Table des matières

A. ANNEXE : Priorité et associativité des opérateurs

Classement du plus fort au plus faible :

	opérateurs	fonction de l'opérateur	associativité
1	→ . **[ ]** **()**	Accès champ d'une structure Indiçage (accès éléments tableau) Appel de fonction	gauche
2	++ -- + - (type) sizeof ~ ! & *	pré et post incrémentation pré et post décrémentation signe plus et signe moins cast taille complément à 1 NON, négation référence (adresse de) indirection (étoile)	droite
3	* / %	multiplication, division, modulo	gauche
4	+ -	addition et soustraction	gauche
5	<< >>	Décalages gauche et droite	gauche
6	< > <= >=	Inférieur et supérieur inférieur ou égal et supérieur ou égal	gauche
7	== !=	Égal et différent	gauche
8	&	ET bit à bit	gauche
9	^	OU bit à bit	gauche
10	|	OU inclusif bit à bit	gauche
11	&&	Conjonction ET	gauche
12	||	Disjonction OU	gauche
13	?:	Conditionnel	gauche

14	= += - = *= /= %=	affectation et affectations combinées	droite
15	,	évaluation séquentielle (virgule des listes)	gauche

B. ANNEXE : quelques outils d'affichage en console windows

L'objectif ici est de fournir les fonctions de bases essentielles pour la réalisation de beaucoup d'exercices du livre. Il suffit de les recopier et de les intégrer à son code.

Ces fonctions s'appuient sur la librairie <windows.h> qui fournit des moyens intéressants et importants pour programmer en console sous windows. Pour ceux préfèrent travailler sous linux ou mac devront trouver l'équivalent qui existe dans ces environnements.

La librairie doit être incluse :

```
#include <windows.h>
```

ensuite, fonction gotoxy() pour déplacer le curseur en écriture :

```
void gotoxy(int x, int y)
{
HANDLE h=GetStdHandle(STD_OUTPUT_HANDLE);
COORD c;
   c.X=x;
   c.Y=y;
   SetConsoleCursorPosition(h,c);
}
```

Fonction textcolor() pour changer la couleur de fond et de face (lettre) en même temps (fond sur les 4 bits hauts, face sur les 4 bits bas) :

```
void textcolor(int color)
{
HANDLE h=GetStdHandle(STD_OUTPUT_HANDLE);
   SetConsoleTextAttribute(h,color);
}
```

Fonction wherex pour savoir quelle est la position horizontale du curseur en écriture :

```
int wherex ()
{
CONSOLE_SCREEN_BUFFER_INFO info;
```

```
    GetConsoleScreenBufferInfo(GetStdHandle(STD_OUTPUT_HANDLE),
                                                        &info);
    return info.dwCursorPosition.X;
}
```

Fonction wherey() pour savoir où est la position verticale du curseur en écriture :

```
int wherey ()
{
CONSOLE_SCREEN_BUFFER_INFO info;

    GetConsoleScreenBufferInfo(GetStdHandle(STD_OUTPUT_HANDLE),
                                                            &info);
    return info.dwCursorPosition.Y;
}
```

Fonction clrscr() pour effacer la fenêtre dans la couleur de son choix :

```
void clrscr (int color)
{
DWORD written;

    FillConsoleOutputAttribute ( GetStdHandle (STD_OUTPUT_HANDLE),
                            color, 2000, (COORD) {0, 0},&written);

    FillConsoleOutputCharacter ( GetStdHandle
                                  (STD_OUTPUT_HANDLE), ' ',
                                  2000, (COORD) {0, 0}, &written);
    gotoxy (0, 0);
}
```

Pour gérer le clavier deux fonctions indispensables sont dans la librairie conio.h

```
#include <conio.h>
```

Ce sont les fonctions :

```
int kbhit() ;
```

Retourne vrai si une touche du clavier est appuyée (n'importe laquelle) et faux sinon.

```
Int getch()
```

Retourne la dernière touche appuyée en valeur ascii.

C. ANNEXE : Librairie C de création en mode console

Une année un étudiant, M. Julien Batonnet, a développé une librairie de création en mode console "ConLib" qui a permis de monter considérablement le niveau des projets de premier semestre. Cette librairie est téléchargeable sur internet à l'adresse http://libconlib.googlecode.com/svn/trunk/

Mais afin que les suivants puissent continuer de profiter de cette expérience nous avons par ailleurs commencé à développer une autre librairie pour l'heure moins sophistiquée mais qui permet d'atteindre un certain nombre d'objectifs, notamment d'avoir la souris et un bon affichage graphique. C'est la librairie "creaco" qui est présentée ici.

Ces deux librairies sont en open-source. Si des personnes se sentent de pouvoir les faire progresser nous serions très heureux de pouvoir en profiter. En effet, face à la 3D nous considérons le mode console comme un mode en soi très intéressant sur le plan graphique. C'est un mode qui n'impressionne pas et qui rejoint le courant de l'Ascii art. Vous n'avez que des lettres et 16 couleurs pour faire quelque chose ! C'est un défi passionnant pour quelqu'un qui aime la programmation. Tout tient dans le scénario, l'imagination et une écriture.

1. Téléchargement :

Cette librairie de fonctions C est disponible sur http://fdrouillon.free.fr et sur le site des éditions ENI.

Elle se compose de deux fichiers : creaco.c et creaco.h.

2. Installation :

Copier ces deux fichiers dans le dossier include du compilateur.

3. Utilisation

Faire un projet console et ajouter

```
#include <creaco.c> // point C
```

S'il y a plusieurs fichiers C ajouter

```
#include <creaco.h> // point H
```

au début des autres fichiers C.

Attention, un certain nombre de librairies sont incluses dans creaco.h :

```
#include <windows.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <time.h>
```

4. Fonctions disponibles

La version de base de cette librairie propose des fonctions pour manipuler le curseur en écriture, la couleur et l'écran console (fenêtre console).

a. Manipuler le curseur en écriture

```
void gotoxy(int x, int y);
```

Déplacer le curseur à une position dans la fenêtre console.
La fonction contrôle que l'on reste dans la fenêtre console

```
int wherex();
```

Retourne la position horizontale du curseur.

```
int wherey();
```
Retourne la position verticale du curseur.

```
void show_cursor(BOOL visible);
```
Montrer ou cacher le curseur selon que le paramètre visible
prend les valeurs 1 (TRUE) ou 0 (FALSE)

```
void size_cursor(unsigned int size);
```
Permet de modifier la taille du curseur (entre 0 et 100).
le paramètre size correspond à un pourcentage de recouvrement
de la cellule où se trouve le curseur.

b. Manipuler la couleur

```
void set_color(int color);  // ou textcolor(int color);
```
Donne ensemble couleur de fond et couleur de face (lettre).

```
void set_bf_color(int backcolor, int forecolor);
```
Donne distinctement une couleur pour le fond et une couleur pour la face.

```
int get_back_color();
```
Permet de récupérer la couleur du fond active.

```
int get_fore_color();
```
Permet de récupérer la couleur de face active.

```
void set_back_color(int color);
```
Active une couleur pour le fond.

```
void set_fore_Color(int color);
```
Active une couleur de face.

c. Quelques manipulations sur la fenêtre console

```
BOOL resize_console(int width, int height);
```
Redimensionner la console (buffer des données et fenêtre affichée) . width reçoit la largeur voulue pour la fenêtre, height reçoit la hauteur voulue pour la fenêtre. Fenêtre affichée et buffer des données ont la même taille (pour avoir un buffer plus grand il faut modifier un peu la fonction). Il y a une taille minimum qui est de 80 caractères en largeur et 25 caractères en hauteur. Il y a également une taille maximum déterminée automatiquement selon la résolution de l'écran et la taille des caractères (sélectionnée directement par l'utilisateur dans les propriétés d'une fenêtre console ouverte). La fonction retourne TRUE (1) en cas de réussite et FALSE (0) en cas d'erreur.

```
BOOL resize_console_max();
```

Cette taille dépend de la taille et de la police de caractères sélectionnées par l'utilisateur de la console. La fonction retourne TRUE (1) en cas de réussite et FALSE (0) en cas d'erreur.

```
int screen_w();
```
Retourne la largeur maximum de la fenêtre compte tenu de la taille du buffer.

```
int screen_h();
```
Retourne la hauteur maximum de la fenêtre compte tenu de la taille du buffer.

```
COORD screen_size();
```
Retourne une structure COORD qui contient largeur et hauteur maximum de la fenêtre.

```
void title_console(char*title);
```
Donne un titre à la fenêtre console. Le titre souhaité est passé sous forme de chaine de caractères.

```
void fill_console(char asciiChar, int color);
```
Remplit la fenêtre avec le caractère et la couleur _fond et face à la fois _spécifiés.

```
void clear_console();
```
Efface la console en noir.

```
void test_screen_console_info(int x,int y,int color)
```
Affiche les informations, à une position et d'une couleur données, relatives à l'écran de la fenêtre console.

d. La gestion des évènements

```
void poll_event() :
```
Récupération des entrées

```
BOOL key_pressed(int keyCode) :
```
Récupération de touches pressées

```
COORD get_mouse_pos() :
```
Récupération des positions h et v de la souris

```
int mouse_x() :
```
Récupération de la position v de la souris

```
int mouse_y() :
```
Récupération de la position h de la souris

```
int mouse_w() :
```
Récupération de la roulette de la souris

```
BOOL clic_pressed(int button) :
```
Récupération des clics de la souris

e. CHBITMAP : buffer de données pour affichage

```
CHBITMAP* create_chb (int width, int height)
```
Allouer une CHBITMAP selon une largeur et une hauteur données en paramètre.

```
CHBITMAP* free_chb(CHBITMAP*b)
```
Libérer la mémoire précédement allouée pour une chbitmap

```
void fill_part_chb(CHBITMAP*b, int x,int y,int w,int h,int
                                asciiChar, int color)
```
Remplit la chbitmap en entier ou une partie avec une lettre et couleurs (lettre, fond) choisies.

```
void fill_chb(CHBITMAP*b, int asciiChar, int color)
```
Remplit le buffer avec une lettre et une couleur

```
void clear_chb(CHBITMAP*b)
```
initialise à 0 une chbitmap

```
void copy_chb(CHBITMAP*src,CHBITMAP*dst,int srcX, int srcY,
                           int dstX,int dstY, int w, int h )
```
Copie une chbitmap dans une autre entièrement ou une partie

```
void draw_chb(CHBITMAP*dst,CHBITMAP*src, int x, int y)
```
copie un buffer dans un autre à partir d'une position donnée

```
void show_to_console(CHBITMAP*b)
```
Permet d'afficher une chbitmap dans la fenêtre console.
La fonction WriteConsoleOutput respecte le rect de la fenêtre console et n'écrit pas en dehors.

```
CHBITMAP* MemScreen()
```
Permet d'obtenir l'adresse de la chbitmap _screen. Cette chbitmap correspond à l'espace de la fenêtre console accessible en écriture. Elle sert de double buffer. Tous les affichages ont lieu d'abord dans _screen et ensuite _screen est affichée dans la fenêtre console.

```
void show_MemScreen()
```
Affiche la chbitmap _screen dans la fenêtre console

```
void clear_MemScreen()
```

```
Efface à 0 la chbitmap _screen sans toucher à la fenêtre
console
```

f. CHTEXT : opérations de text en mode CHBITMAP

```
void draw_char(CHBITMAP*b, int x, int y, int color, int asciiChar)
```

```
Affiche le caractère spécifié en asciiChar avec la couleur
color à la position x, y dans la CHBITMAP b
```

à suivre...