Министерство цифрового развития, связи и массовых коммуникаций

Российской Федерации Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное

учреждение высшего образования

Московский технический университет связи и информатики

Кафедра «Системного программирования»

Лабораторная работа №1. Часть 2.

по дисциплине

«Операционные системы»

Выполнила: студентка группы БСТ2104

Первухина А. А.

Москва 2023

Оглавление

[Цель работы 3](#_Toc145626754)

[Лабораторное задание 3](#_Toc145626755)

[Код программы 3](#_Toc145626756)

[Результаты работы 6](#_Toc145626757)

[Вывод 7](#_Toc145626758)

# Цель работы

Получение практических навыков по программированию в Win32 API с использованием аппаратных и системных функций.

# Лабораторное задание

Разработать программное обеспечение приложения, обеспечивающего получение следующей системной информации:

* Имя компьютера, имя пользователя;
* Пути к системным каталогам Windows;
* Версия операционной системы;
* Системные метрики (не менее 2 метрик);
* Системные параметры (не менее 2 параметров);
* Системные цвета (определить цвет для некоторых символьных констант и изменить его на любой другой);
* Функции для работы со временем;
* Дополнительные API-функции: GetClipCursor, GetKeyboardType, GetUserDefaultLCID, ShowCursor.

# Код программы

#include <windows.h>

#include <tchar.h>

#include <stdio.h>

#include <iostream>

#include <wdmguid.h>

#include <string>

#include <VersionHelpers.h>

#define ENV\_VAR\_STRING\_COUNT (sizeof(envVarStrings)/sizeof(TCHAR\*))

#define INFO\_BUFFER\_SIZE 32767

void main()

{

TCHAR infoBuf[INFO\_BUFFER\_SIZE];

DWORD bufCharCount = INFO\_BUFFER\_SIZE;

// Имя компьютера

GetComputerName(infoBuf, &bufCharCount);

\_tprintf(TEXT("\nComputer name: %s"), infoBuf);

// Имя пользователя

GetUserName(infoBuf, &bufCharCount);

\_tprintf(TEXT("\nUser name: %s"), infoBuf);

// Системная директория

GetSystemDirectory(infoBuf, INFO\_BUFFER\_SIZE);

\_tprintf(TEXT("\nSystem Directory: %s"), infoBuf);

// Виндовс директория

GetWindowsDirectory(infoBuf, INFO\_BUFFER\_SIZE);

\_tprintf(TEXT("\nWindows Directory: %s"), infoBuf);

// Директория временных файлов

GetTempPath(INFO\_BUFFER\_SIZE, infoBuf);

\_tprintf(TEXT("\nTemp Path Directory: %s\n"), infoBuf);

//Версия операционной системы

OSVERSIONINFO osversion;

memset(&osversion, 0, sizeof(OSVERSIONINFO));

osversion.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);

if (!IsWindows10OrGreater()) { std::cout << "OS Version: Windows 10/11\n" << std::endl; }

else { std::cout << "OS Version: Not win 10" << std::endl; }

//// Системные метрики

// Количество мониторов

int monitorNumbers = GetSystemMetrics(SM\_CMONITORS);

std::cout << "\tSystem Metrics: \nMonitor numbers: " << monitorNumbers << "\n";

// Ширина экрана основного монитора отображения, в пикселях.

int screenWidth = GetSystemMetrics(SM\_CXSCREEN);

std::cout << "Screen width: " << screenWidth << "\n";

// Количество кнопок на мышке

int mouseButtons = GetSystemMetrics(SM\_CMOUSEBUTTONS);

std::cout << "The number of buttons on the mouse: " << mouseButtons << "\n";

//// Системные параметры

// Скорость мышки

unsigned int aMouseInfo;

SystemParametersInfo(SPI\_GETMOUSESPEED, 0, &aMouseInfo, 0);

std::cout << "\n\tSystem Parametrs: \nMouse speed: " << aMouseInfo;

// Размер курсора

int cursorWidth = 0;

int cursorHeight = 0;

SystemParametersInfo(SPI\_GETMOUSE, 0, &cursorWidth, SPIF\_SENDCHANGE);

SystemParametersInfo(SPI\_GETMOUSE, 0, &cursorHeight, SPIF\_SENDCHANGE);

std::cout << "\nMouse cursor size:" << std::endl;

std::cout << "Width: " << cursorWidth << " pixels" << std::endl;

std::cout << "Height: " << cursorHeight << " pixels" << std::endl;

//// Системные цвета

// GetSysColors - для получения системных цветов

// SetSysColors - для изменения системных цветов

COLORREF colorWindow = GetSysColor(COLOR\_WINDOW);

int redWindow = GetRValue(colorWindow);

int greenWindow = GetGValue(colorWindow);

int blueWindow = GetBValue(colorWindow);

std::cout << "System color window RGB:" << "("<< redWindow << ", " << greenWindow << ", " << blueWindow << ")" << std::endl;

int n[] = { COLOR\_3DDKSHADOW, COLOR\_3DDKSHADOW };

COLORREF color = GetSysColor(n[0]); //Темная тень для элементов трехмерного отображения.

COLORREF color2 = GetSysColor(n[1]); //Рабочий стол.

unsigned long p[] = { color, color2 };

COLORREF newcolor[] = { RGB(150, 75, 0), RGB(0, 0, 255) }; //Коричневый и Синий

SetSysColors(2, n, newcolor);

Sleep(10000);

SetSysColors(2, n, p);

//Функции для работы со временем:

SYSTEMTIME var1, var2;

GetLocalTime(&var1);

GetSystemTime(&var2);

std::cout << "\n\tTime information:\nLocal Time: date " << var1.wDay << "." << var1.wMonth << "." << var1.wYear << " time " << var1.wHour << ":" << var1.wMinute

<< "\nSystem time: "

<< "date " << var2.wDay << "." << var2.wMonth << "." << var2.wYear << " time " << var2.wHour << ":" << var2.wMinute << std::endl;

// Индивидуальные такси

// GetClipCursor, GetKeyboardType, GetUserDefaultLCID, ShowCursor

std::cout << "\n\tIndividual task. Option #13";

// Извлекает экранные координаты прямоугольной области, которой ограничен курсор.

RECT mem;

GetClipCursor(&mem);

std::cout << "\n1.GetClipCursor = " << mem.left << "." << mem.bottom << "." << mem.right << "." << mem.top;

// Тип клавиатуры

int keyboardType = GetKeyboardType(0);

std::cout << "\n2.Keyboard type = " << keyboardType << std::endl;

// Получение идентификатора языковой локали (LCID) пользователя

LCID lcid = GetUserDefaultLCID();

std::cout << "3.ID of the operating system's user-defined language = " << lcid << std::endl;

// Показывает или скрывает курсор

std::cout << "4.Cursor visible = " << ShowCursor(1) << std::endl;

system("PAUSE");

}

# Результаты работы

Рисунок 1 – результат работы программы

# Вывод

Я получила практические навыки по программированию в Win32 API с использованием аппаратных и системных функций.

Ссылка на Github репозиторий: https://github.com/alisadex/Operating-Systems