Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

ОТЧЁТ

к лабораторной работе №6

на тему

**НЕКОТОРЫЕ СЛУЖЕБНЫЕ И ТЕХНОЛОГИЧЕСКИЕ ЗАДАЧИ.**

Выполнил: студент гр.253505 Авдошко И.С.

Проверил: ассистент кафедры информатики Гриценко Н.Ю.

Минск 2024

СОДЕРЖАНИЕ

[1 Формулировка задачи 3](#_Toc179742261)

[2 Краткие теоритические сведения 3](#_Toc179742262)

[3 Описание функций программы 6](#_Toc179742263)

[Заключение 8](#_Toc179742264)

[Список использованных источников 9](#_Toc179742265)

[Приложение А (обязательное) 10](#_Toc179742266)

# 1 ФОРМУЛИРОВКА ЗАДАЧИ

Цель лабораторной работы – изучение основ работы с системными ресурсами и технологий получения информации о состоянии компьютера, включая использование системных API, реестра, WMI и других инструментов. В процессе выполнения лабораторной работы изучаются подходы к сбору, обработке и отображению системных данных.

В качестве задачи необходимо разработать программу на языке C++, которая предоставляет пользователю удобный интерфейс для просмотра информации о системе. Утилита должна собирать и отображать следующие сведения:

­– общие данные об операционной системе, включая ее версию, номер сборки и установленный сервис-пак;

– информацию о процессоре, включая его модель и характеристики;

– данные об оперативной памяти, такие как общий объем и доступное количество памяти;

– информацию о дисковых устройствах, включая объем свободного и общего пространства;

– сведения о сетевых интерфейсах, такие как IP-адреса, маски подсетей и шлюзы.

Для достижения этой цели требуется:

1 Использовать системные функции и API Windows, включая GlobalMemoryStatusEx, GetDiskFreeSpaceEx, GetAdaptersInfo или их аналоги, для получения данных.

2 Организовать удобный формат вывода информации в консольном интерфейсе, структурировав сведения по разделам.

3 Реализовать обработку ошибок для отображения корректных сообщений в случае невозможности получения данных из источников.

Архитектура программы предполагает модульность: каждый тип данных (операционная система, процессор, память, диски, сеть) обрабатывается отдельными функциями, что упрощает расширение и сопровождение программы.

2 КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ

Реестр – это иерархическая база данных, содержащая данные, критически важные для работы Windows, а также приложений и служб, работающих в Windows. Данные структурированы в формате дерева. Каждый узел в дереве называется ключом. Каждый ключ может содержать как подразделы, так и записи данных, называемые значениями. Иногда наличие ключа — это все данные, необходимые приложению; в других случаях приложение открывает ключ и использует значения, связанные с ключом. Ключ может иметь любое количество значений, а значения могут быть в любой форме [1].

У каждого ключа есть имя, состоящее из одного или нескольких печатных символов. В именах ключей регистр не учитывается. Имена ключей не могут содержать символ обратной косой черты (\), но можно использовать любой другой печатный символ. Имена значений и данные могут содержать символ обратной косой черты.

Имя каждого подраздела уникально по отношению к ключу, который находится непосредственно над ним в иерархии. Имена ключей не локализованы в других языках, хотя значения могут быть.

Прежде чем помещать данные в реестр, приложение должно разделить данные на две категории: данные для конкретных компьютеров и данные пользователя. Благодаря такому различию приложение может поддерживать нескольких пользователей, а также находить пользовательские данные по сети и использовать эти данные в разных расположениях, что позволяет использовать данные профиля пользователя, не зависящие от расположения. (Профиль пользователя – это набор данных конфигурации, сохраненных для каждого пользователя.)

При установке приложение должно записывать данные для конкретного компьютера в ключ HKEY\_LOCAL\_MACHINE. В частности, он должен создавать ключи для названия компании, названия продукта и номера версии, как показано в следующем примере:

\*\*HKEY\_LOCAL\_MACHINE\Software\\*\*MyCompany\MyProduct\1.0

Если приложение поддерживает COM, оно должно записывать эти данные вHKEY\_LOCAL\_MACHINE\Software\Classes.

Приложение должно записывать пользовательские данные под ключом HKEY\_CURRENT\_USER , как показано в следующем примере:

\*\*HKEY\_CURRENT\_USER\Software\\*\*MyCompany\MyProduct\1.0 [2].

Модель безопасности Windows позволяет управлять доступом к разделам реестра. Дополнительные сведения о безопасности см. в разделе Модель управления доступом.

Дескриптор безопасности для раздела реестра можно указать при вызове функции RegCreateKeyEx или RegSetKeySecurity. Если указать значение NULL, ключ получает дескриптор безопасности по умолчанию. Списки управления доступом в дескрипторе безопасности по умолчанию для ключа наследуются от его прямого родительского ключа.

Чтобы получить дескриптор безопасности раздела реестра, вызовите функцию RegGetKeySecurity, GetNamedSecurityInfo или GetSecurityInfo .

Допустимые права доступа для разделов реестра включают стандартные права доступа DELETE, READ\_CONTROL, WRITE\_DAC и WRITE\_OWNER. Разделы реестра не поддерживают стандартное право доступа SYNCHRONIZE.

При вызове функции RegOpenKeyEx система проверяет запрошенные права доступа по дескриптолю безопасности ключа. Если у пользователя нет правильного доступа к разделу реестра, операция открытия завершится ошибкой. Если администратору требуется доступ к разделу, то решение – включить привилегию SE\_TAKE\_OWNERSHIP\_NAME и открыть раздел реестра с WRITE\_OWNER доступом. Дополнительные сведения см. в разделе Включение и отключение привилегий.

Вы можете запросить ACCESS\_SYSTEM\_SECURITY право доступа к разделу реестра, если хотите прочитать или записать системный список управления доступом ключа (SACL). Дополнительные сведения см. в разделах Списки управления доступом (ACL) и Права доступа saCL [3].

3 ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ

Программа представляет собой утилиту для сбора, анализа и отображения системной информации с использованием реестра Windows. Она предоставляет данные об операционной системе, памяти, текущем пользователе, логических дисках и сетевых интерфейсах, используя системные вызовы и функции Windows API, а также чтение значений из реестра. Такой подход позволяет эффективно собирать информацию из разных источников, включая реестр, файловую систему и сетевые параметры, что способствует более глубокой интеграции с операционной системой.

Основная логика утилиты реализована через несколько функций, каждая из которых выполняет конкретную задачу. Функция PrintSystemInfo() отвечает за сбор сведений об операционной системе. Для этого используются данные, хранящиеся в реестре Windows, что позволяет получить точную информацию о текущей версии операционной системы. Программа обращается к разделу реестра HKEY\_LOCAL\_MACHINE\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion, где хранится информация о названии операционной системы, номере сборки, редакции и Release ID. Это обеспечивает точность и актуальность данных, получаемых об операционной системе. Реестр Windows играет ключевую роль в хранении настроек системы и приложений, что делает его важным источником информации для программы. В реестре хранятся не только параметры операционной системы, но и данные о настроенных компонентах, а также информация о пользователях и безопасности.

Дополнительно программа включает функцию PrintRegistryInfo(), которая извлекает информацию о путях к системным директориям и программным файлам. Эти данные также берутся из раздела реестра HKEY\_LOCAL\_MACHINE\\SOFTWARE\\Microsoft\\Windows\\CurrentVersionТаким образом, с помощью реестра можно получить точные сведения о расположении системных файлов, что важно для диагностики и мониторинга состояния системы. Взаимодействие с реестром позволяет утилите работать с настройками и параметрами, которые являются частью конфигурации системы и приложений.

Функция PrintDiskInfo() занимается анализом данных о логических дисках. Здесь используется стандартный вызов API GetLogicalDriveStringsA, который позволяет получить список всех доступных логических дисков. Для каждого диска вызывается функция GetDiskFreeSpaceExA, которая предоставляет подробную информацию о диске, включая его общий объем, доступное пространство и свободное место. Эта информация полезна для анализа состояния файловой системы, и она выводится в мегабайтах, что удобно для пользователя.

Функция PrintNetworkInfo() отвечает за сбор данных о сетевых интерфейсах системы. Она использует функцию GetAdaptersInfo, которая извлекает информацию о каждом сетевом адаптере, включая его имя, описание, IP-адрес, маску подсети и адрес шлюза. Эти данные выводятся в текстовом формате, что помогает диагностировать сетевые подключения и получить информацию о доступных интерфейсах.

Функция PrintMemoryInfo() анализирует память системы с помощью вызова GlobalMemoryStatusEx, который предоставляет данные о физической и виртуальной памяти. С помощью этих данных программа может вывести информацию о общем объеме памяти, доступной и используемой памяти, что дает полное представление о состоянии памяти в системе.

Общий список всех параметров системы представлен на рисунке 3.1.

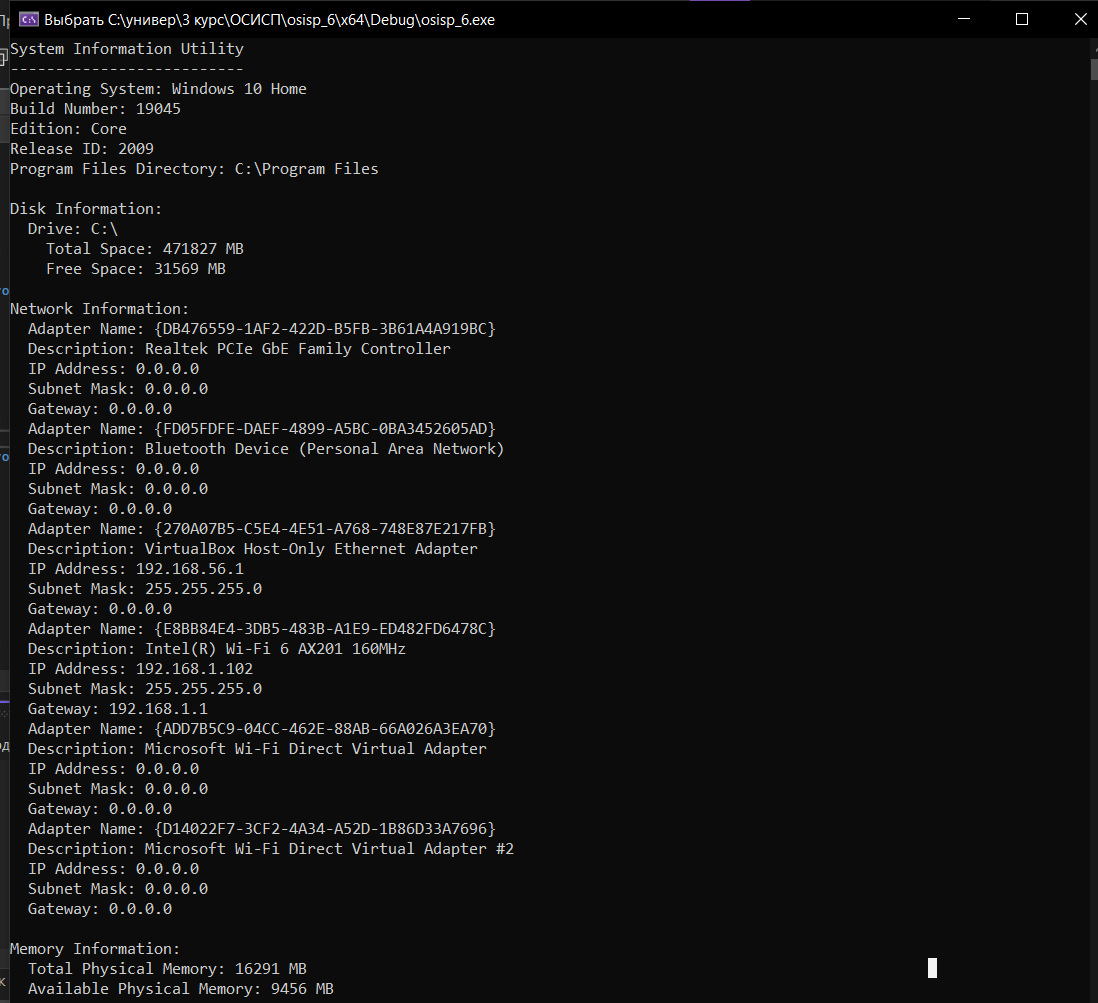


Рисунок 3.1 – Список системных параметров

Все эти функции взаимодействуют с операционной системой с использованием Windows API и данных из реестра, что позволяет получить точную и актуальную информацию о системных ресурсах. Реестр играет важную роль в работе программы, так как хранит настройки и параметры, которые обеспечивают взаимодействие с операционной системой и различными приложениями. Работая с реестром, программа получает доступ к ключевым данным, таким как информация об операционной системе, конфигурации сети и расположении системных файлов. Такой подход позволяет утилите быть гибкой и эффективной, обеспечивая сбор и отображение системной информации с высокой степенью точности.

# ЗАКЛЮЧЕНИЕ

Целью разработки данной программы было создание утилиты для сбора, анализа и отображения системной информации о операционной системе, памяти, дисках и сетевых интерфейсах с использованием реестра Windows и Windows API. В процессе работы была разработана программа, которая эффективно использует возможности Windows для извлечения важных системных данных, таких как версия операционной системы, параметры памяти, информация о логических дисках и сетевых интерфейсах. Особое внимание было уделено использованию реестра Windows, который является ключевым компонентом операционной системы и служит источником информации о конфигурации системы, настройках приложений и других важных данных.

Для реализации поставленной задачи использовались системные вызовы Windows API, такие как GlobalMemoryStatusEx, GetDiskFreeSpaceExA, GetAdaptersInfo, а также прямое обращение к реестру Windows для получения информации о версии и параметрах операционной системы. Это позволило собрать полную картину состояния системы, обеспечив точность и актуальность данных. Использование реестра добавляет дополнительный уровень интеграции с операционной системой, позволяя программе извлекать сведения, которые недоступны через стандартные API-вызовы.

Реестр Windows сыграл важную роль в работе программы, предоставив информацию о версии операционной системы и ее настройках, а также о пути к ключевым системным файлам. Реестр является неотъемлемой частью конфигурации Windows, и его использование в программе позволило не только получить точную информацию о системе, но и продемонстрировать возможности работы с этим важным компонентом операционной системы.

Таким образом, утилита выполняет свою задачу эффективно, обеспечивая глубокий доступ к системным данным и предоставляя пользователю удобный способ получения актуальной информации о состоянии операционной системы и ресурсов. Работа с реестром и API Windows обеспечила полное и точное представление о системе, что делает утилиту полезным инструментом для диагностики и мониторинга.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Microsoft. "Структура реестра" [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/sysinfo/structure-of-the-registry.

[2] Microsoft. "Категории данных" [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/sysinfo/categories-of-data.

[3] Microsoft. "Безопасность раздела реестра и права доступа" [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/sysinfo/registry-key-security-and-access-rights

ПРИЛОЖЕНИЕ А

(обязательное)

**Исходный код программы**

#include <windows.h>

#include <iostream>

#include <string>

#include <iphlpapi.h>

#include <vector>

#pragma comment(lib, "iphlpapi.lib")

void PrintSystemInfo();

void PrintRegistryInfo();

void PrintDiskInfo();

void PrintNetworkInfo();

void PrintMemoryInfo();

void PrintUserInfo();

int main() {

std::cout << "System Information Utility\n";

std::cout << "--------------------------\n";

PrintSystemInfo();

PrintRegistryInfo();

PrintDiskInfo();

PrintNetworkInfo();

PrintMemoryInfo();

PrintUserInfo();

std::cout << "\nPress any key to exit...";

std::cin.get();

return 0;

}

void PrintSystemInfo() {

// Получение версии ОС через реестр

HKEY hKey;

if (RegOpenKeyEx(HKEY\_LOCAL\_MACHINE, L"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion", 0, KEY\_READ, &hKey) == ERROR\_SUCCESS) {

DWORD dwType = 0;

TCHAR szValue[512];

DWORD dwSize = sizeof(szValue);

if (RegQueryValueEx(hKey, L"ProductName", NULL, &dwType, (LPBYTE)szValue, &dwSize) == ERROR\_SUCCESS) {

std::wcout << "Operating System: " << szValue << std::endl;

}

if (RegQueryValueEx(hKey, L"CurrentBuild", NULL, &dwType, (LPBYTE)szValue, &dwSize) == ERROR\_SUCCESS) {

std::wcout << "Build Number: " << szValue << std::endl;

}

if (RegQueryValueEx(hKey, L"EditionID", NULL, &dwType, (LPBYTE)szValue, &dwSize) == ERROR\_SUCCESS) {

std::wcout << "Edition: " << szValue << std::endl;

}

if (RegQueryValueEx(hKey, L"ReleaseId", NULL, &dwType, (LPBYTE)szValue, &dwSize) == ERROR\_SUCCESS) {

std::wcout << "Release ID: " << szValue << std::endl;

}

RegCloseKey(hKey);

}

else {

std::cerr << "Failed to open registry key for OS info.\n";

}

}

void PrintRegistryInfo() {

// Получение информации о Windows через реестр

HKEY hKey;

if (RegOpenKeyEx(HKEY\_LOCAL\_MACHINE, L"SOFTWARE\\Microsoft\\Windows\\CurrentVersion", 0, KEY\_READ, &hKey) == ERROR\_SUCCESS) {

DWORD dwType = 0;

TCHAR szValue[512];

DWORD dwSize = sizeof(szValue);

if (RegQueryValueEx(hKey, L"ProgramFilesDir", NULL, &dwType, (LPBYTE)szValue, &dwSize) == ERROR\_SUCCESS) {

std::wcout << "Program Files Directory: " << szValue << std::endl;

}

if (RegQueryValueEx(hKey, L"CommonFilesDir", NULL, &dwType, (LPBYTE)szValue, &dwSize) == ERROR\_SUCCESS) {

std::wcout << "Common Files Directory: " << szValue << std::endl;

}

if (RegQueryValueEx(hKey, L"CommonFilesDir (x86)", NULL, &dwType, (LPBYTE)szValue, &dwSize) == ERROR\_SUCCESS) {

std::wcout << "Common Files Directory (x86): " << szValue << std::endl;

}

RegCloseKey(hKey);

}

else {

std::cerr << "Failed to open registry key for program files directory.\n";

}

}

void PrintDiskInfo() {

std::cout << "\nDisk Information:\n";

char driveStrings[256];

DWORD driveLength = GetLogicalDriveStringsA(sizeof(driveStrings), driveStrings);

char\* drive = driveStrings;

while (\*drive) {

ULARGE\_INTEGER freeBytesAvailable, totalBytes, totalFreeBytes;

if (GetDiskFreeSpaceExA(drive, &freeBytesAvailable, &totalBytes, &totalFreeBytes)) {

std::cout << " Drive: " << drive << "\n";

std::cout << " Total Space: " << totalBytes.QuadPart / (1024 \* 1024) << " MB\n";

std::cout << " Free Space: " << totalFreeBytes.QuadPart / (1024 \* 1024) << " MB\n";

}

drive += strlen(drive) + 1;

}

}

void PrintNetworkInfo() {

std::cout << "\nNetwork Information:\n";

ULONG bufferSize = 0;

GetAdaptersInfo(nullptr, &bufferSize);

std::vector<BYTE> buffer(bufferSize);

IP\_ADAPTER\_INFO\* adapterInfo = reinterpret\_cast<IP\_ADAPTER\_INFO\*>(buffer.data());

if (GetAdaptersInfo(adapterInfo, &bufferSize) == ERROR\_SUCCESS) {

while (adapterInfo) {

std::cout << " Adapter Name: " << adapterInfo->AdapterName << "\n";

std::cout << " Description: " << adapterInfo->Description << "\n";

std::cout << " IP Address: " << adapterInfo->IpAddressList.IpAddress.String << "\n";

std::cout << " Subnet Mask: " << adapterInfo->IpAddressList.IpMask.String << "\n";

std::cout << " Gateway: " << adapterInfo->GatewayList.IpAddress.String << "\n";

adapterInfo = adapterInfo->Next;

}

}

else {

std::cerr << "Failed to retrieve network information.\n";

}

}

void PrintMemoryInfo() {

MEMORYSTATUSEX memInfo = { sizeof(MEMORYSTATUSEX) };

GlobalMemoryStatusEx(&memInfo);

std::cout << "\nMemory Information:\n";

std::cout << " Total Physical Memory: " << memInfo.ullTotalPhys / (1024 \* 1024) << " MB\n";

std::cout << " Available Physical Memory: " << memInfo.ullAvailPhys / (1024 \* 1024) << " MB\n";

}

void PrintUserInfo() {

char username[256];

DWORD username\_len = sizeof(username);

if (GetUserNameA(username, &username\_len)) {

std::cout << "\nUser Information:\n";

std::cout << " Logged In User: " << username << "\n";

}

else {

std::cerr << "Failed to get username.\n";

}

}