Chapitre 5 : La représentation des données

PLAN

1 – La représentation des caractères
- 1.1 - Le code ASCII
- 1.2 - Le code ANSI
- 1.3 - Le code EBCDIC
- 1.4 - Le code UNICODE
- 1.5 - Notion de contrôle de parité

2 – La représentation des informations multimédia
- 2.1 – La représentation du son
- 2.2 – La représentation des images fixes
- 2.3 – La représentation des images animées
- 2.4 – La compression des données

1 - La représentation des caractères

Les informations que doivent traiter les ordinateurs sont composées de nombres, lettres, chiffres ou symboles particuliers. On doit représenter l'information à traiter, quelle qu'elle soit, de manière à ce qu'elle puisse être utilisable par la machine. Pour cela, on doit coder ces informations afin, qu'assimilables par l'homme elles le deviennent par la machine.

Avec un code théorique à 1 bit, on pourrait représenter 2 états, notés 0 ou 1, ainsi que nous l'avons vu lors de l'étude de la numération binaire, soit 2^1 combinaisons. Il nous faut donc un code à 4 bits pour représenter les 10 chiffres (0 à 9) utilisés dans le système décimal. Si nous voulons représenter, en plus des chiffres, les lettres de l'alphabet, il faut un code capable de représenter les 26 combinaisons, qui correspondent aux lettres, plus les 10 combinaisons, qui correspondent aux chiffres, soit 36 combinaisons différentes, ce qui implique un code composé au minimum de 6 bits. On rencontrera ainsi différents codes permettant la codification d'alphabets plus ou moins importants.

1.1 – Le code ASCII (American Standard Code of Information Interchange)

C'est le système de codage quasi universel. C'est un code à 7 positions, le 8ième bit étant réservé au bit de parité ce qui fait 2^7 = 128 caractères représentables. Ce code comprend :

- les chiffres,
- les majuscules,
- les minuscules,
- quelques symboles usuels en informatique ($,@,*,..),
- des fonctions de commandes (tabulations, Retour chariot, sonnette..),
- des symboles de ponctuations (!,",},...).

Remarques :

- Chiffres < Majuscules <Minuscules
  C'est l'ordre qui sera respecté dans la plupart des utilitaires de tri.
  Dans beaucoup de langages de programmation, on peut écrire : if a<b pour des caractères, c'est cet ordre qui sera utilisé pour évaluer l'expression.

- Certains constructeurs, dont IBM suivis par tous les fabricants de compatibles, ont enrichi cette table ASCII en utilisant le $8^{ième}$ bit, ce qui double le nombre de caractères représentables (256). Les caractères supplémentaires sont essentiellement :
  - les caractères accentués utilisés dans la langue française notamment,
  - un jeu de caractères utilisés dans quelques langues (≈,√,...),
  - quelques symboles mathématiques (∃,∫),
  - caractères semi-graphiques qui permettent de réaliser des petits dessins géométriques (cadres, soulignés, etc..).

- Le code ASCII 8 bits existe en 2 variantes : le jeu de caractères **IBM PC** (c'est le jeu de caractères standard du DOS) et le jeu de caractères **ISO-ANSI** (jeu de caractères international utilisé dans Windows). Ces deux jeux diffèrent notamment au niveau des caractères nationaux accentués et des caractères semi-graphiques.

Comment coder la lettre A ? → à l'aide du tableau
Dans le tableau, on s'aperçoit que la lettre A se trouve à l'intersection de la colonne de valeur hexadécimale 4 et de la ligne de valeur hexadécimale 1. Le code ASCII de la lettre A est donc 41 en hexadécimal (souvent noté 41H). Certains logiciels ou langages peuvent utiliser une codification des caractères ASCII en décimal et non en hexadécimal.

Les caractères ASCII représentés ou non sur le clavier s'obtiennent par ALT + code ASCII.

		000	001	010	011	100	101	110	111
		0	1	2	3	4	5	6	7
0000	0	NUL	DLE	SP	0	@	P	`	p
0001	1	SOH	DC1	|	1	A	Q	a	q
0010	2	STX	DC2	″	2	B	R	b	r
0011	3	ETX	DC3	#	3	C	S	c	s
0100	4	EOT	DC4	$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v
0111	7	BEL	ETB	‘	7	G	W	g	w
1000	8	BS	CAN	(	8	H	X	h	x
1001	9	HT	EM	)	9	I	Y	i	y
1010	10	LF	SUB	*	:	J	Z	j	z
1011	11	VT	ESC	+	;	K	[	k	{
1100	12	FF	FS	,	<	L	\	l	!
1101	13	CR	GS	-	=	M	]	m	}
1110	14	SO	RS	.	>	N	‘	n	~
1111	15	SI	US	/	?	O	-	o	DEL

TABLE DES CODES ASCII

NUL	Absence de caractère,blanc,espace
SOH	Start of Heading : début en-tête
STX	Start of Text
ETX	End of Text
EOT	End of Transmission
ENQ	Enquiry Demande
ACK	Acknowledge, accusé réception
BEL	Bell, sonnette
BS	Backspace marche arrière 1 caractère
HT	Horizontale Tabulation
LF	Line Fed retour à la nvelle ligne
VT	Vertical Tabulation
FF	Form Fed, passage page suivante
CR	Carriage Return, retour chariot
SO	Shift Out caractère suivant non std
SI	Shift In retour au caractères std
DLE	DataLink Escape chgmt de signific.
NAK	Negative Acknoledgment
SYN	Synchronous, caractère de synchro.
ETB	End Of Transmission Block
CAN	Cancel annulation de la donnée précédente
SUB	Substitute remplacement
ESC	Escape caractère de ctrl d'extension
FS	File Separator
GS	Groupe Separator
RS	Record Separator
US	United Separator
SP	Space Espace
DEL	Delete, suppression

DC1 à DC4 : caractères de commandes

1.2 - Code ANSI

Les logiciels sous Windows utilisent la norme ANSI, qui reprend en grande partie le code ASCII, et propose des extensions différentes selon le « code de page » retenu.
Ainsi, le code page 850 est très employé en France, alors que le code page 864 définit un jeu de caractères « arabe ».
L'utilisation du code ANSI se fait de la même manière que pour un code ASCII (intersections colonne-ligne).

Page de codes 850

		0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
		0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
0	-0		►		0	@	P	`	p	Ç	É	á	░	└	ð	Ó	-
1	-1	☺	◄	!	1	A	Q	a	q	ü	æ	í	▒	┴	Đ	β	±
2	-2	☻	↕	"	2	B	R	b	r	é	Æ	ó	▓	┬	Ê	Ô	‗
3	-3	♥	‼	#	3	C	S	c	s	â	ô	ú	│	├	Ë	Ò	¾
4	-4	♦	¶	$	4	D	T	d	t	ä	ö	ñ	┤	─	È	õ	¶
5	-5	♣	§	%	5	E	U	e	u	à	ò	Ñ	Á	┼	ı	Õ	§
6	-6	♠	▬	&	6	F	V	f	v	å	û	ª	Â	ã	Í	µ	÷
7	-7	•	↨	'	7	G	W	g	w	ç	ù	º	À	Ã	Î	þ	¸
8	-8	◘	↑	(	8	H	X	h	x	ê	ÿ	¿	©	╚	Ï	Þ	°
9	-9	○	↓	)	9	I	Y	i	y	ë	Ö	®	╣	╔	┘	Ú	¨
10	-A	◙	→	*	:	J	Z	j	z	è	Ü	¬	║	╩	┌	Û	·
11	-B	♂	←	+	;	K	[	k	{	ï	ø	½	╗	╦	█	Ù	¹
12	-C	♀	∟	,	<	L	\	l	|	î	£	¼	╝	╠	▄	ý	³
13	-D	♪	↔	-	=	M	]	m	}	ì	Ø	¡	¢	═	¦	Ý	²
14	-E	♫	▲	.	>	N	^	n	~	Ä	×	«	¥	╬	Ì	¯	■
15	-F	☼	▼	/	?	O	_	o	⌂	Å	ƒ	»	┐	¤	▀	´	

Page de codes 864

		0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
		0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
0	-0		►		0	@	P	`	p	°	β		٠	¢	ذ	ـ	ﹽ
1	-1	☺	◄	!	1	A	Q	a	q	·	∞	.	١	ء	ر	ﻓ	ّ
2	-2	♪	↕	"	2	B	R	b	r	∙	φ	ﺂ	٢	آ	ز	ﻗ	ن
3	-3	♫	‼	#	3	C	S	c	s	√	±	£	٣	أ	ﺳ	ﻛ	ه
4	-4	☼	¶	$	4	D	T	d	t	▒	½	¤	٤	ؤ	ﺷ	ﻟ	ﻬ
5	-5	═	§	%	5	E	U	e	u	─	¼	ﺄ	٥	ﻊ	ﺻ	ﻣ	ﻰ
6	-6	║	▬	&	6	F	V	f	v	│	≈		٦	ﺋ	ﺿ	ﻧ	ﻲ
7	-7	╬	↨	'	7	G	W	g	w	┼	«		٧	ا	ط	ﻫ	ﻐ
8	-8	╣	↑	(	8	H	X	h	x	┤	»	ﺎ	٨	ﺑ	ظ	و	ق
9	-9	╦	↓	)	9	I	Y	i	y	┬	ﻷ	ﺏ	٩	ة	ﻋ	ى	ﻵ
10	-A	╠	→	*	:	J	Z	j	z	├	ﻸ	ﺕ	ﻑ	ﺗ	ﻏ	ﻳ	ﻶ
11	-B	╩	←	+	;	K	[	k	{	┴		ﺙ	؛	ﺛ	¦	ض	ل
12	-C	╗	∟	,	<	L	\	l	|	┐		،	ﺱ	ﺟ	¬	ﻌ	ك
13	-D	╔	↔	-	=	M	]	m	}	┌	ﻻ	ﺝ	ﺵ	ﺣ	÷	ﻎ	ﻱ
14	-E	╚	▲	.	>	N	^	n	~	└	ﻼ	ﺡ	ﺹ	ﺧ	×	غ	■
15	-F	╝	▼	/	?	O	_	o	⌂	┘	[illegible]	ﺥ	؟	د	ع	م	

1.3 – Le code EBCDIC (Exended Binary Coded Decimal Interchange Code)

Code décimal codé binaire étendu, il est utilisé essentiellement par IBM. C'est un code à 8 bits.
Ce code peut être assimilé à un code à 9 bits quand il fait usage d'une clé d'imparité (bit supplémentaire destiné à contrôler la validité de l'octet associé).

Son utilisation se fait sensiblement de la même façon que le tableau ASCII, à savoir qu'un caractère est codé par la lecture des valeurs binaires des intersections ligne/colonne. Ainsi, le caractère A se codera C1 en hexadécimal soit la suite binaire 11000001. Les caractères de commandes ont en principe la même signification qu'en ASCII. Ainsi, SP indique l'espace, CR le retour chariot, …

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX		PT			GE				FF	CR		
1	DLE	SBA	EUA	1C		NL				EM			DUP	SF	FM	ITB
2							ETB	ESC						ENQ		
3			SYN					EOT					RA	NAK		
4	SP										¢	.	<	(	+	
5	&										!	$	*	)	;	¬
6	-	/									¦	,	%	_	>	?
7											:	#	@	'	=	"
8		a	b	c	d	E	f	g	h	I						
9		j	k	l	m	N	o	p	q	r						
A		~	s	t	u	v	w	x	y	z						
B																
C	{	A	B	C	D	E	F	G	H	I						
D	}	J	K	L	M	N	O	P	Q	R						
E	\		S	T	U	V	W	X	Y	Z						
F	0	1	2	3	4	5	6	7	8	9						
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Le code EBCDIC Standard

1.4 – Unicode (UNIversal CODE)

Compte-tenu de l'extension mondiale de l'informatique et de la diversité de plus en plus importante des caractères à stocker, les organismes de normalisation ISO travaillent depuis 1988 à la création d'un code universel (UNIversal CODE).
Basé sur les pages ASCII, ce code est utilisé sous Windows 95 et Windows NT.

Il se représente sous 2 formes : Une forme 31 bits (UCS-4) pour Universal Charactere Set – 4 octets.
Et une forme 16 bits (UCS-2) ce qui permet de représenter théoriquement 65 535 caractères différents.
Ces caractères couvrent la majeure partie des principaux langages écrits du monde.

1.5 – Notion de contrôle de parité

A l'intérieur de l'ordinateur les informations sont sans cesse « véhiculées », du clavier vers la mémoire, de la mémoire vers le processeur, de la mémoire vers l'écran, … À l'heure actuelle, les ordinateurs sont de plus en plus reliés entre eux au travers de réseaux locaux ou étendus et l'information est donc constamment en circulation.
Il est donc nécessaire d'assurer une transmission convenable des informations. Pour cela, on utilise divers moyens allant du simple contrôle de parité jusqu'à l'élaboration de codes très sophistiqués.

Nous allons voir en quoi consiste le contrôle de parité.

Pourquoi un bit de contrôle de parité ?
Pour détecter des erreurs de transmissions.

Rq : Un caractère ASCII est sur 7 bits, or un octet contient 8 bits, donc le bit de poids le plus fort est inutilisé.
Il va servir au contrôle de parité.

Code dit à parité ou abusivement à parité paire

On complète la chaîne de n bits de façon à ce que la suite de bits transmise comporte un nombre pair de 1.

Donc le bit de poids le plus fort est mis à 1 si le nombre de 1 de l'octet est impair, à 0 sinon.

Exemple sur 8 bits

0	0	1	1	1	0	0	1

Bit de
parité → Il est à zéro (0) car le nombre de 1 est pair (il y a 4 un)

Code dit à imparité ou abusivement à parité impaire

On complète la chaîne de n bits de façon à ce que la suite de bits transmise comporte un nombre impair de 1.

Donc le bit de poids le plus fort est mis à 1 si le nombre de 1 de l'octet est pair, à 0 sinon.

Exemple sur 8 bits

1	0	1	1	1	0	0	1

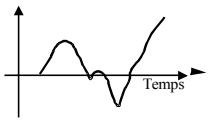
Bit de

parité → Il est à un (1) car le nombre de 1 est pair (il y a 4 un)

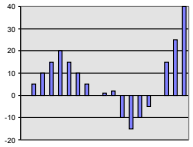
Rq :
Cette méthode très utilisée et généralement suffisante n'est en fait efficace que dans la mesure où il n'y a pas d'erreurs simultanées sur 2, 4, 6, ou 8 bits, ce qui ne changerait pas la parité. (ex. Si nous émettions la suite binaire 10010000 en parité et que nous recevions la suite 01010000, il serait impossible de dire s'il s'agit bien de ce qui a été envoyé ou non car, bien que la suite binaire reçue soit différente de celle émise, la parité est bien respectée).
Pour accroître la précision de la détection d'erreurs de transmissions, il existe d'autres méthodes plus complexes que nous ne ferons qu'énumérer ici : les codes de blocs et les codes cycliques.

2 – La représentation des informations multimédia

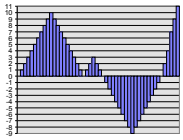
2.1 – Représentation du son



Comment représenté un son ? A priori, un son est un signal analogique (c'est à dire que le signal peut prendre une infinité de valeur au cours du temps). On mesure à intervalle régulier la valeur de l'amplitude d'un signal analogique.
Pour pouvoir stocker ce son dans un ordinateur, il va falloir l'échantillonner
Un échantillon sont des mesures à intervalles réguliers, de la valeur de l'amplitude d'un signal analogique



Pour être codé sous forme numérique, le son doit tout d'abord être échantillonné : la courbe va être remplacée par des segments de droites, l'infinité de valeurs possibles va être remplacé par un ensemble de niveaux.

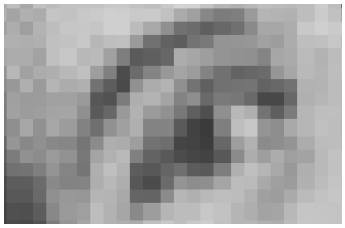


Pendant cette opération, on va forcément perdre de l'information. Afin que cette perte soit le moins sensible possible, on s'attachera à choisir une fréquence d'échantillonnage très grande (beaucoup de valeurs pour une courte durée) : il faut que la fréquence d'échantillonnage soit au moins le double de la plus haute fréquence contenue dans le signal. Pour un CD audio, la fréquence d'échantillonnage est de 44 100 Hz, c'est à dire qu'une seconde de musique est représentée par 44 100 valeurs différentes !

2.2 – Représentation par des images fixes

Toute image affichée sur un écran ou sur une imprimante est constituée de pixels (abréviation de Picture Element).
Il existe 2 grands modes de stockage des images : BITMAP et VECTORIEL.

BITMAP
Chaque point de l'image est mémorisé. L'image devra être utilisée dans la dimension dans laquelle elle a été créé, sinon un effet d'escalier sera inévitable.



Ces images sont très gourmandes en mémoire car chaque pixel est codé par un bit dans le cas d'une image noir et blanc, par 3 octets dans le cas d'une image couleur.
Pour les images en couleurs, chaque octet correspond à une des trois couleurs primaires (Rouge, Vert, Bleu). Les 256 valeurs codables sur un octet correspondent au poids de chacune des couleurs primaires qui compose le point coloré. Ce système donne un total de 16,5 millions de couleurs codables (2553), ce qui est grandement suffisant car l'œil humain est loin de pouvoir en discerner autant.

1 point de couleur : F0 A9 4C
Pour une image 640 x 480, si chaque point est codé sur 3 octets, cela donne une image qui va nécessiter 900 KO de mémoire, cette taille passera à plus de 2 MO pour une image 1024 x 768 !

C'est pourquoi, on utilise souvent un autre système de codage des images bitmaps, le codage à l'aide de la palette de couleur : On détermine 256 couleurs importantes dans l'image, ces couleurs sont codées sur 3 octets (mais une seule fois), ensuite chaque pixel est codé sur un octet qui correspond à une entrée dans la palette des couleurs. De cette façon, on divise à peu près par trois l'encombrement mémoire d'une image, mais on perd quelque peu en nuance et en contraste :

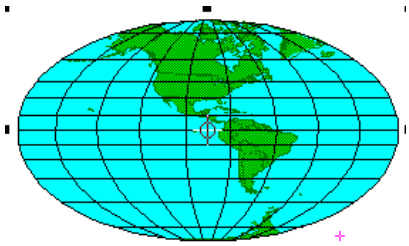
Exemple :
1 point de couleur : A6

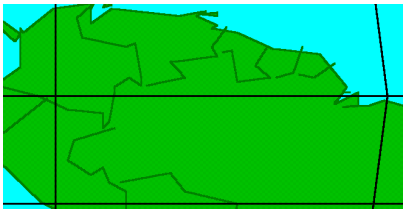
…	…
A5	F2 A6 78
A6	19 FE 6A
A7	05 3F 87
…	…3 octets de couleur

C'est cette technique qui est utilisée pour les images bmp.
Lorsqu'on veut représenter une image en mémoire, on est donc en face d'un choix : doit on privilégier le nombre de pixels d'une image (640x480 ou 1024x768), c'est à dire sa définition ou au contraire le nombre de couleurs utilisées (256 ou 16,5 millions) ? La plupart du temps, un meilleur résultat sera obtenu en privilégiant le nombre de couleurs.

VECTORIEL
Ce sont les équations mathématiques des portions de droites et de courbes qui sont mémorisées.
Un cercle, par exemple, sera déterminé complètement par les coordonnées du centre et la valeur du rayon avec éventuellement la couleur du trait et l'épaisseur de ce trait.

L'intérêt de cette méthode est que l'on peut modifier la taille du dessin sans en altérer la définition et les proportions. De plus si le dessin est composé de plusieurs objets, ils pourront être manipulés séparément.





2.3 – Représentation des images animées

Un image animée n'est autre qu'une suite d'images fixes se succédant rapidement. Ainsi, au cinéma, les images défilent à la cadence de 25 images par secondes. Un téléviseur affiche cinquante trames par seconde. (Une trame est constituée des lignes paires ou des lignes impaires de l'image, la télévision utilisant un affichage entrelacé.) Le résultat correspond à 25 images par seconde. Pour afficher des images animées, l'ordinateur doit être capable d'en faire autant. Or chaque image doit être calculée, puis transférée dans la mémoire vidéo. Ces opérations doivent se faire à la même vitesse, soit 1/25 de seconde pour une animation de haute qualité.

Animations d'objets simples

Entre deux images successives d'un film, tous les objets ne sont pas déplacés. L'image du décor est calculée une fois pour toutes. Puis l'image de l'objet à déplacer est superposée au décor ; La partie masquée du décor est mémorisée. Pour l'image suivante, le décor est reconstitué et l'objet superposé avec un décalage, dans le sens du déplacement. Les objets ainsi manipulés sont appelés *sprites*.
Si l'objet lui-même est animé, il est représenté par plusieurs sprites qui s'affichent à tour de rôle ; Par exemple, un animal qui court sera représenté par 10 à 20 sprites différents représentant les diverses attitudes de la marche.

Le scrolling

Parfois, le fond lui-même doit être animé. L'animation la plus courante est un défilement continu appelé *scrolling*. Dans ce cas, une image du fond dépassant largement la taille de l'écran est placée dans la mémoire vidéo. Un pointeur désigne la partie de l'image qui apparaît à l'écran . Tout se passe alors comme si le fonc défilait derrière une fenêtre. Des objets animés peuvent être superposée à l'image. Pour donner un résultat plus réaliste, on utilise la technique du *scrolling différentiel*. Elle consiste à présenter le fond sur plusieurs plans défilant à des vitesses différentes, pour donner une impression de relief (cas des dessins animés et jeux video).

Animation par image complète

Il s'agit d'images précalculées, c'est le cas des images vidéo, qui sont filmées par une caméra numérique avant d'être stockées sur disque. L'ordinateur n'a alors qu'à transférer des images dans la mémoire vidéo, au rythme choisi. L'idéal est biensûr d'afficher des images de la taille de l'écran au rythme de 24 par seconde. Cependant, cela n'est pas possible à cause de la quantité d'informations à traiter. Une image en 640x480 points et 16 millions de couleurs représente 900Ko de données.

Pour 24 images par seconde, cela donne 21 Mo de données transmises et traitées par seconde. Pour réduire la quantité d'informations plusieurs solutions sont utilisées :
Diminuer la taille de l'image en réduisant la fenêtre d'affichage,
Diminuer le nombre de couleurs (256 couleurs est une quantité raisonnable),
Diminuer le nombre d'image à la seconde (18 au lieu de 24),
Compresser les données et décompresser pendant le temps de traitement par une carte d'extension spéciale.

Animation par image de synthèse

Il est également possible de réaliser une animation en affichant des images synthétisées par l'ordinateur ; Elles peuvent avoir été calculées préalablement. On se rapproche alors du cas précédent. L'ordinateur doit calculer les images en temps réel, à mesure qu'elles sont affichées. Cette méthode est employée dans la plupart des programmes de simulation. Elle nécessite une puissance de calcul très élevée et des techniques de programmation très élaborées.

2.4 – La compression des données

Les problèmes de stockage du son, de l'image ou plus généralement des fichiers binaires posent des problèmes de place mémoire pour le stockage ou la transmission, c'est pourquoi avec l'ère du multimédia les techniques de compression sont d'actualité.

Une photo 24x36 cms, avec une définition de l'ordre de 150 points par mm^2 (qualité photo classique) représente 13 millions de points, à 3 octets par points on dépasse les 40 MO !

Un document vidéo c'est 25 images par secondes avec en plus du son ! !

On va donc être obligé d'utiliser des techniques de compression, il en existe 2 grandes catégories :

- **La compression avec perte d'informations** : Image et son peuvent se passer d'une restitution parfaite car de toute façon l'œil ne perçoit pas toutes les nuances, l'oreille n'entend pas toutes les fréquences. Les taux de compression peuvent atteindre 100 pour 1 !
  Les formats s'appelle par exemple **JPEG** (Joint Photographic Expert Group) pour les images fixes, **MPEG** (Moving Picture Expert Group) pour la vidéo. Pour la vidéo, les techniques utilisées englobent le découpage d'images en blocs, mais aussi la prédiction de mouvements pour déterminer les changements à opérer d'une image à l'autre.

- **La compression sans perte d'informations** : Un fichier binaire correspondant à un exécutable, ou à un texte ne peut se permettre la moindre altération.
  Les logiciels de compression s'appelle **ARJ** ou **PKZIP** (Phil Katz) ou **DRIVESPACE** intégré à MS-DOS. Les images GIF font également partie de cette catégorie.
  Le principe de la compression sans perte est de repérer les séquences répétitives (suite de valeurs identiques très fréquentes dans les images bitmap par exemple).
  Les techniques vont du simple codage de ces séquences répétitives sous la forme [valeur-nb d'occurrence], à des méthodes plus subtiles qui font intervenir la fréquence d'apparition de ces valeurs pour déterminer le nombre de bits qui va servir à coder ces différentes valeurs (méthode Huffman).