Лабораторна робота №3

Створення утіліти «DiskInfo»

Хід роботи:

Завдання :

У даній лабораторній роботі, використовуючи виклики системних функцій, отримати інформацію про дискову підсистему:

Список усіх логічних дисків в системі.

Отримати тип кожного диску присутнього в системі, та дати пояснення для кожного типу диску.

Отримати інформацію про диски в системі та про файлові системи Які Використовують на них.

Отримати інформацію про зайнятості та вільне місце на кожному з дисків.

Отримати інформацію про системну пам'ять.

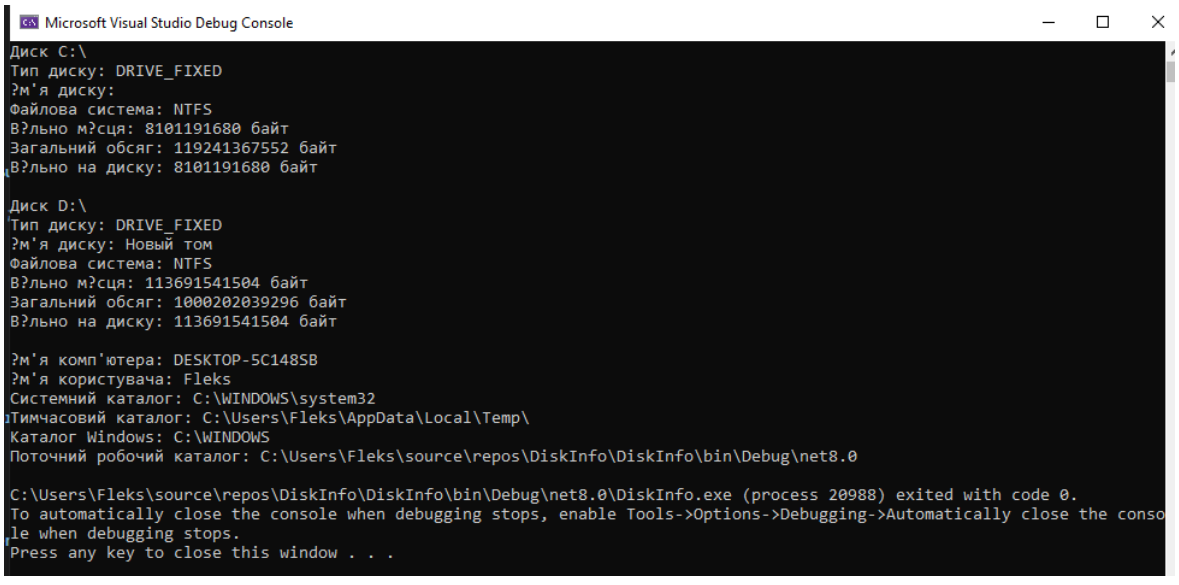
Отримати інформацію про Назву комп'ютера

Отримати Назву поточного користувача

Отримати інформацію про поточний системний каталог, Тимчасовий каталог, поточний робочий каталог.

Для обраних каталогу на диску, Включити спостереження за змінами, продемонструвати відслідковування більше однієї зміни. Зміни записувати в лог файл.

Для данної роботи використовую C# з утилітою DllImport для виклику системних функцій:



```
using System;
using System.IO;
using System.Runtime.InteropServices;
using System.Text;

class DiskInfo
{
    [DllImport("kernel32.dll")]
    static extern uint GetLogicalDrives();

    [DllImport("kernel32.dll")]
    static extern DriveType GetDriveType(string lpRootPathName);

    [DllImport("kernel32.dll", SetLastError = true, CharSet = CharSet.Auto)]
    static extern bool GetVolumeInformation(string lpRootPathName, StringBuilder
lpVolumeNameBuffer, uint nVolumeNameSize, out uint lpVolumeSerialNumber, out uint
lpMaximumComponentLength, out uint lpFileSystemFlags, StringBuilder
lpFileSystemNameBuffer, uint nFileSystemNameSize);

    [DllImport("kernel32.dll", SetLastError = true, CharSet = CharSet.Auto)]
    static extern bool GetDiskFreeSpaceEx(string lpDirectoryName, out ulong
lpFreeBytesAvailable, out ulong lpTotalNumberOfBytes, out ulong
lpTotalNumberOfFreeBytes);

    [DllImport("kernel32.dll", SetLastError = true)]
    static extern void GlobalMemoryStatus(out MEMORYSTATUS lpBuffer);

    [DllImport("kernel32.dll")]
    static extern bool GetComputerName(StringBuilder lpBuffer, ref uint lpnSize);

    [DllImport("advapi32.dll", SetLastError = true)]
    static extern bool GetUserName(System.Text.StringBuilder sb, ref Int32 length);

    [DllImport("kernel32.dll", SetLastError = true, CharSet = CharSet.Auto)]
    static extern uint GetSystemDirectory(StringBuilder lpBuffer, uint uSize);

    [DllImport("kernel32.dll", SetLastError = true, CharSet = CharSet.Auto)]
    static extern uint GetTempPath(uint nBufferLength, StringBuilder lpBuffer);

    [DllImport("kernel32.dll", SetLastError = true, CharSet = CharSet.Auto)]
    static extern uint GetWindowsDirectory(StringBuilder lpBuffer, uint uSize);

    [DllImport("kernel32.dll", SetLastError = true, CharSet = CharSet.Auto)]
    static extern uint GetCurrentDirectory(uint nBufferLength, StringBuilder
lpBuffer);

    [DllImport("kernel32.dll", SetLastError = true)]
    static extern IntPtr FindFirstChangeNotification(string lpPathName, bool
bWatchSubtree, uint dwNotifyFilter);

    [DllImport("kernel32.dll", SetLastError = true)]
    static extern bool FindNextChangeNotification(IntPtr hChangeHandle);

    [DllImport("kernel32.dll", SetLastError = true)]
    static extern bool FindCloseChangeNotification(IntPtr hChangeHandle);

    // ...

    static void Main()
    {
        uint drivesBitMask = GetLogicalDrives();
        for (int i = 0; i < 26; ++i)
        {
            if ((drivesBitMask & (1 << i)) != 0)
            {
                string driveLetter = $"{(char)('A' + i)}:\\";
                Console.WriteLine($"Диск {driveLetter}");
```

```
                DriveType driveType = GetDriveType(driveLetter);
                Console.WriteLine($"Тип диску: {driveType}");

                StringBuilder volumeNameBuffer = new StringBuilder(261);
                StringBuilder fileSystemNameBuffer = new StringBuilder(261);
                uint serialNumber, maxComponentLen, fileSystemFlags;
                if (GetVolumeInformation(driveLetter, volumeNameBuffer,
(uint)volumeNameBuffer.Capacity, out serialNumber, out maxComponentLen, out
fileSystemFlags, fileSystemNameBuffer, (uint)fileSystemNameBuffer.Capacity))
                {
                    Console.WriteLine($"Ім'я диску: {volumeNameBuffer}");
                    Console.WriteLine($"Файлова система: {fileSystemNameBuffer}");
                }

                ulong freeBytesAvailable, totalNumberOfBytes, totalNumberOfFreeBytes;
                if (GetDiskFreeSpaceEx(driveLetter, out freeBytesAvailable, out
totalNumberOfBytes, out totalNumberOfFreeBytes))
                {
                    Console.WriteLine($"Вільно місця: {freeBytesAvailable} байт");
                    Console.WriteLine($"Загальний обсяг: {totalNumberOfBytes} байт");
                    Console.WriteLine($"Вільно на диску: {totalNumberOfFreeBytes}
байт");
                }

                Console.WriteLine();
            }
        }

       /* MEMORYSTATUS memoryStatus;
        GlobalMemoryStatus(out memoryStatus);
        Console.WriteLine($"Загальний обсяг пам'яті: {memoryStatus.dwTotalPhys}
байт");
        Console.WriteLine($"Вільно пам'яті: {memoryStatus.dwAvailPhys} байт");*/

        StringBuilder computerNameBuffer = new StringBuilder(261);
        uint size = (uint)computerNameBuffer.Capacity;
        if (GetComputerName(computerNameBuffer, ref size))
        {
            Console.WriteLine($"Ім'я комп'ютера: {computerNameBuffer}");
        }

        StringBuilder userNameBuffer = new StringBuilder(261);
        Int32 userNameLength = userNameBuffer.Capacity;
        if (GetUserName(userNameBuffer, ref userNameLength))
        {
            Console.WriteLine($"Ім'я користувача: {userNameBuffer}");
        }

        StringBuilder systemDirectoryBuffer = new StringBuilder(261);
        if (GetSystemDirectory(systemDirectoryBuffer,
(uint)systemDirectoryBuffer.Capacity) != 0)
        {
            Console.WriteLine($"Системний каталог: {systemDirectoryBuffer}");
        }

        StringBuilder tempPathBuffer = new StringBuilder(261);
        if (GetTempPath((uint)tempPathBuffer.Capacity, tempPathBuffer) != 0)
        {
            Console.WriteLine($"Тимчасовий каталог: {tempPathBuffer}");
        }

        StringBuilder windowsDirectoryBuffer = new StringBuilder(261);
        if (GetWindowsDirectory(windowsDirectoryBuffer,
(uint)windowsDirectoryBuffer.Capacity) != 0)
        {
            Console.WriteLine($"Каталог Windows: {windowsDirectoryBuffer}");
```

```
        }

        StringBuilder currentDirectoryBuffer = new StringBuilder(261);
        if (GetCurrentDirectory((uint)currentDirectoryBuffer.Capacity,
currentDirectoryBuffer) != 0)
        {
            Console.WriteLine($"Поточний робочий каталог: {currentDirectoryBuffer}");
        }

        // Відстеження змін на диску...
    }


}

enum DriveType : uint
{
    DRIVE_UNKNOWN = 0,
    DRIVE_NO_ROOT_DIR = 1,
    DRIVE_REMOVABLE = 2,
    DRIVE_FIXED = 3,
    DRIVE_REMOTE = 4,
    DRIVE_CDROM = 5,
    DRIVE_RAMDISK = 6
}

[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Auto)]
public class MEMORYSTATUS
{
    public uint dwLength;
    public uint dwMemoryLoad;
    public uint dwTotalPhys;
    public uint dwAvailPhys;
    public uint dwTotalPageFile;
    public uint dwAvailPageFile;
    public uint dwTotalVirtual;
    public uint dwAvailVirtual;
}
```