```c
/*
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/
////////////////////////////////////////////////////////////////
//  ACLLib - Advanced C Lab Library
//    Ver 2014-07
//For students' Lab at Zhejiang University
//Created 2008 by Gao Yuan
//Modified 2009 by Cui Liwei
//    2010 by Lan Huidong
//Revised2012 by Li Rui
//  Modified  2014 by Weng Kai for MOOC
////////////////////////////////////////////////////////////////
/*
For Dev C++, these lib files need to be added into linker options.
Be sure to change the leading folders as your installation.
"C:/Program Files/Dev-Cpp/MinGW32/lib/libwinmm.a"
"C:/Program Files/Dev-Cpp/MinGW32/lib/libmsimg32.a"
"C:/Program Files/Dev-Cpp/MinGW32/lib/libkernel32.a"
"C:/Program Files/Dev-Cpp/MinGW32/lib/libuser32.a"
"C:/Program Files/Dev-Cpp/MinGW32/lib/libgdi32.a"
"C:/Program Files/Dev-Cpp/MinGW32/lib/libole32.a"
"C:/Program Files/Dev-Cpp/MinGW32/lib/liboleaut32.a"
"C:/Program Files/Dev-Cpp/MinGW32/lib/libuuid.a"
*/

#ifndef __ACLLIB_H__
#define __ACLLIB_H__
#ifdef _UNICODE
#undef _UNICODE
#endif
#ifdef UNICODE
#undef UNICODE
#endif
#include <Windows.h>
#define BLACKRGB(0, 0, 0)
#define REDRGB(255, 0, 0)
#define GREENRGB(0, 255, 0)
#define BLUERGB(0, 0, 255)
#define CYANRGB(0, 255, 255)
#define MAGENTARGB(255, 0, 255)
#define YELLOWRGB(255, 255, 0)
#define WHITERGB(255, 255, 255)
#define EMPTY0xffffffff
#define DEFAULT-1
typedef enum
{
PEN_STYLE_SOLID,
PEN_STYLE_DASH,/* -------  */
PEN_STYLE_DOT,/* .......  */
PEN_STYLE_DASHDOT,/* _._._._  */
PEN_STYLE_DASHDOTDOT,/* _.._.._  */
PEN_STYLE_NULL
} ACL_Pen_Style;
typedef enum
{
BRUSH_STYLE_SOLID = -1,
BRUSH_STYLE_HORIZONTAL,/* ----- */
BRUSH_STYLE_VERTICAL,/* ||||| */
BRUSH_STYLE_FDIAGONAL,/* \\\\\ */
BRUSH_STYLE_BDIAGONAL,/* ///// */
BRUSH_STYLE_CROSS,/* +++++ */
BRUSH_STYLE_DIAGCROSS,/* xxxxx */
BRUSH_STYLE_NULL
} ACL_Brush_Style;
typedef enum
{
    NO_BUTTON = 0,
    LEFT_BUTTON,
    MIDDLE_BUTTON,
    RIGHT_BUTTON
} ACL_Mouse_Button;
```

```
typedef enum
{
    BUTTON_DOWN,
    BUTTON_DOUBLECLICK,
    BUTTON_UP,
    ROLL_UP,
    ROLL_DOWN,
    MOUSEMOVE
} ACL_Mouse_Event;
typedef enum
{
KEY_DOWN,
KEY_UP
} ACL_Keyboard_Event;
typedef struct
{
HBITMAP hbitmap;
int width;
int height;
} ACL_Image;
//typedef enum
//{
//TM_NO = 0x00,
//TM_COLOR = 0x01,
//TM_ALPHA = 0x02
//} ACL_TransparentMode;
typedef COLORREF ACL_Color;
typedef int ACL_Sound;
typedef void (*KeyboardEventCallback) (int key,int event);
typedef void (*CharEventCallback) (char c);
typedef void (*MouseEventCallback) (int x, int y, int button, int event);
typedef void (*TimerEventCallback) (int timerID);
//
int Setup();
void initWindow(const char title[], int left, int top, int width, int height);
void msgBox(const char title[],const char text[],int flag);
void registerKeyboardEvent(KeyboardEventCallback callback);
void registerCharEvent(CharEventCallback callback);
void registerMouseEvent(MouseEventCallback callback);
void registerTimerEvent(TimerEventCallback callback);
void startTimer(int timerID, int timeinterval);
void cancelTimer(int timerID);
// Sound
void loadSound(const char *fileName,ACL_Sound *pSound);
void playSound(ACL_Sound soundID,int repeat);
void stopSound(ACL_Sound soundID);
// Paint
void beginPaint();
void endPaint();
void clearDevice(void);
int getWidth();
int getHeight();
// Pen
void setPenColor(ACL_Color color);
void setPenWidth(int width);
void setPenStyle(ACL_Pen_Style style);
// Brush
void setBrushColor(ACL_Color color);
void setBrushStyle(ACL_Brush_Style style);
// Text
void setTextColor(ACL_Color color);
void setTextBkColor(ACL_Color color);
void setTextSize(int size);
void setTextFont(const char *pFontName);
void paintText(int x, int y, const char *pStr);
void setCaretSize(int w,int h);
void setCaretPos(int x,int y);
void showCaret();
void hideCaret();
// Pixel
void putPixel(int x, int y, ACL_Color color);
ACL_Color getPixel(int x, int y);
// the Point
int getX(void);
int getY(void);
void moveTo(int x, int y);
void moveRel(int dx,int dy);
// Lines and Curves
void arc(int nLeftRect, int nTopRect, int nRightRect, int nBottomRect, \
int nXStartArc, int nYStartArc, int nXEndArc, int nYEndArc);
void line(int x0, int y0, int x1, int y1);
void lineTo(int nXEnd, int nYEnd);
void lineRel(int dx, int dy);
void polyBezier(const POINT *lppt,int cPoints);
void polyLine(const POINT *lppt, int cPoints);
// Filled Shapes
void chrod(int nLeftRect, int nTopRect, int nRightRect, int nBottomRect, \
int nXRadial1, int nYRadial1, int nXRadial2, int nYRadial2);
void ellipse( int nLeftRect, int nTopRect, int nRightRect, int nBottomRect);
void pie(int nLeftRect, int nTopRect, int nRightRect, int nBottomRect, \
int nXRadial1, int nYRadial1, int nXRadial2, int nYRadial2);
void polygon(const POINT *lpPoints, int nCount);
void rectangle(int nLeftRect, int nTopRect, int nRightRect, int nBottomRect);
void roundrect(int nLeftRect, int nTopRect, int nRightRect, int nBottomRect, \
int nWidth, int nHeight);
// Image
void loadImage(const char *pImageFileName, ACL_Image *pImage);
void freeImage(ACL_Image *pImage);
void putImage(ACL_Image *pImage, int x, int y);
void putImageScale(ACL_Image *pImage,int x,int y,int width,int height);
void putImageTransparent(ACL_Image *pImage,int x,int y,int width,int height,ACL_Color bkColor);
//void putImageEx(ACL_Image *pImage,int dx,int dy,int dw,int dh, \
//int sx,int sy,int sw,int sh);
//void setTransparentMode(ACL_TransparenetMode);
//void setTransparentColor(ACL_Color);
//void setTransparetnAlpha(int alpha);
void initConsole(void);
#endif
```