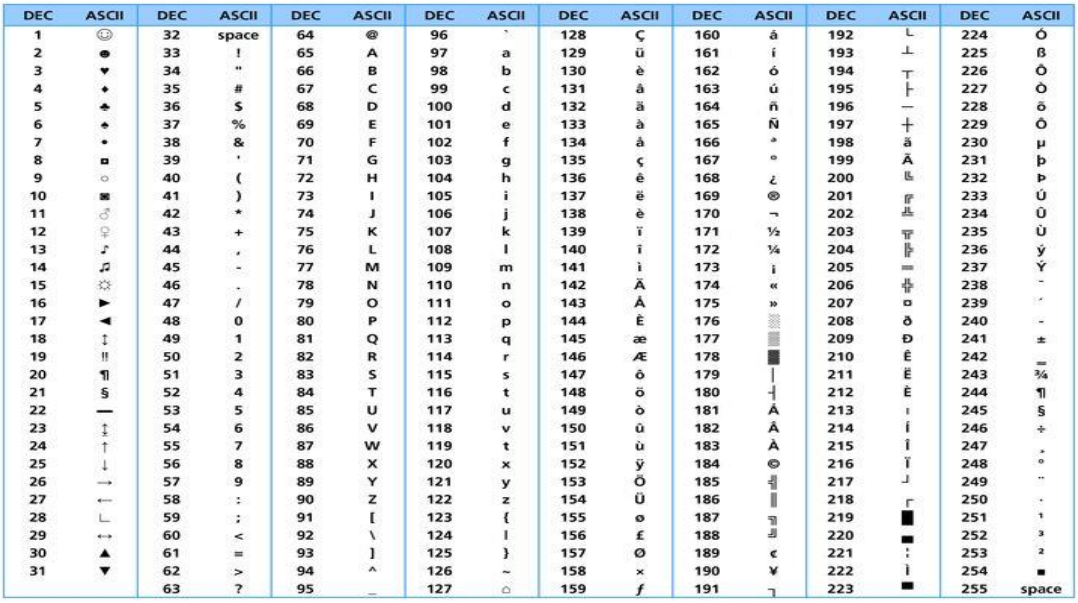
ENCODING

Encoding is the process of converting data from one form to another, following certain rules. There are different types of encoding, such as image encoding, audio and video encoding, and character encoding[1]. Encoding can be used for various purposes, such as compression, encryption, conversion, and memory[2][3].

Some examples of encoding are:

- ASCII and Unicode are character encoding schemes that assign a number to each character used in text files[1].
- JPEG and PNG are image encoding formats that compress and store image data in a binary file[1].
- MP3 and WAV are audio encoding formats that reduce the size and quality of sound data in a digital file[1].
- MPEG and AVI are video encoding formats that compress and store video data in a digital file[1].
- Base64 and Hexadecimal are encoding methods that convert binary data into a readable text format[3].

Encoding is different from encryption, which is a process of hiding or securing data using a secret key or algorithm[2]. Encoding can be reversed without a key, while encryption requires a key to decrypt the data[2].



DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII
1	☺	32	space	64	@	96	`	128	Ç	160	á	192	└	224	Ó
2	☻	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
3	♥	34	"	66	B	98	b	130	è	162	ó	194	┬	226	Ô
4	♦	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
5	♣	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
6	♠	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
7	•	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
8	◘	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
9	○	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
10	◙	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
11	♂	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
12	♀	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
13	♪	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
14	♫	45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
15	☼	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
16	►	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
17	◄	48	0	80	P	112	p	144	É	176	░	208	ð	240	
18	↕	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
19	‼	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
20	¶	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
21	§	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
22	▬	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
23	↨	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
24	↑	55	7	87	W	119	t	151	ù	183	À	215	Î	247	¸
25	↓	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
26	→	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
27	←	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
28	∟	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
29	↔	60	<	92	\	124	|	156	£	188	╝	220	▄	252	³
30	▲	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
31	▼	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
		63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	space

ENCODING VS UNICODE

Encoding and Unicode are two different concepts that are related to how text data is represented and processed in computers. Encoding is a method of converting characters, such as letters, digits, and symbols, into bytes, which are units of data that can store 256 different values. Unicode is a standard that assigns a unique number, called a code point, to each character in the world's languages and scripts[1].

There are many encoding schemes that can be used to encode Unicode characters, such as UTF-8, UTF-16, and UTF-32. These encoding schemes differ in how many bytes they use to encode a character, and how they arrange the bits in each byte. For example, UTF-8 uses one to four bytes per character, and it is compatible with ASCII, which is an older encoding that uses one byte per character for the English alphabet and some control codes. UTF-16 uses either two or four bytes per character, and it can encode most of the commonly used characters in two bytes. UTF-32 always uses four bytes per character, and it can encode any Unicode character directly[2].

The choice of encoding scheme depends on various factors, such as the size and speed of the data, the compatibility with other systems, and the support for different languages and symbols. Different encoding schemes may have different advantages and disadvantages for different purposes. For example, UTF-8 can save space and bandwidth for texts that mostly use ASCII characters, such as English, but it can take more space and bandwidth for texts that use many non-ASCII characters, such as Chinese or Arabic. UTF-16 can encode most of the commonly used characters in two bytes, which can make it faster and easier to process, but it can waste space and bandwidth for texts that mostly use ASCII characters. UTF-32 can encode any Unicode character directly, which can make it simpler and more consistent, but it can be very inefficient and wasteful for most texts[3].

For more information, you can read these articles:

- What are Unicode, UTF-8, and UTF-16?
- Comparison of Unicode encodings
- FAQ - UTF-8, UTF-16, UTF-32 & BOM

ENCODE TEXT INTO BINARY

Steps to convert text in binary system:

- Create a new Windows Forms Application project in Visual Studio
- Design your form using the Windows Forms Designer
- Add a label, a text box, and a button to your form
- Set the properties of the label, text box, and button, such as name, text, font, size, and location
- Double-click the button to generate the click event handler in the Code Editor
- Write the code for encoding the text from the text box into binary and displaying it in the label
- Build and run your application

```
            // Get the text from the text box
            string text = textBox1.Text;

            // Convert the text to a byte array using ASCII encoding
            byte[] bytes = Encoding.ASCII.GetBytes(text);

        // Convert each byte to a binary string and concatenate them
            string binary = "";
            foreach (byte b in bytes)
            {
                binary += Convert.ToString(b, 2).PadLeft(8, '0');
            }

            // Display the binary string in the label
            label1.Text = binary;
```

DECODE FROM BINARY TO TEXT:

```
string binaryText = txtBinary.Text;
            string result = "";

            for (int i = 0; i < binaryText.Length; i += 8)
            {
                string byteString = binaryText.Substring(i, 8);
                int byteValue = Convert.ToInt32(byteString, 2);
                char charValue = Convert.ToChar(byteValue);
                result += charValue;
```

```
            }

            txtText.Text = result;
```

HEXADECIMAL:

Encode text to hexadecimal:

```
string text = txtText.Text;
            string result = "";

            byte[] bytes = Encoding.UTF8.GetBytes(text);
            StringBuilder sb = new StringBuilder();

            foreach (byte b in bytes)
            {
                sb.Append(b.ToString("X2"));
            }

            result = sb.ToString();
            txtHexaDecimal.Text = result;
```

Decode hexa to text :

```
string hexString = txtHexaDecimal.Text;
            string result = "";

            for (int i = 0; i < hexString.Length; i += 2)
            {
                string hexValue = hexString.Substring(i, 2);
                byte byteValue = Convert.ToByte(hexValue, 16);
                result += Convert.ToChar(byteValue);
            }

            txtText.Text = result;
```

THANKYOU ☺

ENG / AHMED ABDULQADER