14 à 17 oct = (an)

IUT du limousin – Département Informatique **Année 2024/2025**
TM/LD – R.1.03

Ressource R1.03 – TP 1 - CODAGE DES CARACTERES

A) Principe

Afin d'introduire en machine des informations de type ***« alphanumériques »,*** lettres, chiffres ou tout autre caractère spécial, il faut disposer d'un code binaire qui établisse une **correspondance unique** entre chaque caractère d'un ensemble de caractères et une représentation binaire donnée.
Il faut retenir du codage informatique des caractères que les caractères sont, pour l'ordinateur, des entités numériques qu'il va associer à des chiffres et/ou lettres répertoriés dans une table
Ainsi par exemple, avec :

- **4 bits** soit 16 combinaisons binaires pour représenter les dix caractères chiffres du '0' au '9'
- **6 bits** , soit 64 combinaisons binaires, pour représenter en plus des chiffres, les 26 lettres de l'alphabet.

Il est nécessaire (au minimum) de coder en machine :

➢ **Les caractères éditables :**

- lettres minuscules et majuscules de l'alphabet
- les chiffres (0 à 9)
- les ponctuations : . ; , ? ! etc
- les signes mathématiques : + - * / etc
- les caractères spéciaux : @ & # $ etc

➢ **les caractères non éditables, caractères de service ou caractères de contrôle »**

Ils permettent de réaliser des actions particulière telles que:

- retour à la ligne
- Bip sonore
- Ajouter une tabulation horizontale ou verticale
- Fin de texte
- Fin de communication

Le code ASCII standard ou code ISO à 7 bits (code ISO 646)

Le code **ASCII**, acronyme qui signifie ***« American Standard Code for Information Interchange »***, est l'un des codes les plus utilisé en informatique. .
Le code ASCII standard est universel dans le monde du matériel et du logiciel.
Il a été élaboré en 1963 aux Etats Unis par l'organisme de normalisation américain appelé **ANSI** : ***« American National Standard Institute »*** puis adopté comme norme internationale par l'**ISO** « ***International Standard Organization »*** et par le **CITT** ***« Commission Consultative Internationale des téléphones et télécommunications »*** qui en ont fait le **code ISO à 7 bits** ou **code CITT n°5.**

Le CITT est devenu par après l'UIT-T ***« Union Internationale des Télécommunications - standardisation du secteur des Télécommunications ».***

Le code **ASCII** à 7 bits ou code **ISO 646** définit **128 combinaisons binaires** différentes offrant la codification de 128 caractères (voir la table ASCII : man ASCII sous Linux)

Dans ce tableau, les colonnes et les lignes sont désignés par des numéros en numération binaire et décimale. Ainsi pour coder le caractère **'A'** en **ASCII**, la consultation du tableau montre que **'A'** est à l'intersection **de la colonne de valeur décimale 4** et de **la ligne de valeur décimale 1**.

Le code ASCII de **'A'** est donc :

* $(41)_{16}$ ***en hexadécimal***
* $(1000001)_2$ ***en binaire***
* $(65)_{10}$ ***en décimal***

Caractéristique du code ASCII :

1) Il offre 31 caractères de contrôle se décomposant en quatre sous-ensembles :

- 11 caractères de contrôle de transmission ou **TC** (Transmission Control), de $\mathbf{TC_0}$ à $\mathbf{TC_{10}}$
- 6 caractères de formatage ou **FE** (Format Effector), de $\mathbf{FE_0}$ à $\mathbf{FE_5}$.
- 4 caractères de séparation d'information ou **IS** (Information Separator), de $\mathbf{IS_1}$ à $\mathbf{IS_4}$.
- 4 caractères de commande de périphérique ou **DC** (Device Control), de $\mathbf{DC_1}$ à $\mathbf{DC_4}$.
- 6 autres caractères de commande

Voir le tableau pour la signification de chacun des caractères de contrôle, exemple pour la classe FE :

LF « Line Feed » ou « saut de ligne »

FF « Form Feed » ou « page suivante »

- Le dernier caractère du tableau, DEL de code $(127)_{10}$ (touche suppression du clavier), est aussi un caractère de contrôle.

2) les caractères imprimables, de $(32)_{10}$ à $(126)_{10}$ sont classés dans l'ordre numérique pour les chiffres et dans l'ordre alphabétique pour les lettres de l'alphabet :
   - les chiffres commencent à $(48)_{10}$ et finissent à $(57)_{10}$
   - les lettres majuscules commencent à $(65)_{10}$
   - les lettres minuscules commencent à $(97)_{10}$

Ce classement facilite les opérations de :

- passage d'un caractère au suivant : **'a' + 1 = 'b'**
- de tri alphanumérique en permettant la comparaison entre caractères comme la condition qui détermine si le caractère **'c'** est un caractère minuscule : $c \geq 'a' \; ET \; c \leq 'z'$
- passage des caractères minuscules en majuscules. La différence entre majuscules et minuscules étant de $(32)_{10}$, on a : $'A'+32='a'$

En mémoire ou sur tout support de stockage, on **réserve un octet** pour le code ASCII d'un caractère. Le **8ème bit est souvent inutilisé et donc mis à zéro**. Sinon ce bit sert de bit de ***« contrôle de parité »*** pour la **détection d'erreur** de transmission entre deux dispositifs informatiques.

https://fr.wikibooks.org/wiki/Les_ASCII_de_0_%C3%A0_127/La_table_ASCII

man ASCII

Annexe : Table ASCII

Rappel : pour insérer un caractère absent du clavier dans un texte il suffit de taper son code tout en pressant sur la touche {Alt}

Table ASCII standard (codes de caractères de 0 à 127)

000	(nul)	016	(dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001	(soh)	017	(dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002	(stx)	018	(dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003	(etx)	019	(dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004	(eot)	020	¶(dc4)	036	$	052	4	068	D	084	T	100	d	116	t
005	(enq)	021	§(nak)	037	%	053	5	069	E	085	U	101	e	117	u
006	(ack)	022	(syn)	038	&	054	6	070	F	086	V	102	f	118	v
007	(bel)	023	(etb)	039	'	055	7	071	G	087	W	103	g	119	w
008	(bs)	024	(can)	040	(	056	8	072	H	088	X	104	h	120	x
009	(tab)	025	(em)	041	)	057	9	073	I	089	Y	105	i	121	y
010	(lf)	026	(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011	(vt)	027	(esc)	043	+	059	;	075	K	091	[	107	k	123	{
012	(np)	028	(fs)	044	,	060	<	076	L	092	\	108	l	124	|
013	(cr)	029	(gs)	045	-	061	=	077	M	093	]	109	m	125	}
014	(so)	030	(rs)	046	.	062	>	078	N	094	^	110	n	126	~
015	¤(si)	031	(us)	047	/	063	?	079	O	095	_	111	o	127	

Table ASCII étendue (codes de caractères de 128 à 255)

128	Ç	144	É	160	á	176	_	192	+	208	ð	224	Ó	240	&SHY;
129	ü	145	æ	161	í	177	_	193	-	209	Ð	225	ß	241	±
130	é	146	Æ	162	ó	178	_	194	-	210	Ê	226	Ô	242	_
131	â	147	ô	163	ú	179	¦	195	+	211	Ë	227	Ò	243	¾
132	ä	148	ö	164	ñ	180	¦	196	-	212	È	228	õ	244	¶
133	à	149	ò	165	Ñ	181	Á	197	+	213	ı	229	Õ	245	§
134	å	150	û	166	ª	182	Â	198	ã	214	Í	230	µ	246	÷
135	ç	151	ù	167	º	183	À	199	Ã	215	Î	231	þ	247	¸
136	ê	152	ÿ	168	¿	184	©	200	+	216	Ï	232	Þ	248	°
137	ë	153	Ö	169	®	185	¦	201	+	217	+	233	Ú	249	¨
138	è	154	Ü	170	¬	186	¦	202	-	218	+	234	Û	250	·
139	ï	155	ø	171	½	187	+	203	-	219	_	235	Ù	251	¹
140	î	156	£	172	¼	188	+	204	¦	220	_	236	ý	252	³
141	ì	157	Ø	173	¡	189	¢	205	-	221	¦	237	Ý	253	²
142	Ä	158	×	174	«	190	¥	206	+	222	Ì	238	&hibar;	254	_
143	Å	159	ƒ	175	»	191	+	207	¤	223	_	239	´	255	

B) Manipulations : Connectez-vous sur la machine Linux 164.81.120.23

a) Affichage de caractères

Saisissez et compilez le programme suivant dans un fichier d'extension .c. Exécutez-les ensuite.
Ordre de compilation du fichier toto.c:

cc toto.c –o toto ➔ cette instruction compile le fichier source toto.c et crée l'exécutable toto.

Exécution : ./toto

```
#include <stdio.h>
int main (void)
{
        char car;    /* 8 bits */

        printf("Entrer un caractere:");
        scanf("%c", &car);

        printf (" Affichage mode caractere: %c\n",car);
        printf ("Affichage caractere en decimal : %d\n",car);
        printf (" Affichage caractere en octal: %o\n",car);
        printf (" Affichage caractere en Hexa: %x\n",car);

        printf (" Affichage car+2 en mode caractere puis decimal: %c,     %d\n",car +2, car+2);
}
```

Exécutions

Entrer un caractère: A
Affichage mode caractère: A
Affichage caractère en décimal : 65
Affichage caractère en octal: 101
Affichage caractère en Hexa: 41
Affichage car+2 en mode caractère puis décimal: C , 67
Affichage car+10 en mode caractère puis décimal: K , 75
::
Entrer un caractere:9
Affichage mode caractere: 9
Affichage caractere en decimal : 57
Affichage caractere en octal: 71
Affichage caractere en Hexa: 39
Affichage car+2 en mode caractere puis decimal: ; , 59
Analysez les résultats !

b) Manipulation sur entiers

- Soit les programmes C suivant que vous devez saisir, compiler et exécuter (cf jeux d'essais).

```
#include <stdio.h>
int main (void)
{
        unsigned short int n;          /* Entiers NON signés courts codés sur 16 bits */

        printf("Entrer un entier:");
        scanf("%hu",&n);

        printf (" Affichage entier en décimal et Hexa: %hu,    %hx\n",n,n );
        printf ("Affichage n+1000 en decimal et Hexa : %hu,   %hx\n",n+1000, n+1000);
}
```

Exécutions

```
Entrer un entier: 50
Affichage entier en décimal et Hexa: 50,    32
Affichage n+1000 en decimal et Hexa : 1050,   41a
:::::::::::::::::::::::::::::::::::::::::
Entrer un entier: -10
Affichage entier en décimal et Hexa: 65526,    fff6
Affichage n+1000 en decimal et Hexa : 990,   3de
```

Analysez les résultats !

```
:::::::::::::::::::::::::::::::::::::::::
Entrer un entier: 65000
Affichage entier en décimal et Hexa: 65000,   fde8
Affichage n+1000 en decimal et Hexa : 464,   1d0
```

Analysez les résultats !

:::

```
#include <stdio.h>
int main (void)
{
        short int n;           /* entiers courts codés sur 16 bits */

        printf("Entrer un entier:");
        scanf("%hd",&n);

        printf (" Affichage entier en décimal et Hexa: %hd      %hx\n",n,n) ;
        printf ("Affichage n+1000 en decimal et Hexa : %hd      %hx\n",n+1000, n+1000);
```

```
        printf ("Affichage n-1000 en decimal et Hexa : %hd      %hx\n",n-1000, n-1000);
}
```

Exécutions

```
Entrer un entier: 50
Affichage entier en décimal et Hexa: 50     32
Affichage n+1000 en decimal et Hexa : 1050     41a
Affichage n-1000 en decimal et Hexa : -950     fc4a
::::::::::::::::::::::::::::::::::::::::::::::
Entrer un entier: -10
Affichage entier en décimal et Hexa: -10     fff6
Affichage n+1000 en décimal et Hexa : 990     3de
Affichage n-1000 en décimal et Hexa : -1010     fc0e

::::::::::::::::::::::::::::::::::::::::
Entrer un entier: 32000
Affichage entier en décimal et Hexa: 32000     7d00
Affichage n+1000 en décimal et Hexa : -32536     80e8
Affichage n-1000 en décimal et Hexa : 31000     7918
```

Car en binaire il commence par 1 donc de signe —

Analysez les résultats !

```
::::::::::::::::::::::::::::::::::::::::
Entrer un entier: -32000
Affichage entier en décimal et Hexa: -32000     8300
Affichage n+1000 en décimal et Hexa : -31000     86e8
Affichage n-1000 en décimal et Hexa : 32536     7f18
```

Dépassement de capacité dans les négatifs

```
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
#include <stdio.h>
int main (void)
{
        short int n;
        short int n1 ;

        printf("Entrer l'entier n:");
        scanf("%hd",&n);
        printf("Entrer l'entier n1:");
        scanf("%hd",&n1);

        printf (" Affichage entier n en décimal et Hexa: %hd,    %hx\n",n,n) ;
        printf (" Affichage entier n1 en décimal et Hexa: %hd,    %hx\n",n1,n1) ;

        printf ("Affichage n+n1 en decimal et Hexa : %hd      %hx\n",n+n1, n+n1);
```

Entier3.c

printf ("Affichage n-n1 en decimal et Hexa : %hd   %hx\n",n-n1, n-n1);
}

Exécutions
Entrer l'entier n: 50
Entrer l'entier n1: -10
Affichage entier n en décimal et Hexa: 50,   32
Affichage entier n1 en décimal et Hexa: -10,   fff6
Affichage n+n1 en decimal et Hexa : 40   28
Affichage n-n1 en decimal et Hexa : 60   3c

::
Entrer l'entier n: 32000
Entrer l'entier n1: 2000
Affichage entier n en décimal et Hexa: 32000,
Affichage entier n1 en décimal et Hexa: 2000,
Affichage n+n1 en decimal et Hexa : -31536 ← dépassement de capacité avec l'addition 32000 > 32767
Affichage n-n1 en decimal et Hexa : 30000   7530

::
Entrer l'entier n: **25000**
Entrer l'entier n1: **8000**
Affichage entier n en décimal et Hexa: 25000,   61a8
Affichage entier n1 en décimal et Hexa: 8000,   1f40
Affichage n+n1 en decimal et Hexa : -32536   80e8 ← pareil qu'au dessus
Affichage n-n1 en decimal et Hexa : 17000   4268
::
Entrer l'entier n: **25000**
Entrer l'entier n1: **-12000**
Affichage entier n en décimal et Hexa: 25000,   61a8
Affichage entier n1 en décimal et Hexa: -12000,   d120
Affichage n+n1 en decimal et Hexa : 13000   32c8
Affichage n-n1 en decimal et Hexa : **-28536**   9088 ← aussi

::

c) Manipulation sur réels

Soit le programme C suivant que vous devez à nouveau saisir, compiler et exécuter ?

```
#include <stdio.h>
int main (void)
{
        float n;
        float n1 ;
```

réel.

```
        printf("Entrer le réel n:");
        scanf("%f",&n);
        printf("Entrer le réel n1:");
        scanf("%f",&n1);

        printf (" Affichage  n en décimal: %e \n",n) ;
        printf (" Affichage n1 en décimal: %e\n",n1) ;

        printf (" Affichage n+n1 en décimal: %e\n",n+n1) ;
}
```

Exécutions

```
Entrer le réel n:50
Entrer le réel n1:1e40
Affichage  n en décimal: 50.000000
Affichage n1 en décimal: inf
Affichage n+n1 en décimal: inf
::::::::::::::::::::::::::::::::::::::::
Entrer le réel n:32000
Entrer le réel n1:4e-48
Affichage  n en décimal: 32000.000000
Affichage n1 en décimal: 0.000000
Affichage n+n1 en décimal: 32000.000000
::::::::::::::::::::::::::::::::::::::::
Entrer le réel n:3e38
Entrer le réel n1:2e38
Affichage  n en décimal: 3.000000 e+38
Affichage n1 en décimal: 2.000000 e+38
Affichage n+n1 en décimal: inf
```

trop grand pour élever (*)

trop petit donc assimilé à zéro car en dessous de $7{,}4 \times 10^{(-45)}$

trop grand (*) aussi

Analyse les résultats ! (*) assimilé à $+\infty$ car au dessus de $(3{,}4 \times 10^{38})$

SSH (. . . . IP . . . . .)

Doe . . . # crée un fichier