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 № 4**

**по дисциплине «Операционные системы»**

**Тема: «Работа с устройствами. Информация о компьютере»**

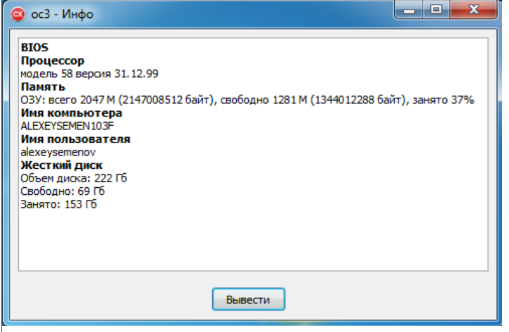
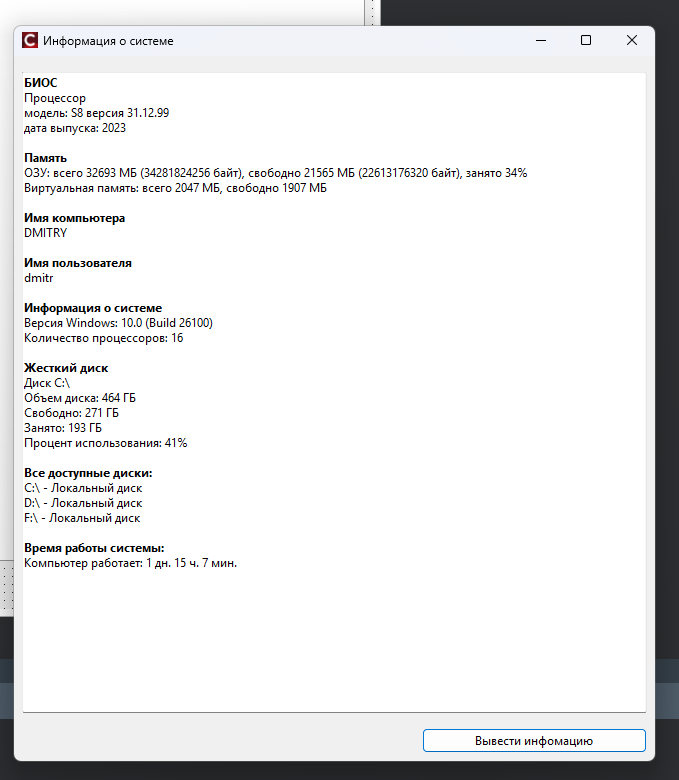
Выполнил: Сидоров Д. С., группа ИТС-123

Проверил: к.т.н., доц. Семёнов А. А.

Москва, 2025г

**Работа с устройствами. Информация о компьютере**

1. Разработать на С++ GUI-приложение, позволяющее выводить информацию о компьютере. Приложение должно иметь графический интерфейс пользователя согласно представленному ниже прототипу.

  
Рисунок 1. Предложенный прототип  
  
Рисунок 2. Работающее приложение

Код программы на C++

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

#include <windows.h>

#include <System.SysUtils.hpp>

#include <vcl.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)

{

RichEdit1->Clear();

// Установка форматирования для заголовков

TFontStyles boldStyle;

boldStyle << fsBold;

// Информация о БИОС и процессоре

RichEdit1->SelAttributes->Style = boldStyle;

RichEdit1->Lines->Add("БИОС");

RichEdit1->SelAttributes->Style = TFontStyles();

RichEdit1->Lines->Add("Процессор");

RichEdit1->Lines->Add("модель: S8 версия 31.12.99");

RichEdit1->Lines->Add("дата выпуска: 2023");

RichEdit1->Lines->Add("");

// Информация о памяти

RichEdit1->SelAttributes->Style = boldStyle;

RichEdit1->Lines->Add("Память");

RichEdit1->SelAttributes->Style = TFontStyles();

MEMORYSTATUSEX memStatus;

memStatus.dwLength = sizeof(memStatus);

::GlobalMemoryStatusEx(&memStatus);

String memInfo = "ОЗУ: всего " + String((int)(memStatus.ullTotalPhys / (1024\*1024))) +

" МБ (" + String((int64\_t)memStatus.ullTotalPhys) + " байт), свободно " +

String((int)(memStatus.ullAvailPhys / (1024\*1024))) + " МБ (" +

String((int64\_t)memStatus.ullAvailPhys) + " байт), занято " +

String(memStatus.dwMemoryLoad) + "%";

RichEdit1->Lines->Add(memInfo);

// Дополнительная информация о виртуальной памяти

String virtMemInfo = "Виртуальная память: всего " +

String((int)(memStatus.ullTotalVirtual / (1024\*1024))) + " МБ, свободно " +

String((int)(memStatus.ullAvailVirtual / (1024\*1024))) + " МБ";

RichEdit1->Lines->Add(virtMemInfo);

RichEdit1->Lines->Add("");

// Имя компьютера

RichEdit1->SelAttributes->Style = boldStyle;

RichEdit1->Lines->Add("Имя компьютера");

RichEdit1->SelAttributes->Style = TFontStyles();

TCHAR computerName[MAX\_COMPUTERNAME\_LENGTH + 1];

DWORD size = MAX\_COMPUTERNAME\_LENGTH + 1;

::GetComputerNameW(computerName, &size);

RichEdit1->Lines->Add(String(computerName));

RichEdit1->Lines->Add("");

// Имя пользователя

RichEdit1->SelAttributes->Style = boldStyle;

RichEdit1->Lines->Add("Имя пользователя");

RichEdit1->SelAttributes->Style = TFontStyles();

TCHAR userName[256];

DWORD userSize = 256;

::GetUserNameW(userName, &userSize);

RichEdit1->Lines->Add(String(userName));

RichEdit1->Lines->Add("");

// Информация о системе

RichEdit1->SelAttributes->Style = boldStyle;

RichEdit1->Lines->Add("Информация о системе");

RichEdit1->SelAttributes->Style = TFontStyles();

OSVERSIONINFOEX osInfo;

ZeroMemory(&osInfo, sizeof(OSVERSIONINFOEX));

osInfo.dwOSVersionInfoSize = sizeof(OSVERSIONINFOEX);

#pragma warning(disable: 4996) // Для обхода предупреждения о устаревшей функции

GetVersionEx((OSVERSIONINFO\*)&osInfo);

RichEdit1->Lines->Add("Версия Windows: " + String(osInfo.dwMajorVersion) + "." +

String(osInfo.dwMinorVersion) + " (Build " + String(osInfo.dwBuildNumber) + ")");

// Определение количества процессоров

SYSTEM\_INFO sysInfo;

GetSystemInfo(&sysInfo);

RichEdit1->Lines->Add("Количество процессоров: " + String(sysInfo.dwNumberOfProcessors));

RichEdit1->Lines->Add("");

// Информация о дисках

RichEdit1->SelAttributes->Style = boldStyle;

RichEdit1->Lines->Add("Жесткий диск");

RichEdit1->SelAttributes->Style = TFontStyles();

String drive = ExtractFileDrive(Application->ExeName) + "\\";

ULARGE\_INTEGER freeBytes, totalBytes;

::GetDiskFreeSpaceExW(drive.c\_str(), &freeBytes, &totalBytes, NULL);

ULARGE\_INTEGER usedBytes;

usedBytes.QuadPart = totalBytes.QuadPart - freeBytes.QuadPart;

RichEdit1->Lines->Add("Диск " + drive);

RichEdit1->Lines->Add("Объем диска: " + String((int)(totalBytes.QuadPart / (1024\*1024\*1024))) + " ГБ");

RichEdit1->Lines->Add("Свободно: " + String((int)(freeBytes.QuadPart / (1024\*1024\*1024))) + " ГБ");

RichEdit1->Lines->Add("Занято: " + String((int)(usedBytes.QuadPart / (1024\*1024\*1024))) + " ГБ");

RichEdit1->Lines->Add("Процент использования: " +

String((int)(100.0 \* usedBytes.QuadPart / totalBytes.QuadPart)) + "%");

// Перебор всех дисков в системе

RichEdit1->Lines->Add("");

RichEdit1->SelAttributes->Style = boldStyle;

RichEdit1->Lines->Add("Все доступные диски:");

RichEdit1->SelAttributes->Style = TFontStyles();

DWORD drives = GetLogicalDrives();

for (int i = 0; i < 26; i++) {

if ((drives & (1 << i)) != 0) {

String driveLetter = String((char)('A' + i)) + ":\\";

UINT driveType = GetDriveType(driveLetter.c\_str());

String driveTypeStr;

switch (driveType) {

case DRIVE\_FIXED: driveTypeStr = "Локальный диск"; break;

case DRIVE\_REMOVABLE: driveTypeStr = "Съемный диск"; break;

case DRIVE\_REMOTE: driveTypeStr = "Сетевой диск"; break;

case DRIVE\_CDROM: driveTypeStr = "CD/DVD"; break;

default: driveTypeStr = "Другой тип"; break;

}

RichEdit1->Lines->Add(driveLetter + " - " + driveTypeStr);

}

}

// Информация о времени работы системы

RichEdit1->Lines->Add("");

RichEdit1->SelAttributes->Style = boldStyle;

RichEdit1->Lines->Add("Время работы системы:");

RichEdit1->SelAttributes->Style = TFontStyles();

DWORD tickCount = GetTickCount();

int days = tickCount / (1000 \* 60 \* 60 \* 24);

int hours = (tickCount % (1000 \* 60 \* 60 \* 24)) / (1000 \* 60 \* 60);

int minutes = (tickCount % (1000 \* 60 \* 60)) / (1000 \* 60);

RichEdit1->Lines->Add("Компьютер работает: " + String(days) + " дн. " +

String(hours) + " ч. " + String(minutes) + " мин.");

}

2. Выполнить аналогичное приложение на питоне. Я буду использовать psutil для простого взаимодействия

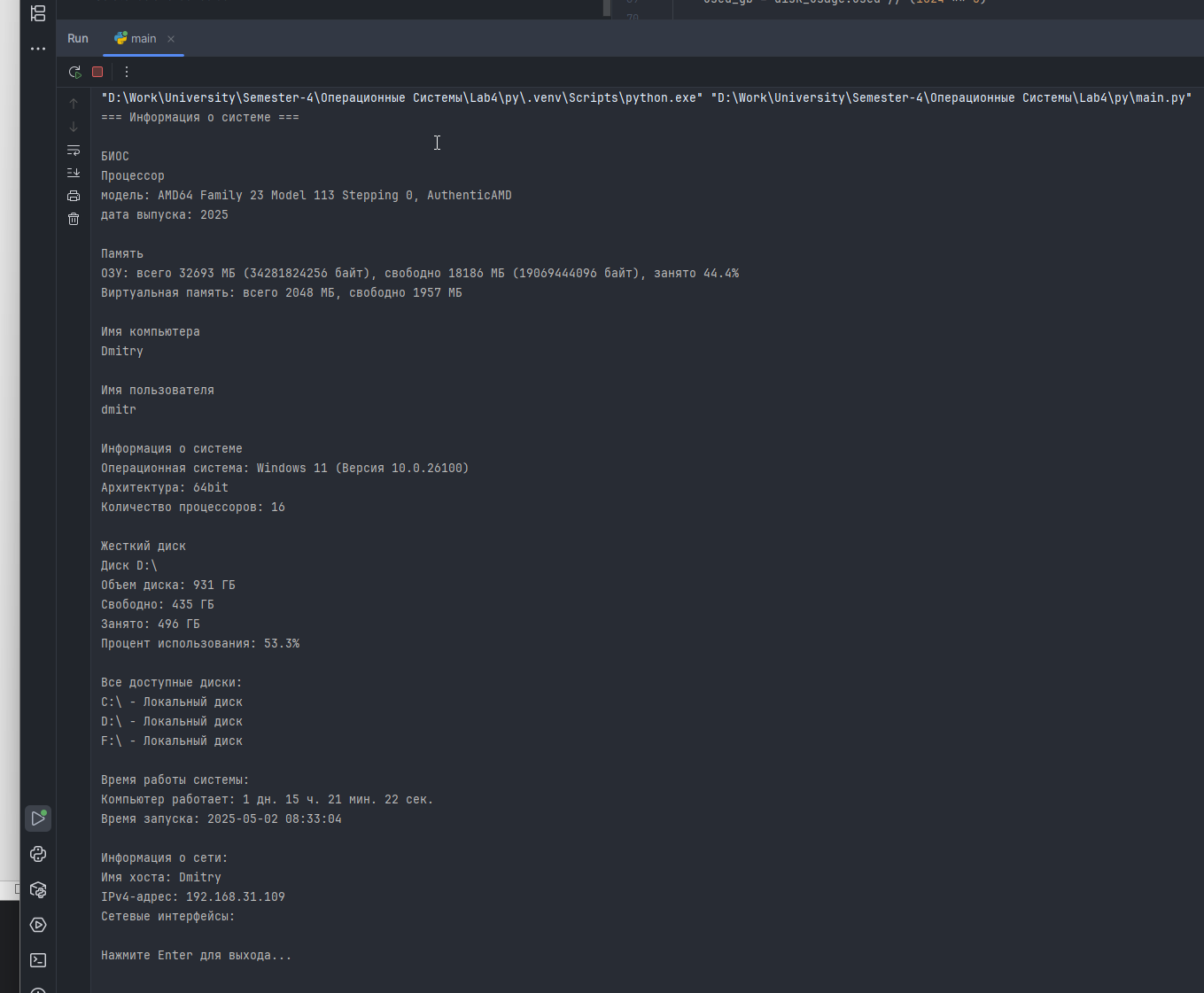


Рисунок 3. Работающая программа на питоне

Как мы видим, программа на питоне пишется проще, но работает медленнее.

Код программы на питоне:

import platform

import os

import psutil

import socket

import getpass

import datetime

def print\_header(text):

"""Функция для печати заголовка с выделением (жирный шрифт в консоли)"""

# ANSI-код для жирного текста: \033[1m

# ANSI-код для сброса форматирования: \033[0m

if os.name == 'nt': # Windows может не поддерживать ANSI в некоторых консолях

print(text)

else:

print(f"\033[1m{text}\033[0m")

def show\_system\_info():

"""Выводит информацию о системе в консоль"""

# Информация о БИОС и процессоре

print\_header("БИОС")

print("Процессор")

print(f"модель: {platform.processor()}")

print(f"дата выпуска: {datetime.datetime.now().year}")

print()

# Информация о памяти

print\_header("Память")

mem = psutil.virtual\_memory()

total\_mb = mem.total // (1024 \* 1024)

available\_mb = mem.available // (1024 \* 1024)

print(

f"ОЗУ: всего {total\_mb} МБ ({mem.total} байт), свободно {available\_mb} МБ ({mem.available} байт), занято {mem.percent}%")

# Дополнительная информация о виртуальной памяти

swap = psutil.swap\_memory()

swap\_total\_mb = swap.total // (1024 \* 1024)

swap\_free\_mb = swap.free // (1024 \* 1024)

print(f"Виртуальная память: всего {swap\_total\_mb} МБ, свободно {swap\_free\_mb} МБ")

print()

# Имя компьютера

print\_header("Имя компьютера")

print(f"{socket.gethostname()}")

print()

# Имя пользователя

print\_header("Имя пользователя")

print(f"{getpass.getuser()}")

print()

# Информация о системе

print\_header("Информация о системе")

print(f"Операционная система: {platform.system()} {platform.release()} (Версия {platform.version()})")

print(f"Архитектура: {platform.architecture()[0]}")

print(f"Количество процессоров: {os.cpu\_count()}")

print()

# Информация о дисках

print\_header("Жесткий диск")

# Основной диск (где запущена программа)

root\_path = os.path.abspath(os.sep)

disk\_usage = psutil.disk\_usage(root\_path)

total\_gb = disk\_usage.total // (1024 \*\* 3)

free\_gb = disk\_usage.free // (1024 \*\* 3)

used\_gb = disk\_usage.used // (1024 \*\* 3)

print(f"Диск {root\_path}")

print(f"Объем диска: {total\_gb} ГБ")

print(f"Свободно: {free\_gb} ГБ")

print(f"Занято: {used\_gb} ГБ")

print(f"Процент использования: {disk\_usage.percent}%")

print()

# Все доступные диски

print\_header("Все доступные диски:")

for partition in psutil.disk\_partitions():

if os.name == 'nt':

# Windows

drive\_type = "Неизвестно"

try:

if "fixed" in partition.opts:

drive\_type = "Локальный диск"

elif "cdrom" in partition.opts:

drive\_type = "CD/DVD"

elif "removable" in partition.opts:

drive\_type = "Съемный диск"

print(f"{partition.device} - {drive\_type}")

except:

pass

else:

# Unix/Linux/Mac

try:

usage = psutil.disk\_usage(partition.mountpoint)

print(f"{partition.mountpoint} - {partition.fstype} - {usage.percent}% использовано")

except:

print(f"{partition.mountpoint} - {partition.fstype}")

print()

# Информация о времени работы системы

print\_header("Время работы системы:")

boot\_time = datetime.datetime.fromtimestamp(psutil.boot\_time())

now = datetime.datetime.now()

uptime = now - boot\_time

days, remainder = divmod(uptime.total\_seconds(), 86400)

hours, remainder = divmod(remainder, 3600)

minutes, remainder = divmod(remainder, 60)

seconds = int(remainder)

print(f"Компьютер работает: {int(days)} дн. {int(hours)} ч. {int(minutes)} мин. {seconds} сек.")

print(f"Время запуска: {boot\_time.strftime('%Y-%m-%d %H:%M:%S')}")

# Информация о сети

print()

print\_header("Информация о сети:")

# Получение IP-адресов

hostname = socket.gethostname()

print(f"Имя хоста: {hostname}")

try:

print(f"IPv4-адрес: {socket.gethostbyname(hostname)}")

except:

print("IPv4-адрес: не доступен")

# Сетевые интерфейсы

print("Сетевые интерфейсы:")

for interface\_name, interface\_addresses in psutil.net\_if\_addrs().items():

for address in interface\_addresses:

if str(address.family) == 'AddressFamily.AF\_INET':

print(f" {interface\_name}: {address.address}")

print("=== Информация о системе ===\n")

try:

show\_system\_info()

except Exception as e:

print(f"Произошла ошибка при получении информации о системе: {e}")

# Ждем ввода пользователя перед закрытием программы

if os.name == 'nt': # Windows

input("\nНажмите Enter для выхода...")

**Вывод:** В ходе выполнения лабораторной работы были успешно разработаны два приложения для вывода системной информации: GUI-приложение на C++ Builder и консольное приложение на Python. Оба приложения позволяют получить и отобразить основные характеристики компьютера, включая информацию о процессоре, оперативной памяти, дисковом пространстве, операционной системе и другие системные параметры. Для C++ приложения был разработан графический интерфейс с использованием компонентов Button и RichEdit, где данные выводятся с форматированием текста и выделением заголовков жирным шрифтом. Консольная версия на Python использует библиотеку psutil, что значительно упрощает процесс получения системной информации. Разработанные приложения демонстрируют различные подходы к получению системных данных и могут быть полезны как для обычных пользователей, так и для системных администраторов.