

Cette activité de groupe doit être comprise et réalisée en **autonomie**. Chacun de vous doit être capable de faire et d'expliquer les exercices d'applications

## 1. Encodage de textes

Comment la lettre 'Z' est-elle stockée dans la mémoire de l'ordinateur ? Comment la différencier de la lettre 'A' ?

Pour répondre à ces questions, il a fallu créer une **table de conversion** entre un caractère et sa représentation en binaire.

Première table utilisée (début 1960) : **ASCII**

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]

**ACTIVITÉ 1**

Quelle est la représentation en **décimal** du caractère Z ? en **binaire** ?

Quelle sont celles de la caractère z ? Comment représenter le caractère é ?

**CORRECTION**

Z est représenté par le nombre décimal  $(90)_{10}$ . Ce qui donne en binaire  $(1011010)_2$  car  $90 = 64 + 16 + 8 + 2 = 2^6 + 2^4 + 2^3 + 2^1$

z est représenté par le nombre décimal  $(122)_{10}$ . Ce qui donne en binaire  $(1111010)_2$  car  $122 = 64 + \mathbf{32} + 16 + 8 + 2 = 2^6 + 2^5 + 2^4 + 2^3 + 2^1$

Impossible avec la table ASCII. C'est pour cela qu'à la fin des années 1960 ont été introduites d'autres tables (par exemple ISO-8859).



**Conversions texte ↔ binaire**

La norme Unicode (début 1990) permet une correspondance entre chaque caractère de l'*humanité* utilisé dans le monde (passé et présent) et un nombre entier appelé **point de code**.

Le point de code d'une taille maximale de 32 bits peut être représenté par des nombres binaires de tailles différentes :

- 1 à 4 paquets de taille 8 bits : UTF-8
- 1 à 2 paquets de taille 16 bits : UTF-16
- 1 paquet de taille 32 bits : UTF-32.

Dans la suite de ce document, nous allons étudier *uniquement* l'encodage en UTF-8 des caractères suivants :

caractère	origine	point de code décimal	point de code binaire
A		65	0100 0001
Ω	lettre grecque	937	0011 1010 1001
文	idéogramme chinois	25 991	0110 0101 1000 0111
	ougaritique (Syrie) des années -1400 à -1100	66 436	0001 0000 0011 1000 0100

**Méthode UTF-8** 4 cas en fonction de la longueur du point de code :

**Cas 1 : point de code de longueur 7 bits ou moins :**

- encodage sur 7 bits identique au point de code
- complété par des 0 à gauche pour atteindre une taille de 8 bits (*padding*)

point de code en binaire	valeur en UTF-8
0xxx xxxx	0xxx xxxx

**Cas 2 : point de code de longueur 8 bits à 11 bits :**

- encodage sur  $2 \times 8$  bits
- utilisation des préfixes : 110 puis 10

point de code en binaire	valeur en UTF-8
0yyy yyxx xxxx	110y yyyy 10xx xxxx

**Cas 3 : point de code de longueur 12 bits à 16 bits :**

- encodage sur  $3 \times 8$  bits
- utilisation des préfixes : 1110 puis 10 et 10

point de code en binaire	valeur en UTF-8
zzzz yyyy yyxx xxxx	1110 zzzz 10yy yyyy 10xx xxxx

**Cas 4 : point de code de longueur supérieur à 16 bits :**

- encodage sur  $4 \times 8$  bits
- utilisation des préfixes : 11110 puis 10, 10 et 10

point de code en binaire	valeur en UTF-8
000u uuuu zzzz yyyy yyxx xxxx	1111 0uuu 10uu zzzz 10yy yyyy 10xx xxxx

**2. Exemples**

	point de code en binaire	valeur en UTF-8	remarque
A	0100 0001	0100 0001	(cas 1) 1×8 bits
Ω	0011 1010 1001	1100 1110 1010 1001	(cas 2) 2×8 bits
文	0110 0101 1000 0111	1110 0110 1001 0110 1000 0111	(cas 3) 3×8 bits
𐄣	0001 0000 0011 1000 0100	1111 0000 1001 0000 1000 1110 1000 0100	(cas 4) 4×8 bits

**Remarques** UTF-8 est optimisé pour les caractères occidentaux (utilisation de 1 octet) **mais** pas pour les caractères asiatiques (utilisation de 3 octets).

**3. Exercice d'application****ACTIVITÉ 2**

Compléter le tableau ci-dessous :

caractère	point de code	UTF-8
𐄣	0000 0001 0000 0010 0100	
N	0000 0000 0000 0100 1110	
§		1100 0101 1001 1111
≡	0001 1101 0011 0011 1101	