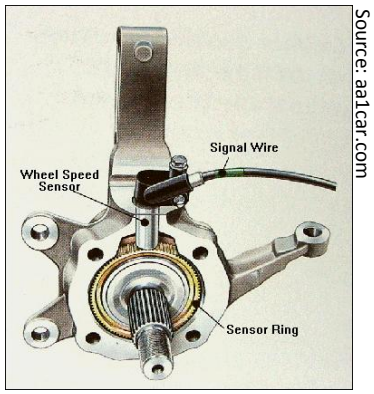
EEE243 – Applied Computer Programming

Input and outputs

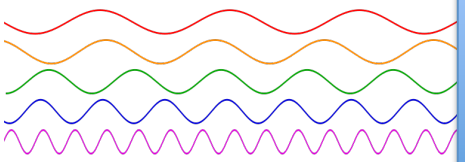
Outline

1. Encoding
2. Peripherals
3. Data streams
4. File manipulations

Encoding

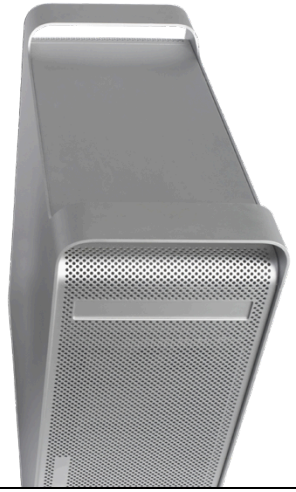


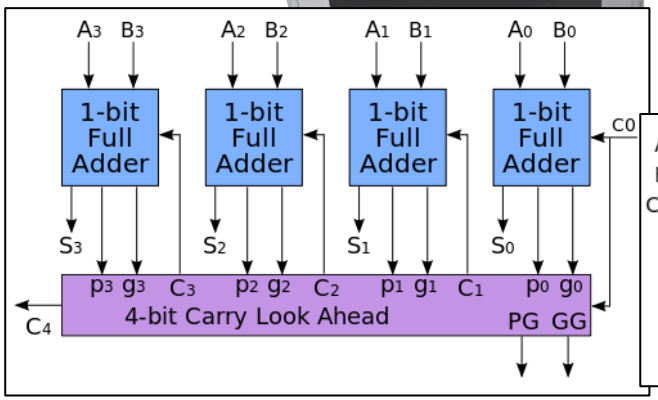
Source: aa1car.com

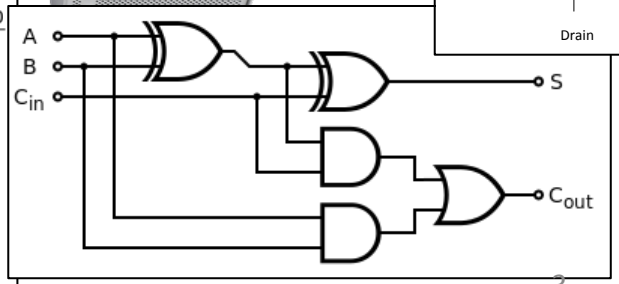


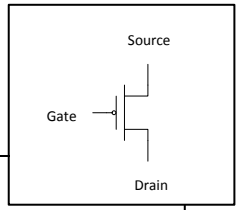
Converter

0101010101010101010000111000O









Encoding

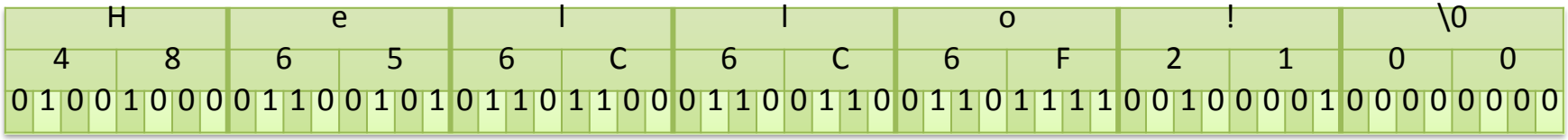
ASCII TABLE

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]

Source: wikimedia.org

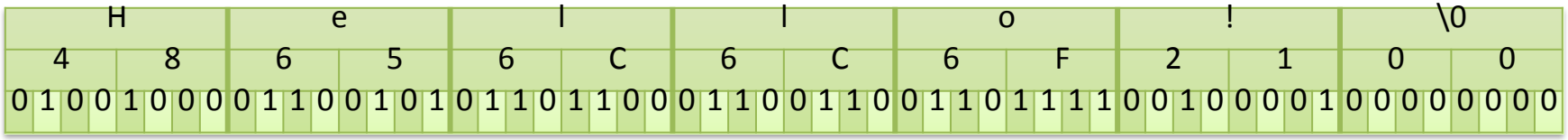
Encoding

```
Hello!
```



Encoding

```
Hello!
```

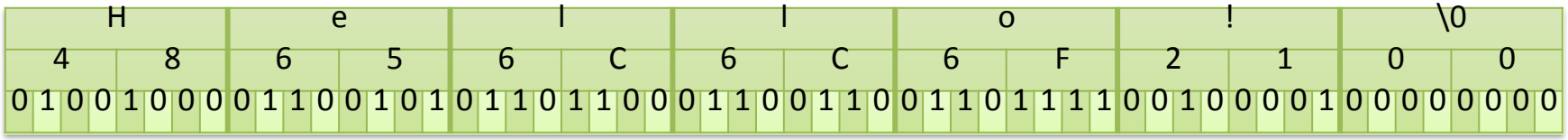



```
Allô!
```

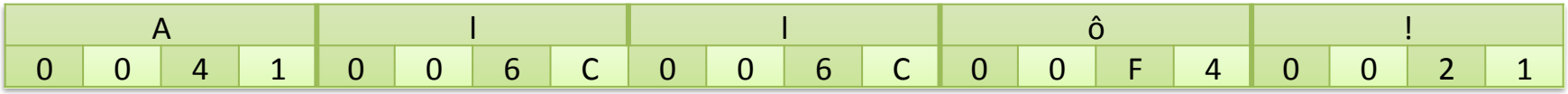
?

Encoding

```
Hello!
```




```
Allô!
```



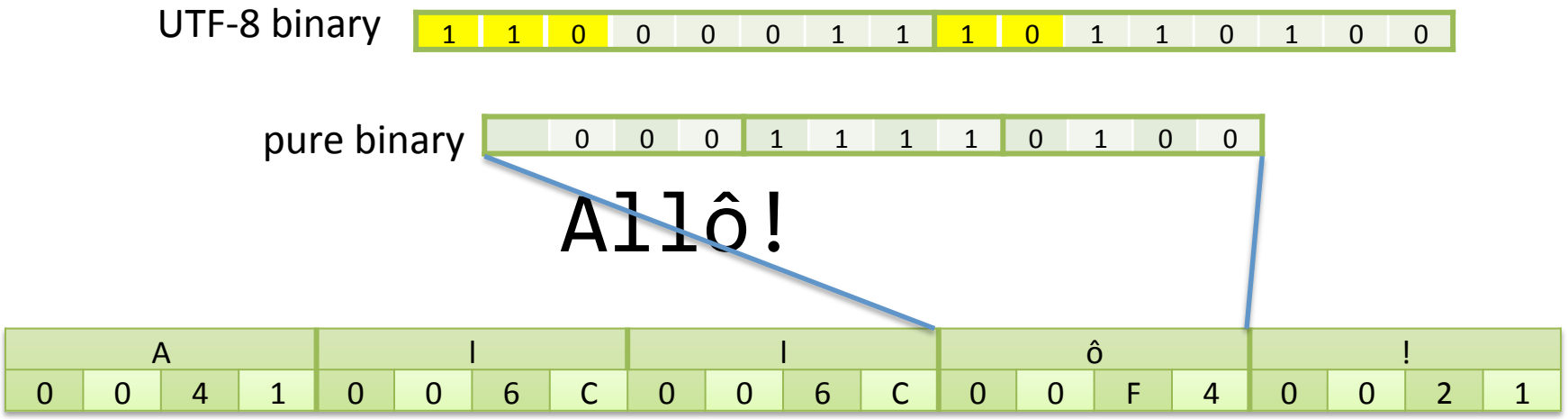
Encoding

Allô!

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

From [1]

Encoding

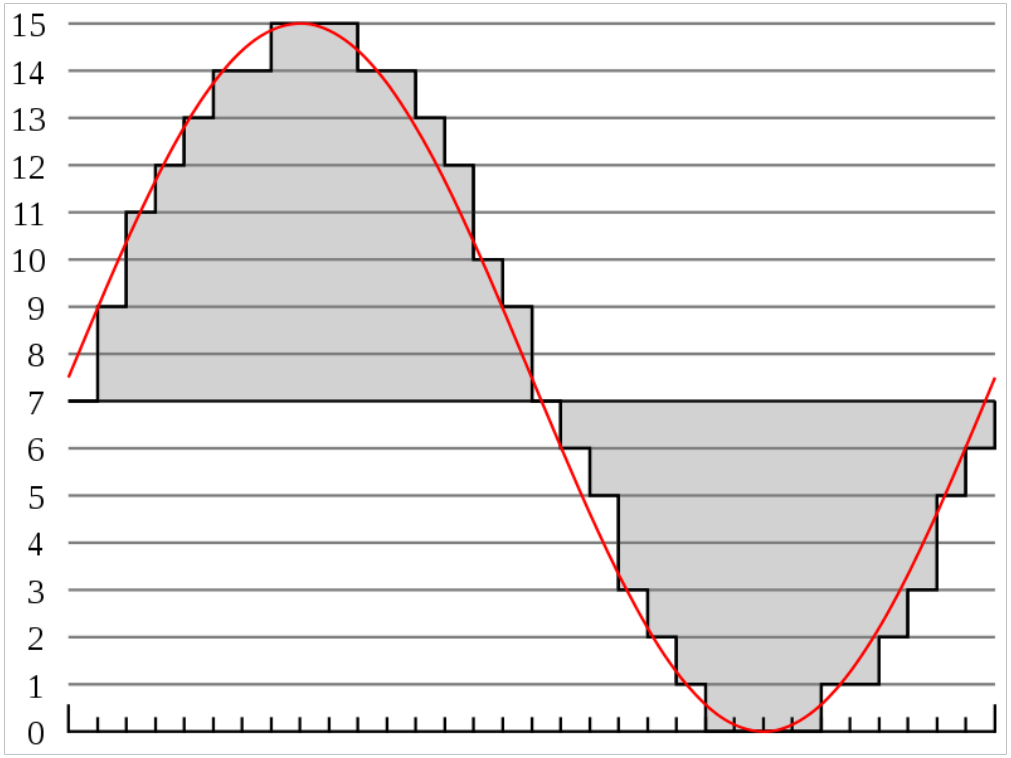


Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

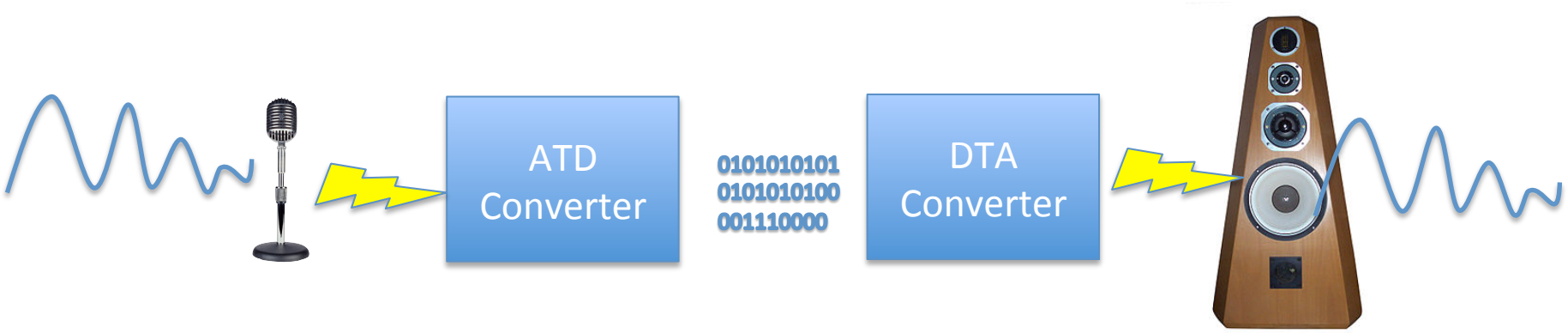
From [1]

Encoding and Digitization

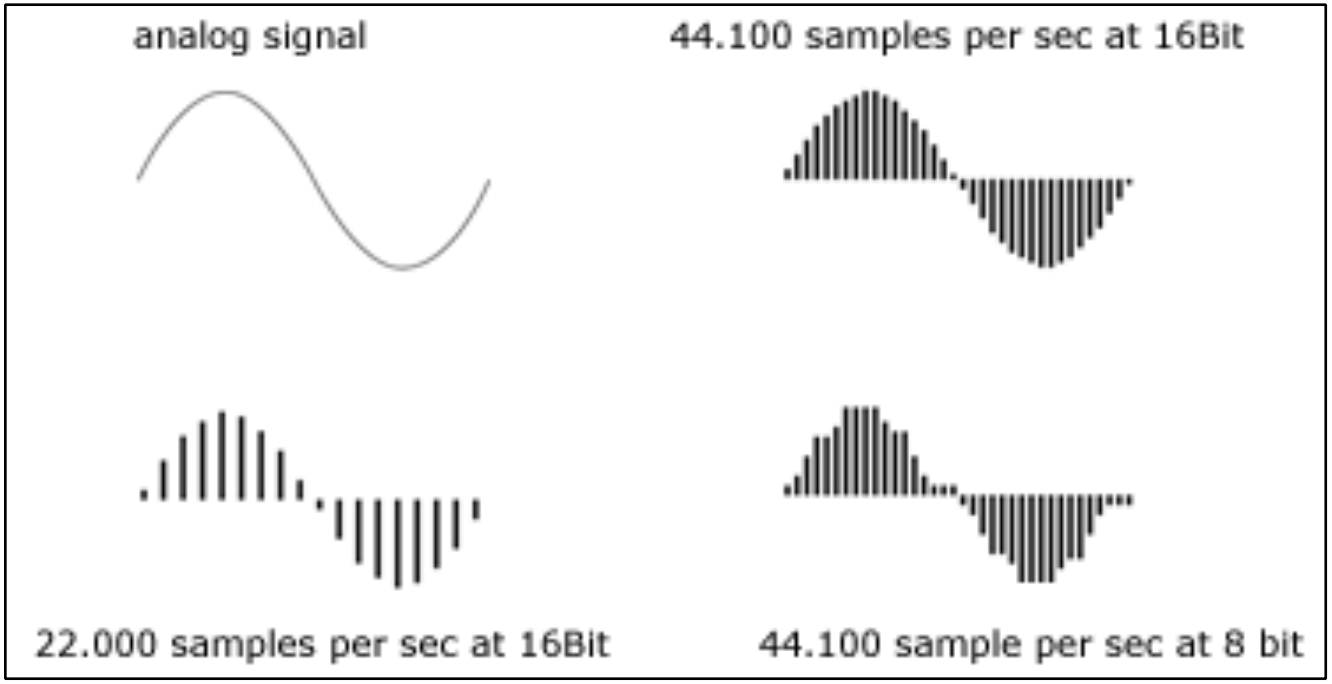
Digitization: going from an analog representation of something to a digital form



Encoding and Digitization



Audio Digitization



Audio Digitization

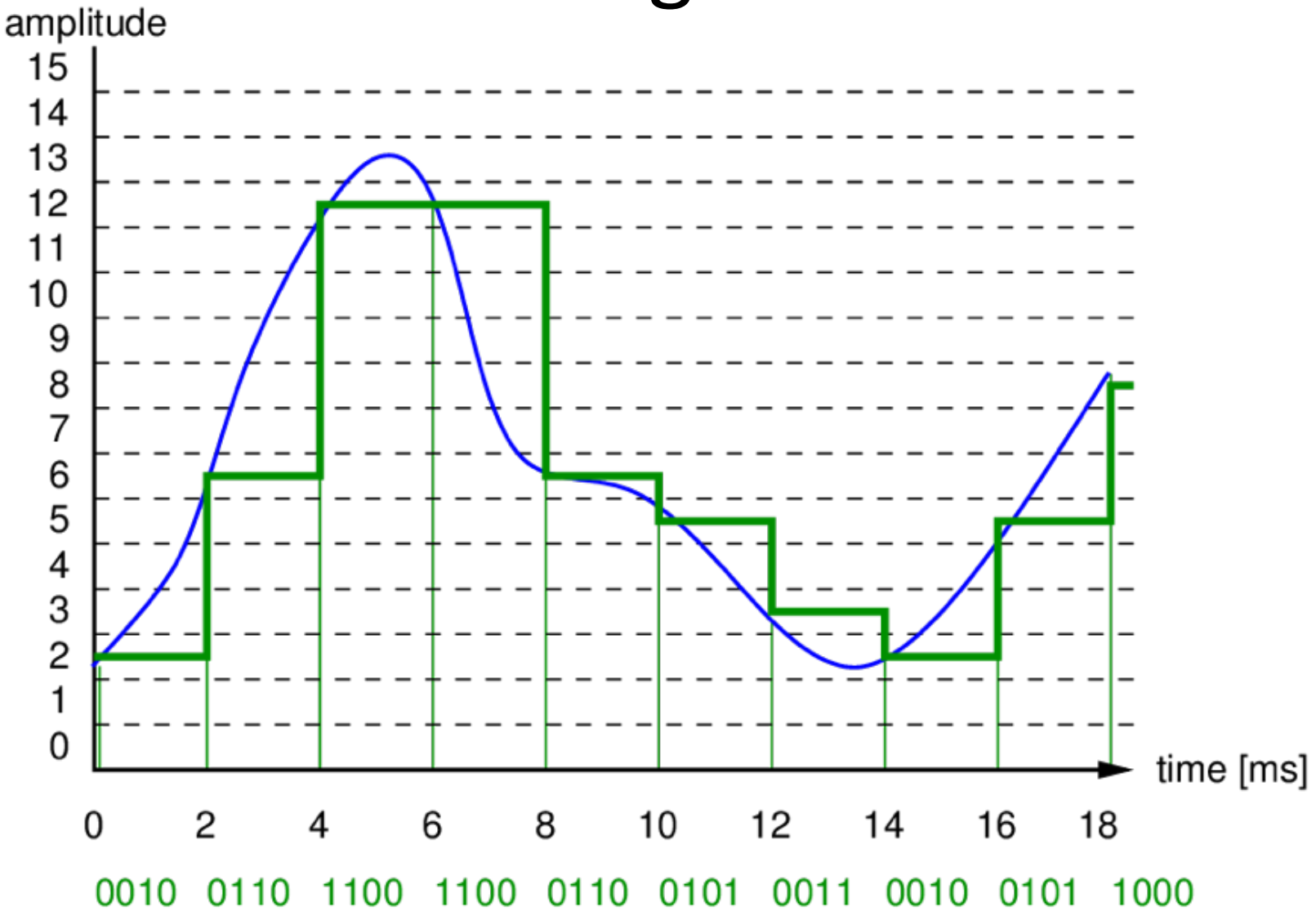


Image Digitization

Image Digitization

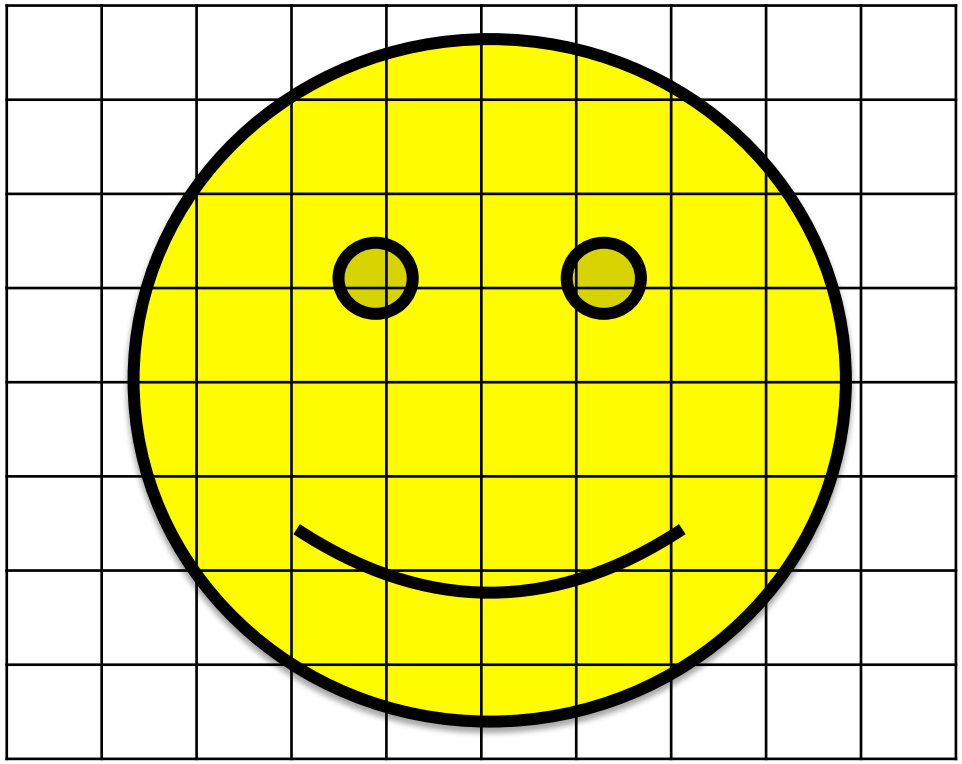


Image Digitization

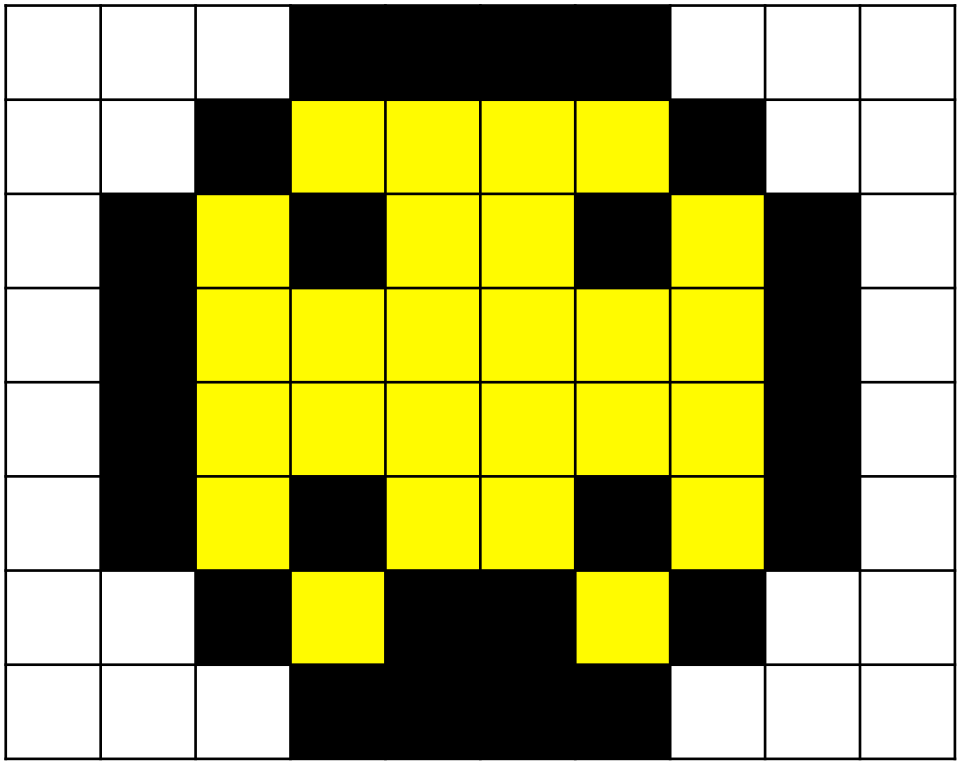


Image Digitization

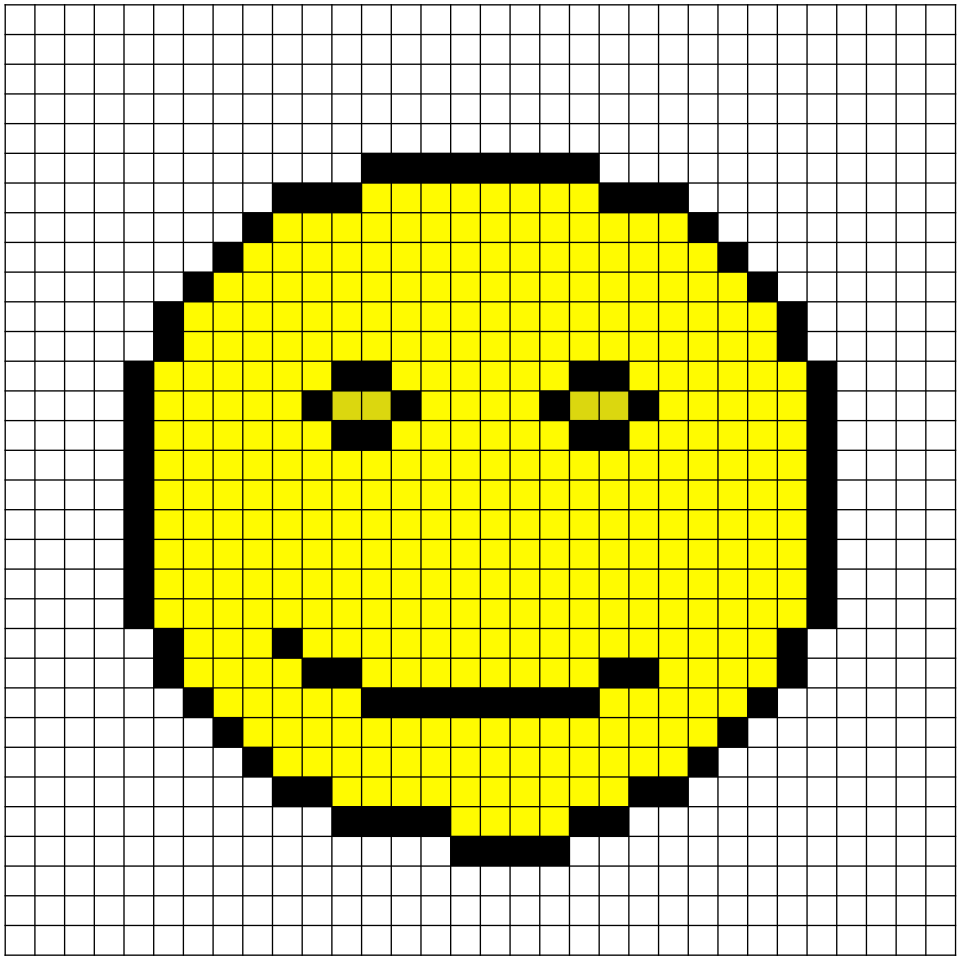
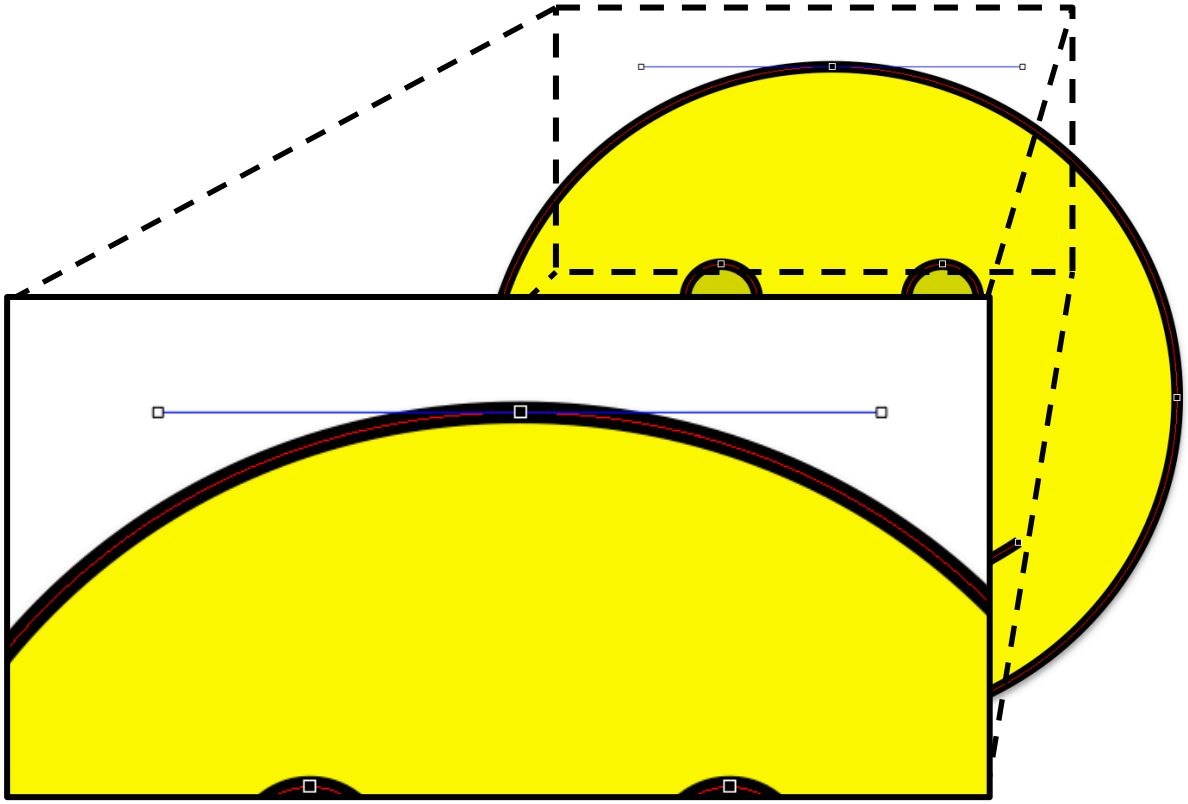


Image Digitization



File Encoding

File: “a computer resource for encoding data discretely in a computer storage device.”[2]

- Allows for persistent data.
- Allows to logically group together bits
- Segregates interpretation

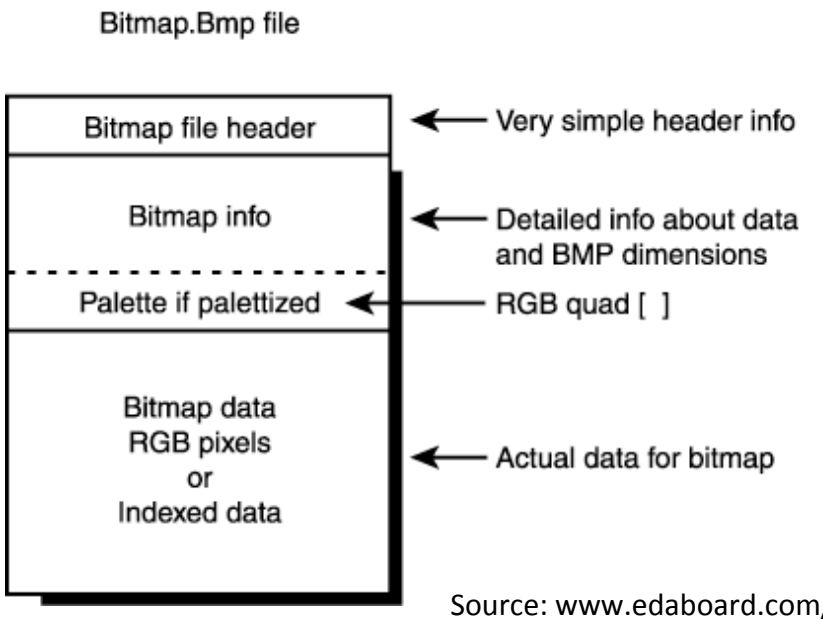
File Encoding

File format: “a standard way that information is encoded for storage in a computer file. It specifies how bits are used to encode information in a digital storage medium.”[1]

- Png, bmp, jpeg, mp3, mp4, html, etc.

File Encoding

- Identifying file type
  - OS specific
  - most use the extension (*.txt)
  - metadata



Source: www.edaboard.com/entry764.html

File Structure Example

Basic BMP File Format		
Name	**Size**	**Description**
Header	14 bytes	Windows Structure: BITMAPFILEHEADER
Signature	2 bytes	'BM'
FileSize	4 bytes	File size in bytes
reserved	4 bytes	unused (=0)
DataOffset	4 bytes	File offset to Raster Data
InfoHeader	40 bytes	Windows Structure: BITMAPINFOHEADER
Size	4 bytes	Size of InfoHeader =40
Width	4 bytes	Bitmap Width
Height	4 bytes	Bitmap Height
Planes	2 bytes	Number of Planes (=1)
BitCount	2 bytes	Bits per Pixel 1 = monochrome palette. NumColors = 1 4 = 4bit palletized. NumColors = 16 8 = 8bit palletized. NumColors = 256 16 = 16bit RGB. NumColors = 65536 (?) 24 = 24bit RGB. NumColors = 16M
Compression	4 bytes	Type of Compression 0 = BI_RGB no compression 1 = BI_RLE8 8bit RLE encoding 2 = BI_RLE4 4bit RLE encoding
ImageSize	4 bytes	(compressed) Size of Image It is valid to set this =0 if Compression = 0
XpixelsPerM	4 bytes	horizontal resolution: Pixels/meter
YpixelsPerM	4 bytes	vertical resolution: Pixels/meter
ColorsUsed	4 bytes	Number of actually used colors
ColorsImportant	4 bytes	Number of important colors 0 = all
ColorTable	4 * NumColors bytes	present only if Info.BitsPerPixel <= 8 colors should be ordered by importance
Red	1 byte	Red intensity
Green	1 byte	Green intensity
Blue	1 byte	Blue intensity
reserved	1 byte	unused (=0)
repeated NumColors times		
Raster Data	Info.ImageSize bytes	The pixel data

Source: www.ue.eti.pg.gda.pl/fpgalab/zadania.spartan3/zad_vga_struktura_pliku_bmp_en.html

Streams

"The logical channels on which I/O is performed." [3]

- Text
  - characters are organized in lines (terminated by new line indicator)
  - Read and write operations can affect the content (new line indicator modified depending on implementation)
- Binary
  - Ordered sequence of characters
  - No modification of content by implementation.

Standard Streams

Automatically created when a program starts

- stdin
  - points to the keyboard
  - `scanf` uses it
- stdout
  - points to the screen (console)
  - `printf` uses it
- stderr
  - points to the screen (console)

Standard Streams

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    char name[30];
    scanf("%s", name); // uses stdin
    printf("Hello World!\n"); // uses stdout
    fprintf(stderr, "An error"); //uses stderr
    return EXIT_SUCCESS;
}
```

Output

```
Hello World!
An error
```

Handling Files in C

- `stdio` contains a definition of the type `FILE`
- **`fopen`:** opens a file

```
FILE *fopen(const char *filename, const char *mode);
```

r	open a text file for reading
w	truncate to zero length or create a text file for writing
a	append; open or create text file for writing at end-of-file
rb	open binary file for reading
wb	truncate to zero length or create a binary file for writing
ab	append; open or create binary file for writing at end-of-file
r+	open text file for update (reading and writing)
w+	truncate to zero length or create a text file for update
a+	append; open or create text file for update
r+b or rb+	open binary file for update (reading and writing)
w+b or wb+	truncate to zero length or create a binary file for update
a+b or ab+	append; open or create binary file for update

- **`fclose`:** closes a file

```
int fclose(FILE *stream);
```

- **`fseek`:** sets the file position indicator

```
int fseek(FILE *stream, long int offset, int whence);
```

`SEEK_SET`, `SEEK_CUR`, or `SEEK_END`

Handling Files in C

- **`ftell`:** obtains the current value of the file position indicator.

```
long int ftell(FILE *stream);
```

- **`fgetc`:** obtains the next character and advances the file position indicator

```
int fgetc(FILE *stream);
```

- **`fputc`:** writes the character to the stream and advances the file position indicator

```
int fputc(int c, FILE *stream);
```

Example

```
#include <stdio.h>
#include <stdlib.h>

void double_space(FILE *in, FILE *out);

int main() {
   FILE *in, *out;

   in = fopen("lorem.txt", "r");
   out = fopen("new-lorem.txt", "w");

   double_space(in, out);

   int err = fclose(in);
   err = fclose(out);

   return EXIT_SUCCESS;
}

void double_space(FILE *in, FILE *out) {
   int c;
   while((c = fgetc(in)) != EOF) {
      fputc(c, out);
      if(c == '\n'){
         fputc('\n', out);
      }
   }
}
```

```
Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui
officia deserunt mollit anim id est laborum.
```

```
Lorem ipsum dolor sit amet,

consectetur adipiscing elit,

sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur.

Excepteur sint occaecat cupidatat non proident, sunt in culpa qui
officia deserunt mollit anim id est laborum.
```

example inspired from [4]

Handling Files in C

- **`fread`:** used to read binary streams and put the information into an array. Advances the file position indicator

```
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream)
```

- **`fwrite`:** writes from an array to a binary stream and advances the file position indicator

```
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
```

Handling Files in C

- **`fprintf`:** used to print formatted text to a text stream. Works similarly to `printf`, but it is possible to specify the stream instead of the default `stdout`.

```
int fprintf(FILE *stream, const char *format, ...);
```

For more details on the stream/file manipulation functions, refer to [5]

Example

```
#include <stdio.h>
#include <stdlib.h>

#define LINE_LEN 10

int main() {
    char buffer;
    int count = 0;

    FILE *input = fopen("lorem.txt", "rb");
    FILE *output = fopen("lorem-bin.txt", "w");

    while (fread(&buffer, 1, 1, input) == 1) {
        if (count == LINE_LEN) {
            fprintf(output, "\n");
            count = 0;
        }
        fprintf(output, "%x ", buffer);
        count++;
    }

    return EXIT_SUCCESS;
}
```

```
Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui
officia deserunt mollit anim id est laborum.
```

```
4c 6f 72 65 6d 20 69 70 73 75
6d 20 64 6f 6c 6f 72 20 73 69
74 20 61 6d 65 74 2c 20 a 63
6f 6e 73 65 63 74 65 74 75 72
20 61 64 69 70 69 73 63 69 6e
67 20 65 6c 69 74 2c 20 a 73
65 64 20 64 6f 20 65 69 75 73
6d 6f 64 20 74 65 6d 70 6f 72
20 69 6e 63 69 64 69 64 75 6e
74 20 75 74 20 6c 61 62 6f 72
65 20 65 74 20 64 6f 6c 6f 72
65 20 6d 61 67 6e 61 20 61 6c
69 71 75 61 2e 20 a 55 74 20
65 6e 69 6d 20 61 64 20 6d 69
6e 69 6d 20 76 65 6e 69 61 6d
2c 20 71 75 69 73 20 6e 6f 73
74 72 75 64 20 65 78 65 72 63
69 74 61 74 69 6f 6e 20 75 6c
6c 61 6d 63 6f 20 6c 61 62 6f
72 69 73 20 6e 69 73 69 20 75
74 20 61 6c 69 71 75 69 70 20
65 78 20 65 61 20 63 6f 6d 6d
6f 64 6f 20 63 6f 6e 73 65 71
75 61 74 2e a 44 75 69 73 20
61 75 74 65 20 69 72 75 72 65
20 64 6f 6c 6f 72 20 69 6e 20
72 65 70 72 65 68 65 6e 64 65
```

Exercise

- Write a program that reads the `lorem.txt` file and creates a `lorem-updated.txt` file. The program puts all the text in uppercase and limits the length of the lines to 20 characters (without cutting the words).

- Note that you must choose the appropriate option to open your stream (r, w, r+, etc.) and you may need to create temporary files.

Questions?

References

[1] Wikipedia contributors. UTF-8. Wikipedia, The Free Encyclopedia. November 8, 2017, 18:48 UTC. Available at: https://en.wikipedia.org/w/index.php?title=UTF-8&oldid=809377359. Accessed November 8, 2017.

[2] Wikipedia contributors. Computer file. Wikipedia, The Free Encyclopedia. October 4, 2017, 13:12 UTC. Available at: https://en.wikipedia.org/w/index.php?title=Computer_file&oldid=803760508. Accessed November 9, 2017.

References

[3] Jaescheke, R., The Dictionary of Standard C. Prentice-Hall. 2001.

[4] Wikibooks contributors. C Programming/File IO. Wikibooks, The Free Textbook Project. October 28, 2017, 23:20 UTC. Available from: https://en.wikibooks.org/w/index.php?title=C_Programming/File_IO&oldid=3320864. Accessed November 15, 2017.

[5] Kelley A., Pohl I., C by Dissection, 4th. ed. Addison Wesley Longman. 2001.