МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Российский государственный университет им. А.Н. Косыгина   
(Технологии. Дизайн. Искусство)»**

Институт ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ЦИФРОВЫЕ ТРАНСФОРМАЦИИ

**Отчет по лабораторной работе № 7**

**по дисциплине «Операционные системы»**

**Тема: «Команды ОС и Win32 API (C++)»**

Выполнил: Сидоров Д. С., группа ИТС-123

Проверил: к.т.н., доц. Семёнов А. А.

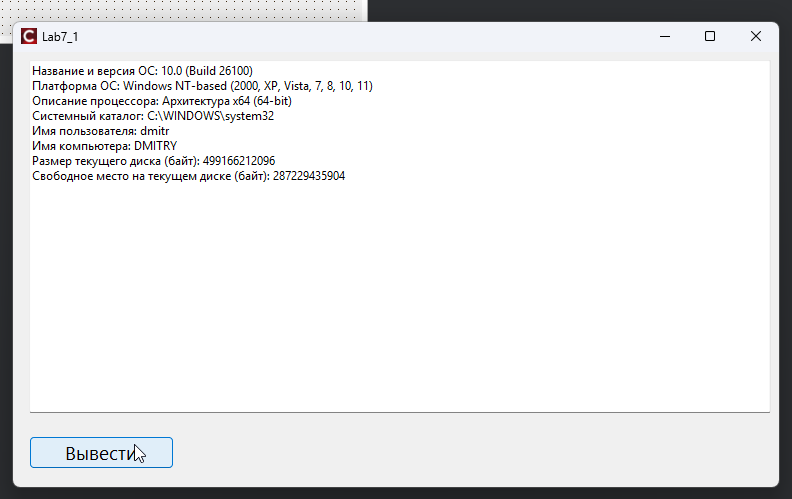
Москва, 2025г

**Команды ОС и Win32 API (C++)**

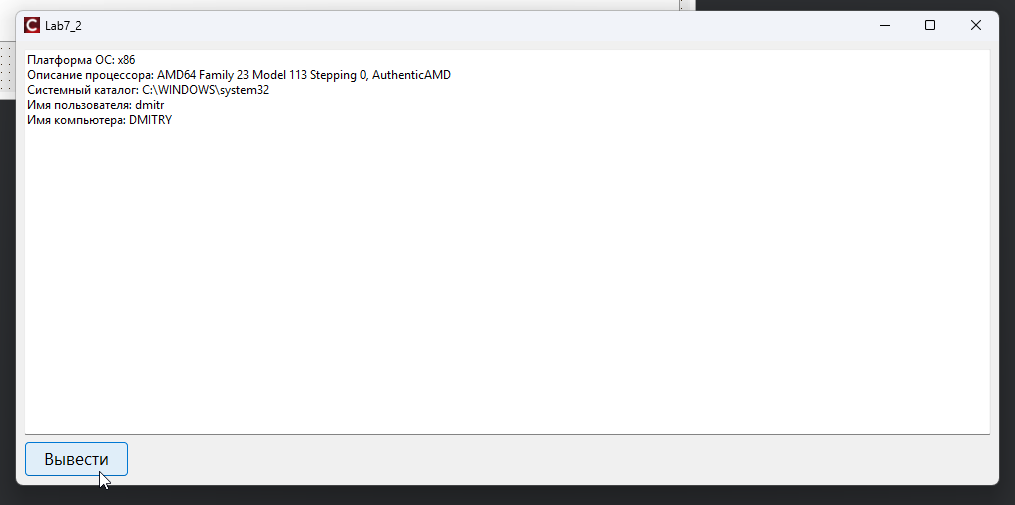
1. Создать приложение на С++, которое позволяет вывести на экран следующую информацию о компьютере: 1) название и версия ОС; 2) платформа ОС; 3) описание процессора; 4) размещение системного каталога; 5) имя пользователя; 6) имя компьютера; 7) размер текущего диска в байтах; 8) размер свободного места на текущем диске в байтах. Сведения, указанные в п. 7, 8, следует получать с помощью функций \_\_int64 DiskSize(Drive) и \_\_int64 DiskFree(Drive) <SysUtils.hpp>, где Drive = 0 – текущий диск, 1 – А, 2 – B, 3 – C, 4 – D и т.д.

2. Написать программу на С++, которая позволяет вывести на экран следующую информацию: 1) платформа ОС; 2) описание процессора; 3) размещение системного каталога; 4) имя пользователя; 5) имя компьютера. Сведения, указанные в п. 1, 2, следует передавать в строковый массив с помощью функции GetEnvironmentVariable(LPCTSTR ("имя\_системной\_переменной"), LPTSTR (имя\_строкового\_массива), размерность\_массива) #include <processenv.h>. Сведения, указанные в п. 3-5, следует передавать в строковый массив с помощью соответствующих (из рассмотренных на лекции) функций Win32 API.

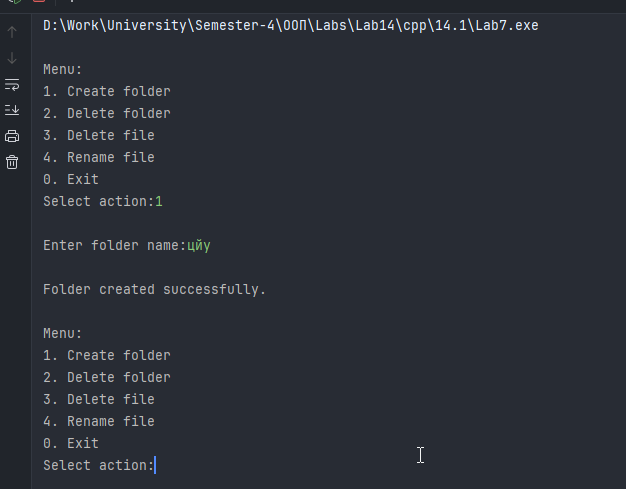
3. Написать программу на С++, позволяющую создавать папку (CreateDir(имя\_массива\_типа\_char)), удалять пустую папку (RemoveDir(имя\_массива\_типа\_ char)), удалять файл (DeleteFile(имя\_массива\_типа\_char)), переименовывать файл (RenameFile(старое\_имя\_массива\_ типа\_char, новое\_имя\_массива\_типа\_char)). Перед выполнением любого действия необходимо проверять наличие файла или папки с помощью функций FileExists(имя\_массива\_типа\_char) и DirectoryExists(имя\_массива\_типа\_char). Все указанные здесь функции требуют подключения заголовочного файла <SysUtils.hpp>. В случае успеха они все возвращают true.



*Рисунок 1. Работающее приложение из п. 1.*



*Рисунок 2. Работающее приложение из п. 2.*



*Рисунок 3. Работающее приложение из п. 3.*

**Листинг кода**

**1.cpp**//---------------------------------------------------------------------------

#include <vcl.h>

#include <SysUtils.hpp>

#include <VersionHelpers.h>

#include <windows.h>

#include <lmcons.h>

#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

//---------------------------------------------------------------------------

typedef LONG(WINAPI \*RtlGetVersionPtr)(PRTL\_OSVERSIONINFOW);

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

RTL\_OSVERSIONINFOW osvi = { 0 };

osvi.dwOSVersionInfoSize = sizeof(osvi);

HMODULE hMod = GetModuleHandle(L"ntdll.dll");

if (hMod) {

RtlGetVersionPtr fPtr = (RtlGetVersionPtr)GetProcAddress(hMod, "RtlGetVersion");

if (fPtr) {

fPtr(&osvi);

}

}

wchar\_t systemDir[MAX\_PATH];

GetSystemDirectoryW(systemDir, MAX\_PATH);

DWORD size = MAX\_COMPUTERNAME\_LENGTH + 1;

wchar\_t computerName[MAX\_COMPUTERNAME\_LENGTH + 1];

GetComputerNameW(computerName, &size);

size = UNLEN + 1;

wchar\_t userName[UNLEN + 1];

GetUserNameW(userName, &size);

SYSTEM\_INFO siSysInfo;

GetNativeSystemInfo(&siSysInfo);

\_\_int64 totalDiskSpace = DiskSize(3);

\_\_int64 freeDiskSpace = DiskFree(3);

String platform;

switch (osvi.dwPlatformId) {

case VER\_PLATFORM\_WIN32\_NT: platform = "Windows NT-based (2000, XP, Vista, 7, 8, 10, 11)"; break;

case VER\_PLATFORM\_WIN32\_WINDOWS: platform = "Windows 95, 98, Me"; break;

case VER\_PLATFORM\_WIN32s: platform = "Windiows 3.x"; break;

default: platform = L"Неизвестная платформа"; break;

}

String cpuArch;

switch (siSysInfo.wProcessorArchitecture) {

case PROCESSOR\_ARCHITECTURE\_AMD64: cpuArch = "x64 (64-bit)"; break;

case PROCESSOR\_ARCHITECTURE\_INTEL: cpuArch = "x86 (32-bit)"; break;

case PROCESSOR\_ARCHITECTURE\_ARM: cpuArch = "ARM"; break;

case PROCESSOR\_ARCHITECTURE\_ARM64: cpuArch = "ARM64"; break;

default: cpuArch = L"Неизвестная архитектура"; break;

}

Memo1->Lines->Clear();

Memo1->Lines->Add(L"Название и версия ОС: " + String(osvi.dwMajorVersion) + "." + String(osvi.dwMinorVersion) + " (Build " + String(osvi.dwBuildNumber) + ")");

Memo1->Lines->Add(L"Платформа ОС: " + platform);

Memo1->Lines->Add(L"Описание процессора: Архитектура " + cpuArch);

Memo1->Lines->Add(L"Системный каталог: " + String(systemDir));

Memo1->Lines->Add(L"Имя пользователя: " + String(userName));

Memo1->Lines->Add(L"Имя компьютера: " + String(computerName));

Memo1->Lines->Add(L"Размер текущего диска (байт): " + String(totalDiskSpace));

Memo1->Lines->Add(L"Свободное место на текущем диске (байт): " + String(freeDiskSpace));

}

//---------------------------------------------------------------------------

**1.h**//---------------------------------------------------------------------------

#ifndef Unit1H

#define Unit1H

//---------------------------------------------------------------------------

#include <System.Classes.hpp>

#include <Vcl.Controls.hpp>

#include <Vcl.StdCtrls.hpp>

#include <Vcl.Forms.hpp>

//---------------------------------------------------------------------------

class TForm1 : public TForm

{

\_\_published: // IDE-managed Components

TMemo \*Memo1;

TButton \*Button1;

void \_\_fastcall Button1Click(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TForm1(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TForm1 \*Form1;

//---------------------------------------------------------------------------

#endif

**2.cpp**//---------------------------------------------------------------------------

#include <vcl.h>

#include <windows.h>

#include <processenv.h>

#include <lmcons.h>

#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

wchar\_t platform[128];

wchar\_t processor[128];

wchar\_t systemDir[MAX\_PATH];

wchar\_t userName[UNLEN + 1];

wchar\_t computerName[MAX\_COMPUTERNAME\_LENGTH + 1];

DWORD size;

GetEnvironmentVariableW(L"PROCESSOR\_ARCHITECTURE", platform, 128);

GetEnvironmentVariableW(L"PROCESSOR\_IDENTIFIER", processor, 128);

GetSystemDirectoryW(systemDir, MAX\_PATH);

size = UNLEN + 1;

GetUserNameW(userName, &size);

size = MAX\_COMPUTERNAME\_LENGTH + 1;

GetComputerNameW(computerName, &size);

Memo1->Lines->Clear();

Memo1->Lines->Add(L"Платформа ОС: " + String(platform));

Memo1->Lines->Add(L"Описание процессора: " + String(processor));

Memo1->Lines->Add(L"Системный каталог: " + String(systemDir));

Memo1->Lines->Add(L"Имя пользователя: " + String(userName));

Memo1->Lines->Add(L"Имя компьютера: " + String(computerName));

}

//---------------------------------------------------------------------------

**2.h**//---------------------------------------------------------------------------

#ifndef Unit1H

#define Unit1H

//---------------------------------------------------------------------------

#include <System.Classes.hpp>

#include <Vcl.Controls.hpp>

#include <Vcl.StdCtrls.hpp>

#include <Vcl.Forms.hpp>

//---------------------------------------------------------------------------

class TForm1 : public TForm

{

\_\_published: // IDE-managed Components

TMemo \*Memo1;

TButton \*Button1;

void \_\_fastcall Button1Click(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TForm1(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TForm1 \*Form1;

//---------------------------------------------------------------------------

#endif

**3.cpp**

#include <iostream>

#include <windows.h>

#include <direct.h>

#include <io.h>

using namespace std;

bool FileExists(const char\*);

bool DirectoryExists(const char\*);

void CreateDir(const char\*);

void RemoveDir(const char\*);

void DeleteFileCustom(const char\*);

void RenameFile(const char\*, const char\*) ;

int main() {

int choice;

char name1[260], name2[260];

do {

cout << "\nMenu:\n";

cout << "1. Create folder\n";

cout << "2. Delete folder\n";

cout << "3. Delete file\n";

cout << "4. Rename file\n";

cout << "0. Exit\n";

cout << "Select action: ";

cin >> choice;

cin.ignore();

switch (choice) {

case 1:

cout << "Enter folder name: ";

cin.getline(name1, 260);

CreateDir(name1);

break;

case 2:

cout << "Enter folder name: ";

cin.getline(name1, 260);

RemoveDir(name1);

break;

case 3:

cout << "Enter file name: ";

cin.getline(name1, 260);

DeleteFileCustom(name1);

break;

case 4:

cout << "Enter current file name: ";

cin.getline(name1, 260);

cout << "Enter new file name: ";

cin.getline(name2, 260);

RenameFile(name1, name2);

break;

case 0:

cout << "Exit.\n";

break;

default:

cout << "Incorrect choice. Try again.\n";

}

} while (choice != 0);

return 0;

}

bool FileExists(const char\* filename) {

return (\_access(filename, 0) != -1 && !(GetFileAttributesA(filename) & FILE\_ATTRIBUTE\_DIRECTORY));

}

bool DirectoryExists(const char\* dirname) {

DWORD attr = GetFileAttributesA(dirname);

return (attr != INVALID\_FILE\_ATTRIBUTES && (attr & FILE\_ATTRIBUTE\_DIRECTORY));

}

void CreateDir(const char\* dirname) {

if (DirectoryExists(dirname)) {

cout << "Folder already exists.\n";

} else {

if (\_mkdir(dirname) == 0) {

cout << "Folder created successfully.\n";

} else {

perror("Error creating folder");

}

}

}

void RemoveDir(const char\* dirname) {

if (DirectoryExists(dirname)) {

if (\_rmdir(dirname) == 0) {

cout << "Folder deleted successfully.\n";

} else {

perror("Error deleting folder (possibly not empty)");

}

} else {

cout << "Folder does not exist.\n";

}

}

void DeleteFileCustom(const char\* filename) {

if (FileExists(filename)) {

if (remove(filename) == 0) {

cout << "File deleted successfully.\n";

} else {

perror("Error deleting file");

}

} else {

cout << "File does not exist.\n";

}

}

void RenameFile(const char\* oldName, const char\* newName) {

if (FileExists(oldName)) {

if (rename(oldName, newName) == 0) {

cout << "File renamed successfully.\n";

} else {

perror("Error renaming file");

}

} else {

cout << "File to rename does not exist.\n";

}

}

**Вывод:** В рамках лабораторной работы были успешно реализованы программы на C++ для взаимодействия с операционной системой Windows с использованием Win32 API. Первая программа позволяет получать и отображать различную системную информацию, включая характеристики ОС, процессора и дисковых ресурсов. Вторая программа демонстрирует работу с системными переменными окружения и получение системной информации через специализированные функции Win32 API. В третьей части была создана консольная утилита для выполнения базовых файловых операций, таких как создание и удаление папок, удаление и переименование файлов, с предварительной проверкой существования объектов для обеспечения безопасной работы. Данная работа наглядно показала возможности программирования на C++ для взаимодействия с операционной системой на низком уровне.