le PPCM, sinon on cherche le multiple successive de **Max** et **Min** qui est égale à **Max + A + B – Min** est ainsi de suite jusqu'à ce qu'on trouve comme reste de division de **Max** par **Min** égale à zéro. On constate que c'est un traitement répétitif à condition d'arrêt, la boucle utiliser sera TANT QUE … FAIRE

Les paramètres formels de cette fonction sont **Af**, **Bf**

4. **Algorithme de la fonction PPCM :**

0) **Début fonction** PPCM (Af, Bf : Entier) **: Entier**
1) **Si** (Af > Bf) **Alors**
   Max ← Af
   Min ← Bf
   Sinon
   Max ← Bf
   Min ← Af
   Finsi
2) **Tant que** (Max **MOD** Min ≠ 0) **Faire**
   Max ← Max + Af + Bf - Min
   Fin Tant que
3) **PPCM** ← Max
4) **Fin** PPCM

Tableau de déclaration des Objets locaux

Objets	Type/Nature
Max, Min	Entier

III. Les nombres premiers:

a) Définition :

Un nombre est dit premier s'il ne se divise que par 1 eu lui-même, exemple 7.

Activité : Ecrire un programme qui permet de saisir un entier **A** strictement positif, puis vérifier s'il est premier ou non et afficher le résultat.

b) Analyses et algorithmes du problème :

1. **Analyse du programme principale :**

- **Résultat :** Afficher le nombre est premier ou non
- **Traitements :** Il faut vérifier si le nombre est premier ou non, en utilisant une fonction **Premier**
- **Données :** Il faut saisir un entier strictement positif A, en utilisant la procédure **Saisie**

2. **Algorithme du programme principal :**

0) **Début** Verif_Premier
1) **Saisie** (A)
2) **Si Premier** (A) = Vrai **Alors**
   Ecrire (A, " est un nombre premier")
   Sinon
   Ecrire (A, " n'est pas un nombre premier")
   Finsi
3) **Fin** Verif_Premier
4)

Tableau de déclaration des Objets

Objets	Type/Nature
A	Entier
Saisie	Procédure
Premier	Fonction

3. **Analyse de la fonction Premier :**
   - **Résultat :** Vérifier si un nombre **Af** est premier ou non par un résultat booléen.
   - **Traitements :** Il s'agit de vérifier s'il y a un diviseur de Af dans l'intervalle [2..Af **div** 2]

   Cette fonction possède un seul paramètre formel **Af.**

4. **Algorithme de la fonction Premier :**

```
0) Début fonction Premier (Af : Entier) : Booléen
1) [i ← 2, Verif ← Vrai] Tant que (i<= Af div 2) et (verif = Vrai) Faire
       Si (Af Mod i = 0) Alors
              Vrif ← Faux
       Sinon
              i ← i + 1
       Finsi
   Fin Tant que
2) Premier ← Verif
3) Fin Premier
```

Tableau de déclaration des Objets locaux

Objets	Type/Nature
i	Entier
Verif	Booléen

IV. La décomposition en facteurs premiers :

a) Définition :

La décomposition d'un entier en produits facteurs premiers consiste à écrire cet entier sous la forme de produits de ces diviseurs.

Exemple : 124 = 31 * 2 * 2

Activité : Ecrire un programme qui permet de saisir un entier **A** strictement supérieur à 1, puis chercher et afficher la décomposition en produits de facteurs premiers de cet entier.

b) Spécifications et algorithmes du problème :

1. **Analyse du programme principale :**
   - **Résultat :** Afficher la décomposition en utilisant la procédure **Affiche**.
   - **Traitements :** Il faut remplir un tableau contenant les facteurs premiers de l'entier A dans l'ordre croissant, en utilisant une procédure **Remplir_FP**
   - **Données :** Il faut saisir un entier strictement supérieur à 1, en utilisant la procédure **Saisie**

2. **Algorithme du programme principal :**

```
0) Début Fact_Premier
1) Saisie (A)
2) Remplir_FP (A, N, F)
3) Affiche (F, A, N)
4) Fin Fact_Premier
```

Tableau de déclaration des nouveaux types

Types
TAB = Tableau de 100 entiers

Tableau de déclaration des Objets

Objets	Type/Nature
A, N	Entier
Saisie, Remplir_FP, Affiche	Procédure
F	TAB

3. **Analyse de la procédure Remplir_FP :**
   - **Résultat :** Remplir le tableau **Ff** par les facteurs premiers d'un entier **Af**.
   - **Traitements :** il s'agit d'un traitement répétitif jusqu'à ce que la valeur de Af devienne égale à 1, la boucle adéquate est REPETER … JUSQU'A. A chaque fois on divise Af par un compteur i, s'il est divisible on range le contenu du compteur dans le tableau et on incrémente le compteur de 1.

   Cette procédure admet un trois paramètres formels qui sont **Af**, **Nf et Ff.**

4. **Algorithme de la procédure Remplir_FP :**

```
0) Début procédure Remplir_FP (Af : Entier ; VAR Nf : Entier ; VAR Ff : TAB)
1) [i ← 2, Nf ← 0] Répéter
```

```
        Si (Af Mod i) = 0 Alors
          Af ← Af div i
          Nf ← Nf + 1
          Ff[Nf] ← i
        Sinon
          i ← i + 1
        Finsi
      Jusqu'à (Af =1)
2) Fin Remplir_FP
```

Tableau de déclaration des Objets locaux

Objets	Type/Nature
i	Entier

5. **Analyse de la procédure Affiche :**
   - **Résultat :** Afficher le tableau **Ff**
   - **Traitements :** Il s'agit d'un traitement répétitif pour afficher chaque élément du tableau **Ff** , donc l'instruction d'affichage va être exécuter **Nf** fois, le nombre de répétition est connu d'avance, d'où utilisation de la boucle POUR … FAIRE…

   Les paramètres formels de cette procédure sont **Ff**, **Nf** et **Af**.

6. **Algorithme de la procédure Affiche :**

```
0) Début procédure Affiche (Ff : TAB ; Af, Nf : Entier)
1) Ecrire (Af, " = ", Ff[1])
2) Pour i de 2 à Nf Faire
     Ecrire (" * ", Ff[i])
   Fin Pour
3) Fin Affiche
```

Tableau de déclaration des Objets locaux

Objets	Type/Nature
i	Entier

V. Le calcul de factoriel :

a) Définition :

La factorielle d'un entier donné est le produit de tous les entiers compris entre 1 et cet entier. Mathématiquement on le note par **ident_entier !**

Exemple : 5 ! = 5 * 4 * 3 * 2 * 1 = 120

Activité : Ecrire un programme qui permet de saisir un entier **A** positif, puis calculer et afficher la factorielle de cet entier.

b) Spécifications et algorithmes du problème :

1. **Analyse du programme principale :**
   - **Résultat :** Afficher la factorielle.
   - **Traitements :** Il faut calculer la factorielle de l'entier A, en utilisant une fonction **Fact**
   - **Données :** Il faut saisir un entier A positif, en utilisant la procédure **Saisie**

2. **Algorithme du programme principal :**

```
0) Début Factoriel
1) Saisie (A)
2) Ecrire (A, " ! = ", Fact(A))
3) Fin Factoriel
```

Tableau de déclaration des Objets

Objets	Type/Nature
A	Entier
Saisie	Procédure
Fact	Fonction

3. **Analyse de la fonction Fact :**
   - **Résultat :** la factorielle d'un entier **Af**.
   - **Traitements :** Il s'agit de déterminer la valeur du produit de Af*(Af-1)*…*3*2*1. donc c'est une traitement répétitif qui nécessite une structure itérative complète puisqu'on connaît le nombre répétition du traitemet, d'où utilisation de la boucle POUR …. FAIRE

   Cette fonction possède un seul paramètre formel **Af.**

4. **Algorithme de la fonction fact :**

```
0) Début fonction Fact (Af : Entier) : Entier
1) [F ← 1] Pour i de 1 à Af Faire
     F ← F * i
   Fin Pour
2) Fact ← F
3) Fin Fact
4)
```

Tableau de déclaration des Objets locaux

Objets	Type/Nature
i, F	Entier

VI. Conversion entre les bases de numération :

a) Exemple :

Exemple1 :

(**38**)/10 = (**100110**)/2, on obtient ce résultat en effectuant des divisions successives par 2 et en écrivant les reste en ordre inverse de leur apparition.

Quotient	**38**	**19**	**9**	**4**	**2**	**1**	**0**
Reste	**0**	**1**	**1**	**0**	**0**	**1**	**Arrêt**

Exemple2 :

(**142**)/10 = (**8E**)/16, on obtient ce résultat en effectuant des divisions successives par 16 et en écrivant les reste en ordre inverse de leur apparition. Si le reste est supérieur à 10 le chiffre se présente par une lettre qu'on l'obtient par la formule suivante : **CHR** (**ORD** ('A') + Reste – 10)

Quotient	**142**	**8**	**0**
Reste	**14 (E)**	**8**	**Arrêt**

b) Définition :

La conversion d'un entier positif consiste à chercher sa représentation dans une base **b** au moins égale à **2**. La représentation est formée par naturels compris entre **0** et **b-1** (où b est la base de numérotation).

Activité : Ecrire un programme qui permet de saisir un entier **A** positif, puis le convertir en base 2 et afficher le résultat.

c) Analyses et algorithmes du problème :

1. **Analyse du programme principale :**
   - **Résultat :** Afficher la conversion.
   - **Traitements :** Il faut convertir l'entier A en base 2, en utilisant une fonction **Convert**
   - **Données :** Il faut saisir un entier A positif, en utilisant la procédure **Saisie.**

2. **Algorithme du programme principal :**

```
0) Début Conversion
1) Saisie (A)
2) Ecrire (" (", A, ")/10 = (", Convert (A), ")/2")
3) Fin Conversion
```

Tableau de déclaration des Objets

Objets	Type/Nature
A	Entier
Saisie	Procédure
Convert	Fonction

3. **Analyse de la fonction Convert :**
   - **Résultat :** la conversion en base 2 d'un entier **Af**.
   - **Traitements :** Il s'agit de calculer le reste de la division de **Af** par 2, la convertir en chaîne et l'introduire dans une chaîne résultat, puis on répète ce traitement jusqu'à ce que **Af** devienne égale à 0. donc nous utilisation de la boucle REPETER … JUSQU'A

   Cette fonction possède un seul paramètre formel **Af.**

4. **Algorithme de la fonction Convert :**

```
0) Début fonction Convert (Af : Entier) : Chaîne
1) [ch ← ""] Répéter
       r ← Af Mod 2
       Convch (r, ch1)
       Ch ← ch1 + ch
       Af ← Af Div 2
   Jusqu'à (Af = 0)
2) Convert ← ch
3) Fin Convert
```

Tableau de déclaration des Objets locaux

Objets	Type/Nature
r	Entier
ch, ch1	chaîne