Universitatea Tehnică a Moldovei
Facultatea de Calculatoare, Informatică şi Microelectronică
Filiera Anglofonă „Computer Science”

Aplicații Windows

Raport pentru Laboratorul #1

Windows Applications

Laboratory Work #1 Report

Student:

gr. FAF-111
Nume Prenume

Conducător:

lector asistent
Truhin Alexandr

Chișinău 2013

Introduction

Here you have to describe what you did in this laboratory work, and why it is important. This template has some predefined styles (if you are using MS Office Word) that you can find under Home-Styles box. Mainly your laboratory work should include:

– Introduction;
– Theoretical aspects of laboratory work;
– Implementation description;
– Conclusions;
– Bibliography;
– Appendices.

It is very handy to use MS Office Word references, as you can use references and not to bother about their order. There are few types of references, but basically you’ll use citations and cross-references to tables and figures.

You code solution should be mainly in appendixes. In case that you need to give reference about one or few lines, you can add it directly after the paragraph in which you are giving reference.

How to submit you report you can find on Submission Process page [1]. You will get your mark for laboratory work based on Grading Policy [2]. More details about anything are on course wiki page [3].

1 Description and Analysis of the Domain

Write here you theory about this laboratory work. You also can use subheadings to divide information into more logical blocks. Following blocks are just examples.

1.1 About Windows

Write useful info here.

1.2 Windows Handlers

You’ll meet handles often in win32 API.

1.2.1 Good Windows Handlers

It is about good handlers.

1.2.2 Bad Windows Handlers

It is about bad handlers

2 Implementation

Write here step by step what you did in laboratory work, and why you did certain decisions. If you need to give a reference to a code, add it into Appendixes section, and give a reference to it.

Conclusions

Please write reasonable conclusions. I’m not interested in reading trash. In case that you have no conclusions about what you’ve done, than we may wait until you’ll have something to say.

References

[1] bumbu. WP Submission Process. GitHub wiki. https://github.com/TUM-FAF/WP/wiki/Submission-Process

[2] bumbu. WP Grading Policy. GitHub wiki. https://github.com/TUM-FAF/WP/wiki/Grading-Policy

[3] bumbu. WP Wiki. GitHub wiki. https://github.com/TUM-FAF/WP/wiki

Appendix A

Application Source Code

```
#include <windows.h>

/*  Declare Windows procedure  */
LRESULT CALLBACK WindowProcedure (HWND, UINT, WPARAM, LPARAM);

/*  Make the class name into a global variable  */
char szClassName[ ] = "CodeBlocksWindowsApp";

int WINAPI WinMain (HINSTANCE hThisInstance, HINSTANCE hPrevInstance, LPSTR
lpszArgument, int nCmdShow){
  HWND hwnd;               /* This is the handle for our window */
  MSG messages;            /* Here messages to the application are saved */
  WNDCLASSEX wincl;        /* Data structure for the windowclass */

  /* 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;

  /* The class is registered, let's create the program*/
  hwnd = CreateWindowEx (
    0,                     /* Extended possibilites for variation */
    szClassName,           /* Classname */
    "WP Lab#1 example",    /* 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);

  /* 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;
}


/*  This function is called by the Windows function DispatchMessage()   */

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;
}
```