Low Level Programming Laboratory, Exercise 4

1) Exercise specification

Write Single Document Interface (SDI) application with graphical interface (GUI) using low level programming techniques available in Windows operating system (Windows API). Program should display in its main window some arbitrarily selected geometrical figure using non standard pen and brush objects. Some interaction with displayed graphical objects should be also available using mouse or keyboard. Moreover program should contain menu bar with basic functionality such as: File->Exit and About entries. This task can be realized using any (also high level programming) language that contains all necessary mechanisms to invoke Windows API functions.

2) Windows API

Behind the acronym Windows API (Application Programming Interface) stands the set of routines that allow to refer to any operating system service. Hence Windows API constitute the complete and what must be emphasized the aggregation of low level routines publicized for application programmers in Windows.

3) Single Document Interface GUI application in Windows

Any application equipped with graphical interface in Windows must contain the following components:

- message loop: takes incoming messages from application FIFO queue and sends them to all windows within application,

Example (C++):

```
int WINAPI WinMain (HINSTANCE hThisInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR lpszArgument,
                     int nCmdShow)
{
    MSG messages;            /* Here messages to the application are saved */

    /* Run the message loop. It will run until GetMessage() returns 0 */
    while (GetMessage (&messages, NULL, 0, 0))
    {
        /* Translate virtual-key messages into character messages */
        TranslateMessage(&messages);
        /* Send message to WindowProcedure */
        DispatchMessage(&messages);
    }

    /* The program return-value is 0 - The value that PostQuitMessage() gave */
    return messages.wParam;
}
```

- message handling routine: is connected with the class of windows and performs all the necessary actions required by the specific message. Unhandled messages are send to default handling routine DefWindowProc().

Example (C++):

```
LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message)                  /* handle the messages */
    {
        case WM_DESTROY:
            PostQuitMessage (0);       /* send a WM_QUIT to the message queue */
            break;
        default:                      /* for messages that we don't deal with */
            return DefWindowProc (hwnd, message, wParam, lParam);
    }

    return 0;
}
```

Before SDI application displays its window the following data structure (window class) must be properly filled and registered:

Example (C++):

```
   WNDCLASSEX wincl;        /* Data structure for the window class */

    /* The Window structure */
    wincl.hInstance = hThisInstance;
    wincl.lpszClassName = szClassName;
    wincl.lpfnWndProc = WindowProcedure;      /* This function is called by windows */
    wincl.style = CS_DBLCLKS;                 /* Catch double-clicks */
    wincl.cbSize = sizeof (WNDCLASSEX);

    /* Use default icon and mouse-pointer */
    wincl.hIcon = LoadIcon (NULL, IDI_APPLICATION);
    wincl.hIconSm = LoadIcon (NULL, IDI_APPLICATION);
    wincl.hCursor = LoadCursor (NULL, IDC_ARROW);
    wincl.lpszMenuName = NULL;                 /* No menu */
    wincl.cbClsExtra = 0;                      /* No extra bytes after the window class */
    wincl.cbWndExtra = 0;                      /* structure or the window instance */
    /* Use Windows's default colour as the background of the window */
    wincl.hbrBackground = (HBRUSH) COLOR_BACKGROUND;

    /* Register the window class, and if it fails quit the program */
    if (!RegisterClassEx (&wincl))
        return 0;
```

Now it is possible to create new window and display it using DisplayWindow routine.

Example (C++):

```
    /* The class is registered, let's create the program*/
    hwnd = CreateWindowEx (
           0,                   /* Extended possibilites for variation */
           szClassName,         /* Classname */
           "Code::Blocks Template Windows App",       /* Title Text */
           WS_OVERLAPPEDWINDOW, /* default window */
           CW_USEDEFAULT,       /* Windows decides the position */
           CW_USEDEFAULT,       /* where the window ends up on the screen */
           544,                 /* The programs width */
           375,                 /* and height in pixels */
           HWND_DESKTOP,        /* The window is a child-window to desktop */
           NULL,                /* No menu */
           hThisInstance,       /* Program Instance handler */
           NULL                 /* No Window Creation data */
           );

    /* Make the window visible on the screen */
    ShowWindow (hwnd, nCmdShow);
```

The execution of code drawing in the client area of window should be triggered by WM_PAINT message.

The preparation of code structures responsible for menu events and interaction handling lies fully within the competence of person executing this task and hence only the following hints cab be provided, i.e.:

- mouse messages: WM_LBUTTONDOWN, WM_RBUTTONDOWN, WM_MOUSMOVE,
- keyboard messages: WM_KEYDOWN.

For further information refer to:
http://msdn.microsoft.com/en-us/library/ff657751%28v=VS.85%29.aspx

Possible tools: Dev-Cpp or CodeBlockse IDEs with MinGW free C++ compiler for Windows.