|  |
| --- |
| ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ |
| (повна назва вищого навчального закладу) |
| Кафедра «Електронних обчислювальних машин» |
| (повна назва кафедри) |

КУРСОВИЙ ПРОЕКТ

|  |  |
| --- | --- |
| по курсу: | Системне програмне забезпечення |
|  | (назва дисципліни) |
| на тему: | Розробка програмного забезпечення, призначеного |
|  | для моніторингу стану вільного простору на системних дисках |
|  |  |
|  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Студента | 3 | | | курсу | | КІУКІ-18-4 | групи |
|  | напрямку підготовки | | | | Комп’ютерна інженерія | | | |
|  | спеціальності | | Системне програмування | | | | | |
|  | Кравченко Н.С. | | | | | | | |
|  | (прізвище та ініціали) | | | | | | | |
|  | Керівник | Фомічов О.О. | | | | | | |
|  | К.т.н., ст. викладач | | | | | | | |
|  | (посада, вчене звання, наукова степінь, прізвище та ініціали) | | | | | | | |
|  | Національна шкала | | | |  | | | |
|  | Кількість балів: | | |  | | Оцінка: ECTS | |  |
|  |  | | | | | | | |

Харків 2020

|  |
| --- |
| Харківський національний університет радіоелектроніки |

(повна назва вищого учбового закладу)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Кафедра: | Електронних обчислювальних машин | | | | |
| Дисципліна: | Системне програмне забезпечення | | | | |
| Спеціальність: | Системне програмування | | | | |
| Курс: | 3 | Група: | КІУКІ-18-4 | Семестр: | 5 |

**З А В Д А Н Н Я**

на курсовий проект (роботу) студента

Кравченко Назар Сергійович

(прізвище, ім’я, по-батькові)

|  |  |  |
| --- | --- | --- |
| 1. Тема проекту (роботи) | Створення програми, призначеної для моніторингу | |
| стану вільного простору на системних дисках. | | |
| 2. Термін здачі студентом закінченого проекту (роботи) До 30.12.2019 | | |
| 3. Вихідні данні до проекту (роботи) | | Програма, пояснювальна записка. |
| Розробити програмне забезпечення, призначеного для моніторингу вільного | | |
| простору на системних дисках . Програма періодично виконує опитування | | |
| системи і виводить дані системних дисків. | | |
| 4. Зміст пояснювальної записки (перелік які підлягають розробці питань): | | |
| 1. Постановка задачі 2. Теоретичні відомості 3. Технічне завдання | | |
| 4. Розробка алгоритму роботи програми 5. Опис програми 6. Інструкція користувача 7. Інструкція оператора 8. Висновки | | |
|  | | |
|  | | |
| 6. Дата видачі завдання: | 29.09.2020 | |

КАЛЕНДАРНИЙ ЛИСТ

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Зміст виконуваних робіт | Період (в тиж.) | Примітка |
| 1 | Видача завдань | 2 тиждень |  |
| 2 | Ознайомлення з літературними джерелами, аналіз та вибір методу вирішення поставленої задачі | 3-4 тижні |  |
| 3 | Розробка алгоритмів рішення, вибір  системних засобів рішення задач курсової роботи | 5-6 тижні |  |
| 4 | Проектування програмного додатку | 7-11 тижні |  |
| 5 | Налагодження та тестування програми | 12-14 тижні |  |
| 6 | Оформлення пояснювальної записки | 15 тиждень |  |
| 7 | Захист курсового проекту | 16-17 тижні |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | | |  |  | |
|  | | | (підпис) |  | |
| Керівник | | |  |  |  |
|  | | | (підпис) |  | (прізвище, ім’я, по-батькові) |
| « » |  | 20\_\_\_ | |  | |

РЕФЕРАТ

Пояснювальна записка до курсового проекту курсу містить 34 сторінки, 8 розділів, 5 таблиць, 2 ілюстрації, 4 лістингу, 7 джерел.

Завданням курсового проекту є розробка програми моніторингу вільного місця на системних дисках. Розроблене програмне забезпечення повинно мати такі функції:

* Пошук і оновлення даних про усі системні диски;
* Відображення даних у зручному для користувача вигляді;
* Встановлення ліміту на зайнятий обсяг пам’яті диску;
* У випадку порушення ліміту виводити повідомлення.

Методом розробки програми є використання мови програмування C#, а також системи побудови додатків WindowsForms.

Результатом є працездатне програмне забезпечення, яке дозволяє відстежувати і контролювати зайняте місце на системних дисках.

Ключові слова: WINDOWS FORMS, АНАЛІЗ, ПРОГРАМА, МОНІТОРИНГ ВІЛЬНОГО МІСЦЯ НА СИСТЕМНИХ ДИСКАХ, ДИСК, ТАЙМЕР, ПОВІДОМЛЕННЯ, C#, .NET,

ЗМІСТ

[ВСТУП 4](#_Toc27823446)

[1. ПОСТАНОВКА ЗАДАЧІ 5](#_Toc27823447)

[2. ТЕОРЕТИЧНІ ВІДОМОСТІ 6](#_Toc27823448)

[2.1 Огляд ООП 6](#_Toc27823449)

[2.2 Огляд C# 7](#_Toc27823450)

[2.3 Розробка програмного забезпечення на платформі .NET 7](#_Toc27823451)

[2.4 Клас Control 8](#_Toc27823452)

[2.5 Огляд елементів керування 9](#_Toc27823453)

[2.6 Статичні елементи керування 11](#_Toc27823454)

[2.7 Кнопки і прапорці 12](#_Toc27823455)

[2.8 Смуги прокрутки 14](#_Toc27823456)

[2.9 Елементи керування з підтримкою редагування тексту 15](#_Toc27823457)

[2.10 Розміщення елементів 15](#_Toc27823458)

[2.11 Користувацькі елементи керування 16](#_Toc27823459)

[3. ТЕХНІЧНЕ ЗАВДАННЯ 18](#_Toc27823460)

[4. РОЗРОБКА АЛГОРИТМУ РОБОТИ ПРОГРАМИ 19](#_Toc27823461)

[4.1 Структура програми 19](#_Toc27823462)

[4.2 Опис змінних 23](#_Toc27823463)

[4.3 Опис алгоритму програми 29](#_Toc27823464)

[5. ОПИС ПРОГРАМИ 30](#_Toc27823465)

[6. ІНСТРУКЦІЯ СИСТЕМНОГО ПРОГРАМІСТА 31](#_Toc27823466)

[7. ПОСІБНИК ОПЕРАТОРА 32](#_Toc27823467)

[8. ВИСНОВКИ 33](#_Toc27823468)

[ПЕРЕЛІК ДЖЕРЕЛ 34](#_Toc27823469)

# ВСТУП

Мета курсової роботи - створення програми, яка надає можливість моніторингу вільного простору на системних дисках. Теоретична і практична значущість полягає в закріпленні навичок складання алгоритмів поставлених завдань і програмування елементів системних утиліт.

В ході виконання курсового проекту необхідно провести теоретичне дослідження основних принципів побудови і функціонування системних утиліт, практичне освоєння методів розробки сучасного системного програмного забезпечення і реалізація елемента системної утиліти.

Для створення цієї роботи було використано мову програмування C#, оскільки вона дозволяє вирішувати широкий спектр завдань, має гнучкість і великою кількістю зручних у використанні бібліотек. Крім цього, він дозволяє використовувати як об'єктно-орієнтований, так і структурний підхід. Як середовище розробки була обрана Visual Studio 2019.

# ПОСТАНОВКА ЗАДАЧІ

Основна задача співзвучна з темою курсової роботи і полягає в розробці програмного забезпечення для моніторингу вільного простору на системних дисках для ОС Windows, за допомогою мови програмування C# та платформи .NET.

Програмне забезпечення повинно

* відображати інформацію про всі системні диски у зручному для користувача вигляді
* мати можливість встановлення обмеження на вільний простір диску
* виводити службове повідомлення, якщо обмеження було порушено
* відображати розмір диску, зайнятий простір, вільний простір та обмеження.

Для цього необхідно розробити користувацький інтерфейс, який буде зручно надавати можливості для моніторингу будь якої кількості системних дисків.

# ТЕОРЕТИЧНІ ВІДОМОСТІ

## Огляд ООП

Об’єктно-орієнтоване програмування - одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою.

Клас – базове поняття в об’єктна-орієнтованому програмуванні. Примірник класу(екземпляр) - це об'єкт. Об'єкт - це сукупність даних (властивостей) і функцій (методів) для їх обробки. Властивості і методи називаються членами класу. Взагалі, об'єктом є все те, що підтримує інкапсуляцію.

ООП має три основні принципи.

Інкапсуляція - це механізм, що поєднує дані і обробляє їх код як єдине ціле.

Інкапсуляцією називається включення різних дрібних елементів в більш великий об'єкт, у результаті чого програміст працює безпосередньо з цим об'єктом. Це призводить до спрощення програми, оскільки з неї вилучаються другорядні деталі.

У ООП ця можливість забезпечується класами, об'єктами і різними засобами вираження ієрархічних зв’язків між ними.

Поліморфізм дозволяє використовувати одні й ті ж імена для схожих, але технічно різних завдань. Головним у поліморфізмі є те, що він дозволяє маніпулювати об'єктами шляхом створення стандартних інтерфейсів для схожих дій.

Успадкування дозволяє одному об'єкту набувати властивості іншого об'єкта, не плутайте з копіюванням об'єктів. При копіюванні створюється точна копія об'єкта, а при спадкуванні точна копія доповнюється унікальними властивостями, які характерні тільки для похідного об'єкта.

## Огляд C#

C# - мова програмування, що поєднує об'єктно-орієнтовані і аспектно-орієнтовані концепції. Розроблено в 1998-2001 роках групою інженерів під керівництвом Андерсa Хейлсбергa в компанії Microsoft як основна мова розробки додатків для платформи Microsoft .NET. Компілятор з C # входить в стандартну установку самої .NET, тому програми на ньому можна створювати і компілювати навіть без інструментальних засобів на кшталт Visual Studio.

C # відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до С ++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато від своїх попередників - мов С ++, Java, Delphi, Модула і Smalltalk - С #, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем: так, C # не підтримує множинне успадкування класів ( на відміну від C ++).

C # розроблявся як мова програмування прикладного рівня для CLR і, як такий, залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C #, яка відображає FCL. Присутність або відсутність тих чи інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльований в відповідні конструкції CLR.

## Розробка програмного забезпечення на платформі .NET

Розробку програмного забезпечення для Windows на платформі .NET можна здійснювати за допомогою фреймворків Windows Presentation Foundation (WPF), Windows Forms (WinForms), Universal Windows Platform (UWP), та Xamarin.

Оскільки ми розроблюємо програмне забезпечення виключно для Windows, можливості Xamarin та UWP, які забезпечують створення крос-платформених програм, не знадобляться. WPF та WinForms використовуються виключно для розробки програмного забезпечення для Windows.

WinForms - це інтерфейс програмування додатків (API), відповідальний за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за допомогою створення обгортки для Win32 API в керованому коді.

WPF – це графічна (презентаційна) підсистема (аналог WinForms), яка починаючи з .NET Framework 3.0 в складі цієї платформи. Має пряме відношення до XAML.

Це перше реальне оновлення технологічного середовища призначеного для користувача інтерфейсу з часу випуску Windows 95. Воно включає нове ядро для заміни GDI і GDI+, використовувані в Windows Forms. WPF є високорівневим об'єктно-орієнтованим функціональним шаром що дозволяє створювати двовимірні та тривимірні інтерфейси.

Оскільки WPF є більш складною технологією ніж WinForms і наше програмне забезпечення має відображати інформацію о вільному просторі на системних дисках елементів, для якої Win32 API буде достатньо, ми будемо використовувати фреймворк WinForms.

## Клас Control

Control - найважливіший клас в Windows Forms - він один дозволяє отримати незліченну кількість інструментів для створення додатків Windows Forms. Control не тільки базовий клас для таких елементів управління, як кнопки, дерева, панелі інструментів і меню, але і для Form - класу, що інкапсулює головне вікно програми Windows Forms і виконує інші функції в діалогових вікнах.

Об'єкт Form за замовчуванням містить рядок заголовка з системним меню зліва і кнопки згортки, розгортки та закриття вікна - справа. По периметру вікна проходить рамка визначення розміру вікна. Всередині вікна розташована клієнтська область - в ній відображається інформація, і містяться інструменти для користувача. Програми можуть відображати текст і графічну інформацію безпосередньо в клієнтської області і отримувати команди з клавіатури або миші з застосуванням таких методів, як OnKeyPress і OnMouseDown. Але в більшості сучасних програм в клієнтської області розташовуються дочірні елементи управління - обробка введення / виводу інформації про дії користувача делегується їм. Крім того, програмісти мають право створювати власні елементи управління, розширюючи або комбінуючи існуючі.

## Огляд елементів керування

Практично все на екрані Microsoft Windows складається з елементів керування. Дійсно, в загальному випадку, елемент керування можна визначити як візуальний об'єкт. Зазвичай елементи керування займають прямокутну область екрану, хоча можуть бути непрямокутними або навіть прихованими і виконувати команди, що надходять з клавіатури або миші. Клас Control підтримує безліч подій (і відповідних методів введення) для обробки введення інформації, наприклад події клавіатури KeyDown, KeyUp і KeyPress і миші - MouseDown, MouseUp і MouseMove. Також елементи керування повинні «перемальовувати» себе на екрані - для цього служить метод OnPaint.

Перетворюючи інформацію про операції користувача в прості події, елементи керування служать рівнем абстрагування між користувачем і додатком.

Одна з найважливіших визначених класом Control властивостей - Parent; вона вказує на інший об'єкт типу Control. У процесі виконання програми на дисплеї відображаються лише елементи керування з коректно визначеною властивістю Parent. Положення елемента завжди задається щодо «батька» і відображається на поверхні останнього. Частина елемента керування, яка виступає за краї батька, не відображається.

Клас Form є винятком з цього правила. Значення властивості Parent об'єкта типу Form звичайно дорівнює null, тобто батьківською властивістю форми є робочий стіл. Однак об'єкт Form, у якого значення властивості TopLevel дорівнює false (тобто батько форми - не робочий стіл), може бути нащадком будь-якого іншого елемента керування. Така методика використовується в додатках з багатодокументним інтерфейсом (Multiple Document Interface, MDI). (В класі Form є властивість Owner, в якому можна задати інший об'єкт типу Form. Форма на екрані завжди відображається поверх свого власника, і якщо останній згортається або закривається, то ж відбувається з формою. Цей механізм корисний в немодальних діалогових вікнах. Власник немодального діалогового вікна повинен зіставлятися з створившим його додатком.)

За замовчуванням всі елементи керування видимі і активні. Якщо властивості Visible привласнити значення false, елемент керування зникне з поверхні свого батька. Він нікуди не пропаде і як і раніше буде залишатися членом батьківського набору елементів керування, але картина буде такою, як ніби його немає. Усе нащадки невидимого елементу керування також невидимі.

Однак найбільш часто змінюють властивість Enabled. Елемент з властивістю Enabled рівним false відображається на екрані, але виглядає знебарвленим, або відключеним. Такий елемент керування не отримує фокус введення і не реагує на команди клавіатури або миші.

Кожному елементи керування дається певний розмір і положення щодо лівого верхнього кута його батька. Розташування визначається властивістю Location - об'єктом Point з двома властивостями, X та Y. Координати встановлюються в пікселях, і визначають положення верхнього лівого кута елемента керування, відносно верхнього лівого кута батька. Розмір елементу встановлюється властивістю Size об'єктом типу Size з властивостями Width і Height.

У всіх елементів управління є властивість Text, хоча деякі з них (Наприклад, смуга прокрутки) не відображають ніякого тексту. Для об'єкту Form, властивість Text - це текст, який відображається в рядку заголовку форми.

Елемент керування успадковує вихідні властивості Font, BackColor і ForeColor у свого батька. При зміні властивостей батька властивості «дочок» також змінюються. Однак якщо елементам управління явно задати інші властивості, вони збережуть їх і перестануть залежати від батьків.

## Статичні елементи керування

Група елементів керування (GroupBox).

Група елементів керування по периметру виділена лінією і вгорі містить текст, визначений у властивості Text. Рамка групи темніше кольору фону.

Зазвичай групи використовуються в якості «батька» кнопок-перемикачів (Radio buttons). Всі перемикачі всередині однієї групи взаємовиключні. Натискання кнопок зі стрілками переміщує фокус введення і мітку вибору між перемикачами. GroupBox -, не нащадок ContainerControl, оскільки в ContainerControl навігація виконується клавішею Tab.

Перемикачам зовсім не обов'язково бути нащадками GroupBox. Загальним батьком групи взаємовиключних перемикачів може служити будь-який елемент керування. Логікою активізації перемикачів керує не група, а самі перемикачі.

Мітка (Label).

Label - це елемент керування, що відображає нередагуємий текст. Сам текст задається у властивості Text. Хоча мітка може відображати текст в декількох рядках, вона не виводить смуг прокрутки, якщо текст не вміщується в елементі керування. Якщо в елементі керування треба розмістити об'ємний текст, але робити його редагованим не можна, використовується елемент керування TextBox з властивістю Readonly рівною true.

Крім тексту, мітка може відображати об'єкт Image, який можете бути як об'єктом Bitmap, так і Metafile. (Ці класи визначені в просторах імен System.Drawing і System.Drawing.lmaging).

Крім цього, можна використовувати зображення, задані в якості ресурсів і прикріплені до виконуваного файлу програми.

## Кнопки і прапорці

Елемент керування «кнопка» називається так не тому, що його можна клацати, - в кінцевому підсумку, всі елементи при натисканні ініціюють подію Click, а головним чином з тієї причини, що при натисканні він візуально поводиться як кнопка. Усі три типи кнопок походять від абстрактного класу ButtonBase.

Зазвичай Button використовується для ініціювання дій, CheckBox – для установки і зняття прапорців, a RadioButton зазвичай служить для вибору одного з декількох взаємовиключних варіантів.

На кнопці як правило присутній текст, але деякі з них – особливо екземпляри класу Button - містять зображення, з текстом або без. Клас ButtonBase містить властивість Image, успадковану кнопками всіх трьох типів і що дозволяє призначити відображаємий на кнопці растр або метафайл. Крім цього, можна задати властивість ImageList об'єкта ButtonBase і задати конкретне зображення, використовуючи властивості Imagelndex або ImageKey. (Застосування ImageList доцільно, тільки якщо ви працюєте з безліччю кнопок із зображеннями. Можна, призначити один і той же об'єкт ImageList для всіх кнопок, але з різними властивостями Imagelndex або ImageKey для кожної кнопки.)

Якщо кнопка повинна відображати і текст, і зображення, можна задати визнвчену в об'єкті ButtonBase властивість TextlmageRelation, якій присвоюють одне з значень перерахування TextlmageRelation, що складається з членів ImageAboveText, ImageBeforeText, TextAbovelmage, TextBeforelmage і Overlay.

Властивості TextAlign и ImageAlign задають положення тексту або зображення в елементі управління. Обидва приймають значення перерахування ContentAlignment, що складається з дев'яти членів - комбінацій Top, Middle і Bottom з Left, Center і Right. Значення за замовчуванню обох властивостей – ContentAlignment.MiddleCenter.

Кнопка (Button).

У порівнянні з ButtonBase клас Button привносить мало нового. В ньому практично завжди визначається обробник події Click. Єдиний виняток привласнення властивості DialogResult значення з перерахування DialogResult (зазвичай DialogResult.OK або DialogResult.Cancel).

Прапорець (CheckBox)

Зазвичай елемент керування CheckBox - це невеликий прямокутник, розташований зліва від тексту. Однак якщо властивості Appearance задати значення Арреarance.Button (за замовчуванням Appearance.Normal), елемент управління буде схожий з кнопкою, за винятком того, що буде після натискання «залипати» і «відлипати». Вид елемента керування CheckBox можна змінити, задавши властивості CheckAlign значення з перерахування ContentAlignment. За замовчуванням використовується значення ContentAlignment.MiddleLeft, тобто прапорець по вертикалі вирівнюється по осі тексту і розташовується зліва від нього.

Булева властивість Checked вказує поточний стан прапорця - «встановлений» або «скинутий». Подія CheckedChanged інформує про зміну стану.

За замовчуванням значення AutoCheck дорівнює true, тобто по команді з клавіатури або по написканню миші кнопка автоматично змінює свій стан і ініціює подію CheckedChanged. Якщо відгук необхідний при будь-якій зміні стану прапорця (наприклад, для активізації і відключення інших елементів управління), встановіть обробник події CheckedChanged. Якщо відгук для кожної зміни не потрібен, зазвичай досить отримати значення властивості Checked, коли користувач натисканням кнопку ОК закриває форму.

Властивості AutoCheck зазвичай задають значення false, коли планують захоплювати подія Click і керувати установкою / скиданням прапорця самостійно, задаючи значення властивості Checked з програми. Така зміна властивості Checked також ініціює подію CheckedChanged.

Іноді недостатньо одного стану прапорця - встановлений або скинутий, - потрібен проміжний стан.

Для використання цього третього стану, перш за все, треба булевій властивості ThreeState привласнити значення true. Замість властивості Checked потрібно використовувати CheckState. Це значення з перерахування CheckState, що складається з членів Unchecked, Checked і Indeterminate. Також, замість установки обробника CheckedChange, використовують CheckStateChanged. Якщо викликати метод EnableVisualStyles класу Application проміжний стан буде виглядати як невеликий квадрат, а інакше в елементі керуання буде відображатися сірий прапорець.

Якщо AutoCheck дорівнює true, одним натисканням елемент керування буде по циклу змінювати свій стан: «встановлений-скинутий-невизначений». Якщо необхідний інший порядок, надайте AutoCheck значення false і вручну закодуйте в обробнику події Click порядок зміни значень CheckState.

## Смуги прокрутки

Смуги прокрутки зазвичай встановлюються з правого боку або знизу таких елементів керування як ListBox, ComboBox і TextBox (при використанні в багаторядковому режимі, як в Windows Notepad). Ці елементи керування автоматично відображають смуги прокрутки. Похідні від ScrollableControl елементи керування, в тому числі Form і Panel, також відображають смуги прокрутки, коли їх властивість AutoScroll дорівнює true, а дочірні елементи керування організовані так, що зайнята ними область виходить за рамки батька.

## Елементи керування з підтримкою редагування тексту

Текстове поле (TextBox) - це найпростіший елемент керування з підтримкою редагування тексту, але разом з тим він досить складний, щоб стати основою блокнота Windows.

TextBox додає лише кілька властивостей до наявних в TextBoxBase. Напевно, найважливіше з них - це ScrollBars, якому присвоюються значення з перерахування ScrollBars.Vertical, ScrollBars.Horizontal або ScrollBars.Both. значення за замовчуванням - ScrollBars.None, тобто, смуги прокрутки не відображаються, навіть якщо цього вимагає довжина тексту. (Якщо значення властивості Wordwrap доівнює true, горизонтальні смуги прокрутки не відображаються незалежно від значення ScrollBars).

Якщо TextBox використовується для введення пароля, потрібно вказати в PasswordChar символ, який буде відображатися при введенні. В цьому відношенні також корисно властивість CharacterCasing. Якщо привласнити цій властивості значення з перерахування CharacterCasing.Lower або CbaracterCasing.Upper, символи що вводяться будуть приводитися в нижній або верхній регістр відповідно.

## Розміщення елементів

Панель TableLayoutPanel розміщує дочірні елементи в сітці рядків і стовпців. Зазвичай кожен дочірній елемент керування займає одну клітинку сітки, але ніщо не забороняє розміщувати окремі елементи на декількох суміжних клітинах. Звичайно, інші панелі, що містять інші елементи TableLayoutPanel, також можуть займати ці осередки. Однак оскільки підтримка і управління таблицями досить ресурсномістке заняття, краще не захоплюватися вкладенням таблиць. У багатьох випадках елемент FlowLayoutPanel справляється з подібним завданням не гірше, ніж дочірні елементи TableLayoutPanel

Є два основних способи використання TableLayoutPanel У більшості випадків значенням true ініціюється властивість AutoSize елемента TableLayoutPanel і контейнера панелі. Таким чином, розміри таблиці і її контейнера підганяються для розміщення елементів управління.

В якості альтернативи можна надати користувачам можливість змінювати розмір контейнера таблиці або в вигляді переміщуваного кордону, або з використанням елемента SplitContainer. В цьому випадку властивості Dock елементу TableLayoutPanel присвоюється значення DockStyle.Fill. Документація попереджує, що це не найефективніший спосіб використання TableLayoutPanel, але це ідеальний варіант для виконання деяких нестандартних завдань.

## Користувацькі елементи керування

З точки зору програмування, нестандартний (призначений для користувача) елемент керування - це визначений програмістом клас, який безпосередньо або побічно успадковує класу Control. Призначений для користувача елемент управління може бути як модифікацією існуючого, так і абсолютно новим. Елемент керування можна істотно адаптувати, просто встановивши в ньому додаткові обробники подій. Але якщо потрібно повністю змінити стандартний механізм обробки подій, буде потрібен новий клас.

Елемент керування - це, по суті, фільтр, що обробляє інформацію, що надходить від користувача інформацію і перетворює її в дії додатка, наприклад в події. Найчастіше елемент керування виконує три важливі завдання. По-перше, він відображається на екрані у вигляді впізнаваного користувачем візуального об'єкту. По друге, він обробляє інформацію, що надходить від користувача (зазвичай з клавіатури і миші). (Є спеціальні елементи керування, що підтримують обробку голосового введення або введення зі стилосом в планшетних ПК.) І, по-третє, він ініціює події, сповіщаючи додаток про конкретні зміни.

Готові елементи керування були розроблені і протестовані на предмет коректного виконання всіх трьох функцій. Тому набагато простіше удосконалити готовий елемент управління, створивши, похідний клас від існуючого елементу керування, ніж починати з нуля, створюючи підклас Control.

Дуже часто нові елементи керування створюють шляхом комбінування готових. Щоб скористатися перевагами логіки готових елементів керування, рекомендується створювати такі елементи керування як спадкоємці класу UserControl. Серед іншого, цей клас підтримує перемикання між декількома елементами керування з клавіатури.

Є елементи керування, які сильно відрізняються від стандартних, - для них доводиться писати власний код промальовування і обробки користувальницького введення. Їх створення краще почати з похідного від Control класу.

# ТЕХНІЧНЕ ЗАВДАННЯ

Основною вимогою для реалізації даної курсової роботи є розробка програмного забезпечення для моніторингу вільного простору системних дисків: загальний розмір, вільний простір, використаний простір. Крім цього реалізувати встановлення обмеження на використаний простір. У випадку досягнення або перевищення значень обмеження повідомляти користувача окремим діалоговим вікном. Граничні значення для обмеження задаються в абсолютному (байти, мегабайти або гігабайти) або відсотковому значенні.

Програма повинна мати зрозумілий інтерфейс та детальну інформацію про систему, а також захищена від усіх можливих помилок користувача у процесі введення значень і в разі, якщо користувач робить щось не так, то вона повідомляє його про це за допомогою діалогових вікон.

# РОЗРОБКА АЛГОРИТМУ РОБОТИ ПРОГРАМИ

## Структура програми

Клас DriveAndRule містить системний диск та члени необхідні для контролю зайнятого простору. Усі маніпуляції з системними дисками у програмі здійснюються через екземпляри даного класу.

public class DriveAndRule

{

DriveInfo drive { get; set; }

public bool isTracking { get; set; }

public Int64 TrackingAmount { get; set; }

public AboutDrive representaition { get; set; }

public DriveAndRule(DriveInfo d)

{

drive = d;

isTracking = false;

TrackingAmount = 0;

representaition = null;

}

//Получить имя диска

public string DriveName

{

get { return drive.Name; }

}

//Получить свободное место диска

public Int64 DriveFreeSpace

{

get { return drive.TotalFreeSpace; }

}

//Получить размер диска

public Int64 DriveSize

{

get { return drive.TotalSize; }

}

//Получить занятое место на диске

public Int64 DriveUsedSpace

{

get { return Tools.UsedSpace(drive); }

}

//Обновить диск и отображение

public void Update(DriveInfo data)

{

drive = data;

representaition.DrawSeries();

}

}

Лістинг 3.1 – клас DriveAndRule

Статичний клас Tools містить суфікси розмірів дисків, методи для перетворення розміру дисків і розширюючий метод класу DriveInfo для отримання зайнятого простору.

static class Tools

{

//Определенные размеры

public static string[] SizeSuffixes = { "percents", "bytes", "KB", "MB", "GB", "TB", "PB", "EB", "ZB", "YB" };

//Получение оптимальной приставки для байтов

public static string SizeSuffixString(Int64 value)

{

if (value < 0) { return "-" + SizeSuffixString(-value); }

if (value == 0) { return "0.0 bytes"; }

int mag = (int)Math.Log(value, 1024);

decimal adjustedSize = (decimal)value / (1L << (mag \* 10));

return string.Format("{0:n1} {1}", adjustedSize, SizeSuffixes[mag + 1]);

}

//Перевод из определенного размера в байты

public static long RawData(int index, Int64 value)

{

index--;

long adjustedSize = value \* (1L << (index \* 10));

return adjustedSize;

}

public static Int64 UsedSpace(this DriveInfo drive)

{

return (drive.TotalSize - drive.TotalFreeSpace);

}

}

Лістинг 3.2 – клас Tools

Головне вікно MainForm відображає системні диски оновлює дані та слідкує за обмеженням простору. Диски зберігаються у масиві типу DriveAndRule, графічне представлення дисків розміщається у вікні за допомогою контейнеру типу TableLayoutPanel.

private void timer\_Tick(object sender, EventArgs e)

{

//Обновление информации, проверка контроля и удаление неактуальных диков

bool IsExist;

for (int i = drivesAndRules.Count - 1;i>= 0; i--)

{

IsExist = false;

foreach (DriveInfo drive in DriveInfo.GetDrives())

{

if (drive.IsReady && drivesAndRules[i].DriveName == drive.Name)

{

//Проверка наличия диска и обновление данных

IsExist = true;

drivesAndRules[i].Update(drive);

}

}

//Проверка контроля

if (drivesAndRules[i].isTracking && drivesAndRules[i].TrackingAmount < drivesAndRules[i].DriveUsedSpace)

{

//Проверка наличия окна о диске

if (!drivesAndRules[i].representaition.TrackInfo.IsDisposed)

drivesAndRules[i].representaition.TrackInfo.Dispose();

//Остановка контроля и вывод соосбщения

drivesAndRules[i].isTracking = false;

MessageBox.Show($"Drive {drivesAndRules[i].DriveName} memory Tracking rule overwlow", "Warning");

drivesAndRules[i].TrackingAmount = 0;

drivesAndRules[i].representaition.chkBox.Checked = false;

}

//Удаление неактуальных диков

if (!IsExist)

{

//Проверка наличия окна о диске

if (!drivesAndRules[i].representaition.TrackInfo.IsDisposed)

drivesAndRules[i].representaition.TrackInfo.Dispose();

//Уничтожение графического представления и элемента списка

drivesAndRules[i].representaition.Dispose();

drivesAndRules.RemoveAt(i);

}

}

foreach (DriveInfo drive in DriveInfo.GetDrives())

{

//Поиск новых дисков

IsExist = false;

foreach (DriveAndRule data in drivesAndRules)

if (drive.IsReady && data.DriveName == drive.Name)

IsExist = true;

//Добавление новых дисков

if (!IsExist)

{

drivesAndRules.Add(new DriveAndRule(drive));

drivesAndRules[drivesAndRules.Count-1].representaition = new AboutDrive(drivesAndRules[drivesAndRules.Count - 1]);

tableLayoutPanel.Controls.Add(drivesAndRules[drivesAndRules.Count - 1].representaition);

}

}

}

Лістинг 3.3 – оновлення даних у таймері MainForm

Користувацький компонент AboutDrive забезпечує зручне відображення даних про системний диск.

Форма «Активировать мониторинг» дозволяє встановити, змінити або відмінити обмеження простору.

private void bttnTrackInfo\_Click(object sender, EventArgs e)

{

Int64 tmp;

//Проверка корректности ввода

if (!Int64.TryParse(txtBox.Text, out tmp))

{

MessageBox.Show("Input correct value", "Warning");

return;

}

//Определение контрольного значения

if (comboBox.SelectedIndex == 0)

{

if (tmp <= 0 || tmp > 100)

{

MessageBox.Show("Percents must be between 0 and 100", "Warning");

return;

}

if ((Int64)(about.driveAndRule.DriveSize \* ((double)tmp / 100)) < (about.driveAndRule.DriveSize - about.driveAndRule.DriveFreeSpace))

{

MessageBox.Show("Tracking amount less then used space", "Warning");

return;

}

about.driveAndRule.TrackingAmount = (Int64)(about.driveAndRule.DriveSize \* ((double)tmp / 100));

about.driveAndRule.isTracking = true;

about.chkBox.Checked = about.driveAndRule.isTracking;

}

else

{

if (tmp <= 0 || Tools.RawData(comboBox.SelectedIndex, tmp) >= about.driveAndRule.DriveSize)

{

MessageBox.Show("Value must be between 0 and Capacity", "Warning");

return;

}

if (Tools.RawData(comboBox.SelectedIndex, tmp) <= about.driveAndRule.DriveUsedSpace)

{

MessageBox.Show("Tracking amount less then used space", "Warning");

return;

}

about.driveAndRule.TrackingAmount = Tools.RawData(comboBox.SelectedIndex, tmp);

about.driveAndRule.isTracking = true;

about.chkBox.Checked = about.driveAndRule.isTracking;

}

}

Лістинг 3.4 – встановлення обмеження простору

## Опис змінних

Клас DriveAndRule.

|  |  |  |
| --- | --- | --- |
| Назва змінної | Тип | Призначення |
| drive | DriveInfo | Зберігається системній диск. |
| isTracking | bool | Визначає чи контролюється диск. |
| TrackingAmount | Int64 | Визначає розмір контролюємого простору. |
| representaition | AboutDrive | Посилання на графічне представлення диску. |

Таблиця 3.2.1 – змінні класу DriveAndRule.

Клас MainForm.

|  |  |  |
| --- | --- | --- |
| Назва змінної | Тип | Призначення |
| Max | System.Drawing.Size | Зберігається максимальний розмір, при якому вікно перестане збільшуватись і з’являться смуги прокрутки. |
| drivesAndRules | System.Collections.Generic.List<DriveAndRule> | Колекція в якій зберігаються системні диски. |
| tableLayoutPanel | System.Windows.Forms.TableLayoutPanel | Панель для розміщення графічного представлення дисків. |
| timer | System.Windows.Forms.Timer | Таймер у події якого оновлюються дані. |

Таблиця 3.2.2 – змінні класу MainForm.

Клас AboutDrive.

|  |  |  |
| --- | --- | --- |
| Назва змінної | Тип | Призначення |
| driveAndRule | DriveAndRule | Посилання на дані системного диску, які відображатимуться |
| TrackingInfo | TrackingInfo | Вікно керування обмеженням простору даного диску |
| lblDriveLetter | System.Windows.Forms.Label | Мітка, що відображає текст на формі. |
| lblDriveLetterValue | System.Windows.Forms.Label | Мітка на формі, яка відображає ім’я системного диску. |
| lblDriveCapacity | System.Windows.Forms.Label | Мітка, що відображає текст на формі. |
| lblDriveCapacityValue | System.Windows.Forms.Label | Мітка на формі, яка відображає розмір системного диску. |
| chart | System.Windows.Forms.DataVisualization.Charting.Chart | Діаграма, що у графічному вигляді відображає співвідношення вільного і зайнятого місця. |
| groupBox | System.Windows.Forms.GroupBox | Забезпечує рамку і заголовок. |
| chkBox | System.Windows.Forms.CheckBoxIndex | Відображає чи існує обмеження простору для цього системного диску. |

Таблиця 3.2.3 – змінні класу AboutDrive.

Клас TrackingInfo.

|  |  |  |
| --- | --- | --- |
| Назва змінної | Тип | Призначення |
| bindingSuffix | System.Windows.Forms.BindingSource | Інкапсулює джерело даних для comboBox. |
| about | DriveTrackInfoer.AboutDrive | Посилання на графічне представлення диску, з якого була викликана ця форма. |
| bttnTrackInfo | System.Windows.Forms.Button | Встановити обмеження простору з вказаними парарметрами. |
| bttnCancel | System.Windows.Forms.Button | Скасувати обмеження простору. |
| tableLayoutPanel | System.Windows.Forms.TableLayoutPanel | Панель у якій знаходяться елементи інтерфейсу. |
| lblAmount | System.Windows.Forms.Label | Мітка, що відображає текст на формі. |
| lblAmountValue | System.Windows.Forms.Label | Мітка на формі, яка відображає обмеження простору системного диску. |
| lblLetter | System.Windows.Forms.Label | Мітка, що відображає текст на формі. |
| lblLetterValue | System.Windows.Forms.Label | Мітка на формі, яка відображає ім’я системного диску. |
| lblTracking | System.Windows.Forms.Label | Мітка, що відображає текст на формі. |
| chkBox | System.Windows.Forms.CheckBox | Відображає чи існує обмеження простору для цього диску. |
| txtBox | System.Windows.Forms.TextBox | Елемент для введення користувачем розміру обмеження простору. |
| comboBox | System.Windows.Forms.ComboBox | Елемент для вибору користувачем величини обмеження простору. |

Таблиця 3.2.4 – змінні класу TrackingInfo.

Клас Tools.

|  |  |  |
| --- | --- | --- |
| Назва змінної | Тип | Призначення |
| SizeSuffixes | System.String | Суфікси в яких може відображатися/задаватися розмір диску/обмеження |

Таблиця 3.2.5 – змінні класу Tools.

## Опис алгоритму програми

Форма MainForm відображає, оновлює дані системних дисків і слідкує за обмеженням простору, якщо обмеження було порушене виводиться системне повідомлення.

Оновлення даних здійснюється при загрузці головної форми, а далі за подією таймера.

Спочатку програма перевіряє чи усі збережені диски досі актуальні, неактуальні диски позначаються для видалення. Далі перевіряється чи є порушення обмежень простору, якщо є пов’язані вікна контролі обмежень закриваються і виводиться системне повідомлення, а обмеження знімається. Після цих операцій неактуальні диски видаляються, попередньо видаливши свої графічні представлення і закривши вікна контролю обмежень, якщо ті біли відкриті. Завдяки такій послідовності дій, навіть якщо диск став неактуальним, а обмеження було порушено – системне повідомлення буде виведено.

Наступним етапом оновлення даних є додавання нових дисків у системі. Якщо у системі з’являється новий диск він додається до сховища дисків, створюється графічне представлення цього диску, яке виводиться у головній формі.

Обмеження можуть бути встановлені у будь-який момент.

# ОПИС ПРОГРАМИ

Запуск програми відкриває MainForm - це головна форма, в якій відбуваються всі основні дії: відображення даних системних дисків та контроль за обмеженнями простору, завдяки таймеру. Форма містить панель TableLayaoutPanel, яка відображає елементи AboutDrive, що здійснюють графічне представлення дисків.

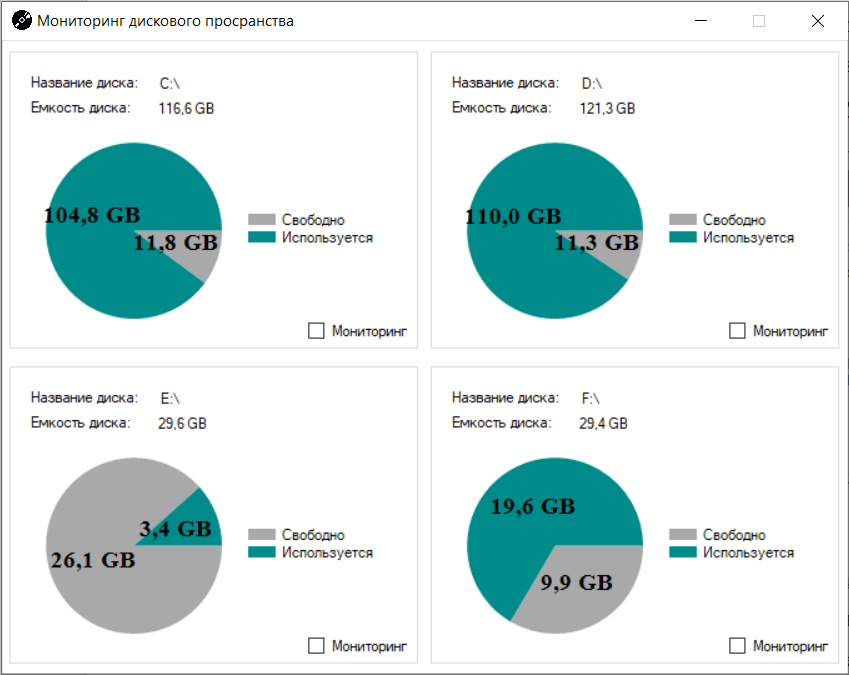


Рисунок 4.1 – головне вікно програми.

Натискання на мітку відкриває нову форму «Активировать мониторинг», основна задача якої полягає в тому, що користувач при введені обмеження встановлює значення обмеження в елементі який його викликав. При досягненні обмеження відображається повідомлення угорі головної форми на нетривалий час та окреме вікно у вигляді MessageBox.

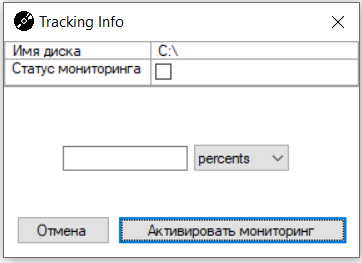


Рисунок 4.2 – вікно керування обмеженням простору

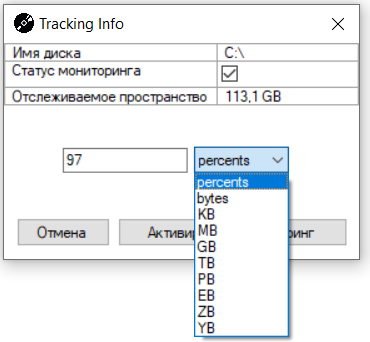


Рисунок 4.3 – вікно керування обмеженням простору із активним моніторингом

# ІНСТРУКЦІЯ СИСТЕМНОГО ПРОГРАМІСТА

## Тестування розробленої програми

У тестуванні цієї програми повинні входити перевірки:

- працездатність усіх кнопок меню;

- правильності обробки представлених керуючих кнопок;

- обробки можливих помилок;

- коректної обробки винятків;

- працездатності всіх додаткових функцій, які були враховані розробником.

Під час тестування цієї програми були виявлені деякі помилки, які, однак, були успішно виправлені.

Були реалізовані наступні виправлення помилок:

- перевірка на заповнення обов’язкових полів;

- надання неправдивих даних (наприклад, негативне значення простору);

- некоректне відображення помилок користувачу;

- хаотичні місцезнаходження допоміжних вікон на екрані при їх виклику.

## Системні вимоги розробленої програми

Мінімальні системні вимоги:

* Операційна система Microsoft Windows 10 з встановленим .NET Framework 4.7.2 або вище;
* монітор з розділовою здатністю 1024х768 або вище.
* 512 MB RAM;
* 5 MB вільного місця на системному диску.

# ПОСІБНИК ОПЕРАТОРА

Після запуску програми з’явиться головне вікно, яке відображатиме поточні системні диски.

Оновлення системних дисків здійснюється автоматично, підключіть або відключіть диск від системи та зачекайте.

Встановлення обмеження простору для системного диску здійснюється наступним чином:

* Знайдіть графічне представлення потрібного диску;
* Натисніть мітку «Мониторинг»;
* З’явиться вікно керування обмеженням простору, введіть потрібне обмеження і натисніть кнопку «Активировать мониторинг», у випадку некоректного введення виведеться повідомлення і вікно закриється;
* Якщо обмеження було задано коректно у графічному представлені диску перед міткою «Мониторинг» буде встановлений прапорець.

Оновлення системних дисків здійснюється автоматично, підключ

Прибирання обмеження простору для системного диску здійснюється наступним чином:

* Знайдіть графічне представлення потрібного диску;
* Натисніть мітку «Мониторинг»;
* З’явиться вікно керування обмеженням простору, натисніть кнопку Cancel;
* У графічному представлені диску перед міткою «Мониторинг» прапорець не буде встановлений.

# ВИСНОВКИ

У процесі виконання курсового проекту було розроблено програмне забезпечення, призначене для моніторингу вільного простору системних дисків комп’ютера. Програмне забезпечення виводить в окреме вікно інформацію про всі системні диски та періодично виконує опитування системи і оновлює дані, якщо для диску було встановлено обмеження простору і воно було порушено виводить спливаюче вікно з повідомленням.

Програма написана на С# за допомогою середовища Visual Studio 2019 і відповідає всім вимогам заданим на курсове проектування.

Інтерфейс програми дуже зручний у використанні і інформативний для моніторингу вільного простору необмеженої кількості системних дисків у системі. Всі кнопки інтуїтивно доступні для розуміння користувача.

У подальшому є можливість удосконалити даний програмний продукт, а саме замінити діючий інтерфейс на ще більш функціональний та сучасний, наприклад переписати форми у системі створення клієнтських додатків WPF або UWP. Також можна виконувати моніторинг за допомогою TPL, що збільшить швидкодію програми при великій кількості системних дисків.

У ході виконання курсового проекту були закріплені знання та навички, отримані при вивченні курсу "Системне програмне забезпечення". Придбано нові знання у роботи з середовищем розробки «Microsoft Visual Studio 2019», було отримано досвід програмування клієнтських додатків в операційній системі Windows.

# ПЕРЕЛІК ДЖЕРЕЛ

1. Троелсен Е. Мова програмування C # 5.0 і платформа .NET 4.5, 6-е видання - Pro C # 5.0 and the .NET 4.5 Framework, 6th edition. [Текст] / Троелсен Е. – «Вільямс», 2013. - 1312 с.
2. Справочник MSDN [Електронний ресурс]: справочник. – Режим доступа: [http://msdn.com](http://msdn.com/)
3. Програмування на платформі Microsoft .NET Framework 4.0 на мові C# [Текст] / Д. Ріхтер. – Санкт-Петербург: «Пітер», 2012. – 1136 с.
4. Петцольд Ч. Программирование с использованием Microsoft Windows Forms [Текст] / Ч. Петцольд – Русская Редакция, 2006. – 433с.
5. Jeffrey Richter CLR via C#, 4th edition [Текст], 2012. - 896 с.
6. Mark J Price.C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development, 4th edition [Текст], 2019. - 820 с.