La mémoire

Mais avant, un petit détour...

Les bases de numération

On reviendra à la mémoire un peu plus tard 😉

Numération égyptienne (~ -3000)

https://en.wikipedia.org/wiki/Egyptian_numerals

Value	1	10	100	1,000	10,000	100,000	1 million, or many
Hieroglyph							
Description	Single stroke	Cattle hobble	Coil of rope	Water lily (also called lotus)	Bent finger	Tadpole	Heh[3]

Numération babylonienne (~ -1800)

https://fr.wikipedia.org/wiki/Numération_mésopotamienne

Exemples de nombres écrits en numération babylonienne sexagésimale.

Valeur décimale	Écriture babylonienne cunéiforme	Décomposition en base 60
1		1 x 1
17		17 x 1
44		44 x 1
60		60 = 1 x 60 + 0 x 1
85		1 × 60 + 25 x 1
3600		3600 = 1 x 60² + 0 x 60 + 0 x 1
11327		3 × 60² + 8 × 60 + 47 x 1
7000,2525		1 x 60² + 56 x 60 + 40 x 1 + 15/60 + 9/60²

Numération indo-arabe (~ 300)

0
1
2
3
4
5
6
7
8
9

$$133742_{(10)} = 1.10^5 + 3.10^4 + 3.10^3 + 7.10^2 + 4.10 + 2$$

Numération indo-arabe (~ 300)

0
1
2
3
4
5
6
7
8
9

$$133742_{(10)} = 1.10^5 + 3.10^4 + 3.10^3 + 7.10^2 + 4.10 + 2$$

Numération hexadécimale

0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F

$$3CAFE5_{(16)} = 3.16^5 + 12.16^4 + 10.16^3 + 16.16^2 + 15.16 + 5$$

Numération binaire

0

1

$$110011_{(2)} = 1.2^5 + 1.2^4 + 0.2^3 + 0.2^2 + 1.2 + 1$$

La mémoire

Et la représentation des données

Numération binaire

0

1

$$110011_{(2)} = 1.2^5 + 1.2^4 + 0.2^3 + 0.2^2 + 1.2 + 1$$

Nombres négatifs

0

1

$$110011_{(2)} = 1.2^5 + 1.2^4 + 0.2^3 + 0.2^2 + 1.2 + 1$$

$\pm$

Nombres négatifs

$0001_{(2)}$ $1_{(10)}$

$1001_{(2)}$ $-1_{(10)}$

Nombres négatifs

$$
\begin{array}{rr}
 & 0001_{(2)} \quad\quad 1_{(10)} \\
+ & 1001_{(2)} \quad\quad -1_{(10)} \\
\hline
\end{array}
$$

Nombres négatifs

$$
\begin{array}{rr}
 & 0001_{(2)} \quad 1_{(10)} \\
+ & 1001_{(2)} \quad -1_{(10)} \\
\hline
 & 1010_{(2)} \quad -2_{(10)}
\end{array}
$$

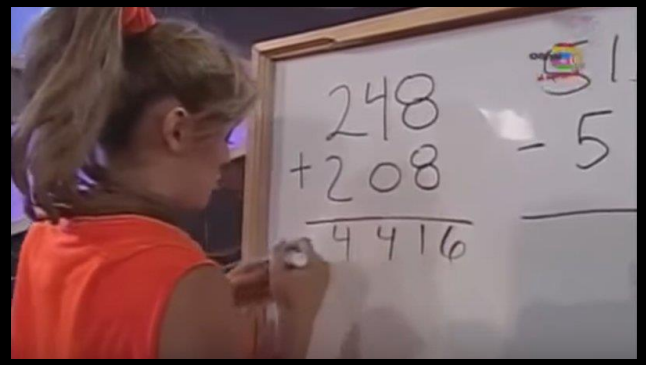
Nombres négatifs

$0001_{(2)}$ $1_{(10)}$

$+\ 1001_{(2)}$ $-1_{(10)}$

$1010_{(2)}$ $-2_{(10)}$

n-n ≠ n+(-n)



Nombres négatifs

$$n-n = n+(-n)$$

$0001_{(2)}$	$1_{(10)}$
$+$ $_{(2)}$	$-1_{(10)}$

$0000_{(2)}$	$0_{(10)}$

Nombres négatifs

Binaire	N	Z
0000	0	0
0001	1	1
1111	15	-1

$$n-n = n+(-n)$$

$$\begin{array}{rr} 0001_{(2)} & 1_{(10)} \\ + \ 1111_{(2)} & -1_{(10)} \\ \hline 0000_{(2)} & 0_{(10)} \end{array}$$

Nombres négatifs

Binaire	N	Z
0000	0	0
0001	1	1
1111	15	-1

$$n-n = n+(-n)$$

$$\begin{array}{rr} 0010_{(2)} & 2_{(10)} \\ + \quad (2) & -2_{(10)} \\ \hline 0000_{(2)} & 0_{(10)} \end{array}$$

Nombres négatifs

Binaire	N	Z
0000	0	0
0001	1	1
0010	2	2
1110	14	-2
1111	15	-1

$$n-n = n+(-n)$$

$$\begin{array}{rr} 0010_{(2)} & 2_{(10)} \\ +\ 1110_{(2)} & -2_{(10)} \\ \hline 0000_{(2)} & 0_{(10)} \end{array}$$

Nombres négatifs

Binaire	N	Z
0000	0	0
0001	1	1
0010	2	2
1110	14	-2
1111	15	-1

$$n-n = n+(-n)$$

$$\begin{array}{rr} 0101_{(2)} & 5_{(10)} \\ + \quad (2) & -5_{(10)} \\ \hline 0000_{(2)} & 0_{(10)} \end{array}$$

Nombres négatifs

Binaire	N	Z
0000	0	0
0001	1	1
0010	2	2
0101	5	5
1011	11	-5
1110	14	-2
1111	15	-1

$$n-n = n+(-n)$$

$$\begin{array}{rr} 0101_{(2)} & 5_{(10)} \\ +\ 1011_{(2)} & -5_{(10)} \\ \hline 0000_{(2)} & 0_{(10)} \end{array}$$

Nombres négatifs

Binaire	N	Z
0000	0	0
0001	1	1
0010	2	2
0101	5	5
1011	11	-5
1110	14	-2
1111	15	-1

$$n-n = n+(-n)$$

Complément à 2 : inversion bit à bit +1

Nombres négatifs

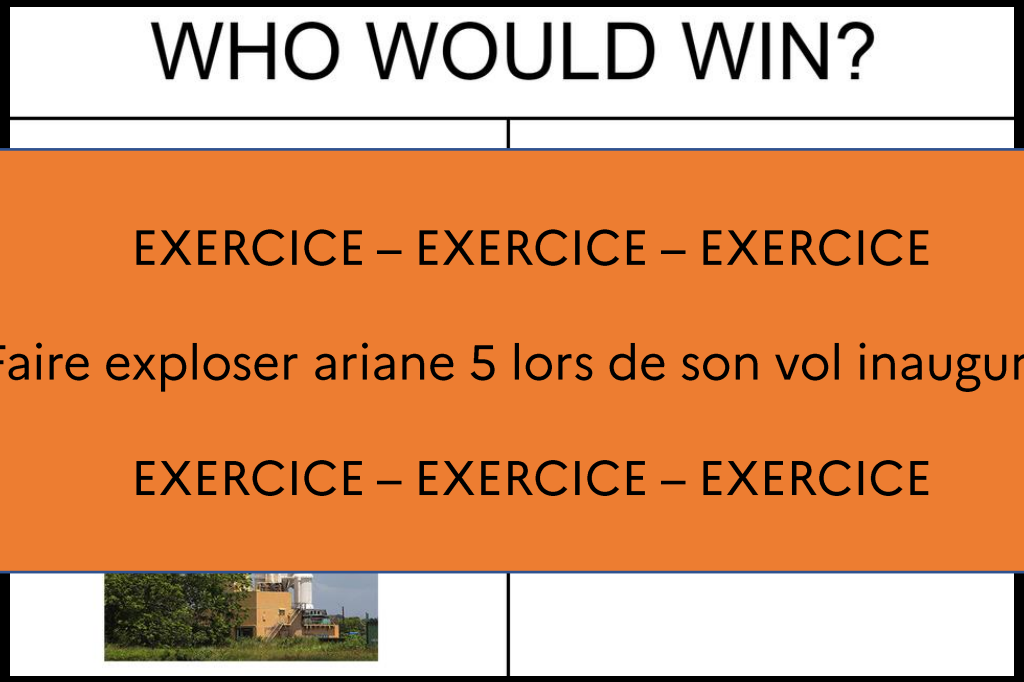
Binaire	N	Z
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	-8
1001	9	-7
1010	10	-6
1011	11	-5
1100	12	-4
1101	13	-3
1110	14	-2
1111	15	-1

Nombres négatifs

Binaire	N	Z
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	-8
1001	9	-7
1010	10	-6
1011	11	-5
1100	12	-4
1101	13	-3
1110	14	-2
1111	15	-1

Nombres négatifs

Binaire	N	Z
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	-8
1001	9	-7
1010	10	-6
1011	11	-5
1100	12	-4
1101	13	-3
1110	14	-2
1111	15	-1



Afficher la donnée (en C)

```
%d      int
%ld     long int
%lld    long long int
%f      float
%lf     double
%Lf     long double
```

Les lettres

American Standard Code for Information Interchange (ASCII) Table

LOW ASCII

000:	013:♪	026:→	039:'	052:4
001:☺	014:♫	027:←	040:(	053:5
002:☻	015:☼	028:∟	041:)	054:6
003:♥	016:►	029:↔	042:*	055:7
004:♦	017:◄	030:▲	043:+	056:8
005:♣	018:↕	031:▼	044:,	057:9
006:♠	019:‼	032:	045:-	058::
007:•	020:¶	033:!	046:.	059:;
008:◘	021:§	034:"	047:/	060:<
009:○	022:▬	035:#	048:0	061:=
010:◙	023:↨	036:$	049:1	062:>
011:♂	024:↑	037:%	050:2	063:?
012:♀	025:↓	038:&	051:3	064:@

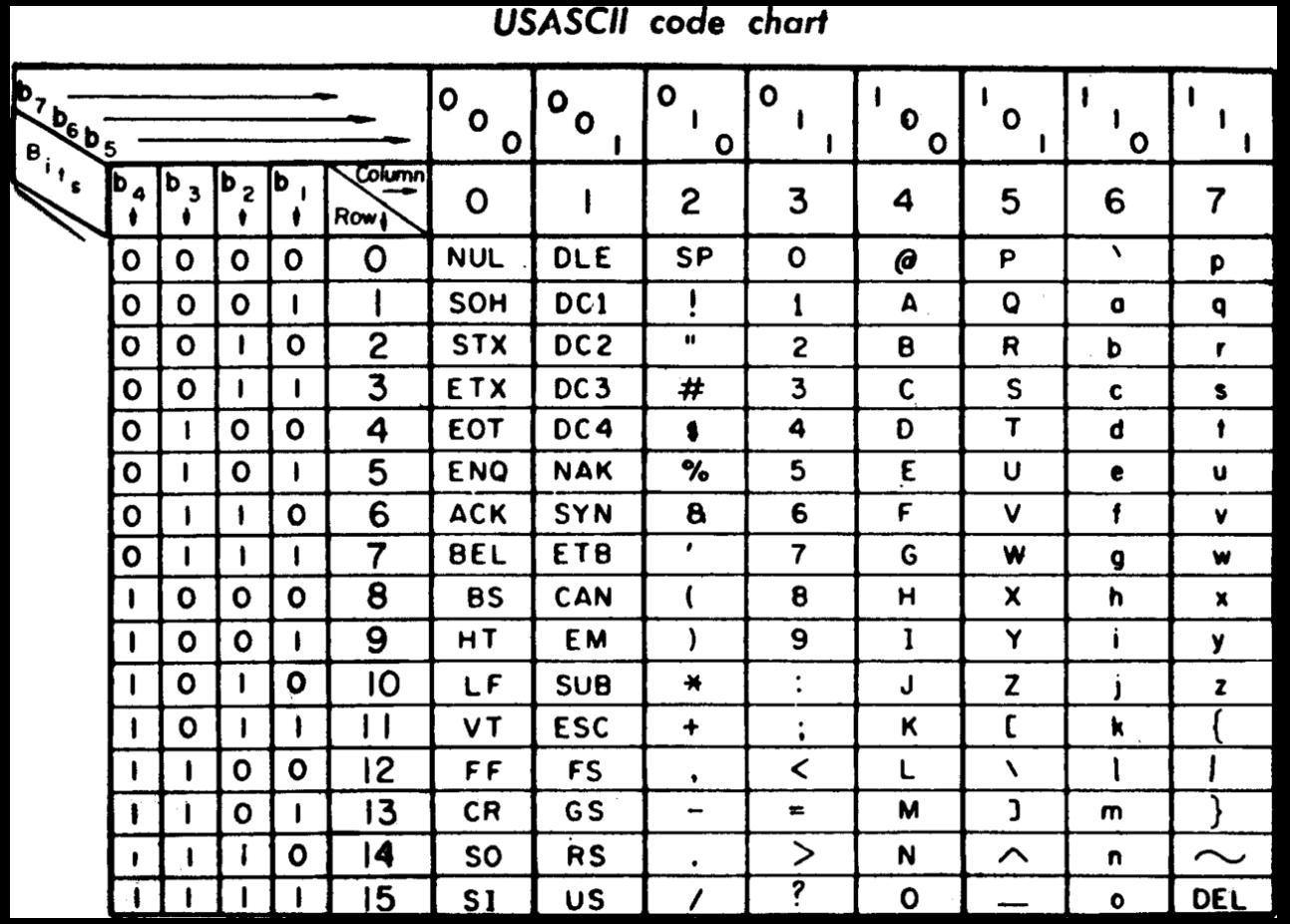
065:A	078:N	091:[	104:h	117:u
066:B	079:O	092:\	105:i	118:v
067:C	080:P	093:]	106:j	119:w
068:D	081:Q	094:^	107:k	120:x
069:E	082:R	095:_	108:l	121:y
070:F	083:S	096:`	109:m	122:z
071:G	084:T	097:a	110:n	123:{
072:H	085:U	098:b	111:o	124:¦
073:I	086:V	099:c	112:p	125:}
074:J	087:W	100:d	113:q	126:~
075:K	088:X	101:e	114:r	127:⌂
076:L	089:Y	102:f	115:s	
077:M	090:Z	103:g	116:t	

HIGH ASCII

128:Ç	141:ì	154:Ü	167:º	180:┤
129:ü	142:Ä	155:¢	168:¿	181:╡
130:é	143:Å	156:£	169:⌐	182:╢
131:â	144:É	157:¥	170:¬	183:╖
132:ä	145:æ	158:₧	171:½	184:╕
133:à	146:Æ	159:ƒ	172:¼	185:╣
134:å	147:ô	160:á	173:¡	186:║
135:ç	148:ö	161:í	174:«	187:╗
136:ê	149:ò	162:ó	175:»	188:╝
137:ë	150:û	163:ú	176:▒	189:╜
138:è	151:ù	164:ñ	177:▒	190:╛
139:ï	152:ÿ	165:Ñ	178:▒	191:┐
140:î	153:Ö	166:ª	179:│	192:└

193:┴	206:╬	219:█	232:Φ	245:J
194:┬	207:╧	220:▄	233:Θ	246:÷
195:├	208:╨	221:▌	234:Ω	247:≈
196:─	209:╤	222:▐	235:δ	248:°
197:┼	210:╥	223:▀	236:∞	249:·
198:╞	211:╙	224:α	237:φ	250:·
199:╟	212:╘	225:ß	238:∈	251:√
200:╚	213:╒	226:Γ	239:∩	252:ⁿ
201:╔	214:╓	227:π	240:≡	253:²
202:╩	215:╫	228:Σ	241:±	254:■
203:╦	216:╪	229:σ	242:≥	255:
204:╠	217:┘	230:μ	243:≤	
205:═	218:┌	231:τ	244:⌠	

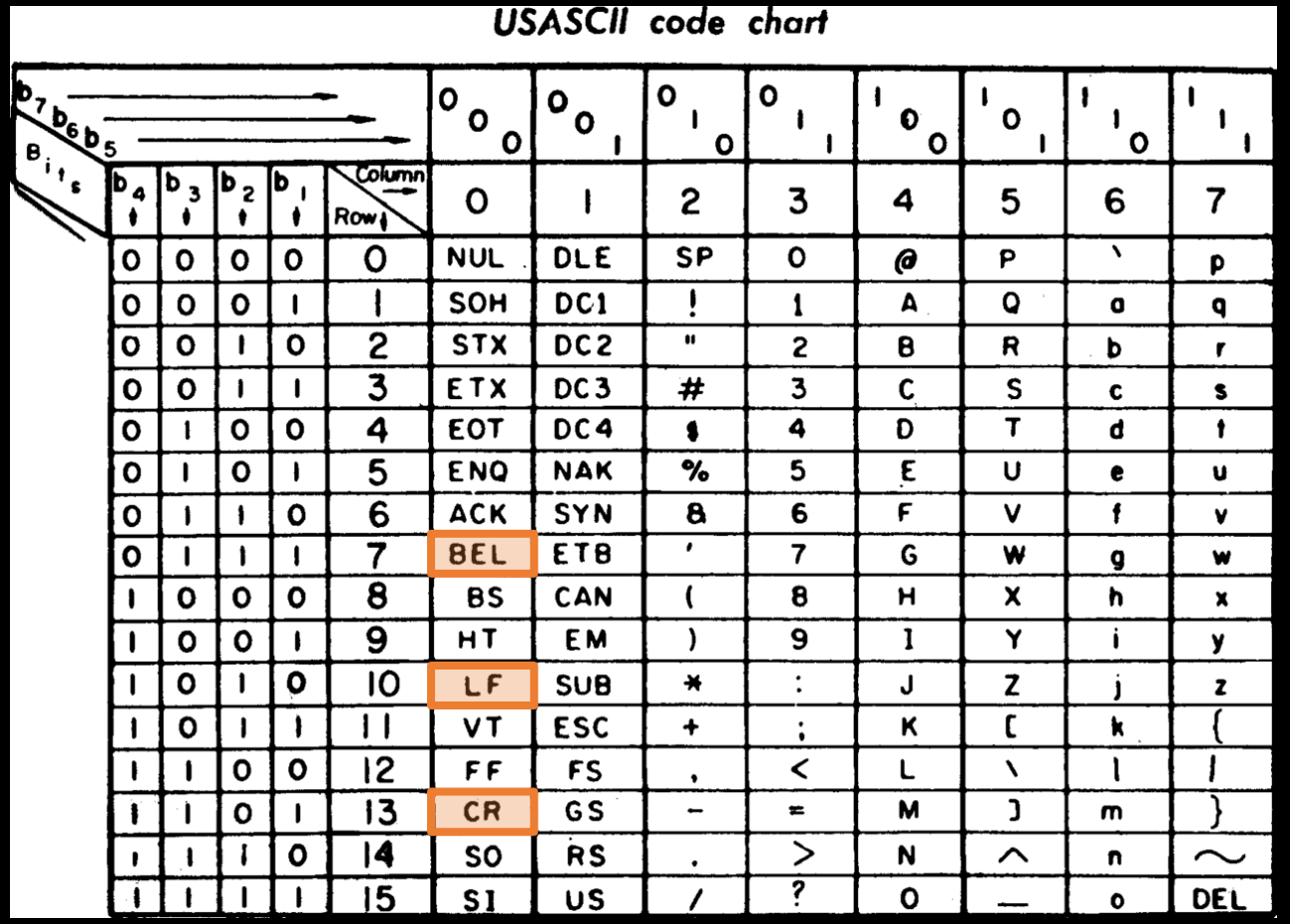
La table ASCII



USASCII code chart

b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
Bits b_4 b_3 b_2 b_1 ↓	Column →							
Row ↓	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

La table ASCII



USASCII code chart

b7 b6 b5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
Bits ↓ b4 b3 b2 b1 / Row ↓ Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

Parenthèse historique

https://www.youtube.com/watch?v=jxkygWI-Wfs

« Teletype Model 19 (and Model 15) Demonstration »

https://www.youtube.com/watch?v=qv5b1Xowxdk

« Altair 8800 - Vidéo #7.1 –
Loading Altair 4K BASIC with a Teletype »

Parenthèse historique

https://www.youtube.com/watch?v=jxkygWI-Wfs

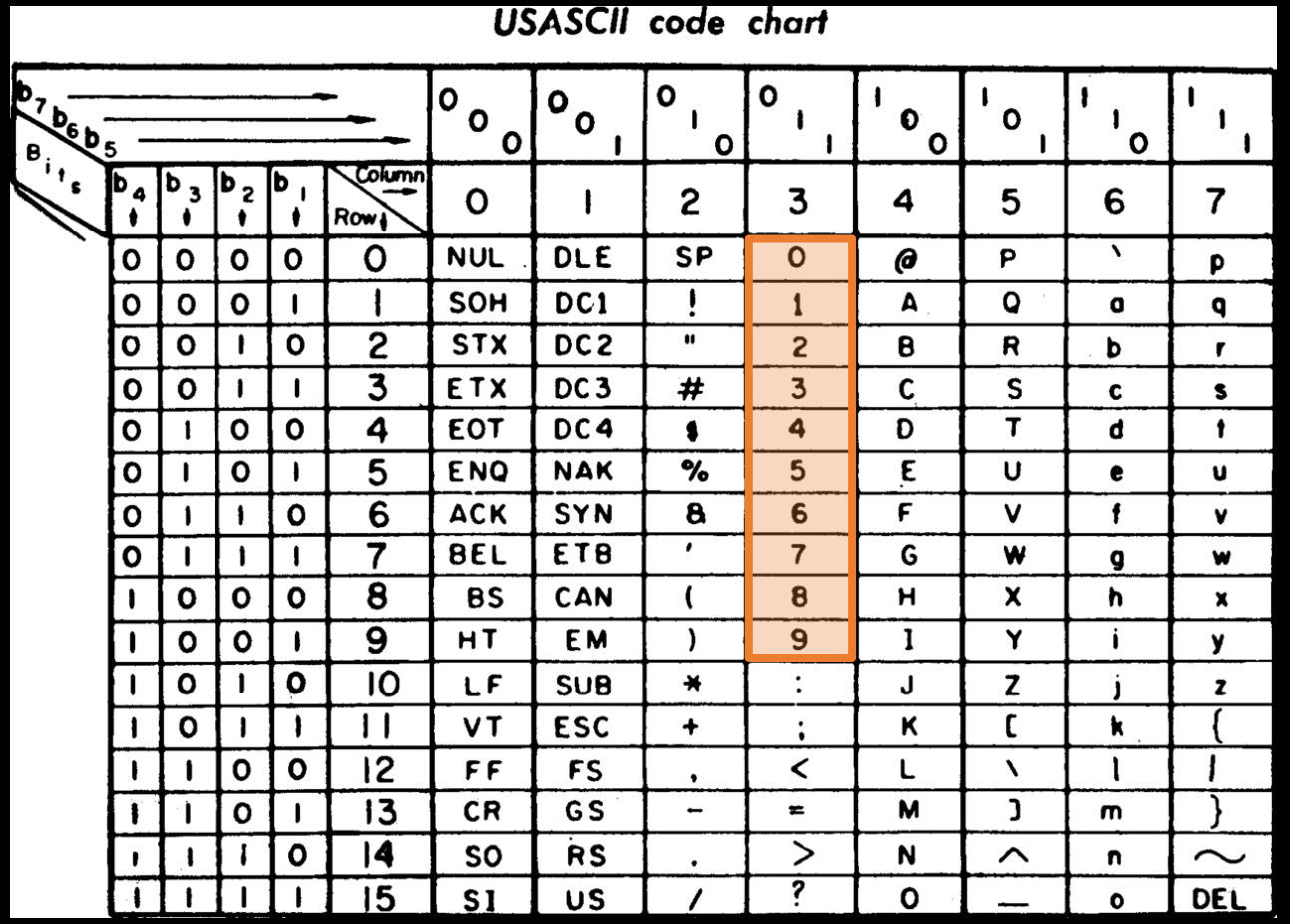
« Teletype Model 19 (and Model 15) Demonstration »

Parenthèse historique

https://www.youtube.com/watch?v=qv5b1Xowxdk

« Altair 8800 - Vidéo #7.1 –
Loading Altair 4K BASIC with a **T**ele**ty**pe »

La table ASCII

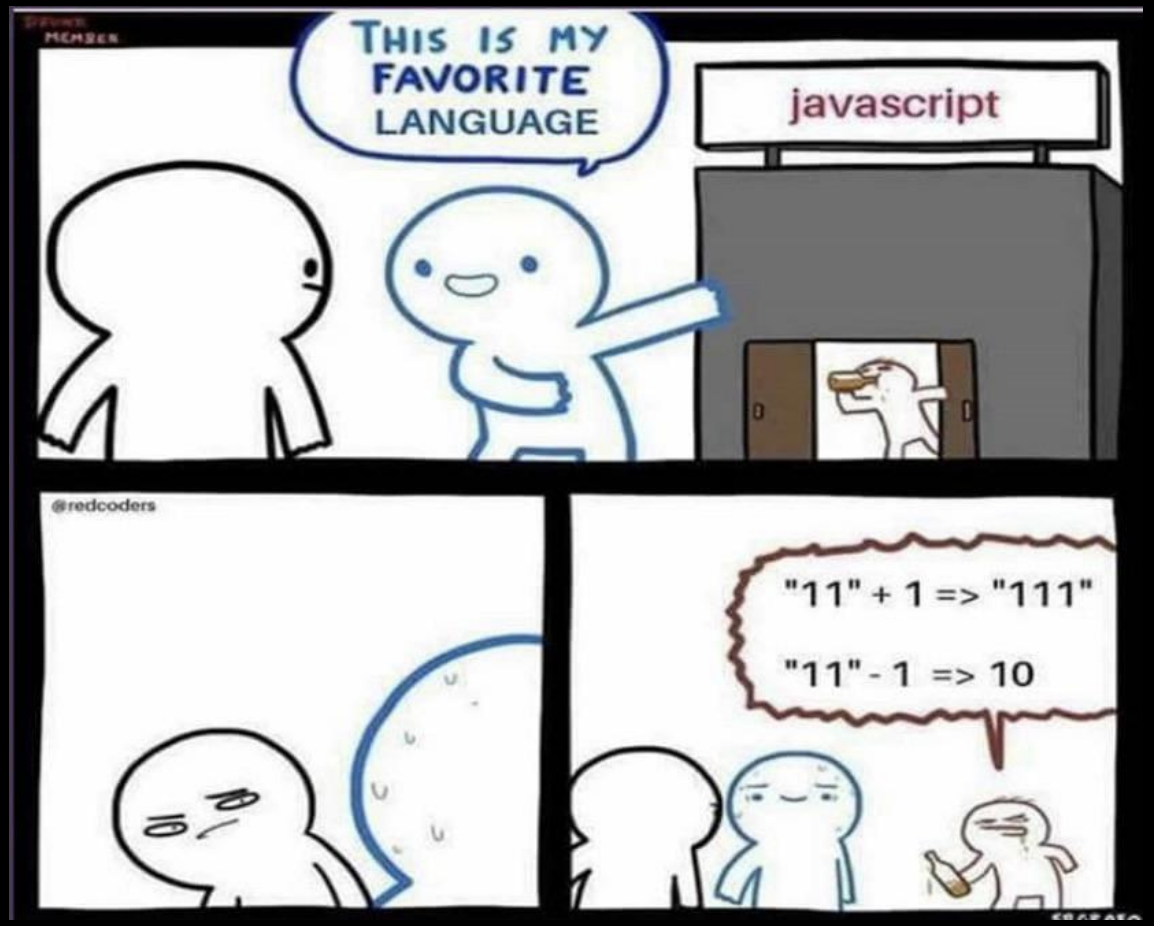


USASCII code chart

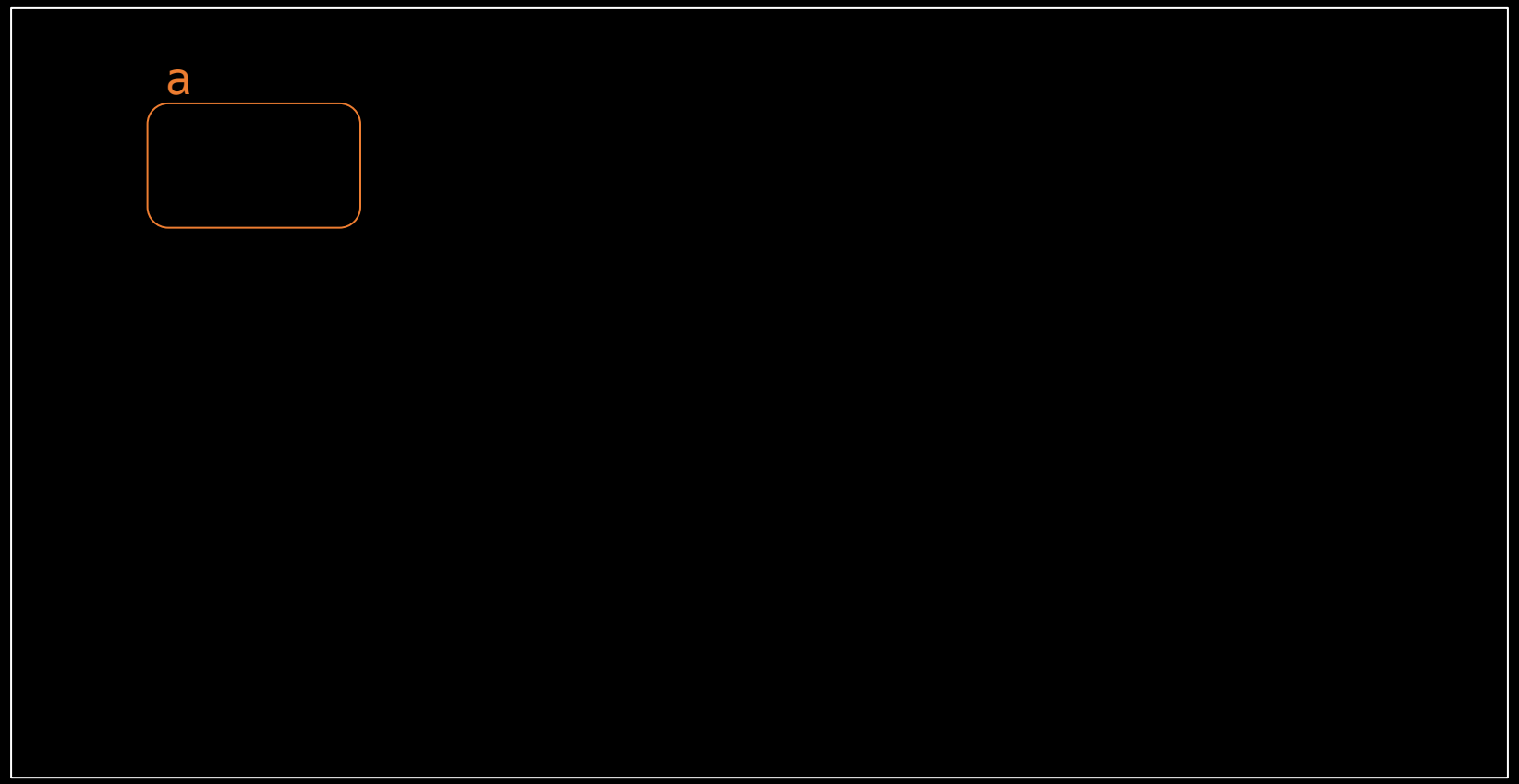
Bits b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 b_3 b_2 b_1 ↓ Row / Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

Faire des calculs avec les données

ou encore les transtyper…

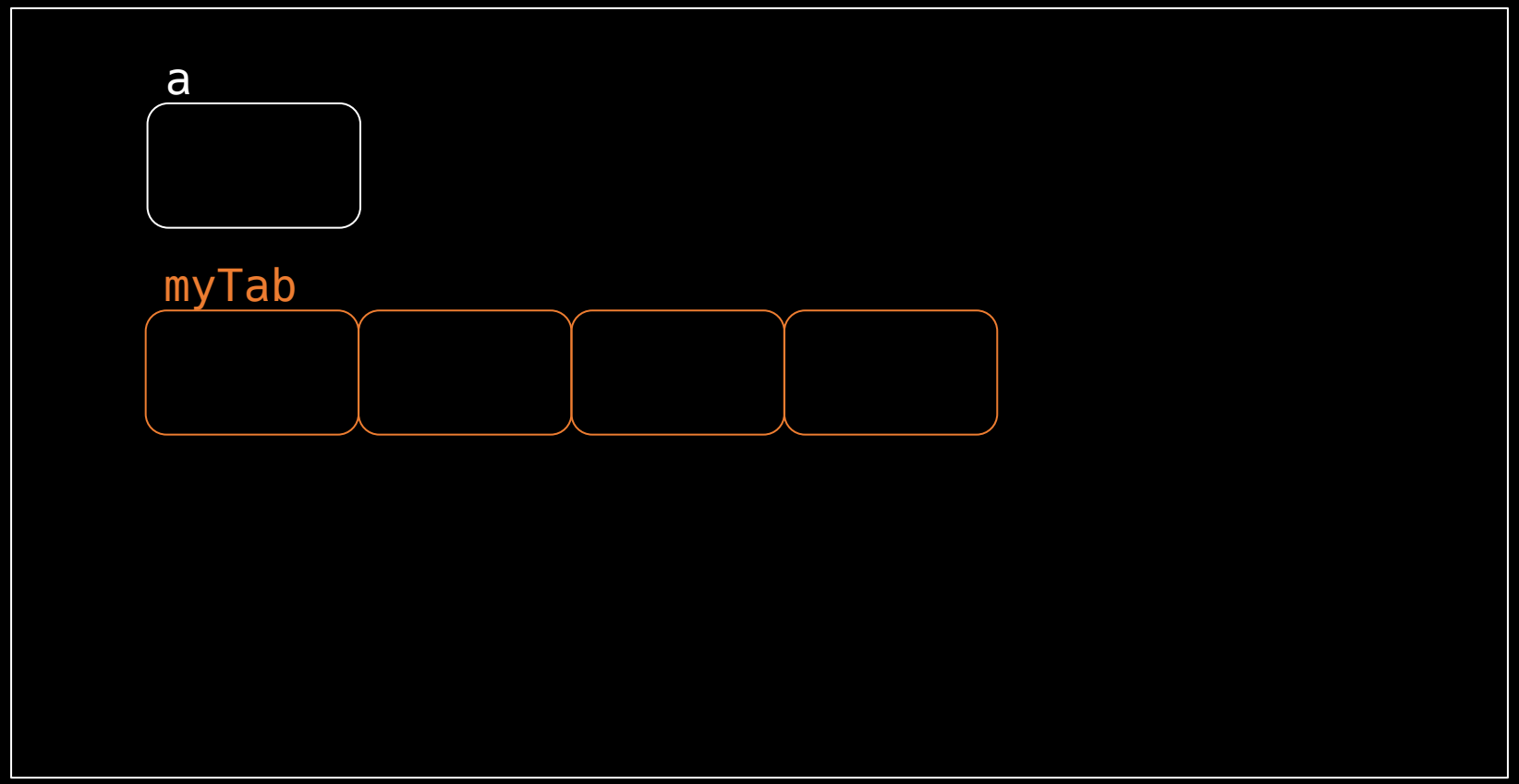


Les tableaux



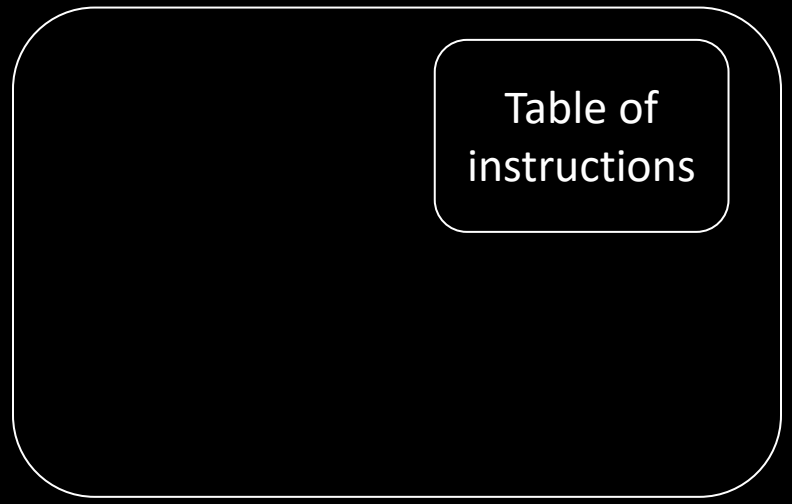

```
int a;
```

Les tableaux

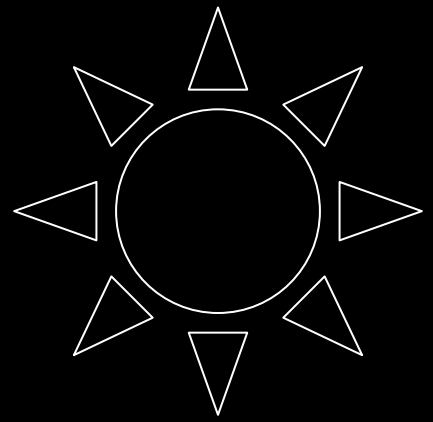



```
int myTab[5];
```

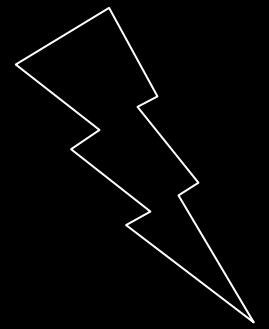
Retrouver sa donnée



Store



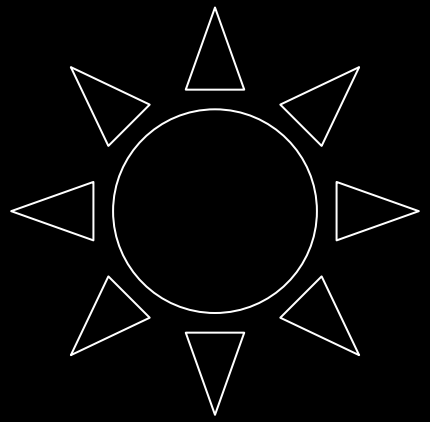
Control

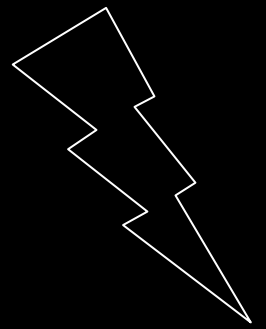


Executive unit

Retrouver sa donnée

01101010100101010111101011100000010101111000000000000101010111111000000000001101010110111101010110101111010101000101010100101010101100101100101010010011010101111111000000010101011110101010000110101011



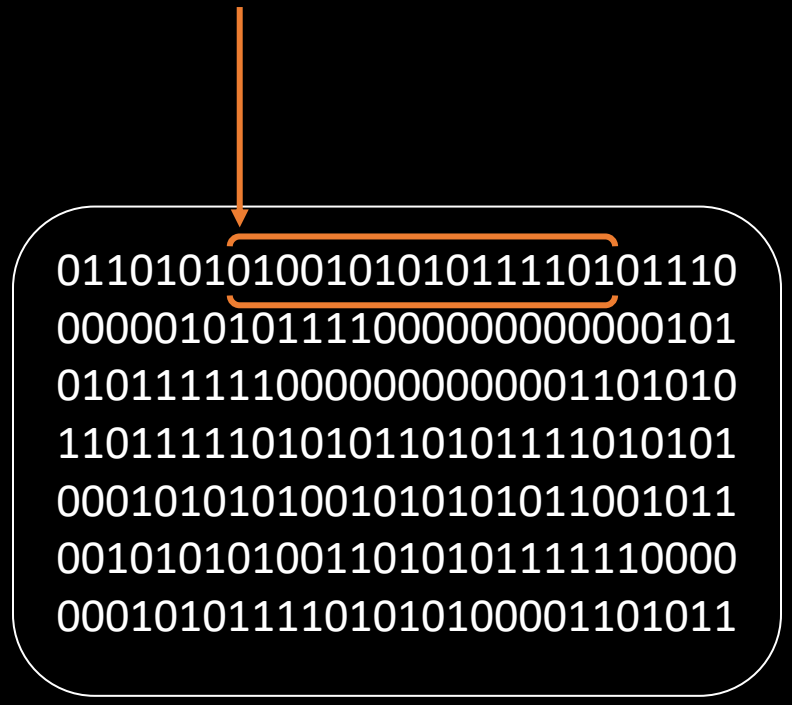


Store

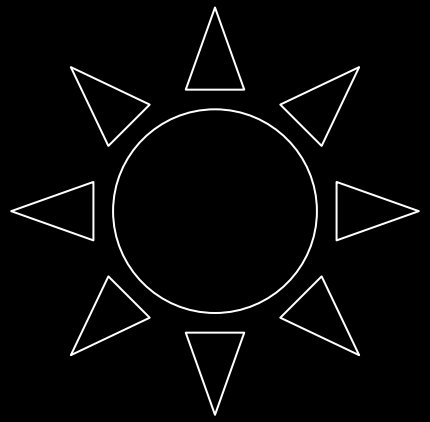
Control

Executive unit

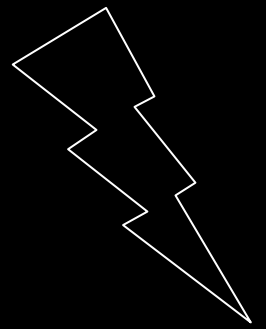
Retrouver sa donnée



Store



Control



Executive unit

i/o

Entrée utilisateur

Les paramètres

```
#include <stdio.h>

int main(int argc, char *argv[]) {

    printf("Program name %s\n", argv[0]);
    printf("Argument count: %d\n", argc);


    return 0;
}
```

Entrée utilisateur

```c
#include <stdio.h>

int main() {
    int varInt = 0;
    printf("Veuillez saisir un entier : ");

    if ( scanf("%d", & varInt) != 1 ) {
        printf("Erreur de saisie !\n");
        return 1; // exit main with error 1
    }

    printf("Vous avez saisi : %d\n", varInt);
}
```

Entrée utilisa

EXERCICE – EXERCICE – EXERCICE

Écrire un programme qui, selon le choix fait par l'utilisateur :
--- donne le code ASCII d'un caractère ---
--- affiche le caractère ASCII correspondant au code donné ---

EXERCICE – EXERCICE – EXERCICE

```
#include <stdio.h>

int main() {
    int varInt = 0;
    printf("Veuillez saisir un entier : ");

    if ( scanf("%d", & varInt) != 1 ) {
        printf("Erreur de saisie !\n");
        return 1; // exit main with error 1
    }

    printf("Vous avez saisi : %d\n", varInt);
}
```

Écrire un fichier

```c
#include <stdio.h>

int main() {
	FILE *filePointer;

	filePointer = fopen("output.txt", "w");

	fprintf(filePointer, "Hello file\n");

	fclose(filePointer);

	return 0;
}
```

Lire un fichier

```
#include <stdio.h>
int main() {
        FILE * file_pointer;

        fp = fopen("input.txt", "r");

        int int1=0;
        int int2=0;
        char str1[10];
        char str2[10];

        fscanf(file_pointer, "%d %s %d %s", &int1, str1, &int2, str2);
        printf("Read int1 |%d|\n", int1);
        printf("Read str1 |%s|\n", str1);
        printf("Read int2 |%d|\n", int2);
        printf("Read str2 |%s|\n", str2);

        fclose(file_pointer);
        return 0;
}
```

Lire un fic

EXERCICE – EXERCICE – EXERCICE

Écrire un programme qui lit un fichier et attend un entier par ligne,
puis écrit un second fichier avec pour contenu
ces mêmes entiers, multipliés par 2.

EXERCICE – EXERCICE – EXERCICE

```c
#include <stdio.h>
int main() {
        FILE * file_pointer;

        fp = fopen("input.txt", "r");

        int int1=0;
        int int2=0;
        char str1[10];
        char str2[10];

        fscanf(file_pointer, "%d %s %d %s", &int1, str1, &int2, str2);
        printf("Read int1 |%s|\n", int1);
        printf("Read str1 |%s|\n", str1);
        printf("Read int2 |%s|\n", int2);
        printf("Read str2 |%s|\n", str2);

        fclose(file_pointer);
        return 0;
}
```