CPP4	Programmation C : Utilitaire de conversion de donnée	
Système : CodeBlock	Durée : 3 heures	Travail individuel

G.COLIN

Compétences :

Tester et valider un module logiciel.
Réaliser la conception détaillée d'un module logiciel.

Savoirs :

Développement logiciel (Principe de base – Algorithme – Programmation procédurale)

Contexte et démarche

L'objectif est de créer des fonctions logicielles de conversion de données, qui seront utiles lors de la réalisation du projet. Ces fonctions logicielles seront placées dans le fichier "utiles.cpp" et la déclaration de ces fonctions dans le fichier "utiles.h".
On demande de tester des fonctions données et de valider leur fonctionnement.

Une explication sera donnée au préalable par le prof.

1 – Conversion d'une chaîne de caractères en nombre

1.1 – Ascii hexa vers octet

La première fonction à réaliser convertit une chaîne de 2 caractères représentant un octet en notation hexa, en un nombre sur 8 bits.

La fonction est la suivante :

La fonction ascii_hexa_to_byte reçoit une chaine de 2 caractères représentant un octet en notation hexa
Elle retourne un octet correspondant à ce nombre.

Algorithme	Programme à placer dans utiles.cpp

Déclarer variable "a" sur 8 bits
Déclarer variable "b" sur 8 bits
Mettre dans a le code ascii du 1er caractère - 48
SI a > 9
| Retirer 7 à a
FIN_SI
Mettre dans b la valeur de a décalée de 4 bits vers la gauche
Mettre dans a le code ascii du 2ème caractère - 48
SI a > 9
| Retirer 7 à a
FIN_SI
Ajouter a à b
Retourner b
	`#include <string>`
	
	`unsigned char`
	`ascii_hexa_to_byte(std::string str)`
	`{`
	`  unsigned char a,b;`
	`  a=str[0]-48;`
	`  if (a>9) a=a-7;`
	`  b=a*16;`
	`  a=str[1]-48;`
	`  if (a>9) a=a-7;`
	`  b=b+a;`
	`  return b;`
	`}`

Le fichier d'entête utiles.h contient la déclaration de la fonction:

```
unsigned char ascii_hexa_to_byte(std::string str);
```

Le programme principal de test sera le suivant (à placer dans le fichier main.cpp):

```
#include <iostream>
#include <string>

#include "utiles.h"

using namespace std;

int main()
{
    string st;
    int nombre;
    cout<<"Saisir le nombre a convertir sous forme de caracteres"<<endl;
    cin>>st;
    nombre=ascii_hexa_to_byte(st);
    cout << "Resultat en decimal:"<< nombre << endl;
    cout << "Resultat en hexa:"<< hex << nombre << endl;
    return 0;
}
```

On demande de:

- ⇨ créer un projet (test_utiles) sous CodeBlock contenant les 3 fichiers (main.cpp; utiles.cpp; utiles.h)
- ⇨ tester le programme avec les valeurs suivantes : 12 ; 27; 1C; AE
- ⇨ tester le programme avec les valeurs suivantes : 1a; 5; f; 156; AT
- ⇨ conclure sur le format que doit avoir la chaîne de caractères à saisir
- ⇨ expliquer les résultats obtenus en faisant référence au programme de la fonction ascci_hexa_to_byte et du tableau des codes ascii donné ci-dessous.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

1.2 – Ascii hexa vers entier

On souhaite tester une fonction qui converti une chaîne de caractère représentant un nombre en hexadécimal de 4 chiffres, en un nombre entier correspondant à cette représentation.
En entrée la fonction reçoit une chaîne de 4 caractères et retourne un entier.
Le programme de cette fonction est le suivant :

```
unsigned int ascii_hexa_to_int16(std::string str)
{
    unsigned char a;
    unsigned int i;

    a=str[0]-48;
    if (a>9) a=a-7;
    i=a*4096;

    a=str[1]-48;
    if (a>9) a=a-7;
    i=i+a*256;

    a=str[2]-48;
    if (a>9) a=a-7;
    i=i+a*16;

    a=str[3]-48;
    if (a>9) a=a-7;
    i=i+a;

    return i;
}
```

On demande de:

- ⇨ ajouter le programme de la fonction au fichier utiles.cpp
- ⇨ compléter le fichier utiles.h
- ⇨ modifier le programme principal pour tester cette nouvelle fonction
- ⇨ tester la fonction avec différentes valeurs
- ⇨ préciser le format de la chaîne de caractères à saisir

1.3 – Ascii décimal vers octet

On souhaite tester une fonction qui converti une chaîne de caractère représentant un nombre en décimal de 3 chiffres au maximum, en un octet correspondant à cette représentation.
En entrée la fonction reçoit une chaîne de caractères et retourne un octet.
Le programme de cette fonction est le suivant :

```
unsigned char ascii_dec_to_byte(std::string st)
{
  unsigned char a,b,c;
  if (st.size()==0) return 0;
  else
      if (st.size()==1) {a=st[0]-0x30;return a;}
      else
          if (st.size()==2) {a=st[0]-0x30;b=st[1]-0x30;a=10*a+b;return a;}
          else
              if (st.size()==3) {a=st[0]-0x30;b=st[1]-0x30;c=st[2]-0x30;a=100*a+b*10+c;return a;}
              else return 0;
}
```

On demande de :

- ⇨ ajouter le programme de la fonction au fichier utiles.cpp
- ⇨ compléter le fichier utiles.h
- ⇨ modifier le programme principal pour tester cette nouvelle fonction
- ⇨ tester la fonction avec différentes valeurs
- ⇨ préciser le format et les limites de la chaîne de caractères à saisir

1.4 – Ascii décimal vers entier

On demande de

⇨ créer une fonction : unsigned int ascii_dec_to_int(std::string st) qui converti une chaîne de caractère représentant un nombre en décimal de 4 chiffres au maximum (de 0 à 9999) en un entier correspondant à cette représentation.
⇨ compléter le fichier utiles.h
⇨ modifier le programme principal pour tester cette nouvelle fonction
⇨ tester la fonction avec différentes valeurs

2 – Vérification du format d'une chaîne de caractères

2.1 – Chaîne d'un nombre hexa de 2 caractères

La fonction ci-dessous, teste le format d'une chaîne de 2 caractères représentant un nombre en notation hexadécimale. En entrée la fonction reçoit une chaîne de 2 caractères, elle renvoie une chaîne de 2 caractères avec un format correct (avec des majuscules) ou une chaîne vide si le format est incorrect.

```
std::string format_octet_hexa(std::string str)
{
    int i,j;
    char caract_ok[17]="0123456789ABCDEF";
    for (i=0;i<2;i++)
    {
        switch(str[i])
        {
            case 'a':str[i]='A';break;
            case 'b':str[i]='B';break;
            case 'c':str[i]='C';break;
            case 'd':str[i]='D';break;
            case 'e':str[i]='E';break;
            case 'f':str[i]='F';break;
        }
    }
    j=0;
    for (i=0;i<16;i++) {if (str[0]==caract_ok[i]) j++;}
    for (i=0;i<16;i++) {if (str[1]==caract_ok[i]) j++;}
    if (j!=2) {str="";}
    return str;
}
```

On demande de :

⇨ ajouter le programme de la fonction au fichier utiles.cpp
⇨ compléter le fichier utiles.h
⇨ modifier le programme principal pour tester cette nouvelle fonction
⇨ tester la fonction avec différentes valeurs
⇨ préciser le format des caractères acceptés.

2.2 – Chaîne d'un nombre hexa de 4 caractères

On demande de

⇨ créer une fonction : std::string format_int16_hexa(std::string str) qui teste le format d'une chaîne de caractère représentant un nombre hexadécimal de 4 chiffres..
⇨ compléter le fichier utiles.h
⇨ modifier le programme principal pour tester cette nouvelle fonction
⇨ tester la fonction avec différentes valeurs

3 – Conversion d'un nombre en une chaîne de caractères

La fonction ci-dessous convertit un nombre sur 8 bits en 2 caractères représentant ce nombre en hexadécimal.
En entrée la fonction reçoit un octet (ici a). En sortie la fonction retourne les 2 caractères ascii.

Remarque importante : Etant donné que la fonction renvoie 2 données, on passe par des pointeurs (*asc1 et *asc2)

Dans le programme principal, pour tester cette fonction :

- On déclare 1 caractère pour stocker le nombre à convertir : unsigned char a;
- On déclare 2 caractères pour stocker le résultat de la conversion : char b,c;
- On place la donnée saisie au clavier dans a
- On transmet à la fonction les adresses des variables b et c: byte_to_ascii_hexa(a, &b, &c)
- On affiche le contenu des variables b et c, résultat de la conversion.

```
void byte_to_ascii_hexa(unsigned char a, char* asc1, char* asc2)
{
    *asc1=0;
    *asc2=0;
    while (a>15)
    {
        (*asc1)++;
        a=a-16;
    }
    *asc2=a;
    if (*asc2<10) *asc2=*asc2+48; else *asc2=*asc2+55;
    if (*asc1<10) *asc1=*asc1+48; else *asc1=*asc1+55;
}
```

On demande de :

- ⇨ ajouter le programme de la fonction au fichier utiles.cpp
- ⇨ compléter le fichier utiles.h
- ⇨ modifier le programme principal pour tester cette nouvelle fonction
- ⇨ tester la fonction avec différentes valeurs
- ⇨ préciser les limites du nombre accepté.
- ⇨ justifier les 2 dernières lignes du programme (if …) de la fonction.