Plan du cours 4

- Représentation des types de base
  - Les entiers et les mots mémoires
  - Les réels et la précision numérique
  - Les chaînes de caractères
- Les images numériques
  - Types d'image
  - Représentation numériques
- Encodage de l'information
  - détection/correction d'erreur
  - compression
  - encryptage

Quelques définitions

- bit
  - **unité** minimale de l'information numérique: 0 ou 1
- byte ou octet: 8 bits (soit $2^8 = 256$ valeurs possibles) ex: 01101001
  - **groupe élémentaire** très souvent utilisé
  - exemple: pour coder le jeu de caractères d'un langage: 0-9,A-Z,a-z,{},(),+,*,/,,;:!...
- mot-mémoire ou mot (taille définie par l'architecture)
  - unité de base pour le stockage de l'information ou des instructions: aujourd'hui **32bit** ou **64bit** soit 4 ou 8 bytes.
  - une **adresse** en mémoire est associée à tout mot-mémoire
- principe de la représentation numérique
  - établir une correspondance (**codage** et **décodage**) entre un signifiant (chiffres, alphabet,...) et un ou plusieurs mots en binaire.

Représentation des entiers

- principe de l'écriture en base b

$$N = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_1 b^1 + a_0 = \sum_{i=0}^{n} a_i b^i$$

$$a_i \in \{0,1,\ldots b-1\} \text{ et } a_n \neq 0$$

  - base b=2: **binaire**
  - base b=10: **décimale**
  - base b=16: **hexadécimale**
- signe: un bit pour coder le signe
  - 0=moins 1=plus
- valeurs maximales possibles avec k bits
  - *sans* signe 2^k-1 soit ~4.10^9 en 32 bits et 2.10^{19} en 64 bits
  - *avec* signe: 2^{k-1}-1 soit ~2.10^9 en 32 bits et 10^{19} en 64 bits
- overflow
  - **dépassement de capacité**, contraint en particulier les *dimensions des tableaux*

Décimal	Binaire	Hexadécimal	Octal
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	8	10
9	1001	9	11
10	1010	A	12
11	1011	B	13
12	1100	C	14
13	1101	D	15
14	1110	E	16
15	1111	F	17

Représentation des réels

- on ne peut coder en réalité que des nombres fractionnaires
  - 12,398239 mais pas π ou 1/3 exactement, présence de la virgule
  - utilise la base avec puissances négatives b^{-i} : $0{,}12_{10} = 1.10^{-1} + 2.10^{-2}$
  - en binaire : $1{,}25_{10} \Rightarrow 1{,}01_2 = 1 + 0/2 + 1/2^2$
- décomposition d'un réel: principe de la virgule flottante
  - M **mantisse** (nombre fractionnaire avec signe)
  - b **base** (2, 8, 10 ou 16)
  - E **exposant** (un entier relatif codé avec un décalage)
  - similaire à la notation scientifique: 1.2342e+4
  - exemple standard de répartition sur 64 bits avec b=2

$$\boxed{N = M \times b^E}$$

signe: 1bit E: 11bits M: 52bits
0011001011011000101000101101101100011001101101101000101000101101011 00

- valeurs maximales possibles
  - en 32 bits: ~ 2e+308
  - en 64 bits: ~ 2e+4932
  - overflow: en cas de dépassement, retourne **NaN** ou **Inf**

Notion de précision machine

- Origine et ordre de grandeur
  - la distance entre deux réels représentables augmente avec l'exposant mais la distance relative est la même partout
  - nécessité d'**arrondir/tronquer** les résultats d'une opération => introduit une erreur 1.5/4.5 = 0.33333333333333
  - correspond en gros à changer le **dernier bit de la mantisse** de taille M => environ 2^{-M}
- exemples de précision machine (Norme IEEE 754 2008)
  - single precision: 32bits $2^{-23} \sim$ **10^{-6}**
  - double precision: 64bits $2^{-52} \sim$ **2.10^{-16}**
  - quadruple precision: 2x64bits $2^{-112} \sim 2.10^{-34}$
  - un calcul qui devrait retourner 0.0 peut retourner 2.83123e-16
- difficultés
  - l'erreur peut dépendre de l'ordre des opérations, attention aux intermédiaires de calculs (**attention aux overflow et à la précision**)
  - calculs de longues sommes ou produits, **accumulation d'erreurs**

Représentation des caractères

- caractères alphanumériques: A-Z a-z 0-9
- caractères spéciaux:){}[]?!.,:/\&«»''$%@#~...
- caractères de contrôles: liés à des touches du clavier
  - fin/début du texte, espace, retour chariot, tabulation, supprimer...
- caractères diacritiques (accents, cédille...): éèêàçïùœ...
- et tous les autres alphabets, idéogrammes...

少小离家老大回
乡音无改鬓毛摧。
儿童相见不相识、
笑问客从何处来？

- Il existe plusieurs encodages
  - **ASCII** : utilise 7 bits, 128 caractères, avec extension à 8 bits versions: ASCII, latin-1, latin-2
  - **UNICODE**: de 8 à 32bits, près de 100.000 caractères codés versions: utf-8, uft-32...

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Taille d'un fichier

- Les tailles de fichiers sont données en bytes
- Les **préfixes** kilo, méga, giga ont cependant une définition légèrement différente
  - **kilo**: $2^{10} = 1\,024$
  - **méga**: $2^{20} = 1\,048\,576$
  - **giga**: $2^{30} = 1\,073\,741\,824$
  - **téra**: $2^{40} = 1\,099\,511\,627\,776$
- exemples
  - un tableau de 1000 réels double precision (64 bits soit 8 bytes) correspond à 8x1000 bytes = 7.81kb.
  - une matrice 1000x1000 de réels correspond à 8×10^6 bytes = 7.63Mb

Notions générales

- Différents types/natures d'images numériques

  photos, graphiques, dessins, schéma, icônes, fontes...
- Représentation vectorielle

  image définie **à partir de formes géométriques** et **de courbes simples** => lisse à toute échelle
  très utile pour les graphiques/imprimantes
  ex: cercle défini par centre/rayon/épaisseur/couleur

- Représentation bitmap

  pixel: **"pic**ture **el**ement"
  carré élémentaire de couleur donnée
  image définie par une **matrice de pixels**
  typiquement obtenue par un capteur
  effet de **crénelage** ou **pixellisation** lorsque l'on zoome dessus

Propriétés d'une image matricielle

- La **définition** de l'image
  - nombre total de pixels sous la forme NxM (N lignes, M colonnes)
    exemple: 640x480 (définition VGA)
  - **rapport d'aspect**: N/M ou M/N noté en fraction
    exemple: 4:3 pour VGA

- La **résolution** de l'image
  - **nombre de pixels par unité de longueur**
  - **unité** de résolution **ppp** ou **ppi** : pixels par pouce (inches)
    1 pouce (inch) = 2,54 cm
  - le nombre de pixels varie comme le carré de la résolution
  - exemples
    résolution d'un écran pour l'affichage: 72 ou 96 ppi
    résolution d'une imprimante: 300 à 600 ppi
- Métadonnées
  - dimensions (16cmx12cm), date, lieu, propriétaire, appareil...

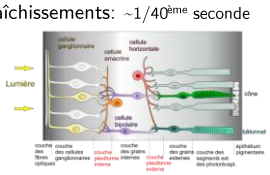
Résolution et vision humaine

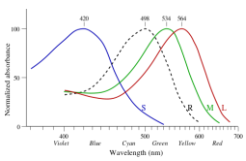
- Le choix dépend de la distance aux objets
- Pour l'**impression**, l'unité est le **dpi**: dot per inches

Distance du support	Résolution à partir de laquelle un œil humain moyen ne voit plus de différence, en points par pouce
6,3 cm	1 200 dpi
12,7 cm	600 dpi
20 cm	380 dpi
25,3 cm	300 dpi
30 cm	253 dpi
50 cm	152 dpi
76 cm	100 dpi
1 m	76 dpi
1,50 m	50 dpi
2 m	38 dpi
3 m	25 dpi
5 m	15 dpi
10 m	7,6 dpi
20 m	3,8 dpi

Perception humaine des couleurs

- Capteur: l'œil, traitement du signal: le cerveau
- Laps de temps entre deux rafraîchissements: ~1/40ème seconde
- Perception de la rétine
  - les **cônes** assurent la perception des couleurs pour la lumière diurne
  - **trois types** de cônes
    3 à 4 millions par œil
  - courbe d'**absorbance** des cônes
    trois couleurs privilégiées:
    rouge, vert, bleu
  - capable de distinguer
    environ 1 million de couleurs

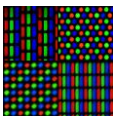




Synthèse additive des couleurs

- Couleur primaire
  - des couleurs sont primaires entre elles si on ne peut les obtenir par mélange des autres. Il en faut au moins 3.
  - exemple: le triplet **RGB**: Rouge Vert Bleu
- Synthèse additive
  - par mélange des trois couleurs primaires en jouant sur les intensités, on peut reconstruire un grand nombre de couleurs

- Utilisation
  - simple à mettre en œuvre pour des ***sources émettrices*** de lumière: écran, vidéoprojecteur,...



Synthèse soustractive des couleurs

- Couleur complémentaire
  - l'addition d'une couleur et de sa complémentaire donne une couleur neutre: noir/blanc/gris
  - **C**yan**M**agenta**Y**ellow (**CMY**) complémentaire de RGB
    cyan =bleu+vert
    magenta=bleu+rouge
    jaune =vert+rouge

- Une surface colorée éclairée en lumière blanche absorbe la couleur complémentaire qui se « soustrait » alors au blanc
- Synthèse soustractive
  - on utilise des **colorants/filtres** qui absorbent la lumière
  - avec le triplet primaire cyan, magenta et jaune, on peut reconstruire une couleur à partir du blanc en lui « *soustrayant* » ce qu'il faut de Rouge Vert Bleu

  - utilisé souvent pour des objets diffusant la lumière (papier): **cartouches d'imprimantes**

Représentation numérique de la couleur

- Profondeur d'une couleur
  - nombre de bits/pixel pour coder la couleur
- images en **noir et blanc**
  - un bit par pixel suffit: 0 = noir, 1 = blanc
- images en **niveaux de gris**
  - 8 bits par pixel => 256 valeurs
    0=noir...255=blanc
- images en **couleur**
  - on peut choisir 16 couleurs: 4 bits suffisent
  - avec la synthèse additive: type RGB
    8bits soit 256 valeurs par couleur primaire
    256^3 = 16,7 millions de couleurs

R	G	B	Couleur
0	0	0	noir
255	0	0	rouge
0	255	0	vert
0	0	255	bleu
128	128	128	gris
255	255	255	blanc
0	255	255	Cyan
255	0	255	magenta
255	255	0	jaune

- principe d'une palette
  - on sélectionne les 256 couleurs les plus pertinentes pour représenter l'image et on n'utilise que celles-ci

Les formats d'images

	Type (matriciel/ vectoriel)	Compression des données	Nombre de couleurs supportées	Affichage progressif	Animation	Transparence
JPEG	matriciel	Oui, réglable (avec perte)	16 millions	Oui	Non	Non
JPEG2000	matriciel	Oui, avec ou sans perte	4 milliards	Oui	Oui	Oui
GIF	matriciel	Oui, Sans perte	256 maxi (palette)	Oui	Oui	Oui
PNG	matriciel	Oui, sans perte	Palettisé (256 couleurs ou moins) ou 16 millions	Oui	Non	Oui (couche Alpha)
TIFF	matriciel	Compression ou pas avec ou sans pertes	de monochrome à 16 millions	Non	Non	Oui (couche Alpha)
SVG	vectoriel	compression possible	16 millions	* ne s'applique pas *	Oui	Oui (par nature)

Encodage de l'information

- Trois types de problèmes techniques/physiques qui se posent avec les données numérisées
  - assurer l'intégrité, **détecter et corriger des erreurs**
  - minimiser la taille, **compresser et décompresser**
  - sécuriser les données, **encrypter**
- Il faut trouver des algorithmes efficaces qui assurent ces tâches
- Il faut connaître l'existence de ces problèmes et savoir utiliser la compression/décompression et éventuellement l'encryptage

Détection d'erreurs

- Problématique
  - Les supports physiques de stockage et de transmission (semi-conducteurs, magnétiques, fibre optique, réseau électrique...) sont soumise à du bruit qui peut **altérer les données**: bit 0 -> 1
- Stratégie
  - **rajouter** de l'information pour pouvoir détecter voire corriger des erreurs, mais on ne veut pas en rajouter trop non plus.
- exemple: le bit de parité
  - on compte la parité du nombre de bits
  - en lisant l'information, on vérifie la parité

données 7bits	Parité	
	paire = 0	impaire = 0
0000000	00000000	10000000
1010001	11010001	01010001
1101001	01101001	11101001
1111111	11111111	01111111

  - exemple d'utilisation lors d'une transmission: s'il y a erreur, **on renvoie l'information**
  - **très rapide** mais **imparfait**: un nombre pair d'erreurs est non détecté

Correction d'erreurs

- une idée toute bête: la redondance
  - on envoie trois fois le mot: 10011 -> 100111001110011
  - on conserve ceux qui apparaissent au moins deux fois
  - pratique pour la transmission, moins pour le stockage
- autre exemple: la **double parité**

Caractère	Numéro de bit							Bit de parité	contrôle transversal
	1	2	3	4	5	6	7		
car. « 1 »	0	1	1	1	0	0	1	0	
car. « 9 »	0	1	1	1	0	0	1	1	
car. « 6 »	0	1	1	0	1	1	0	1	
car. « 8 »	0	1	1	1	0	0	0	0	
bit de parité	1	1	1	1	0	0	1		
contrôle longitudinal									

- en pratique:
  - il existe des solutions plus évoluées mais ces exemples simples illustrent bien la problématique et sa résolution

Compression

- Problématique
  - diminuer la taille de l'information à stocker
  - il faut obligatoirement deux algorithmes: un de **compression** un de **décompression**
- Propriétés essentielles
  - **qualité** de compression: on peut comprimer ***sans*** ou ***avec*** **pertes**
  - **taux** de compression: rapport entre la taille originale et la taille compressée, grand si on accepte les pertes
  - **temps** de compression: rapidité de l'algorithme dans le cas d'une transmission, la compression n'est intéressante que si elle fait gagner du temps.
- exemples de formats de fichiers compressés:
  - sans pertes: zip, rar, bz2,... (tout type)
  - avec pertes: jpeg, mpeg, mp3... (images, video, son)

Quelques idées simples

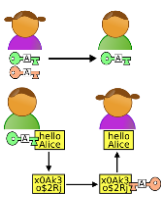
- Run-length encoding (sans perte)
  - 00000001111100111111111
  - remplacé par: 7'0'4'1'2'0'3'1'9'1'
  - adapté aux données avec des longues séquences de 0 et 1: par exemple à des images en noir et blanc
- Codage de Huffmann (sans perte)
  - on crée un **dictionnaire** des mots qui reviennent souvent et on les remplace par un mot qui prend moins de mémoire "maintenant" prend 10 bytes remplacé par « 1 » écrit sur 1 byte et remplacé « 1 » si besoin par autre chose s'il apparaît moins souvent
  - adapté à du texte mais le principe est général
- JPEG (avec pertes)
  - l'image est découpée en **domaines de 8x8 pixels**
  - une prend une sorte de **série de Fourier** du signal dont on ***tronque*** les hautes fréquences => étude en TP traitement du signal

Chiffrement de l'information

- Problématique
  - on veut **protéger** l'accès au contenu des données. **Sureté**: protéger des erreurs accidentelles, **Sécurité**: protéger des actions malveillantes
  - **chiffrement/déchiffrement**: « code secret », on veut rendre le contenu inutilisable pour qui n'est pas censé y accéder
- Code César: on remplace chaque lettre par celle située 3 lettres plus loin dans l'alphabet
  - Veni, vidi, vici => Zhqm, zmgm, zmfm (pas de 'j','u','w' en latin)
- Principe du masque
  - Les interlocuteurs échangent un **masque** 0011001001001000 qui dit quels bits doivent être inversés dans le message envoyé sous forme binaire. Ce masque est **construit aléatoirement**. On dit qu'ils échangent une **clé de chiffrement secrète** qui sert au décodage.
  - **Problème**: la clé fait la même longueur que le message et il faut pouvoir l'échanger sans que la clé soit interceptée.

Méthode clé publique – clé privée

- Objectif
  - pas d'échange préalable de clé de déchiffrement secrète, éviter l'interception de la clé
- Cryptographie asymétrique
  - **analogie**: on envoie un cadenas ouvert mais sans la clé. L'expéditeur peut fermer le cadenas mais pas l'ouvrir
  - Alice veut pouvoir recevoir des messages secrets de n'importe qui
  - **Étape 1**: Alice génère deux clés, une privée (la clé du cadenas) qu'elle conserve en sécurité une publique (le cadenas) qu'elle diffuse librement
  - **Étape 2**: Bob chiffre le message en utilisant la clé publique et l'envoie à Alice qui est la seule à pouvoir le décoder



Protocole RSA

- utilisé le plus couramment (email, connexion sécurisée,...)
  - **clé privée**: trois nombres d, e et n tels que pour tout entier w < n
    $(w^e \% n)^d \% n = w$
    avec % = modulo ou reste de la division entière
  - **clé publique**: e et n uniquement
  - **chiffrement**: pour un bout de message sous forme d'une suite de bits correspondant à un entier < n, on envoie
    $w' = w^e \% n$
  - **déchiffrement**: on utilise d et l'opération
    $(w')^d \% n \Rightarrow w$
  - une personne malveillante peut avoir accès à e et n et chercher d par la définition mais le calcul est **extrêmement long** (plusieurs années) et c'est cela qui protège le protocole.