|  |
| --- |
| 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 обліку працюючих у організації комп’ютерів |
|  |  |
|  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Студента | 3 | | | курсу | | КІУКІ-17-4 | групи |
|  | напрямку підготовки | | | | Комп’ютерна інженерія | | | |
|  | спеціальності | | Системне програмування | | | | | |
|  | Коротецький О.О. | | | | | | | |
|  | (прізвище та ініціали) | | | | | | | |
|  | Керівник | Фомічов О.О. | | | | | | |
|  | ст. викладач | | | | | | | |
|  | (посада, вчене звання, наукова степінь, прізвище та ініціали) | | | | | | | |
|  | Національна шкала | | | |  | | | |
|  | Кількість балів: | | |  | | Оцінка: ECTS | |  |
|  |  | | | | | | | |

Харків 2019

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

(повна назва вищого учбового закладу)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Кафедра: | Електронних обчислювальних машин | | | | |
| Дисципліна: | Системне програмне забезпечення | | | | |
| Спеціальність: | Системне програмування | | | | |
| Курс: | 3 | Група: | КІУКІ-17-4 | Семестр: | 5 |

**З А В Д А Н Н Я**

на курсовий проект (роботу) студента

Коротецький Олександр Олексійович

(прізвище, ім’я, по-батькові)

|  |  |  |
| --- | --- | --- |
| 1. Тема проекту (роботи): | Створення програми, призначеної для обліку | |
| працюючих в організації комп’ютерів. | | |
| 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 обліку працюючих | | |
| в організації комп’ютерів. Програма має зберігати та оперувати даними | | |
| про фізичний вміст комп’ютерів та сам комп’ютер.  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: | 10.10.2019 | |

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 26 сторінки, 8 розділів, 2 ілюстрації, 3 лістинги, 5 джерел.

Завданням курсового проекту є розробка програми обліку та керування комп’ютерами організації. Розроблене програмне забезпечення повинно мати такі функції:

* Додавання сутностей комп’ютерів та керування ними;
* Відображення даних у зручному для користувача вигляді;
* Редагування фізичного вмісту комп’ютерів;
* У випадку несумісності компонентів виводити повідомлення.

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

Результатом є працездатне програмне забезпечення, яке дозволяє проводити облік комп’ютерів та модифікувати їх.

Ключові слова: C#, .Net, WindowsForms, MS SQL DATABASE, ПРОГРАМА, ОБЛІК КОМП’ЮТЕРІВ У ОРГАНІЗАЦІЇ, БАЗА ДАННИХ, ПОВІДОМЛЕННЯ, ІНТЕРФЕЙС.

ЗМІСТ

[ВСТУП 3](#_Toc29998479)

[1. ПОСТАНОВКА ЗАДАЧІ 4](#_Toc29998480)

[2. ТЕОРЕТИЧНІ ВІДОМОСТІ 6](#_Toc29998481)

[2.1 Огляд ООП 6](#_Toc29998482)

[2.2 Огляд C# 7](#_Toc29998483)

[2.3 Розробка програмного забезпечення на платформі .NET 8](#_Toc29998484)

[2.4 Клас SqlClient 9](#_Toc29998485)

[2.5 Огляд елементів керування 9](#_Toc29998486)

[2.6 Статичні елементи керування 11](#_Toc29998487)

[2.7 Кнопки і прапорці 12](#_Toc29998488)

[2.8 Смуги прокрутки 15](#_Toc29998489)

[2.9 Елементи керування з підтримкою редагування тексту 15](#_Toc29998490)

[2.10 Розміщення елементів 16](#_Toc29998491)

[2.11 Користувацькі елементи керування 16](#_Toc29998492)

[3. ТЕХНІЧНЕ ЗАВДАННЯ 18](#_Toc29998493)

[4. РОЗРОБКА АЛГОРИТМУ РОБОТИ ПРОГРАМИ 19](#_Toc29998494)

[4.1 Структура програми 19](#_Toc29998495)

[4.2 Опис алгоритму програми 22](#_Toc29998496)

[5. ОПИС ПРОГРАМИ 22](#_Toc29998497)

[6. ІНСТРУКЦІЯ СИСТЕМНОГО ПРОГРАМІСТА 23](#_Toc29998498)

[7. ПОСІБНИК ОПЕРАТОРА 24](#_Toc29998499)

[8. ВИСНОВКИ 25](#_Toc29998500)

[ПЕРЕЛІК ДЖЕРЕЛ 26](#_Toc29998501)

# ВСТУП

Мета курсової роботи - створення програми, яка надає можливість обліку комп’ютерів у організації. 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 баз даних, практичне освоєння методів розробки сучасного системного програмного забезпечення і реалізація програмного і користувацького інтерфейсу управління базами даних.

Для створення цієї роботи було використано мову програмування C#, оскільки вона дозволяє вирішувати широкий спектр завдань, має гнучкість і велику кількість зручних у використанні бібліотек. Крім цього, вона 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 обліку та управління комп’ютерами у організації для ОС Windows, за допомогою мови програмування C# та платформи .NET.

Програмне забезпечення повинно відображати інформацію про всі комп’ютери у організації в зручному для користувача вигляді і мати можливість модифікації компонентів комп’ютерів, якщо виявлена несумісність компонентів виводити службове повідомлення. Серед відображеної інформації повинні бути інвентарний номер комп’ютеру, ідентифікаційні номери основних компонентів комп’ютерів, історію модифікацій та список периферійних пристроїв.

Для цього необхідно розробити користувацький інтерфейс, який буде зручно надавати можливості для обліку будь-якої кількості комп’ютерів та їх периферійних пристроїв.

# 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 SqlClient

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 DataGridView розміщує дочірні елементи в сітці рядків і стовпців. Цей елемент ідеально підходить для перегляду великої кількості однотипних елементів у вигляді сітки данних. По-замовчанням ширина атрибутів DataGridView є незадовільною, тож у більшості випадків значенням true ініціюється властивість AutoSizeColumnsMode елемента DataGridView. Таким чином, розміри таблиці і її контейнера підганяються для розміщення елементів у ній.

В якості альтернативи можна надати користувачам можливість змінювати розмір контейнера таблиці або в вигляді переміщуваного кордону.

## 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 стилусом 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 обліку комп’ютерів у організації: перегляд та зміна комп’ютерів. Крім цього реалізувати їх модифікацію та відстеження змін. У випадку спроби поєднання несумісних компонентів повідомляти користувача окремим діалоговим вікном. А саме спроби встановлення процесора на материнську плату з іншим типом сокету (або гнізда процесору). Також спроби вставити у материнську плату більшу кількість планок оперативної пам’яті, ніж їх є фізично і навпаки, підібрати материнську плату несумісну або з сокетом процесора, або з кількістю місць для планок оперативної пам’яті.

Програма повинна мати зрозумілий інтерфейс який надає зручне керування та детальну інформацію про комп’ютери у організації, 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

Так як облікові данні повинні бути не тільки оброблені, але й збережені для подальшої праці над ними, то для такої задачі якнайбільше підходить робота з даними на основі бази даних.

Для цього був створений універсальний для усіх баз даних типу Microsoft SQL Database клас який знаходиться у просторі імен System.Data.SqlClient.

На його основі мною був створений клас DB\_interface.

public class DB\_interface

{

private string ConnectionString;

public DB\_interface(DB\_ConstructorMode DBCM) {…}

public object[] SelectAllFrom(string Table) {…}

public object[] SelectRowsWhere(string Table, string WhereParam, string ParamValue) {…}

public bool InsertInto(string Table, string[] ColumnNames, object[] ColumnValues, bool IgnorePKErrorMessage = false) {…}

public void UpdateWhere(string Table, string[] ColumnNames, object[] ColumnNewValues, string WhereParam, string ParamValue) {…}

public int DeleteRows(string Table, string WhereParam, string ParamValue) {…}

public int ClearTable(string Table) {…}

}

Лістинг 4.1.1 – клас DB\_interface

У класу є лише одне, але надважливе поле - private string ConnectionString; Воно необхідне для підключення до поточної бази данних.

У програмі це поле автоматично генерується у конструкторі класу

public DB\_interface(DB\_ConstructorMode DBCM) {…}

Параметр типу DB\_ConstructorMode є типом enum котрий має 2 види значення:

InFolder та OFD. У програмі по-замовчанням використовується конструктор з параметром DB\_ConstructorMode.InFolder. При цьому параметрі відбувається спроба знайти у поточній директорії програми у папці DB файл бази даних CPDatabase.mdf. При невдачі виводиться діалогове вікно з повідомленням про помилку та вікно для обирання файлу бази даних вручну.

public object[] SelectAllFrom(string Table);

Цей метод повертає колекцію об’єктів що є повною вибіркою вказаної у параметрі string Table таблиці.

public object[] SelectRowsWhere(string Table, string WhereParam, string ParamValue);

Цей метод повертає колекцію об’єктів що є вибіркою вказаної у параметрі string Table таблиці з фільтром по параметру string WhereParam зі значенням string ParamValue.

public bool InsertInto(string Table, string[] ColumnNames, object[] ColumnValues, bool IgnorePKErrorMessage = false);

Цей метод дозволяє додати у вказану таблицю рядок та повертає результат чи вдалася вставка. Параметр string Table вказує назву таблиці у базі даних. Параметр string[] ColumnNames вказує назви атрибутів таблиці бази даних. Параметр object[] ColumnValues вказує значення елементів у рядку таблиці, перший елемент відповідає першій назві попереднього параметру, другий – другій, і так далі. Необов’язковий параметр bool IgnorePKErrorMessage при значенні true дозволяє не виводити повідомлення про невдачу вставки.

public void UpdateWhere(string Table, string[] ColumnNames, object[] ColumnNewValues, string WhereParam, string ParamValue);

Цей метод призначений для оновлення існуючих записів. Атрибути string Table, string[] ColumnNames та object[] ColumnNewValues мають таке ж призначення як і у попереднього методу. string WhereParam та string ParamValue виступають фільтром для оновлення певних рядків бази данних.

public int DeleteRows(string Table, string WhereParam, string ParamValue);

Цей метод призначений для видалення записів у таблиці string Table з фільтром string WhereParam, string ParamValue.

public int ClearTable(string Table);

Цей метод призначений для ПОВНОЇ очистки таблиці (видалення усіх її записів).

Після реалізації інтерфейсу між програмою і базою даних потрібно сформувати програмний каркас для комп’ютерів. Цими класами стали: Computer, CPU, Motherboard, Peripheral, RAM, Changelog, Peripheral.

Для того, щоб мати можливість універсально користуватися такими можливостями класів як запис та читання з бази даних ми повинні оголосити інтерфейс, цими інтерфейсами стали:

public interface IDB\_Write

{

bool WriteToDB(DB\_interface DBI\_obj);

}

public interface IDB\_Read

{

object[] ReadFromDB(DB\_interface DBI\_obj)

}

Лістинг 4.1.2 – Інтерфейси для читання та запису у базі даних

Всі вище перелічені класи унаслідуються та імпелентують ці інтерфейси.

Головний клас, який представляє комп’ютер, Computer має набір властивостей які представляють собою елементи комп’ютера:

public List<Changelog> ChangelogList { get; private set; }

public List<Peripheral> PeripheralsList { get; private set; }

public long PeripheralsChangelogs\_id { get; private set; }

public string InventoryNumber{…}

public CPU CurCPU{…}

public Motherboard CurMotherboard{…}

public RAM CurRAM{…}

Лістинг 4.1.3 – властивості класу Computer

Конструктор цього класу приймає лише один параметр – інвентарний номер. Після передачі інвентарного номеру у вигляді рядка, автоматично генерується ідентифікаційний номер списків периферії та журналу змін, а саме: інвентарний номер сприймається як число у 36-ричній системі і конвертується у 10-ричну методом private long Base36ToBase10(string input){…}.

## Опис алгоритму програми

Форма MainForm представляє собою головне вікно програми з інтерфейсом користувача призначена для графічного спостереження бази даних та керування даними.

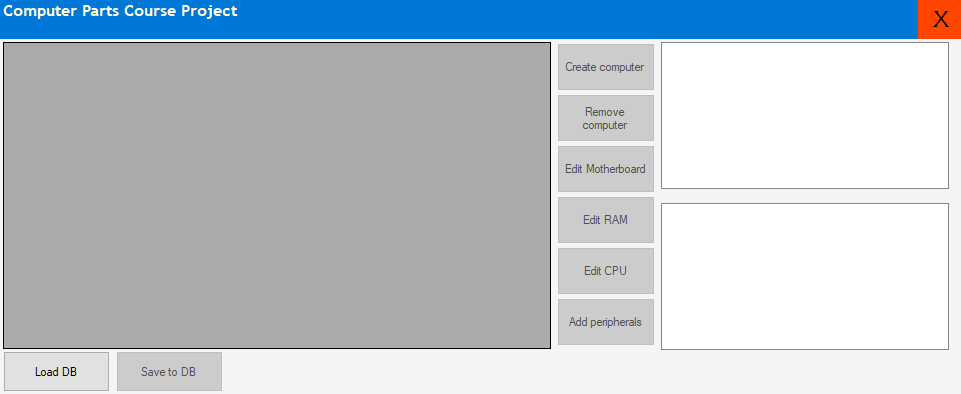
За-замовчанням при старті програми доступна лише кнопка Load DB.

Завантаження бази даних у графічний додаток відбувається за кнопкою Load DB, після натискання якої почитають бути доступні усі інші кнопки.

Далі 3 варіанти: додати новий комп’ютер, видалити комп’ютер і модифікувати існуючий.

Модифікація відбувається наступним чином: натиснути на рядок комп’ютера який збираємося модифікувати та натиснути на одну з кнопок які відповідають за відповідні елементи комп’ютера, далі інтуїтивно ввести характеристики та натиснути кнопку OK, якщо відмінити зміну – Cancel.

# ОПИС ПРОГРАМИ

Запуск програми відкриває MainForm - це головна форма, в якій відбуваються всі основні дії: відображення усіх комп’ютерів, їх модифікація та перегляд списку змін і периферії. Форма містить панелі DataGridView та ListBox у яких у компактному вигляді зібрані відомості про комп’ютери, список змін, периферію.

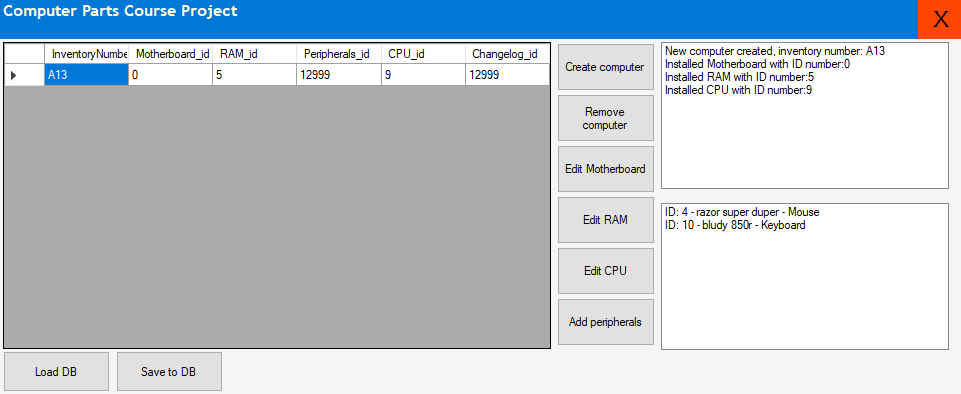


Рисунок 5.1, 5.2 – головне вікно програми.

# ІНСТРУКЦІЯ СИСТЕМНОГО ПРОГРАМІСТА

У тестуванні цієї програми повинні входити перевірки:

- працездатність усіх кнопок меню;

- 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.

Мінімальні системні вимоги:

* Операційна система Microsoft Windows 10 з встановленим .NET Framework 4.6 або вище;
* монітор з розділовою здатністю 1024х768 або вище.
* мишка;
* 512 MB RAM;
* 25 MB вільного місця на жорсткому диску.

# ПОСІБНИК ОПЕРАТОРА

Після запуску програми з’явиться головне вікно, яке відображатиме пусті списки.

* Для завантаження даних з бази даних натисніть кнопку LoadDB.
* Для створення комп’ютера натисніть кнопку Create computer та введіть його майбутній інвентарний номер (тільки латинські букви та цифри).
* Для модифікації комп’ютера натисніть на рядок комп’ютера який потрібно модифікувати, після натисніть на одну з спеціальних кнопок:
* Edit Motherboard – встановити нову материнську плату.
* Edit RAM - встановити нову оперативну пам’ять.
* Edit CPU - встановити новий процесор.
* Add Peripherals – встановити нові периферійні пристрої.
* Для видалення комп'ютера з бази натисніть на рядок потрібного комп’ютера та натисніть кнопку Remove Computer.
* Для збереження даних у базу даних натисніть кнопку Save to DB.

# ВИСНОВКИ

У процесі виконання курсового проекту було розроблено програмне забезпечення, призначене для обліку комп’ютерів у організації. Програмне забезпечення виводить в окреме вікно інформацію про всі комп’ютери у організації та за натисканням кнопки виконує опитування бази даних і оновлює дані, якщо база даних не є сумісною, або не знаходиться у базовій директорії, виводиться діалог відкриття файлу сумісних баз даних.

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

Інтерфейс програми зручний у використанні і інформативний для моніторингу цілісності комп’ютерів та їх наявності в організації. 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 розширити кількість видів сумісних баз даних.

У ході виконання курсового проекту були закріплені знання та навички, отримані при вивченні курсу "Системне програмне забезпечення". Придбано нові знання у роботі 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. Ріхтер Д. Програмування на платформі Microsoft .NET Framework 4.0 на мові C# [Текст] / Д. Ріхтер. – Санкт-Петербург: «Пітер», 2012. – 1136 с.
3. Петцольд Ч. Программирование с использованием Microsoft Windows Forms [Текст] / Ч. Петцольд – Русская Редакция, 2006. – 433с.
4. Довідник MSDN [Електронний ресурс]: довідник. – Режим доступа: [http://msdn.com](http://msdn.com/)
5. Шилдт Г. Полный справочник по С# [Текст] / Шилдт Г. – Киев : «Вильямс», 2004. – 740с.