Cours 9

1. WAZO et arguments du “main”

A. *Le fonctionnement de WAZO*

Programme : page 1

TEXTE.TXT (TAILLE = 51)

X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	eof		

Nombre de WAZO : 5 et positions successives : 22, 42, 8, 17, 2

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
X	X	W	A	Z	O	X	X	W	A	Z	O	X	X	X	X	X	W
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
A	Z	O	X	W	A	Z	O	X	X	X	X	X	X	X	X	X	X
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50			
X	X	X	X	X	X	W	A	Z	O	X	X	X	X	X	eof		

B. *WAZOO : arguments du main*

Programme pages 5-6
Main (int N, char * arg[])
Commande : WAZOO toto.cpp
N = 2

Arg[0]	« WAZOO »
Arg[1]	« toto.cpp »

C. *Lire des arguments entiers : DIT*

Programme page 5.
Commande : DIT 5 Bonjour
N = 3

Arg[0]	« DIT »
Arg[1]	« 5 »
Arg[2]	« Bonjour »

Recuperation d’un nombre inscrit dans une chaine de caracteres :

```
sscanf(chaine, conversion, variables)
```

Retourne (comme scanf) le nombre de conversions0 effectuees avec succes.

```
sscanf(«af 34», «%d%d», &a, &b)     retourne 0
sscanf(«432 fr», «%d%d», &a, &b)    retourne 1 et «a» vaut 432
sscanf(«432 1234», «%d%d», &a, &b) retourne 2 et
                                    «a» vaut 432 et «b» vaut 1234
```

2. Champs statiques et steganographe

A. Champs statiques

(1) Introduction

```
Class marteau{
   Private :
      Static long coupsG;
      Long coupsI; // coups individuel
   Public :
      Marteau(void){ coupsI = 0; }
      Void frappe(void) { coupsI++; coupsG++; }
      Static long nbG(void) { return coupsG; }
      Long nbI(void) { return coupsI; }
};

long marteau::coupsG = 0;

void main(void){
    cout << marteau::nbG(); // valide
    cout << marteau::nbI(); // NON valide

    marteau x, y, z, w;

    x.frappe(); x.frappe();
    z.frappe();
    w.frappe(); w.frappe(); w.frappe();

    cout << marteau::nbG(); // valide, affiche 6
    cout << x::nbG();       // valide, affiche 6
    cout << y::nbG();       // valide, affiche 6
    cout << z::nbG();       // valide, affiche 6
    cout << w::nbG();       // valide, affiche 6


    cout << marteau::nbI(); // NON valide
    cout << x::nbI();       // valide, affiche 2
    cout << y::nbI();       // valide, affiche 0
    cout << z::nbI();       // valide, affiche 1
    cout << w::nbI();       // valide, affiche 3
}
```

(2) Ts_stati.cpp

Programme page 7

Le champ statique « nb_ins » permet de savoir combien d'instances de la classe sont actuellement « vivantes »…

- Notez la presence du constructeur de copie.
- Notez la vie du tableau t1.

B. Steganographe

(1) Principe de base (1 octet remplace 1 octet)

MSG.TXT (TAILLE = 17)

C	H	A	R	G	E	Z		V	O	S		A	R	M	E	S	eof

TEXTE.TXT (TAILLE = 51) decalage = 3

C	X	X	H	X	X	A	X	X	R	X	X	G	X	X	E	X	X
Z	X	X		X	X	V	X	X	O	X	X	S	X	X		X	X
A	X	X	R	X	X	M	X	X	E	X	X	S	X	X	eof		

(2) Amelioration (les 8 bits de chaque octet sont caches dans 8 octets)

Exemple d'un camouflage:

```
if (carSrc & 1) /* le car à cacher se termine par un 1 */
  carDest |= 1;
else            /* le car à cacher se termine par un 0 */
  carDest &= ~1;
```

Etape 1 (le bit à cacher est à zéro)

0	1	0	1	1	0	1	0

&	1	1	0	1	0	0	1	1	carDest
	1	1	1	1	1	1	1	0	~1
	1	1	0	1	0	0	1	0	carDest

Etape 2 (le bit à cacher, après le shift, est à un)

0	0	1	0	1	1	0	1

|	0	1	0	1	0	1	1	0	carDest
	0	0	0	0	0	0	0	1	1
	0	1	0	1	0	1	1	1	carDest

Etape 3 (le bit à cacher est à zéro)

0	0	0	1	0	1	1	0

&	0	1	0	1	0	1	1	0	carDest
	1	1	1	1	1	1	1	0	~1
	0	1	0	1	0	1	1	0	carDest

Etape 4 (le bit à cacher est à un)

0	0	0	0	1	0	1	1

|	0	1	0	1	0	1	1	1	carDest
	0	0	0	0	0	0	0	1	1
	0	1	0	1	0	1	1	1	carDest

Etc…

Exemple d'une récupération

```
if (carSrc & 1) /* le bit à récupérer est un 1 */
  carDest = carDest | (1u << i);
else            /* le car à récupérer est un 0 */
  carDest = carDest & ~(1u << i);
```

On débute avec un carDest = 0

0	0	0	0	0	0	0	0

Voici l'octet contenant le bit de la position 0 à récupérer, celui-ci est à 0 (i vaut 0)

0	0	0	0	0	0	0	0

&	0	0	0	0	0	0	0	0	carDest
	1	1	1	1	1	1	1	0	~(1u << i)
	0	0	0	0	0	0	0	0	carDest

Voici l'octet suivant contenant le 1er bit à récupérer, celui-ci est à 1 (i vaut 1)

0	1	0	1	0	1	1	1

|	0	0	0	0	0	0	0	0	carDest
	0	0	0	0	0	0	1	0	1u << i
	0	0	0	0	0	0	1	0	cardest

Voici l'octet suivant contenant UN bit à récupérer, celui-ci est à 0 (i vaut 2)

0	1	0	1	0	1	1	0

&	0	0	0	0	0	0	1	0	carDest
	1	1	1	1	1	0	1	1	~(1u << i)
	0	0	0	0	0	0	1	0	carDest

Voici l'octet suivant contenant UN bit à récupérer, celui-ci est à 1 (i vaut 3)

0	0	0	0	1	0	1	1

|	0	0	0	0	0	0	1	0	carDest
	0	0	0	0	1	0	0	0	1u << i
	0	0	0	0	1	0	1	0	cardest

Etc…

Programme: ts_stega.cpp page 5… illustre simplement l'utilisation de la classe

Programme stego.cpp page 6… création d'une commande "stego"

```
Commande: stego e msg.txt image.bmp
Commande: stego d _mage.bmp recup.txt
Commande: stego ?
```

Stega.hpp (pages 1-4)

Section publique

- 2 constructeurs, setNomsFichiers, Encoder, Decoder

Section privee:

- afficherBinaire a ete utilisee pour debugger
- ouvrirSource (ouverture en lecture)
- ouvrirDestination (ouverture en lecture/écriture)
- copierFichier (copie du BMP, TXT ou AU avant encodage)
- determinerTypeFichier (nb octets critiques)

Section statique:

- Les messages d'erreurs et le nombre d'octets critiques pour chaque type de fichiers.