Chapitre 5

Les algorithmes d'arithmétique

Objectifs

- Acquérir des habilités de résolution de problèmes à travers l'apprentissage des algorithmes de calcul numérique,
- Proposer des solutions récursives à quelques problèmes arithmétiques.

Plan du chapitre

Chapitre 5

Les algorithmes d'arithmétique

I- Préambule

L'arithmétique est une branche des mathématiques qui étudie les relations entre les nombres. C'est aussi l'étude des nombres et des opérations élémentaires entre eux.
L'origine du mot arithmétique est :

- du grec : arithmétiké
  - arithmos : nombre
  - techné : art
- du latin : arithmetica

L'arithmétique faisait partie de la géométrie des Grecs anciens : études des nombres figurés. L'adoption de la numération de position à base dix fait réaliser d'immenses progrès, même si :

- La base 2 est universelle pour les ordinateurs
- La base 12 subsiste pour les heures
- La base 60 pour les angles et les durées en minutes et secondes

Exemples d'études arithmétiques :

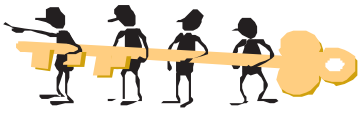
- Test de primalité
- Nombres parfaits
- Test de parité
- Somme de puissances
- Calcul de la factorielle
- Calcul de PGCD et PPCM, etc.

II- Calcul du PGCD (solution récursive)

II.1 Activité 1

Question 1 : Proposez une analyse, puis déduisez l'algorithme d'une fonction permettant de calculer le PGCD (le Plus Grand Commun Diviseur) de deux entiers positifs non nuls a et b, en utilisant la méthode de la différence.
Essayons cette méthode sur l'exemple suivant : a = 22 et b = 6

PGCD (22, 6) = PGCD (22-6, 6) = PGCD (16, 6)	Car 22 > 6
= PGCD (16-6, 6) = PGCD (10, 6)	Car 16 > 6
= PGCD (10-6, 6) = PGCD (4, 6)	Car 10 > 6
= PGCD (4, 6-4) = PGCD (4, 2)	Car 4 < 6
= PGCD (4-2, 2) = PGCD (2, 2)	Car 4 > 2
= 2	Car 2 = 2



Analyse de la fonction PGCD

Résultat : PGCD de a et b

Traitement : Tant que a ≠ b Faire

Si a > b **Alors** a ← a - b

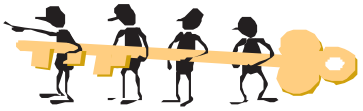
Sinon b ← b - a

❖Nous continuons à remplacer la plus grande valeur parmi a et b par la différence jusqu'à a = b

Algorithme de la fonction PGCD (solution itérative)

0) Fonction PGCD (a, b : Entier) : Entier
1) **Tant que** (a ≠ b) **Faire**
Si (a > b) **Alors** a ← a - b
Sinon b ← b - a
Fin Si
Fin Tant que
2) PGCD ← a
3) Fin PGCD

Question 2 : A partir de l'algorithme donné ci-dessus, dégagez une relation récursive de la fonction PGCD.

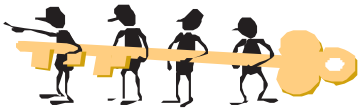


Essayons de dégager cette relation à partir de l'exemple présenté précédemment : a = 22 et b = 6

La relation récursive dégagée est la suivante :

Si a = b alors PGCD (a, b) = a
Sinon si a > b alors PGCD (a, b) = PGCD (a-b, b)
Sinon PGCD (a, b) = PGCD (a, b-a).

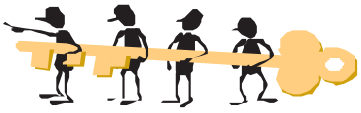
Question 3 : Déduisez l'algorithme récursif de la fonction PGCD.



0) Fonction PGCD (a, b : Entier) : Entier
1) **Si** (a = b) Alors PGCD ← a
Sinon Si (a > b) Alors PGCD ← FN PGCD (a-b,b)
Sinon PGCD ← FN PGCD (a,b-a)
Fin Si
2) Fin PGCD

II.2 Application

Traduisez et testez la solution récursive du problème permettant d'afficher le PGCD de deux entiers positifs et non nuls a et b donnés, en utilisant la méthode de la différence. Enregistrez votre programme sous le nom **PGCD_rec**.



```
Program Pgcd_diff;
Uses Crt;
Var
  a,b : Integer;
Procedure saisie (Var z : Integer);
 Begin
   Repeat
   Write('donner un entier positif et non nul : ');
   Readln(z);
   Until (z>0) ;
 End;
Function pgcd(n, m : Integer) : Integer;
 Begin
   If (n = m) Then pgcd := m
   Else If (n > m) Then pgcd := pgcd(n - m, m)
   Else pgcd := pgcd(n, m - n);
 End;

{Programme Principal}
 Begin
   Saisie(a);
   Saisie(b);
   Write('pgcd(',a,',',b,')= ', pgcd(a,b));
 End.
```

III- Calcul de A_n^p et C_n^p

III.1 Présentation

COMBINATOIRE DE P OBJETS PARMI N

Exemple : p = 3 et n = 5

PERMUTATIONS	**Ordre**	**Per = p!**	**6**
ARRANGEMENTS	**Ordre**	**A = n! / (n-p)!**	**60**
COMBINAISONS	**Sans ordre**	**C = A/P = n!/(n-p)!p!**	**10**

D'où

Arrangement de p éléments parmi n

C'est le nombre de permutations ordonnées possibles de p éléments parmi n
Exemple avec {a,b,c}: A(2,3) = 6
{a,b}, {b,a}, {a,c}, {c,a}, {b,c}, {c,b}

Combinaison de p éléments parmi n

C'est le nombre de permutations sans ordre possibles de p éléments parmi n
Exemple avec {a,b,c}: C(2,3) = 3
{a,b}, {a,c}, {b,c}

III.2 Calcul de A_n^p

Activité 1

D'après vos connaissances en Mathématiques,

1) Définissez un arrangement de p éléments d'un ensemble E de n éléments.

→ Un arrangement de p éléments d'un ensemble E à n éléments est un p-uplet d'éléments distincts de E.

2) Donnez la formule mathématique du nombre d'arrangements, noté $\mathbf{A_n^p}$

→ Le nombre d'arrangements de p éléments de l'ensemble E est représenté par la notation suivante : $\mathbf{A_n^p} = n(n-1)(n-2)\ldots(n-p+1)$

Ou encore $\mathbf{A_n^p} = \dfrac{n!}{(n-p)!}$

3) Quelles conditions doivent vérifier les entiers n et p pour valoir la définition et la formule données précédemment ?

→ n et p sont des entiers qui vérifient la condition suivante : $1 \leq p \leq n$

4) Proposez une analyse, puis déduisez les algorithmes correspondants au problème permettant de chercher puis d'afficher le $\mathbf{A_n^p}$ de deux entiers donnés n et p, avec $(1 \leq p \leq n)$.

a) Analyse du problème

Résultat : Affichage de $\mathbf{A_n^p}$. L'affichage sera représenté sous la forme suivante **A(n, p)** : Ecrire ("A(", n, ",", p, ") = ", **FN** Arrange (n,p))

Données : deux entiers n et p, introduits grâce à la procédure **Saisie**

b) Algorithme du programme principal et codification des objets

0) Début Arrangement
1) **Proc** Saisie (n, p)
2) Ecrire ("A (", n, ",", p, ") = ", **FN** Arrange (n,p))
3) Fin Arrangement

Tableau de codification des objets globaux

Objets	Type / Nature	Rôle
n,p	Entier	Données
Saisie	Procédure	Permet de saisir n et p
Arrange	Fonction	Permet de calculer $\mathbf{A}_{\mathbf{n}}^{\mathbf{p}}$

c) Analyse de la fonction Arrange

Résultat : Arrange
Traitement :
Arrange ← ar
[ar ← 1]
Pour i de a à a-b+1 (pas = -1) **Faire**
ar ← ar * i

} Structure itérative complète utilisée pour calculer le résultat de l'opération a(a-1)(a-2)…(a-b+1)

d) Algorithme de la fonction Arrange

0) Fonction Arrange (a, b : Entier) : Réel
1) ar ← 1
Pour i de a à a-b+1 (pas = -1) **Faire**
ar ← ar * i
FinPour
2) Arrange ← ar
3) Fin Arrange

e) Traduction en Pascal de la fonction Arrange

```
Function Arrange (a, b : Integer) : Real;
Var
  i, ar : Integer;
Begin
   ar := 1;
   For i := a Downto a-b+1 Do
      ar := ar*i;
   Arrange := ar;
End;
```

Questions :

1) Avec l'assistance de votre enseignant, traduisez et testez la solution du problème permettant de chercher puis d'afficher le A_n^p de deux entiers donnés n et p, avec $1 \leq p \leq n$. Enregistrez votre programme sous le nom **Arrang_1.**
2) Exécutez le programme obtenu pour trouver les valeurs suivantes :

A_5^3 = ... A_7^4 = ... A_{10}^2 = ... A_{13}^1 = ...

3) En essayant plusieurs jeux d'exécution :
- Ecrivez plus simplement la valeur A_n^1 = ...
- Comparez les valeurs A_n^n = ... et A_n^{n-1} = ...

III.3 Calcul de C_n^p

Activité 2

D'après vos connaissances en Mathématiques,
1) Définissez une combinaison de p éléments d'un ensemble E fini, non vide et de n éléments.

→ Une combinaison de p éléments d'un ensemble E de n éléments est une partie de E formée par p éléments.

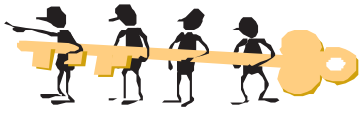
2) Comment représenter le nombre de combinaisons de p éléments de l'ensemble E ? Donnez sa formule mathématique.

→ Le nombre de combinaisons de p éléments de l'ensemble E est représenté par la notation suivante : $C_n^p = \frac{A_n^p}{p!} = \frac{n!}{p!(n-p)!}$

3) Quelles conditions doivent vérifier les entiers n et p pour valoir la définition et la formule données précédemment ?

→ n et p sont des entiers qui vérifient la condition suivante : $0 \leq p \leq n$

Proposez une analyse, puis déduisez les algorithmes correspondants au problème permettant de chercher puis d'afficher le C_n^p de deux entiers donnés n et p, avec $0 \leq p \leq n$.



a) Analyse du problème

Résultat : Affichage de C_n^p . L'affichage sera représenté sous la forme suivante C(n,p) :

Ecrire ("C (", n, ",", p, ") = ", FN Comb (n,p))

Données : deux entiers n et p introduits grâce à la procédure Saisie

b) Algorithme du programme principal et codification des objets

0) Début Combinaison_1
1) **Proc** Saisie (n,p)
2) Ecrire ("C(", n, ",", p, ") = ", **FN** Comb (n,p))
3) Fin Combinaison_1

Tableau de codification des objets globaux

Objets	Type / Nature	Rôle
n,p	Entier	Données
Saisie	Procédure	Permet de saisir n et p
Comb	Fonction	Permet de calculer C_n^p

c) Analyse de la fonction Comb

Résultat : Comb

Traitement : Comb ← **FN** Factorielle(n) / (**FN** Factorielle(p) * **FN** Factorielle(n-p))

NB : La fonction Factorielle a été déjà l'objet d'un apprentissage précédent.

d) Algorithme de la fonction Comb

0) Fonction Comb (n, p : Entier) : Réel
1) Comb ← **FN** Factorielle (n) / (FN Factorielle (p) * FN Factorielle (n-p))
2) Fin Comb

Questions :

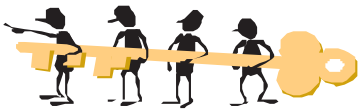
1) Avec l'assistance de votre enseignant, traduisez et testez la solution du problème permettant de chercher puis d'afficher le C_n^p de deux entiers donnés n et p, avec $0 \leq p \leq n$. Enregistrez votre programme sous le nom **Comb_1.**
2) Faîtes les modifications nécessaires pour proposer une solution utilisant la fonction Arrange traitée précédemment. Enregistrez votre nouveau programme sous le nom **Comb_2**.
3) Exécutez le programme obtenu pour trouver les valeurs suivantes :

C_{11}^1 = ... C_{32}^0 = ... C_{21}^{21} = ... C_9^8 = ...

III.4 Application

Nous voulons utiliser un procédé récursif pour chercher puis afficher le C_n^p de deux entiers donnés n et p, avec $0 \leq p \leq n$.

- Proposez une analyse à la fonction Comb,
- Déduisez l'algorithme correspondant,
- Traduisez et testez la solution obtenue. Enregistrez votre programme sous le nom **Comb_rec**.



a) Analyse de la fonction récursive Comb

Résultat : Comb

Traitement : D'après vos connaissances en Mathématiques, vous pouvez dégager la relation suivante :

Si p = 0 **ou** p = n **alors** $C_n^p = 1$

Sinon $C_n^p = C_{n-1}^{p-1} + C_{n-1}^{p}$

D'où **Si** (p = 0) **OU** (p = n) **alors** Comb ← 1
Sinon Comb ← FN Comb (n-1, p) + FN Comb (n-1, p-1)

b) Algorithme de la fonction récursive Comb

0) Fonction Comb (n, p : Entier) : Réel
1) **Si** (p = 0) **ou** (p = n) **alors** Comb ← 1
Sinon Comb ← FN Comb (n-1, p) + FN Comb (n-1, p-1)
FinSi
2) Fin Comb

c) Traduction en Pascal de la fonction récursive Comb

```
Function Comb(n, p : integer) : real;
Begin
      If (p = 0) Or (p = n) Then Comb := 1
      Else Comb := Comb(n-1,p) + Comb(n-1,p-1);
End;
```

IV- Quelques règles de divisibilité

Dans cette partie du chapitre, les problèmes ne seront pas résolus directement avec l'opérateur arithmétique **Mod**. Nous allons, par contre, analyser quelques problèmes répondant aux règles de divisibilité appropriées.

IV.1 Définitions

Un entier n est ***divisible*** par un entier m, si le reste de la division euclidienne de n par m est nul.

Chapitre

5

Une règle ***de divisibilité*** est une séquence d'opérations simples qui permet de reconnaître rapidement si un entier est divisible par un autre sans qu'il soit nécessaire d'effectuer la division. Ces règles sont généralement applicables pour les grands nombres.

IV.2 Divisibilité par 3

Activité 3

Soit les entiers suivants : 6, 42, 191, 39, 41, 30, 20, 13 et 8.
Dans la liste d'entiers donnée ci-dessus, encerclez ceux qui sont divisibles par 3.

Réponse :

(6), (42), 19, (39), 41, (30), 20, 13 et 8

Constatations :

Pouvez-vous trouver une propriété commune entre les entiers encerclés et expliquant la raison pour laquelle ils sont tous divisibles par 3.

Réponse :

Nous constatons que la somme des chiffres de chacun d'eux est divisible par 3 :

6 est divisible par 3,
$4 + 2 = 6 \Rightarrow$ 6 est divisible par 3,
$3 + 9 = 12 \Rightarrow 1 + 2 = 3$ est divisible par 3,
$3 + 0 = 3 \Rightarrow$ 3 est divisible par 3.
Par contre :
$1 + 9 + 1 = 1 + 1 = 2$ n'est pas divisible 3,
$4 + 1 = 5$ n'est pas divisible par 3,
et de même pour 20, 13 et 8.

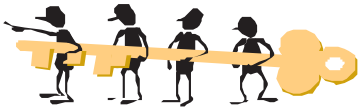
Conclusion :

Un entier est divisible par 3, si la somme des chiffres qui le composent est divisible par 3.

Application :

Nous proposons de vérifier si un entier n donné est divisible par 3, en utilisant la règle de divisibilité citée ci-dessus.

1- Proposez une analyse modulaire au problème,
2- Déduisez les algorithmes des modules dégagés,
3- Traduisez et testez la solution obtenue. Enregistrez votre programme sous le nom **Divi_3**.



a) Analyse du problème

Résultat : Ecrire (s, **FN** Div_3 (s))

Données : un nombre entier représenté sous forme d'une chaîne s, entrée grâce à la procédure Saisie.

b) Algorithme du programme principal et codification des objets

0) Début Divisible_3
1) **Proc** Saisie (s)
2) Ecrire (s, **FN** Div_3 (s))
3) Fin Divisible_3

Tableau de codification des objets globaux

Objets	Type / Nature	Rôle
s	Chaîne	Représente le nombre à tester
Saisie	Procédure	Permet de saisir s
Div_3	Fonction	Permet de retourner un message indiquant la divisibilité ou non du nombre donné par 3.

c) Analyse de la fonction Div_3

Résultat : Div_3

Traitement :

- Si la somme des chiffres composant le nombre à tester est divisible par 3 alors ce nombre l'est aussi sinon il n'est pas divisible par 3.

```
Si Somme Mod 3 = 0 alors
        Div_3 ← " est divisible par 3."
Sinon
        Div_3 ← " n'est pas divisible par 3."
Finsi
```

- [Somme ← 0]

Utiliser une structure itérative complète pour faire un parcours total de la chaîne numérique ch afin de cumuler la somme de ses chiffres.

```
Pour i de 1 à Long (ch) Faire
    Valeur (ch[i], nb, e)
    Somme ← Somme + nb
FinPour
```

d) Algorithme de la fonction Div_3

0) Fonction Div_3 (ch : Chaîne) : Chaîne
1) Somme ← 0
   Pour i de 1 à Long (ch) **Faire**
      Valeur (ch[i], nb, e)
      Somme ← Somme + nb
   FinPour
2) **Si** Somme Mod 3 = 0 **alors** Div_3 ← " est divisible par 3."
   Sinon Div_3 ← " n'est pas divisible par 3."
   Finsi
3) Fin Div_3

Tableau de codification des objets locaux

Objets	Type / Nature	Rôle
Somme	Entier	Somme des chiffres de ch
nb	Entier	Conversion en numérique d'un caractère de ch
e	Entier	Indicateur d'erreurs lors de la conversion en numérique
i	Entier	Compteur

e) Traduction en Pascal du problème Divisible_3

```
Program Divisible_3;
Uses Crt;
Var s : String;

Procedure saisie (Var ch : String);
var num : Longint; e: Integer;
Begin
 Repeat
   Write('donner un entier : ');
   Readln(ch);
   Val(ch, Num, e);
 Until (e = 0)
End;

Function Div_3(ch : String) : String;
Var e, nb, somme, i : Integer;
Begin
  somme := 0;
  For i := 1 To Length(ch) Do
  Begin
    Val(ch[i], nb, e);
    somme := somme + nb
  End;
  If Somme Mod 3 = 0 Then div_3 := ' est divisible par 3.'
  Else div_3 := ' n''est pas divisible par 3.';
End;
```

```
{Programme Principal}
Begin
  Saisie(s);
  Write(s, Div_3(s));
End.
```

IV.3 Divisibilité par 4

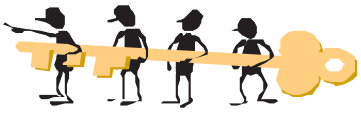
Un entier est divisible par 4, si le nombre composé des deux derniers chiffres est divisible par 4.

Exemples : 6287 n'est pas divisible par 4 (car 87 n'est pas divisible par 4), 4120 l'est (car 20 = 5 x 4, donc divisible par 4).

Application :

Nous proposons de vérifier si un entier n donné est divisible par 4, en utilisant la règle de divisibilité citée ci-dessus.

1- Proposez une analyse modulaire au problème,
2- Déduisez les algorithmes dégagés,
3-Traduisez et testez la solution obtenue. Enregistrez votre programme sous le nom **Divi_4.**



a) Analyse du problème

Résultat : Ecrire (n, **FN** Div_4 (n))
Données : un entier n, entré grâce à la procédure Saisie.

b) Algorithme du programme principal et codification des objets

0) Début Divisible_4
1) **Proc** Saisie (n)
2) Ecrire (n, **FN** Div_4 (n))
3) Fin Divisible_4

Tableau de codification des objets globaux

Objets	Type / Nature	Rôle
n	Entier Long	Nombre à tester
Saisie	Procédure	Permet de saisir n
Div_4	Fonction	Permet de retourner un message indiquant la divisibilité ou non de n par 4.

c) Analyse de la fonction Div_4

Résultat : Div_4

Traitement :

➢ Si le nombre composé des deux derniers chiffres du nombre à tester est divisible par 4 alors le nombre initial l'est aussi sinon ce dernier n'est pas divisible par 4.

Si (m Mod 100) Mod 4 = 0 **Alors**
Div_4 ← " est divisible par 4."
Sinon
Div_4 ← " n'est pas divisible par 4."

d) Algorithme de la fonction Div_4

0) Fonction Div_4 (m : Entier Long) : Chaîne
1) **Si** (m Mod 100) Mod 4 = 0 **Alors** Div_4 ← " est divisible par 4."
Sinon Div_4 ← " n'est pas divisible par 4."
Finsi
2) Fin Div_4

e) Traduction en Pascal de la fonction Div_4

```
Function Div_4(m : Longint) : String;
Begin
  If ((m Mod 100) Mod 4) = 0 Then div_4 := ' est divisible par 4.'
  Else div_4 := ' n''est pas divisible par 4.';
End;
```

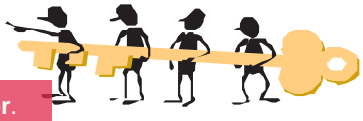
IV.4 Divisibilité par 5

Un entier est divisible par 5, si son chiffre des unités est égal à 0 ou à 5.

Exemples : 6287 n'est pas divisible par 5, car $7 \notin \{0, 5\}$
4120 est divisible par 5, car $0 \in \{0, 5\}$
3635 est divisible par 5, car $5 \in \{0, 5\}$

Application :
Nous voulons vérifier si un entier n donné est divisible par 5, en utilisant la règle de divisibilité citée ci-dessus.

1 - Proposez une analyse modulaire au problème,
2 - Déduisez les algorithmes dégagés,
3 - Traduisez et testez la solution obtenue. Enregistrez votre programme sous le nom **Divi_5**.



a) Analyse du problème

Résultat : Ecrire (n, **FN** Div_5 (n))
Données : un entier n, entré grâce à la procédure Saisie.

b) Algorithme du programme principal et codification des objets

0) Début Divisible_5
1) **Proc** Saisie (n)
2) Ecrire (n, **FN** Div_5 (n))
3) Fin Divisible_5

Tableau de codification des objets globaux

Objets	Type / Nature	Rôle
n	Entier Long	Nombre à tester
Saisie	Procédure	Permet de saisir n
Div_5	Fonction	Permet de retourner un message indiquant la divisibilité ou non de n par 5.

c) Analyse de la fonction Div_5

Résultat : Div_5
Traitement :
Si (2*m) Mod 10 = 0 **Alors**
Div_5 ← " est divisible par 5."
Sinon
Div_5 ← " n'est pas divisible par 5."

d) Algorithme de la fonction Div_5

0) Fonction Div_5 (m : Entier Long) : Chaîne
1) **Si** (2*m) Mod 10 = 0 **Alors** Div_5 ← " est divisible par 5."
Sinon Div_5 ← " n'est pas divisible par 5."
Finsi
2) Fin Div_5

e) Traduction en Pascal de la fonction Div_5

```
Function Div_5(m : Longint) : String;
Begin
  If (2*m) Mod 10 = 0
  Then div_5 := ' est divisible par 5.'
  Else div_5 := ' n''est pas divisible par 5.';
End;
```

IV.5 Autres règles de divisibilité

◈ Un entier est divisible par 2, si son chiffre des unités est divisible par 2.

Exemples : 5287 n'est pas divisible par 2
136 est divisible par 2
94618 est divisible par 2.

- Un entier est divisible par 9, si la somme de ses chiffres est divisible par 9.
- Un entier est divisible par 10, si son chiffre des unités est égal à 0.
- Un entier est divisible par 25, si le nombre composé des deux derniers chiffres est divisible par 25…

Questions :

1) Cherchez d'autres règles de divisibilité.
2) Soit n = 571u où u désigne le chiffre des unités dans l'écriture décimale de n. Ecrivez un programme Pascal permettant d'afficher les valeurs respectives de u et de n pour lesquelles l'entier n sera divisible par 4.
3) Soit n = 10c8u où c et u désignent respectivement le chiffre des centaines et le chiffre des unités dans l'écriture décimale de n. Ecrivez un programme Pascal permettant de déterminer puis d'afficher les couples (c, u) et l'entier n pour lesquels ce dernier sera divisible par 3.
4) Ecrivez un programme Pascal permettant de déterminer puis d'afficher tous les diviseurs d'un entier naturel non nul n donné.

V. Conversions entre bases de numération

V.1 Définition

Un **système de numération** est une méthode de comptage fondée sur une base de numération qui est un entier supérieur ou égal à 2. Soit N une base de numération, le système sera doté de N chiffres allant de 0 à N-1.

V.2 Exemples de bases de numération

- **Base 2** :

Le système de numération à base 2 (binaire) est un moyen de représenter les nombres avec 2 chiffres. Les chiffres utilisés sont 0 et 1.

- **Base 8** :

Dans le système octal, seuls les chiffres 0, 1, 2, 3, 4, 5, 6 et 7 sont utilisés pour coder un triplet de bits.

- **Base 10** :

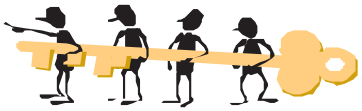
Le système de numération à base 10 est un moyen de représenter les nombres avec 10 symboles : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9. C'est le système couramment utilisé.

- **Base 16** :

Les nombres binaires étant de plus en plus longs, il a fallu introduire une nouvelle base : la base hexadécimale. Elle consiste à compter sur la base 16. Les chiffres utilisés sont 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ainsi que les lettres A, B, C, D, E et F pour représenter respectivement les valeurs de 10, 11, 12, 13, 14 et 15.

Activité :

Nous avons vu en 3ème année comment convertir un nombre décimal en binaire. Proposez une analyse modulaire et déduisez les algorithmes permettant de réaliser ce traitement.



a) Analyse du problème

Résultat : La procédure Afficher (T, c) permet d'afficher le tableau contenant les restes des divisions successives par 2,

Traitement : Le remplissage du tableau des restes est effectué par la procédure Chercher (N, T, c)

Données : N est l'entier à convertir, il est entré grâce à la procédure Saisir (N)

b) Algorithme du programme principal

0) Début Conversion_10_2
1) **Proc** Saisir (N)
2) **Proc** Chercher (N, T, C)
3) **Proc** Afficher (T, C)
4) **Fin** Conversion_10_2

Tableau de codification des nouveaux types

Types
Binaire = 0,1 Tab = Tableau de 100 binaire

Tableau de codification des objets globaux

Objets	Types / Nature	Rôle
N	Entier	Nombre à convertir en binaire
C	Entier	Nombre de divisions par 2
T	Tab	Tableau contenant les restes des divisions successives par 2
Saisir	Procédure	Permet de saisir N
Chercher	Procédure	Permet de remplir T par les restes des divisions successives par 2
Afficher	Procédure	Permet d'afficher les éléments de T

c) Analyse de la procédure Chercher

Résultat : Pour remplir le tableau Reste, nous devons :

- [c ← 0]
- Procéder à des divisions successives par 2 qui s'arrêtent lorsque N = 0. Pour chaque division nous devons : Incrémenter le nombre de divisions c de 1, chercher le reste

de la division de N par 2 et le ranger dans un tableau puis changer la valeur de N par N Div 2.

```
Répéter
   C ← C + 1
   Reste [C] ← n mod 2
   n ← n div 2
Jusqu'à (n = 0)
```

d) Algorithme de la procédure Chercher

```
0) Procédure Chercher (n : entier ; Var Reste : Tab ; Var C : entier)
1) C ← 0
      Répéter
         C ← C + 1
         Reste [C] ← n mod 2
         n ← n div 2
      Jusqu'à (n = 0)
2) Fin Chercher
```

Nous allons voir dans ce qui va suivre d'autres algorithmes de conversion entre bases de numération.

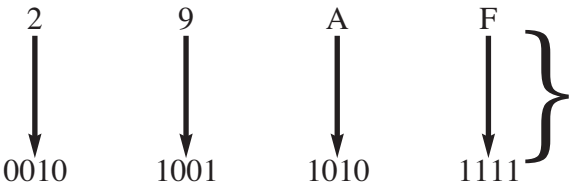
V.3 Conversion d'un nombre hexadécimal en binaire

Nous proposons de convertir un nombre hexadécimal en base 2.

Exemple :

Soit à convertir en binaire le nombre hexadécimal n = 29AF

Nous allons procéder de la manière suivante :



Chaque chiffre de n sera représenté sur 4 bits.

Le nombre binaire obtenu est : **10100110101111**

⇒ En effet, un chiffre hexadécimal correspond à 4 bits. Il suffit donc, de convertir un à un chaque chiffre hexadécimal en binaire et de les mettre les uns à la suite des autres.

Questions :

1) Proposez une analyse modulaire au problème,
2) Déduisez les algorithmes des modules dégagés,
3) Traduisez et testez la solution obtenue. Enregistrez votre programme sous le nom **Hex_Bin**.

a) Analyse du problème

Résultat : Ecrire ("(", Hex, ")16 = (", **FN** Conv_Bin(Hex), ")2")

Données : une chaîne de caractères Hex, entrée grâce à la procédure Saisie. Hex ne peut contenir que des chiffres allant de 0 à 9 et de lettres allant de A à F.

b) Algorithme du programme principal

0) Début Hex_Bin
1) Proc Saisie (Hex)
2) Ecrire ("(", Hex, ")16 = (", **FN** Conv_Bin(Hex), ")2")
3) Fin Hex_Bin

Tableau de codification des objets globaux

Objets	Type / Nature	Rôle
Hex	Chaîne de caractères	Nombre hexadécimal à convertir en binaire
Saisie	Procédure	Permet de saisir Hex
Conv_Bin	Fonction	Permet de retourner une chaîne représentant la conversion en binaire du nombre hexadécimal Hex.

c) Analyse de la fonction Conv_Bin

Résultat : Conv_Bin

Traitement :

- Conv_Bin ← CH_Bin
- Pour supprimer les zéros superflus du début de CH_Bin :

  Tant que CH_Bin[1] = "0" **Faire**
  Efface (CH_Bin,1,1)

- [CH_Bin ← ""]

Ensuite, un parcours total sera effectué sur la chaîne CH_Hex, pour convertir chaque caractère représentant un chiffre hexadécimal en son correspondant binaire :

Pour i **de** 1 à Long (CH_Hex) **Faire**
CH_Bin ← CH_Bin + Bin_Car (CH_Hex[i])

d) Algorithme de la fonction Conv_Bin

0) Fonction Conv_Bin (CH_Hex : Chaîne) : Chaîne
1) CH_Bin ← ""
Pour i de 1 à Long (CH_Hex) **Faire**
CH_Bin ← CH_Bin + FN Bin_Car (CH_Hex[i])
FinPour

2) **Tant que** CH_Bin[1] = "0" **Faire**
Efface (CH_Bin,1,1)
Fin Tant Que
3) Conv_Bin ← CH_Bin
4) Fin Conv_Bin

Tableau de codification des objets locaux

Objets	Type / Nature	Rôle
CH_Bin	Chaîne	Conversion de CH_Hex en binaire
i	Entier	Compteur
Bin_car	Fonction	Déterminer une chaîne de quatre caractères, représentant la conversion en binaire d'un chiffre hexadécimal

e) Analyse de la fonction Bin_Car

Résultat : Bin_Car
Traitement :

- Bin_Car ← CH
- [CH ← "0000"]

Ensuite, des divisions successives par 2 du chiffre hexadécimal seront réalisées, les restes seront affectés directement dans leurs positions dans CH. Cette action est répétée jusqu'à ce que N = 0 :

i ← 4
Répéter
R ← N Mod 2
Convch (R, Ch_R)
Ch[i] ← Ch_R[1]
N ← N Div 2
i ← i-1
Jusqu'à N = 0

- Le correspondant numérique du chiffre hexadécimal est obtenu de deux manières, selon les cas suivants :
  Si le chiffre est numérique, il suffit d'utiliser la procédure prédéfinie Valeur, **sinon** nous utiliserons la formule suivante : Code Ascii de la lettre - 55

Exemple : Le correspondant numérique du chiffre hexadécimal
A = Ord('A') - 55 = 65 - 55 = 10

f) Algorithme de la fonction Bin_Car

0) Fonction Bin_Car (Symb : Caractère) : Chaîne
1) **Si Non** (Symb **Dans** ["0".."9"])
Alors N ← Ord (Symb) - 55
Sinon Valeur (Symb, N, E)
FinSi

```
2) CH ← "0000"
   i ← 4
   Répéter
         R ← N Mod 2
         Convch (R, Ch_R)
         Ch[i] ← Ch_R[1]
         N ← N Div 2
         i ← i-1
   Jusqu'à N = 0
3) Bin_Car ← CH
4) Fin Bin_Car
```

Tableau de codification des objets locaux

Objets	Types / Nature	Rôle
CH	Chaîne[4]	Représente la conversion en binaire d'un chiffre hexadécimal
i	Entier	Compteur
N	Entier	Correspondant numérique du chiffre hexadécimal
E	Entier	Indicateur d'erreurs de la conversion numérique
R	Entier	Reste de la division euclidienne par 2
Ch_R	Chaîne[1]	Représente le reste sous forme d'une chaîne

g) Traduction en Pascal de la fonction Bin_Car

```
Function Bin_Car (Symb : Char) : String ;
Var
  Ch : String[4];
  Ch_R : String[1];
  N, E, R, i : Integer;
Begin
   If Not (Symb In ['0'..'9'])
   Then N := Ord (Upcase(Symb)) - 55
   Else Val(Symb, N, E);
   Ch := '0000';
   i := 4;
   Repeat
      R := N Mod 2;
      Str (R, Ch_R);
      Ch[i] := Ch_R[1];
      N := N Div 2;
      i := i-1;
   Until N = 0;
   Bin_Car := Ch;
End;
```

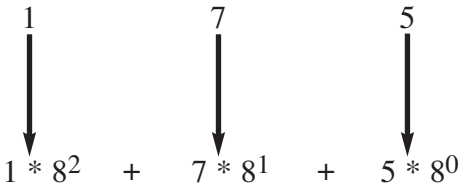
V.4 Conversion d'un nombre octal en décimal

Nous voulons convertir un nombre octal en base 10.

Exemple :

Soit à convertir en décimal le nombre octal n = 175

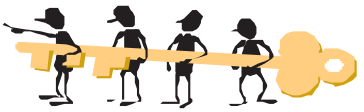
Nous allons procéder de la manière suivante :



Le nombre décimal obtenu est : 125

Questions :

a) Proposez une analyse modulaire au problème,
b) Déduisez les algorithmes des modules dégagés,
c) Traduisez et testez la solution obtenue. Enregistrez votre programme sous le nom **Oct_Dec**.



a) Analyse du problème

Résultat : Ecrire ("(", Oct, ")8 = (", **FN** Conv_Bin(Oct), ")10")

Données : un entier Oct, entré par le biais de la procédure Saisie. Oct ne peut contenir que des chiffres allant de 0 à 7.

b) Algorithme du programme principal

0) Début Oct_Dec
1) **Proc** Saisie (Oct)
2) Ecrire ("(", Oct, ")8 = (", **FN** Conv_Bin(Oct), ")10")
3) Fin Oct_Dec

Tableau de codification des objets locaux

Objets	Type / Nature	Rôle
Oct	Entier Long	Nombre octal à convertir en décimal
Saisie	Procédure	Permet de saisir Oct
Conv_Dec	Fonction	Permet de retourner un entier représentant la conversion en décimal du nombre octal Oct.

c) Analyse de la fonction Conv_Dec

Résultat : Conv_Dec

Traitement :

- Conv_Dec ← Dec
- [Dec ← 0]

Chaque chiffre du nombre octal sera multiplié par 8 élevé à une puissance relative à son rang, en appelant la fonction Puiss_8, puis, ajouté à Dec. Nous allons donc, utiliser la structure itérative complète :

Pour i de Long (CH_Oct) à 1 (pas = -1) **Faire**
Valeur (Ch_Oct[i], N, e)
Dec ← Dec + N * FN Puiss_8 (Long (Ch_Oct) - i)

- Pour pouvoir manipuler les chiffres du nombre octal, nous pouvons le convertir en une chaîne de caractères : ConvCh (N_Oct, CH_Oct)

d) Algorithme de la fonction Conv_Dec

0) Fonction Conv_Dec (N_Oct : Entier Long) : Entier Long
1) ConvCh (N_Oct, CH_Oct)
2) Dec ← 0
Pour i de Long (CH_Oct) à 1 (pas = -1) **Faire**
Valeur (Ch_Oct[i], N, e)
Dec ← Dec + N * FN Puiss_8 (Long (Ch_Oct) - i)
FinPour
3) Conv_Dec ← Dec
4) Fin Conv_Dec

Tableau de codification des objets locaux

Objets	Types / Nature	Rôle
Ch_Oct	Chaîne	Conversion du nombre octal en chaîne
Dec	Entier Long	Conversion du nombre octal en décimal
i	Entier non signé	Compteur
N	Entier	Conversion numérique d'un chiffre de Ch_Oct
e	Entier	Indicateur d'erreurs de la conversion numérique
Puiss_8	Fonction	Permet d'élever 8 à une puissance

e) Traduction en Pascal du problème Oct_Dec

```
Program Oct_Dec;
Uses Crt;
Var Oct : Longint;

Procedure Saisie(Var N_Oct : Longint);
Var
  trouve : Boolean;
  Ch_Oct : String;
  i : Word;
```

```
Begin
   trouve := True;
   While trouve Do
   Begin
      Write('Donner un nombre octal : ');
      Readln(N_Oct);
      Str(N_Oct, Ch_Oct);
      i := 1;
      While (Ch_Oct[i] In ['0'..'7']) And (i <=Length(Ch_Oct)) Do
        i := i + 1;
      If i > Length(Ch_Oct) Then Trouve := false ;
   End;
End;

Function Puiss_8 (M : Integer) : Longint;
Var
   P : Longint;
   j : Integer;
Begin
   P := 1;
   For j := 1 To M Do P := P * 8;
     Puiss_8 := P;
End;

Function Conv_Dec (N_Oct : Longint) : Longint;
Var
  i, e, N : Integer;
  Dec : Longint;
  Ch_Oct : String;
Begin
   Str(N_Oct, Ch_Oct);
   Dec := 0;
   For i := Length(CH_Oct) DownTo 1 Do
   Begin
      Val(Ch_Oct[i],N,e);
      Dec := Dec + N * Puiss_8(Length(Ch_Oct) - i);
   End;
   Conv_Dec := Dec;
End;

{Programme Principal}
Begin
   Saisie(Oct);
   Write('(', Oct, ')8 = (', Conv_Dec(Oct), ')10');
End.
```

V.5 Conversion d'un nombre binaire en octal

Nous proposons de convertir un nombre binaire en base 8.

Exemple :

Soit à convertir en octal le nombre binaire x = 1110101

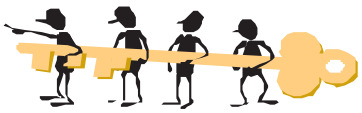
Nous allons procéder de la manière suivante :

3ème partie contenant les bits restants	2ème partie constituée de 3 bits	1ère partie constituée de 3 bits
1	1 1 0	1 0 1
$1*2^0 = 1$	$1*2^2 + 1*2^1 + 0*2^0 = 6$	$1*2^2 + 0*2^1 + 1*2^0 = 5$

Le nombre octal obtenu est : 165

Questions :

a) Proposez une analyse modulaire au problème,
b) Déduisez les algorithmes des modules dégagés,
c) Traduisez et testez la solution obtenue. Enregistrez votre programme sous le nom **Bin_Oct.**



a) Analyse du problème

Résultat : Ecrire ("(", Bin, ")2 = (", **FN** Conv_Oct(Bin), ")8")

Données : une chaîne de caractères représentant le nombre binaire Bin , entrée par le biais de la procédure Saisie. Bin ne peut contenir que des "0" ou "1".

b) Algorithme du programme principal

0) Début Bin_Oct
1) **Proc** Saisie (Bin)
2) Ecrire ("(", Bin, ")2 = (", **FN** Conv_Oct (Bin), ")8")
3) Fin Bin_Oct

Tableau de codification des objets locaux

Objets	Type / Nature	Rôle
Bin	Chaîne	Nombre binaire à convertir en octal
Saisie	Procédure	Permet de saisir Bin
Conv_Oct	Fonction	Permet de retourner un entier représentant la conversion en octal du nombre binaire Bin.

c) Analyse de la fonction Conv_Oct

Résultat : Conv_Oct

Traitement :

- Conv_Oct ← Oct
- [N ← Long (Ch_Bin) Div 3, Oct ← "", L ← Long (Ch_Bin)]
- Extraire une partie de trois bits à partir du dernier bit, en commençant de la droite de CH_Bin; convertir chaque partie en octal en utilisant une fonction Puiss_2; convertir ce résultat en chaîne pour pouvoir le concaténer à l'objet destination de la conversion en octal Oct.

Nous allons utiliser une structure itérative complète dont le nombre d'itérations est égal à N :

```
Pour i de 1 à N Faire
   Ch1 ← Sous_chaîne (Ch_Bin, L - 2, 3)
   A_oct ← 0
   Pour j de 1 à 3 Faire
     Valeur(ch1[j], a, e)
     A_oct ← A_oct + a * FN Puiss_2 (3 - j)

   Convch (A_oct, ch2)
   Oct ← ch2 + Oct
    L ← L - 3
```

- Ajouter, si nécessaire, des "0" à gauche de CH_Bin, de façon à obtenir une longueur multiple de 3 :

```
Tant Que Long (Ch_Bin) Mod 3 ≠ 0 Faire
        Ch_Bin ← "0" + Ch_Bin
```

d) Algorithme de la fonction Conv_Oct

```
0) Fonction Conv_Oct (Ch_Bin : Chaîne) : chaîne
1) Tant Que Long (Ch_Bin) Mod 3 ≠ 0 Faire
        Ch_Bin ← "0" + Ch_Bin
   Fin Tant Que
2) N ← Long (Ch_Bin) Div 3
   Oct ← "", L ← Long (Ch_Bin)
   Pour i de 1 à N Faire
     Ch1 ← Sous_chaîne (Ch_Bin, L-2, 3)
     A_oct ← 0
     Pour j de 1 à 3 Faire
       Valeur (ch1[j], a, e)
       A_oct ← A_oct + a * FN Puiss_2 (3-j)
     FinPour
     Convch (A_oct, ch2)
     Oct ← ch2 + Oct
     L ← L - 3
    FinPour
3) Conv_Oct ← Oct
4) Fin Conv_Oct
```

Tableau de codification des objets locaux

Objets	Types / Nature	Rôle
Ch1, Ch2, oct	Chaîne	Chaînes intermédiaires
i, j, L	Entier	Compteurs
e	Entier	Indicateur d'erreurs de la conversion numérique
a, A_oct	Entier	Variables intermédiaires
Puiss_2	Fonction	Permet d'élever 2 à une puissance

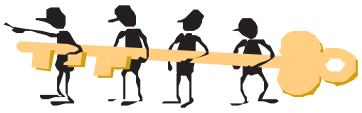
e) Traduction en Pascal de la fonction Conv_Oct

```
Function Conv_Oct(Ch_Bin : String) : String;
Var
  L, j, i, e, a, a_oct : Integer;
  ch1, ch2, oct : String;
Begin
   While Length(Ch_Bin) Mod 3 <> 0 do
      Ch_Bin := '0' + Ch_Bin;
   Oct := '';
   L := Length(Ch_Bin);
   For i := 1 To (length(Ch_Bin) Div 3) Do
   Begin
      ch1 := Copy (Ch_Bin, L - 2,3);
      A_oct := 0;
      For j := 1 To 3 Do
      Begin
        Val(ch1[j], a, e);
        A_oct := A_oct + a * Puiss_2 (3 - j);
      End;
      Str(A_oct, ch2);
      oct := ch2 + oct;
      L := L - 3;
   End;
   conv_oct := oct;
End;
```

V.6 Conversion d'un nombre d'une base B1 en une base B2

Nous voulons convertir un nombre d'une base B1 en base B2, avec $2 \leq b1 \leq 16$ et $2 \leq b2 \leq 16$.

a) Proposez une analyse modulaire au problème,
b) Déduisez les algorithmes des modules dégagés,
c) Traduisez et testez la solution obtenue. Enregistrez votre programme sous le nom **B1_B2**.



a) Analyse du problème

Résultat : Ecrire ("(", N_B1, ")",B1," = (", **FN** Conversion_B2 (N_B1, B1, B2), ")", B2)

Données :

- Une chaîne de caractères N_B1 représentant le nombre de la base B1, entrée par le biais de la procédure Saisie_Nombre. N_B1 ne peut contenir que des chiffres de la base B1 (commençant obligatoirement par le chiffre 0).
- Les bases B1 et B2 sont données par la procédure Saisie_Base. B1 et B2 sont comprises entre 2 et 16.

b) Algorithme du programme principal

0) Début B1_B2
1) **Proc** Saisie_Base (B1)
2) **Proc** Saisie_Base (B2)
3) **Proc** Saisie_Nombre (N_B1, B1)
4) Ecrire ("(", N_B1,")",B1," = (", **FN** Conversion_B2 (N_B1, B1, B2), ")", B2)
3) Fin B1_B2

Tableau de codification des objets globaux

Objets	Types / Nature	Rôle
N_B1	Chaîne	Nombre à convertir
B1, B2	Entier	Les bases de conversion
Saisie_Base	Procédure	Permet de saisir une base de numération
Saisie_Nombre	Procédure	Permet de saisir le nombre à convertir
Conversion_B2	Fonction	Permet de retourner la conversion de N_B1 en base B2

c) Analyse de la fonction Conversion_B2

Résultat : Conversion_B2

Traitement : Ce traitement sera décomposé en deux modules : un passage d'une base B1 en base 10, puis une conversion en base B2 du nombre décimal obtenu.

- CH_B2 est le résultat de la conversion d'un nombre décimal N_10 en un nombre en base B2. Une fonction Conv_B2 traduira ce traitement,
- N_10 est le résultat de la conversion du nombre N_B1 en base 10. Cette tâche est effectuée par la fonction Conv_10.

d) Algorithme de la fonction Conversion_B2

0) Fonction Conversion_B2 (Ch_B1 : Chaîne; B1, B2 : Entier) : Chaîne
1) N_10 ← FN Conv_10 (CH_B1, B1)
2) CH_B2 ← FN Conv_B2 (N_10, B2)
3) Conversion_B2 ← CH_B2
4) Fin Conversion_B2

Tableau de codification des objets globaux

Objets	Type / Nature	Rôle
N_10	Entier Long	Conversion de N_B1 en base 10
Ch_B2	Chaîne	Conversion de N_10 en base B2
Conv_10	Fonction	Permet la conversion d'un nombre de base quelconque en base 10
Conv_B2	Fonction	Permet de convertir un nombre décimal en base quelconque

e) Analyse de la fonction Conv_10

Résultat : Conv_10

Traitement :

- Conv_10 ← Dec
- [Dec ← 0]

Chaque chiffre de N_B1 sera converti en valeur numérique, multiplié par la base élevée à une puissance relative à son rang (en utilisant une fonction Puiss_B1), puis, ajouté à Dec. Nous allons donc, utiliser une structure itérative complète :

Pour i de Long (Ch_B1) à 1 (pas = -1) **Faire**
Si Majuscule (Ch_B1[i]) Dans ["A".."F"]
Alors N ← Ord (Majuscule (Ch_B1[i])) - 55
Sinon Val (Ch_B1[i],N,e)

Dec ← Dec + N * FN Puiss_B1 (Long (Ch_B1) - i, B1)

f) Algorithme de la fonction Conv_10

```
0) Fonction Conv_10 (Ch_B1 : Chaîne; B1 : Entier) : Entier Long
1) Dec ← 0
   Pour i de Long (Ch_B1) à 1 (pas = -1) Faire
      Si Majuscule (Ch_B1[i]) Dans ["A".."F"]
      Alors N ← Ord (Majuscule (Ch_B1[i])) - 55
      Sinon Val (Ch_B1[i],N,e)
      FinSi
      Dec ← Dec + N * FN Puiss_B1 (Long (Ch_B1) - i, B1)
   FinPour
2) Conv_10 ← Dec
3) Fin Conv_10
```

Tableau de codification des objets locaux

Objets	Type / Nature	Rôle
i	Entier	Compteur
e	Entier	Indicateur d'erreurs de la conversion numérique
N	Entier	Conversion numérique de chaque chiffre de Ch_B1
Dec	Entier Long	Conversion de N_B1 en base 10
Puiss_B1	Fonction	Permet d'élever B1 à une puissance

g) Traduction en Pascal de la fonction Conv_10

```
Function Conv_10 (CH_B1 : String; B1 : Integer) : Longint;
Var
  i, e, N : Integer;
  Dec : Longint;
Begin
   Dec := 0;
   For i := Length(CH_B1) DownTo 1 Do
   Begin
      If Upcase(Ch_B1[i]) In ['A'..'F']
      Then N := Ord(Upcase(Ch_B1[i])) - 55
      Else Val(Ch_B1[i],N,e);
      Dec := Dec + N * Puiss_B1(Length(Ch_B1) - i,B1);
   End;
   Conv_10 := Dec;
End;
```

h) Analyse de la fonction Conv_B2

Résultat : Conv_B2

Traitement :

- Conv_B2 ← Ch
- [Ch ← " "]

Diviser successivement N_10 par B2 jusqu'à obtenir un quotient nul. La concaténation des restes forme le nombre converti. Nous utiliserons donc, une structure itérative à condition d'arrêt :

Répéter
R ← N_10 **Mod** B2
Si R ≥ 10
Alors Ch_R ← **Majuscule** (Chr (55 + R))
Sinon Convch (R, Ch_R)

Ch ← Ch_R + Ch
N_10 ← N_10 Div B2
Jusqu'à N_10 = 0

i) Algorithme de la fonction Conv_B2

0) Fonction Conv_B2 (N_10 : Entier Long; B2 : Entier) : Chaîne
1) Ch ← " "
Répéter
R ← N_10 Mod B2
Si R ≥ 10
Alors Ch_R ← Majuscule (Chr (55 + R))
Sinon Convch (R, Ch_R)
FinSi
Ch ← Ch_R + Ch
N_10 ← N_10 Div B2
Jusqu'à N_10 = 0
2) Conv_B2 ← Ch
3) Fin Conv_B2

Tableau de codification des objets locaux

Objets	Type / Nature	Rôle
Ch	Chaîne	Résultat de la conversion en base B2
R	Entier	Reste de la division euclidienne
Ch_R	Chaîne	Conversion en chaîne du reste R

Retenons

- Le PGCD de deux entiers positifs non nuls a et b est le Plus Grand Commun Diviseur. Parmi les méthodes utilisées pour rechercher le PGCD, nous citons la méthode de la différence.
- Le nombre **d'arrangements** dans un ensemble de n éléments noté $\mathbf{A_n^p}$ est égal à $\frac{n!}{(n-p)!}$
- Le nombre de **combinaisons** de p éléments parmi n noté $\mathbf{C_n^p}$ est égal à $\frac{n!}{p!\,(n-p)!}$

- Un entier est **divisible** par un autre, si le reste de sa division entière par ce dernier est nul.
- Une **règle de divisibilité** est une séquence d'opérations simples qui permet de reconnaître rapidement si un entier est divisible par un autre, sans qu'il soit nécessaire d'effectuer la division directe.
- La numération désigne les techniques de représentation des nombres. Les représentations écrites au moyen de signes constituent des systèmes de numération.
- De nombreuses **bases de numération** ont été employées tels que le système binaire (base 2) utilisé par les ordinateurs, le système octal (base 8), le système décimal (base 10), il est de loin le plus répandu de nos jours, le système duodécimal (base 12), le système hexadécimal (base 16) utilisé en électronique et en informatique car la conversion binaire-hexadécimale est très simple, etc.
- La représentation d'un nombre dans une base plus grande devient plus compacte.
- Il est possible de convertir un nombre appartenant à une base B1 donnée, dans une autre base B2.

Citation du Chapitre

« Compter en octal,
c'est comme compter en décimal,
si on n'utilise pas ses pouces »

Tom Lehrer

Exercices

Exercice 1

Mettez la lettre V (Vrai) dans la case qui correspond à chaque proposition si vous jugez qu'elle est vraie sinon mettez la lettre F (Faux).

a. 245025 + 9171 est divisible par

2 ☐ 3 ☐ 4 ☐

b. B43 peut être un nombre

Décimal ☐ Duodécimal (Base 12) ☐ Hexadécimal ☐

c. L'entier a+1 est représenté dans la base a (a >= 2) par le nombre

10 ☐ 11 ☐ Cela dépend de a ☐

d. La conversion du nombre décimal 3243 en hexadécimal est

ABC ☐ BAC ☐ CAB ☐

Exercice 2

Dans un contexte informatique, définir les expressions suivantes :

a) Base de numération
b) Diviseur d'un entier
c) Multiple d'un entier

Exercice 3

Une anagramme d'un mot donné est un mot de même longueur, ayant un sens ou non et formé par les mêmes lettres :

1. Ecrivez un programme Pascal permettant d'afficher le nombre d'anagrammes d'un mot donné composé uniquement de lettres,

2. Ecrivez un programme Pascal permettant d'afficher le nombre d'anagrammes commençant par une voyelle, d'un mot donné composé uniquement de lettres.

Exercice 4

Soient a et b deux réels et n un entier naturel non nul. Ecrivez un programme Pascal permettant de vérifier la formule du binôme exprimée par l'égalité suivante :

$$(a+b)^n = C_n^0\, a^n + C_n^1\, a^{n-1}\, b^1 + C_n^2\, a^{n-2}\, b^2 + \ldots + C_n^{n-1} a^1\, b^{n-1} + C_n^n\, b^n$$

Exercice 5

Soit n un entier naturel non nul. Ecrivez un programme Pascal permettant de calculer puis d'afficher la somme suivante :

$$C_n^0 - C_n^1 + C_n^2 - C_n^3 + \ldots\ldots + (-1)^n\, C_n^n$$

Exercice 6

Soit n = 61d1 où d désigne le chiffre des dizaines dans l'écriture décimale de n. Ecrivez un programme Pascal permettant d'afficher les valeurs respectives de d et de n pour lesquelles l'entier n sera divisible par 9.

Exercice 7

Soit n = m97u où m et u désignent respectivement le chiffre des milliers et le chiffre des unités dans l'écriture décimale de n. Ecrivez un programme Pascal permettant de déterminer puis d'afficher les couples (m, u) et l'entier n pour lesquels ce dernier sera divisible par 5 et par 9.

Exercice 8

Ecrivez un programme Pascal permettant de déterminer puis d'afficher l'ensemble des entiers naturels n compris entre 1 et 100 tels que n divise n + 8.

Exercice 9 (Bac 1995)

"Conversion entre bases "

On se propose de réaliser la conversion d'un nombre décimal N donné (N est un entier naturel strictement positif) en son équivalent dans une base B donnée.

Pour cela on effectue des divisions euclidiennes par B, les restes successifs seront rangés dans un vecteur (tableau) appelé RESTES à RMAX éléments (RMAX = 15).

1) Ecrire un algorithme de la procédure **SAISIE-DEC** qui saisit un entier naturel **N** strictement positif.

2) Ecrire un algorithme de la procédure **SAISIE-BASE** qui saisit un entier naturel **B** tel que $2 \leq B \leq 16$.

3) **a** - Ecrire un algorithme de la procédure **RANGER (N, B, RESTES, R)** , qui permet de :
- ranger dans le vecteur RESTES les restes successifs de la suite des divisions euclidiennes par B .
- calculer le nombre des restes R.

b - Traduire en langage Pascal l'algorithme de cette procédure.

4) Ecrire un algorithme de la procédure **RENVERSER (RESTES, R)** qui renverse les éléments rangés dans le vecteur RESTES.
(permuter **RESTES [1]** avec **RESTES [R],**
RESTES [2] avec **RESTES [R-1]**, etc.).

5) a - Ecrire un algorithme de la fonction **CONVERT (C)** qui permet de retourner le caractère qui correspond à l'entier C (C compris entre 0 et 15).

Exemples : **CONVERT** (0) = "0", **CONVERT** (9) = "9"
CONVERT (10) = "A" **CONVERT** (15) = "F"

b - Traduire en langage Pascal l'algorithme de cette fonction.

6) Ecrire un algorithme de la procédure **CONCATENATION (résultat, RESTES)** qui, en utilisant la fonction CONVERT permet de rassembler les restes en une chaîne de caractères intitulée résultat, représentant le nombre N dans la base B.

7) En utilisant les procédures et la fonction déjà définies, écrire un algorithme du programme principal intitulé **CONVERSION** permettant d'afficher le résultat de la conversion d'un décimal N dans une base **B.**

Pour chacun des exercices suivants,

a) Proposez une analyse modulaire au problème,

b) Déduisez les algorithmes des modules dégagés,

Exercice 10

Nous proposons de vérifier si un entier n donné est divisible par 7, en utilisant la règle de divisibilité suivante : Nous nous appuyons sur le fait que si le nombre mcdu est divisible par 7, alors : mcd-2*u est divisible par 7, et réciproquement.

Prenons un exemple : 7241. Nous conservons tous les chiffres sauf le dernier, et nous lui retranchons deux fois le dernier : 724-2*1=722. Nous procédons de même avec le résultat, soit 722 : 72-2*2=68. Or 68 n'est pas divisible par 7, donc 7241 non plus.

Traduisez et testez la solution obtenue. Enregistrez votre programme sous le nom **Div_7.**

Exercice 11

Nous voulons vérifier si un entier n donné est divisible par 11, en utilisant la règle de divisibilité suivante : Il faut que la différence entre la somme des chiffres de rang pair et la somme des chiffres de rang impair soit divisible par 11.

Exemples : 651 n'est pas divisible par 11 (car (6+1)-5=2, et 2 n'est pas divisible par 11), mais 41679 l'est (car (4+6+9)-(1+7) = 11, et 11 est divisible par 11).

Traduisez et testez la solution obtenue. Enregistrez votre programme sous le nom **Div_11.**

Exercice 12

Nous proposons de vérifier si un entier n donné est divisible par 6, en utilisant la règle de divisibilité suivante : Un nombre est divisible par 6 s'il est divisible à la fois par 2 et par 3.

NB : Ne pas utiliser directement l'opérateur arithmétique Mod.

Traduisez et testez la solution obtenue. Enregistrez votre programme sous le nom **Div_6.**

Exercice 13

Nous voulons convertir un nombre octal en base 2.

Traduisez et testez la solution obtenue. Enregistrez votre programme sous le nom **Oct_Bin.**

Exercice 14

Nous proposons de convertir un nombre binaire en base 16.

Traduisez et testez la solution obtenue. Enregistrez votre programme sous le nom **Bin_Hex.**

Exercice 15

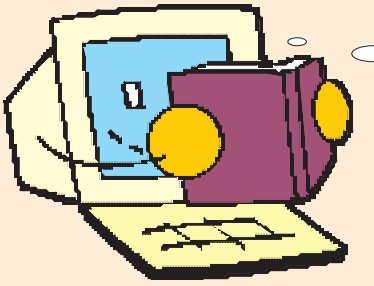
Nous proposons de convertir un nombre décimal en base binaire.

Traduisez et testez la solution obtenue. Enregistrez votre programme sous le nom **DecBin_R.**

Exercice 16

Nous voulons lire deux nombres binaires puis vérifier si l'un est divisible par l'autre.

Traduisez et testez la solution obtenue. Enregistrez votre programme sous le nom **Div_Bin.**



Lecture

Quelques règles de divisibilité

Par 2

Il suffit que le dernier chiffre soit divisible par 2 (c'est-à-dire que ce soit 0, 2, 4, 6, ou 8).
Exemples : 5287 n'est pas divisible par 2, mais 136 et 94618 le sont.

Par 3

Il suffit que la somme des chiffres qui composent le nombre soit divisible par 3.
Exemples : 734 n'est pas divisible par 3 (car 7+3+4=14, 14 n'est pas divisible par 3), mais 81234 l'est (car 8+1+2+3+4=18, 18 est divisible par 3).

Par 4

Il suffit que les deux derniers chiffres soient divisibles par 4.
Exemples : 61934 n'est pas divisible par 4 (34 n'est pas divisible par 4), mais 2028 l'est (28divisible par 4 car 28=4x7).

Par 5

De même que pour 2, il suffit que le dernier chiffre soit divisible par 5 (c'est-à-dire que ce soit 0 ou 5). Exemples : 2497 n'est pas divisible par 5, mais 625 et 4800 le sont.

Par 7

C'est un peu plus compliqué. On s'appuie sur le fait que si le nombre abcd est divisible par 7, alors : abc-2d est divisible par 7, et réciproquement.
Prenons un exemple : 7241. Nous conservons tous les chiffres sauf le dernier, et nous lui retranchons deux fois le dernier : 724-2x1=722. Nous procédons de même avec le résultat, soit 722 : 72-2x2=68. Or 68 n'est pas divisible par 7, donc 7241 non plus.
Autre exemple : 30086. 3008-2x6=2996 ; 299-2x6=287 ; 28-2x7=14 ; 14 étant divisible par 7, 30086 l'est aussi !

Par 9

De même que pour 3, il suffit que la somme des chiffres qui composent le nombre soit divisible par 9.
Exemples : 5019 n'est pas divisible par 9 (5+0+1+9=15, 15 n'est pas divisible par 9), mais 837 l'est (8+3+7=18, 18 est divisible par 9).

Par 11

Il faut que la différence entre la somme des chiffres de rang pair et la somme des chiffres de rang impair soit divisible par 11.
Exemples : 651 n'est pas divisible par 11 (car (6+1)-5=2, et 2 n'est pas divisible par 11), mais 41679 l'est (car (4+6+9)-(1+7)=11, et 11 est divisible par 11).

Par 25

De même que pour 4, il suffit que les deux derniers chiffres soient divisibles par 25.
Exemples : 6287 n'est pas divisible par 25 (87 n'est pas divisible par 25), mais 4150 l'est (50=25x2).