Министерство науки и высшего образования Российской Федерации

Новосибирский государственный технический университет

Индивидуальная работа по курсу

«Управление ресурсами в вычислительных системах»

Управления ресурсами в ОС Windows

Факультет: прикладной математики и информатики

Группа: ПМ-63

Студент: Шепрут И.И.

Вариант: 17

Преподаватели: Стасышин В. М.

Хайленко Е. А.

Новосибирск

2019

1. Задание

Разработка простейшего Windows-приложения определяющего директорию Windows.

Выбран 3 уровень ИР (вариант 17). Данное приложении должно:

* создавать дочерний поток, в котором будет вызываться функция, реализующая требования варианта
* обладать графическим интерфейсом
* функция, реализующая требования выбранного варианта быть в составе динамической библиотеки

1. Анализ задачи

2.1. Определение директории Windows

Чтобы определить директорию Windows, необходимо вызвать функцию **GetWindowsDirectoryW** и передать ей в качестве параметров строковый буфер и размер этого буфера.

1. Алгоритм программы

Регистрация класса окна. Создание окна приложения с заданным классом. Отображение окна. Запуск цикла обработки сообщений.

Создание дочернего потока, запуск функции в динамической библиотеке.

Загрузка динамической библиотеки. Вызов функции реализующей требования варианта:

- Определение директории Windows.

Вывод полученных результатов при помощи графического интерфейса.

1. Описание разработанных функций

4.1. Функция динамической библиотеки

|  |  |
| --- | --- |
| Функция | Описание |
| void Information(wchar\_t\* Buffer, int BufferSize) | Определение директории Windows. Первый параметр – указатель на буфер строки, куда будет помещен результат, второй параметр – размер буфера. |

4.2. Функции основного приложения

|  |  |
| --- | --- |
| Функция | Описание |
| DWORD WINAPI ThreadFunc(void \*) | Функция, выполняющаяся в отдельном потоке, в ней выполняются следующие действия:  1. Подключение динамической библиотеки  2. Загрузка динамической библиотеки  3. Вызов функции.  4. Освобождение ресурсов.  5. Вывод данных в строку. |
| LRESULT CALLBACK WindowProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam) | Функция обработки сообщений окна. При получении различных сообщений, выполняются различные действия:   * При получении сообщении о создании окна, создается отдельный поток и ожидается его завершение. * При получении сообщения об отрисовке, на экран выводится текст. * При получении сообщения об нажатии клавиши, программа завершается, если нажатая клавиша – это Esc. |
| int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd) | Главная функция программы, в ней выполняются следующие действия:  1. Регистрация класса окна.  2. Создание окна.  3. Отображение окна.  3. Цикл обработки сообщений. |

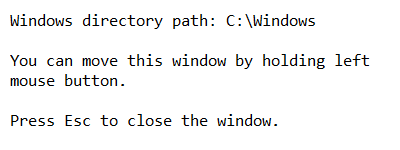
1. Спецификация

Программа находится в папке “rgz”.

Чтобы собрать программу нужно в командной строке разработчика для VS 2017 нужно сначала перейти в папку, а затем ввести скомпилировать командой “compile.bat”.

Для запуска программы нужно запустить файл “main.exe”.

1. Тест



1. Текст программы

Файл info.cpp

|  |
| --- |
| #include <windows.h>  // Функция динамической библиотеки  extern "C" \_declspec(dllexport) void Information(wchar\_t\* Buffer, int BufferSize) {  // Определяем директорию Windows  GetWindowsDirectoryW(Buffer, BufferSize);  } |

Файл main.cpp

|  |
| --- |
| #define UNICODE  #include <windows.h>  //-----------------------------------------------------------------------------  // Размеры окна  int width = 400;  int height = 150;  //-----------------------------------------------------------------------------  // Строка, где будет записано то, что отобразится на экране  wchar\_t Answer[1024];  //-----------------------------------------------------------------------------  // Функция, в которой запускается необходимая функция и выводится информация в строку  DWORD WINAPI ThreadFunc(void \*) {  typedef int (\*ImportFunction)(wchar\_t \*, int);  wchar\_t String[1024];  ImportFunction DLLInfo;  // Считываем dll'ку  HINSTANCE hinstLib = LoadLibrary(TEXT("info.dll"));  // Получаем необходимую функцию  DLLInfo = (ImportFunction) GetProcAddress(hinstLib, "Information");  // Вызываем эту функцию  DLLInfo(String, 1024);  // Освобождаем ресурсы библиотеки  FreeLibrary(hinstLib);  // Вывод информации в строку  wsprintf(Answer, L"Windows directory path: %s\n\nYou can move this window by holding left mouse button.\n\nPress Esc to close the window.", String);  return 0;  }  //-----------------------------------------------------------------------------  // Функция обработки сообщений главного окна  LRESULT CALLBACK WindowProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam) {  HDC hdc;  PAINTSTRUCT ps;  // Создаем шрифт для отображения текста  HFONT hFont = CreateFont(20, 0, 0, 0, FW\_THIN, FALSE, FALSE, FALSE, DEFAULT\_CHARSET, OUT\_OUTLINE\_PRECIS, CLIP\_DEFAULT\_PRECIS, CLEARTYPE\_NATURAL\_QUALITY, VARIABLE\_PITCH, TEXT("Consolas"));  switch (msg) {  // Обрабатываем сообщение, приходящее после создания окна  case WM\_CREATE: {  HANDLE hThread;  DWORD IDThread;  // Создаем поток, где вычислется необходимая информация  hThread = CreateThread(NULL, 0, ThreadFunc, NULL, 0, &IDThread);  WaitForSingleObject(hThread, INFINITE); // Ожидаем закрытия потока  CloseHandle(hThread);  } break;  // Закрытие окна  case WM\_DESTROY: {  PostQuitMessage(0);  } break;  // Обрабатываем сообщение об отрисовке окна  case WM\_PAINT: {  hdc = BeginPaint(hwnd, &ps);  PatBlt(hdc, 0, 0, width, height, WHITENESS); // Закрашиваем всё окно белым цветом  SelectObject(hdc, hFont); // Меняем шрифт  RECT rc;  SetRect(&rc,10,10,width-10, height-10); // Устанавливаем границы для отображения текста  // Отображаем текст в заданном прямоугольнике. При помощи этой функции автоматически расставляются переносы слов и учитывается перенос на новую строку.  DrawText(hdc, Answer, wcslen(Answer), &rc, DT\_LEFT | DT\_EXTERNALLEADING | DT\_WORDBREAK);  EndPaint(hwnd, &ps);  } break;  // Закрываем окно, если была нажата клавиша Esc  case WM\_KEYDOWN: {  if (wParam == VK\_ESCAPE)  PostQuitMessage(0);  } break;  // Обрабатываем это сообщение, чтобы при клике в любое место окна, оно воспринималось как часть заголовка, из чего следует, что окно можно перемммещать, просто зажав левую кнопку мыши на любой части окна.  case WM\_NCHITTEST: {  return HTCAPTION;  } break;  }  // Для всех остальных сообщений вызываем стандартный обработчик с стандартным поведением  return DefWindowProc(hwnd, msg, wParam, lParam);  }  //-----------------------------------------------------------------------------  // Главная функция, которая запускается при запуске программы  int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd) {  // Создаем и регистрируем класс окна  WNDCLASSEX wcl = {0};  wcl.cbSize = sizeof(wcl);  wcl.lpfnWndProc = WindowProc;  wcl.hInstance = hInstance;  wcl.hIcon = LoadIcon(0, IDI\_APPLICATION);  wcl.hCursor = LoadCursor(0, IDC\_ARROW);  wcl.lpszClassName = L"OwnMovedWindow";  if (!RegisterClassEx(&wcl))  return 1;  // Создаем Popup окно - т. е. окно, у которого нет границ, заголовка и кнопок закрытия, открытия. Делаем его положение на переднем плане.  HWND mainHwnd = CreateWindowEx(WS\_EX\_TOPMOST, wcl.lpszClassName, L"Windows path information", WS\_POPUP, 200, 200, width, height, 0, 0, wcl.hInstance, 0);  // Отображаем окно  ShowWindow(mainHwnd, nShowCmd);  UpdateWindow(mainHwnd);  //-------------------------------------------------------------------------  MSG msg;  bool bRet;  // Цикл обработки сообщений созданного окна  while((bRet = GetMessage(&msg, mainHwnd, 0, 0)) != 0) {  if (bRet != -1) {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  }  return 0;  } |

Файл compile.bat

|  |
| --- |
| if exist \*.exe del \*.exe  if exist \*.dll del \*.dll  cl /EHsc main.cpp kernel32.lib user32.lib gdi32.lib advapi32.lib  cl /LD info.cpp kernel32.lib user32.lib gdi32.lib advapi32.lib  del \*.obj \*.lib \*.exp |