МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
КАФЕДРА АВТОМАТИКИ ТА ТЕЛЕКОМУНІКАЦІЙ

МЕТОДИЧНІ ВКАЗІВКИ
ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ
З ДИСЦИПЛІН: «ОПЕРАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМ АВТОМАТИЗАЦІЇ» ТА «ОПЕРАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМ»
ДЛЯ СТУДЕНТІВ ДЕННОЇ ТА ЗАОЧНОЇ ФОРМ НАВЧАННЯ
ЗІ СПЕЦІАЛЬНОСТЕЙ
151 АВТОМАТИЗАЦІЯ ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ ТА
172 ТЕЛЕКОМУНІКАЦІЇ ТА РАДІОТЕХНІКА
ОСВІТНЬОГО СТУПІНЯ «БАКАЛАВР»

Покровськ
2019

УДК 621.39 : 004.4](072)
М 54

Методичні вказівки до лабораторних робіт з дисциплін: «Операційне забезпечення систем автоматизації» та «Операційне забезпечення телекомунікаційних систем» (для студентів денної та заочної форм навчання зі спеціальностей 151 – Автоматизація та комп'ютерно-інтегровані технології та 172 – Телекомунікації та радіотехніка освітнього ступеня «бакалавр») / уклад. Д.О. Жуковська. – Покровськ : ДонНТУ, 2019. – 34 с.

Методичні вказівки складені відповідно до робочих програм, розроблених для навчальних планів підготовки бакалавра та технічного фахівця в галузях системної інженерії, автоматизації та комп'ютерно-інтегрованих технологій, а також. бакалавра та технічного фахівця в галузях телекомунікаційних систем та радіотехніки. Розглядаються основні принципи побудови і функціонування сучасних операційних систем для ПЕОМ. Основну увагу приділено застосуванню операційних систем для управління обчислювальним процесом. Наведено відомості з програмування на мові С++ в операційних системах. Лабораторні роботи охоплюють найбільш важливі теми і спрямовані на формування у студентів базових навичок ефективного використання операційних систем для проектування систем автоматизації. Методичні вказівки націлені на виконання навчально-дослідних робіт студентів під час лабораторного практикуму.

Укладач: _____Жуковська Д.О., ст. викл. каф. автоматики та телекомунікацій
(підпис)

Рецензент: _____Маслова Н.О., доц., к.т.н., доц., каф. прикладної математики та
(підпис) інформатики

Відповідальний за випуск: _____Поцепаєв В.В., к.т.н., доц., завідувач кафедри автоматики та телекомунікацій

Затверджено навчально-методичним відділом ДонНТУ,
протокол № 7 від 26.02.2019 р.

Розглянуто на засіданні кафедри автоматики та телекомунікацій,
протокол № 5 від 29.01.2019р.

ЗМІСТ

ВСТУП

Метою вивчення дисциплін «Операційне забезпечення систем автоматизації» та «Операційне забезпечення телекомунікаційних систем» є вивчення основних принципів побудови сучасних операційних систем (Windows 9x, Windows NT (2000, XP), Unix, Linux, Android) і їх основних підсистем: файлові системи (FAT, NTFS, ext2fs), системи і алгоритми управління пам'яттю, системи управління процесами.

Завдання: отримання знать про основи операційних систем та вміння застосовувати отримані знання для проектування систем автоматизації та телекомунікаційних систем.

В результаті вивчення даного курсу студент повинен

знати: сучасний стан наукової дисципліни, основні функції операційних систем, машинно-незалежні властивості операційних систем, принципи побудови операційних систем, установку і супровід операційних систем;

вміти: застосовувати набуті знання для вирішення практичних завдань; використовувати сервісні засоби, що поставляються з операційними системами; встановлювати різні операційні системи; підключати до операційних систем нові сервісні засоби; забезпечувати базові налаштування операційних систем в середовищі їх функціонування, вирішувати завдання забезпечення захисту операційних систем.

У процесі вивчення навчальної дисципліни «Операційне забезпечення систем автоматизації» та «Операційне забезпечення телекомунікаційних систем» студенти виконують лабораторні роботи. Звіт до лабораторних робіт студенти оформляють на окремих аркушах форматом А4 відповідно до змісту і він повинен містити такі розділи:

– титульна сторінка (зразок оформлення титульної сторінки звіту подано у додадку А);

–тема та мета роботи;

–постановка завдання;

–опис усіх етапів виконання роботи;

–висновки за результатами роботи.

Кожна лабораторна робота, що виконана і захищена за графіком, встановленим робочою програмою курсу, оцінюється за критерієм оцінювання знань за 100-бальною шкалою.

1. ЛАБОРАТОРНА РОБОТА № 1
ПРОЦЕСИ ТА ПОТОКИ

Мета роботи: ознайомитися з теоретичними відомостями про багатопоточні програми, розробити рішення поставленого завдання на мові програмування C#.

Устаткування, комплектуючі та програмне забезпечення персональний комп'ютер, NET Framework, Microsoft Visual Studio**.**

Завдання:

1. Вибрати завдання відповідно до номеру варіанта (таб.1.1).

Таблиця 1.1 – Варіанти завдань

№ варіанта	Завдання
1	Написати програму, в якій буде виконуватись 3 дочірніх потоки. У першому потоці необхідно ввести масив значень *x* [10]. У другому потоці порахувати *y* для кожного *x* за формулою: *y=2*sin(x)/x*. У третьому потоці – вивести масиви значень *x* і y в консоль.
2	Написати програму, в якій буде виконуватись 3 дочірніх потоки. У першому потоці необхідно ввести число *x*. У другому вивести в консоль таблицю множення на це число, в третьому - таблицю додавання.
3	Написати програму, в якій буде виконуватись 2 дочірніх потоки. У першому потоці необхідно ввести масив значень *x* [10]. У другому потоці впорядкувати масив *x* по зменшенню значень і вивести їх в консоль.
4	Написати програму, в якій буде виконуватись 3 дочірніх потоки. У першому потоці необхідно ввести масив значень *x* [10] і число *k*. У другому потоці необхідно видалити всі значення масиву *x*, які менше числа *k* і присвоїти їм значення *0*. У третьому потоці необхідно вивести індекси значень масиву які були змінені.
5	Написати програму, в якій буде виконуватись 3 дочірніх потоки. У першому необхідно ввести числа *a, b, c* – сторони паралелепіпеда. У другому потоці порахувати площу поверхні і об'єм паралелепіпеда. У третьому намалювати в консолі кожну грань будь-яким з символів ASCII таблиці.
6	Написати програму, в якій створити 2 потоки. Перший потік здійснює запис в файл випадкових даних. Другий здійснює читання даних з цього файлу і виведить їх на екран.

Продовження таблиці 1.1

7	Написати програму, в якій буде виконуватись 3 дочірніх потоки. У першому ввести масив значень x [10], у другому ввести масив значень y [10]. У третьому по черзі порівняти значення двох масивів, за умови якщо значення x більше, вивести напис – «значення x більше», інакше – «значення y більше».
8	Написати програму, в якій буде виконуватись 2 дочірніх потоки. У першому потоці ввести три значення – сторони трикутника. У другому знайти площу трикутника і вивести в консоль, якщо трикутник прямокутний, то порахувати синуси, косинуси і тангенси кутів прилеглих до гіпотенузи і вивести на екран.
9	Написати програму, в якій створити масив, який заповнений 50ма випадковими елементами (цілі числа). У першому потоці порахувати суму елементів масиву, а у другому знайти максимальний елемент масиву.
10	Написати програму, в якій створити 2 потоки. У першому потоці задати масив X(10) і Y(10) а у другому написати алгоритм, який змінює послідовно місцями значення елементів X (k) і Y (k).
11	Написати програму, в якій створити 3 потоки. У першому потоці задати число N (N≤1000), в другому порахувати N^N, а в третьому – визначити кількість повторень кожної з цифр 0,1,2, ..., 9 в числі N^N.
12	Написати програму, в якій створити 2 потоки. У першому потоці задати 2 числа в двійковій системі числення і визначити ці числа в десятковій системі. В другому потоці скласти між собою задані 2 числа в двійковій системі і результат представити також в двійковій системі.

2. Розробити рішення поставленого завдання на мові програмування C #.

3. Привести код та результат виконання програми.

4. Зробити висновок.

Теоретичні відомості

Багатопотокова програма складається з двох або більше частин, які виконуються паралельно. Кожна частина такої програми називається потоком і визначає окремий шлях виконання команд. Таким чином, багатопоточна обробка є особливою формою багатозадачності.

Багатопотокове програмування спирається на цілий ряд засобів, передбачених для цієї мети в самій мові C#, а також на класи, визначені в середовищі NET Framework. Завдяки вбудованій в C# підтримки багатопотокової обробки зводяться до мінімуму або взагалі усуваються труднощів, що пов'язані з організацією багатопотокової обробки в інших мовах програмування. Як стане ясно з подальшого, підтримка в C# багатопотокової обробки чітко організована і легко сприймається.

Розрізняють два різновиди багатозадачності: на основі процесів і на основі потоків. У зв'язку з цим важливо розуміти відмінності між ними. Процес фактично являє собою виконувану програму. Тому багатозадачність на основі процесів - це засіб, завдяки якому на комп'ютері можуть паралельно виконуватися дві і більше програми. Так, багатозадачність на основі процесів дозволяє одночасно виконувати програми текстового редактора, електронних таблиць і перегляду сторінок в Інтернет. При організації багатозадачності на основі процесів програма є найменшою одиницею коду, виконання якої може координувати планувальник завдань. Потік являє собою координовану одиницю виконуваного коду. Своїм походженням цей термін зобов'язаний поняттю «потік виконання». При організації багатозадачності на основі потоків у кожного процесу повинен бути принаймні один потік, хоча їх може бути і більше. Це означає, що в одній програмі одночасно можуть вирішуватися два завдання і більше. Наприклад, текст може форматуватися в редакторі тексту одночасно з його виводом на друк, за умови, що обидва ці дії виконуються в двох окремих потоках. Відмінності в багатозадачності на основі процесів і потоків можуть бути зведені до наступного: багатозадачність на основі процесів організується для паралельного виконання програм, а багатозадачність на основі потоків – для паралельного виконання окремих частин однієї програми.

Потік може перебувати в одному з декількох станів. В цілому, потік може бути таким, що виконуються; готовим до виконання, як тільки він отримає час і ресурси ЦП; припиненим, тобто тимчасово виконуваним; відновленим в подальшому; заблокованим в очікуванні ресурсів для свого виконання; а також завершеним, коли його виконання закінчено і не може бути відновлено.

Класи, які підтримують багатопотокове програмування, визначені в просторі імен *System.Threading*. Тому будь-яка багатопоточна програма на C# включає в себе наступний рядок коду:

using System.Threading.

Система багатопотокової обробки ґрунтується на класі *Thread*, який інкапсулює потік виконання. Клас *Thread* є герметичним, тобто він не може успадковуватися. У класі *Thread* визначено ряд методів і властивостей, призначених для управління потоками. У даній лабораторній роботі будуть розглянуті найбільш часто використовувані члени даного класу.

Для створення потоку досить отримати екземпляр об'єкта типу *Thread*, тобто класу, визначеного в просторі імен *System.Threading*. Нижче наведена найпростіша форма конструктора класу *Thread*:

public Thread (ThreadStart запуск),

де запуск – це ім'я методу, що викликається з метою початку виконання потоку, а ThreadStart – делегат, в середовищі .NET Framework, як показано нижче:

public delegate void ThreadStart().

Отже, метод, що вказується в якості точки входу в потік, повинен мати тип, що повертається *void* і не приймати ніяких аргументів. Щоб потік почав своє виконання необхідно викликати метод *Start ()*. Ннайпростіша форма методу *Start ():*

public void Start ().

Також в класі *Thread* є метод *Sleep ()*, який призупиняє виконання.

public void Sleep (int time),

де *time* - час затримки в мілісекундах.

Щоб визначити чи завершено потік в класі *Thread* є властивість, яка доступна тільки для читання і називається *IsAlive*:

public bool IsAlive { get; }.

Властивість повертає значення *true*, якщо потік виконується, в іншому випадку повертає - *false*.

У класі *Thread* є властивість *Name* - ім'я потоку. При створенні потоку ім'я присвоюється за замовчуванням, але при бажанні його можна змінити:

public string Name {set; get; }.

Також в класі *Thread* є властивість *Priority*. Нижче наведена загальна форма даної властивості:

public ThreadPriority Priority {get; set; },

де *ThreadPriority* – позначає перерахування, в якому визначаються наведені нижче значення пріоритетів:

ThreadPriority.Highest;
ThreadPriority.AboveNormal;
ThreadPriority.Normal;
ThreadPriority.BelowNormal;
ThreadPriority.Lowest.

За замовчуванням для потоку встановлюється значення пріоритету *ThreadPriorityNormal.*

Приклад простої програми, що реалізує багатопотоковість:

```
using System;
using System.Threading;


namespace Threads111
{
   class Program
   {
      // Створюємо 3 об'єкти типу Thread
      public static Thread thread1;
      public static Thread thread2;
      public static Thread thread3;
```

```
        // Створюємо три змінні типу bool для перевірки потоками на їх
завершення
        public static bool check1 = false;
        public static bool check2 = false;
        public static bool check3 = false;

        static void Main(string[] args)
        {
            // Ініціалізація об'єкта типу Thread (створення потоку)
            thread1 = new Thread(FirstThread);
            // Виведення рядка в консоль
            Console.WriteLine("Введіть ім'я першого потоку:");
            // Введення імені потоку з консолі
            thread1.Name = Console.ReadLine();

            thread2 = new Thread(SecondThread);
            Console.WriteLine("Введіть ім'я другого потоку:");
            thread2.Name = Console.ReadLine();

            thread3 = new Thread(ThirdThread);
            Console.WriteLine("Введіть ім'я третього потоку:");
            thread3.Name = Console.ReadLine();
            //Запускаємо потоки на виконання
            thread1.Start();
            thread2.Start();
            thread3.Start();
            //Цикл з постумовою для перевірки потоків на завершення
            do
            {
                //Перевірка на завершення першого потоку
                if (!thread1.IsAlive && !check1)
                {
                    Console.WriteLine("Потік " + thread1.Name + "
завершений!!!");
                    check1 = true;
                }
                // Перевірка на завершення другого потоку
                if (!thread2.IsAlive && !check2)
                {
                    Console.WriteLine("Потік " + thread2.Name + "
завершений!!!");
                    check2 = true;
                }
                // Перевірка на завершення третього потоку
                if (!thread3.IsAlive && !check3)
```

```
            {
                Console.WriteLine("Потік " + thread3.Name + "
завершений!!!");
                check3 = true;
            }
            // Створюємо затримку в 500 мс
            Thread.Sleep(500);
        } while (!check1 || !check2 || !check3);

        Console.WriteLine("Основний потік завершений!!!!");
        Console.ReadKey();
    }
    //Метод,що виконується в першому потоці
    public static void FirstThread()
    {
        for (int i = 0; i < 4; i++ )
        {
            Console.WriteLine("{0} ітерація потоку " + thread1.Name, i + 1);
            Thread.Sleep(500);
        }
    }
    //Метод, що виконується в другому потоці
    public static void SecondThread()
    {
        for (int i = 0; i < 4; i++)
        {
            Console.WriteLine("{0} ітерація потоку " + thread2.Name, i + 1);
            Thread.Sleep(1000);
        }
    }
    // Метод,що виконується в третьому потоці
    public static void ThirdThread()
    {
        for (int i = 0; i < 4; i++)
        {
            Console.WriteLine("{0} ітерація потоку " + thread3.Name, i + 1);
            Thread.Sleep(2000);
        }
    }
  }
}
```

Результат роботи програми представлены на рис. 1.1.

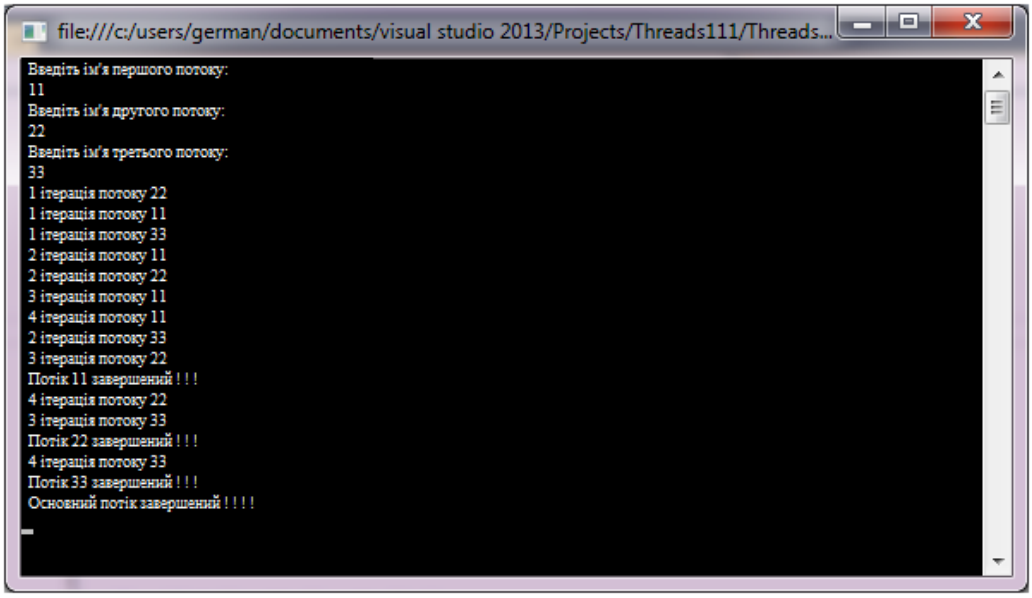

Рисунок 1.1 – Результат роботи програми

Зміст звіту

1. Мета роботи.
2. Розробка рішення поставленого завдання на мові програмування C #.
3. Код та результат виконання програми.
4. Висновки за результатами досліджень лабораторної роботи.

Контрольні питання

1. Що таке потік?
2. Як створити новий потік?
3. Чим потік відрізняється від процесу?
4. Як запустити потік після завершення іншого потоку?
5. Як присвоїти потоку ім'я?
6. Як заблокувати дані від зміни іншими потоками?
7. Як зупинити виконання потоку?

ЛІТЕРАТУРА: [1], [2], [3].

2. ЛАБОРАТОРНА РОБОТА № 2
ФАЙЛИ І ФАЙЛОВА СИСТЕМА

Мета роботи: вивчити методи управління файлами, отримати практичні навички роботи з файловими системами.

Устаткування, комплектуючі та програмне забезпечення персональний комп'ютер, операційна система сімейства Windows**.**

Завдання:

1. Перевірка роботи команд.

1.1. Прийшовши в командний рядок операційної системи Windows виконайте послідовність наступних дій:

a. змініть поточний диск:

D: - перехід на диск D

C: - перехід на диск C

b. подивіться вміст каталогу:

dir (шлях) (ім'я файлу) (/ p) (/ w)

Параметр / p задає виведення інформації в повноекранному режимі, з затримкою до тих пір, поки користувач не клацне по будь-якій клавіші. Це зручно для великих каталогів. Параметр / w задає виведення інформації тільки про імена файлів в каталозі по п'ять імен в рядку.

c. перейдіть інший каталог:

cd <ім'я каталогу>

d. створіть новий каталог:

md <ім'я каталогу>

e. видаліть каталог:

rd <ім'я каталогу>

f. створіть текстовий файл:

copy con <ім'я файлу>

Після введення цієї команди потрібно буде по черзі вводити рядки файлу. В кінці кожного рядка треба клацати клавішею Enter. А після введення останньої - одночасно натиснути Ctrl і Z, а потім Enter. Або клавішу F6, потім Enter.

g. видаліть створений файл:

del (шлях) им'я_файлу.

1.2. Представте результати роботи команд.

2. Робота з командами файлової системи.

2.1. Виконайте наступні команди для роботи з файловою системою:

a. змонтуйте/ роздемонтуйте файлову систему;

b. виведіть інформацію про підмонтовані диски;

c. створіть файлову систему (форматування);

d. розбийте диски;

e. перевірте файлові системи;

f. встановіть відповідність шляху в файловій системі заданому диску;

g. створіть, змініть або видаліть мітки тому (імені) диска;

h. відобразіть мітку тому диску і серійний номер, якщо вони існують;

i. перетворіть томи з файловою системою FAT і FAT32 в томи з файловою системою NTFS;

j. відобразіть або змініть шифрування папок і файлів на томах NTFS.

2.2. Представте результати роботи команд.

3. Локальні файлові системи.

3.1. Створіть папку в файловій системі NTFS і вкладіть в неї кілька файлів. Встановіть права доступу на папку. Які права успадкує файл в папці при встановленому прапорці «Переносити успадковані від батьківського об'єкту дозвіл на цей об'єкт»?

3.2. Встановіть спеціальні дозволи для папки. Яку область дії можна задати для цих дозволів? Перевірте можливість установки спеціальних дозволів для файлу.

3.3. Якщо деякі із дозволів призначені користувачеві особисто, а інші - як члену групи, які підсумкові дозволи отримає користувач? Переконайтеся на прикладі вашої папки. Як в такому випадку діють заборони?

3.4. Хто є власником файлу? Як і кому можна передати володіння файлом?

3.5. Розгляньте роботу з дозволом на доступ до файлу з командного рядка (команда ICACLS):

a. створіть деякий файл;

b. за допомогою команди ICACLS отримаєте файл, який містить інформацію про дозволи для цього файлу. Як утворилися подібні дозволи?

c. дайте будь-якому користувачеві дозвіл на читання файлу, а іншому відмовте в можливості запису. Перевірте, чи виконується це засобами графічного інтерфейсу.

3.6. Як передати володіння файлом іншому користувачеві? Проробіть це через графічний інтерфейс і засобами командного рядка. Чи всім користувачам можлива передача володіння?

3.7. Встановіть ліміти дискового простору, різні для різних користувачів.

3.8. Стисніть вашу папку. Проробіть це двома способами: з командного рядка і з використанням графічного інтерфейсу. Задайте в системі можливість відображення стислих файлів іншим кольором.

3.9. Як можна зашифрувати інформацію деяких файлів на диску? Перевірте, чи був створений сертифікат після шифрування файлу. Як можна зберегти сертифікат в деякому файлі, щоб мати в подальшому можливість дешифрування файлу при будь-яких умовах?

3.10. Створіть символічні і жорсткі посилання на файл і папку. У чому їх відмінність? Що таке точка підключення (з'єднання) для папки?

3.11. Перевірте можливість монтування деякого тому на папку в розділі NTFS (двома способами: з командного рядка і з використанням графічного інтерфейсу).

3.12. Перевірте можливість створення іменованих потоків в файлі. Доведіть, що одночасно можуть існувати іменовані і неіменовані потоки.

3.13. Відмовте в деякому виді доступу певному користувачеві. Призначте аудит спроб цього користувача отримати заборонений доступ. Продемонструйте, що система зафіксувала подібні спроби.

3.14. Який сервіс пропонує система для дисків? Подивіться, наскільки фрагментовані диски на вашому ПК.

4. Спільні файлові ресурси.

4.1. За допомогою вікна «Моє мережне оточення» подивіться склад вашої мережі.

4.2. Запустіть ізольоване оснащення «Загальні папки».

4.3. Виділіть свою папку в спільне використання. Як створити невидимий ресурс? Переконайтеся в його «невидимості».

4.4. Як виділити деяку папку в спільне використання з командного рядка? (див. Додаток Б).

4.5. Встановіть деякі із дозволів на доступ по мережі всім користувачам, окремому користувачеві або групі. Як взаємодіють локальні дозволі і мережні?

4.6. Підключіть папку на іншому комп'ютері в якості свого локального диску. Проробіть це з командного рядка з «невидимим» ресурсом іншого комп'ютера.

4.7. Встановіть можливість роботи з деякими файлами, доступними по мережі в автономному режимі.

4.8. Які методи синхронізації існують при роботі з автономними файлами?

5. Налаштування обробки файлів з певним розширенням.

5.1. Створіть і пропишіть в реєстрі нове розширення.

Приклад:

Створимо оброблювач довільного розширення .rrr.

У розділі HKEY_CLASSES_ROOT додамо новий розділ .rrr.

Параметр, що відповідає цьому розділу, повинен містити посилання на деякий тип файлу, наприклад rrrfile.

Створимо в галузі HKEY_CLASSES_ROOT розділ з ім'ям типу файлу rrrfile.

Створимо в розділі rrrfile підрозділ Shell (рис. 2.1).

Взаємне розташування розділів у реєстрі, що задає обробку файлів з розширенням .rrr, зіставлених з типом файлів rrfile

Рисунок 2.1 – Налаштування обробки файлів з певним розширенням

Далі в підрозділі Shell створимо підрозділи open (команда відкриття) і list (можливо будь-яка інша назва) без параметрів, а в них підрозділи command, параметрами яких є команди обробки файлів з даними розширенням відповідно на відкриття і, наприклад, перегляд. Наприклад, команда відкриття редактором Блокнот може виглядати наступним чином: notepad.exe% 1.

5.2. Через системний реєстр задайте можливість появи команди Зашифрувати / Дешифрувати. Щоб її активізувати, необхідно додати параметр EncryptionContext Menu зі значенням 1 типу REG_DWORD в розділ реєстру

HKEY_LOCAL_MACHINE \ SOFTWARE \ Microsoft \ Windows \ CurrentVersion \ Explorer \ Advanced).

5.3. Розгляньте вміст розділу HKLM \ Software \ Microsoft \ Windows \ CurrentVersion, а саме підрозділи: Run, RunOnce. Для чого зазвичай використовуються ці підрозділи?

5.4. За допомогою команди REG створіть Reg-файл, який містить інформацію про створений тип файлу. Яка структура Reg-файлу? Змініть команду обробки описаного вами розширення і імпортуйте Reg-файл назад до реєстру. Перевірте через редактор реєстру правильність ваших дій.

5.5. Задайте обробку файлу з деяким розширенням іншим способом: за допомогою команд ASSOC і FTYPE (див. Додаток Б).

6. Привести результат роботи команд.

4. Зробити висновок.

Теоретичні відомості. Операційна система Windows оснащена наступними командами для роботи з файлами:

Attrib – дозволяє переглядати, встановлювати або знімати атрибути файлу або каталогу, такі як «Тільки читання», «Архівний», «Системний» і «Прихований»;

Chdir (Cd) – виведення імені поточного каталогу або перехід в іншу папку;

Copy – копіювання одного або декількох файлів;

Del (erase) – видалення файлів;

Dir – виведення списку файлів і підкаталогів каталогу;

Fc – порівняння двох файлів і виведення відмінностей між ними;

Find – пошук заданого рядка тексту в файлі або декількох файлах;

Findstr – пошук зразків тексту в файлах з використанням регулярних виразів;

Ftype – виведення або редагування зв'язку між типом файлу і його розширенням;

Mkdir – створення папки;

Move – служить для переміщення одного або декількох файлів з одного каталогу в інший;

Rename (ren) – змінює ім'я файлу або набору файлів;

Replace – замінює файли в одному каталозі файлами з тими ж іменами з іншого каталогу;

Rmdir (rd) – видаляє каталог;

Tree – представляє графічно дерево каталогів заданого шляху або диску;

Xcopy – копіює файли і каталоги, включаючи підкаталоги.

Операційна система Windows оснащена наступними командами для роботи з файловими системами:

Chkdsk – виведення на екран звіту про стан диску;

Chkntfs – перегляд або задача планування автоматичної перевірки системи для томів файлових систем FAT, FAT32 або NTFS при запуску комп'ютера;

Cipher – відображення або зміна шифрування папок і файлів на томах NTFS;

Compact – виведення відомостей або зміна щільності файлів і каталогів в розділах NTFS;

Convert – перетворення томів з файловою системою FAT і FAT32 в томи з файловою системою NTFS;

Defrag – пошук і об'єднання фрагментованих файлів;

DiskPart – програма DiskPart.exe – це працюючий в текстовому режимі командний інтерпретатор, який дозволяє управляти об'єктами (дисками, розділами чи томами) за допомогою сценаріїв або команд, що вводяться з командного рядка;

Format – форматування диску;

Fsutil (підтримуються тільки з версії Windows 5.1) - є службовою програмою командного рядка, яка використовується для виконання пов'язаних задач файлових систем FAT і NTFS.

Основні підкоманди Windows для роботи з файловими системами:

behavior – запитує, змінює, включає або відключає налаштування для створення імен файлів з довжиною 8,3 символу;

dirty – запит установки «брудного» біту тому (якщо встановлено «брудний» біт, то autochk автоматично перевірить том на наявність помилок при наступному перезавантаженні комп'ютера);

file – пошук файлу за ідентифікатором безпеки;

fsinfo – перераховує всі диски, запитує тип диска, відомості про томи, спеціальні відомості про томи NTFS або статистику файлової системи;

hardlink – створює жорстке посилання;

Objectid – управляє ідентифікаторами об'єктів, які використовуються Windows для відстеження об'єктів, таких як файли і каталоги;

quota – управляє дисковими квотами в томах NTFS;

reparsepoint – робота з точками монтування;

sparse – управління розрідженими файлами;

usn – управління журналом змін, в якому зберігається архів всіх змін файлів в томі;

volume – демонтування тому і відображення вільного місця на диску;

Label – служить для створення, зміни або видалення мітки тому (тобто імені) диску;

Mountvol – служить для створення, видалення та отримання списку точок підключення тому;

Subst – встановлює відповідність шляху в файловій системі заданому диску;

Vol – відображає мітку тому диска і серійний номер, якщо вони існують.

Зміст звіту

1. Мета роботи.
2. Розробка рішення поставленого завдання.
3. Скріншоти результату виконання кожного завдання.
4. Висновки за результатами досліджень лабораторної роботи.

Контрольні питання

1. Які бувають атрибути файлу?
2. Як в операційних системах Unix класу переглянути зміст поточного каталогу?
3. Як в операційних системах DOS / Windows класу змінити ім'я файлу?
4. Які файлові системи підтримуються в операційній системі Windows XP?
5. Яка максимальна довжина файлу в файловій системі FAT32?
6. Чи підтримують файлові системи FAT32 і NTFS операційні системи Unix класу?
7. Сумісності файлових систем NTFS і FAT.
8. Обмеження файлових систем NTFS і FAT.

ЛІТЕРАТУРА: [1], [4], [5].

3. ЛАБОРАТОРНА РОБОТА № 3
ЗАХИСНІ МЕХАНІЗМИ ОПЕРАЦІЙНИХ СИСТЕМ

Мета роботи: використання інструментальних засобів програмного середовища Microsoft Visual C# для проектування WINDOWS FORMS.

Устаткування, комплектуючі та програмне забезпечення персональний комп'ютер, NET Framework, Microsoft Visual Studio, Window Forms.

Завдання:

1. Вибрати завдання відповідно до номеру варіанта.

Таблиця 3.1 – Варіанти завдань

№ варіанта	Завдання
1	Зробити на формі додаткову кнопку, яка б зберігала пароль в файл типу ".rtf".
2	Програмно зробити обмеження на довжину пароля. При виході за допустимі межі видавати вікно про помилку типу «Пароль виходить за допустимі межі».
3	Зробити формат пароля виду «*** - *** - *** ...», тобто після кожного 3-го символу пароля ставити тире.
4	Зробити обмеження на повторення символів в паролі. Додатково до цього заблокувати зміну розмірів вікна форми. Кнопку «На весь екран» так само заблокувати.
5	Зробити так, щоб програми сама брала довжину пароля з файлу. Можливість введення довжини вручну заблокувати, при цьому NumericUpDown з форми не видаляти.
6	Зробити додатковий textBox в який будуть зберігатись всі згенеровані паролі. Кожен новий згенерований пароль повинен бути з нового рядка.
7	Зробити 2 додаткових поля Label, в одному з яких при запуску генератора буде вказуватися актуальна дата, а в другому - системний час. Все в звичному форматі.
8	Сортувати згенерований пароль, щоб спочатку були цифри, потім великі літери, а в кінці малі. Спеціальні символи залишити в кінці.
9	Додати додаткову гілку символів букв кирилиці. Коди букв дивитись в ASCII таблиці.
10	При створеному паролі під час натискання на кнопку «Закрити» запускати вікно, що пропонує зберегти пароль.

2. Розробити рішення поставленого завдання.

3. Привести код та результат програми. Перевірити працездатність основної та доповненої програми.

4. Зробити висновок.

Теоретичні відомості. Для створення графічних інтерфейсів за допомогою платформи .NET застосовуються різні технології - Window Forms, WPF, додатки для магазину Windows Store (для ОС Windows 8 / 8.1 / 10). Однак найбільш простою і зручною платформою досі залишається Window Forms або форми.

Для створення графічного проекту використовується середовище розробки Visual Studio. Після інсталяції середовища та всіх його компонентів, запустимо Visual Studio і створимо проект графічного додатку. Для цього виберемо пункт File (Файл) і виберемо New -> Project (Створити -> Проект). Після цього перед нами відкриється діалогове вікно створення нового проекту (рис. 3.1).

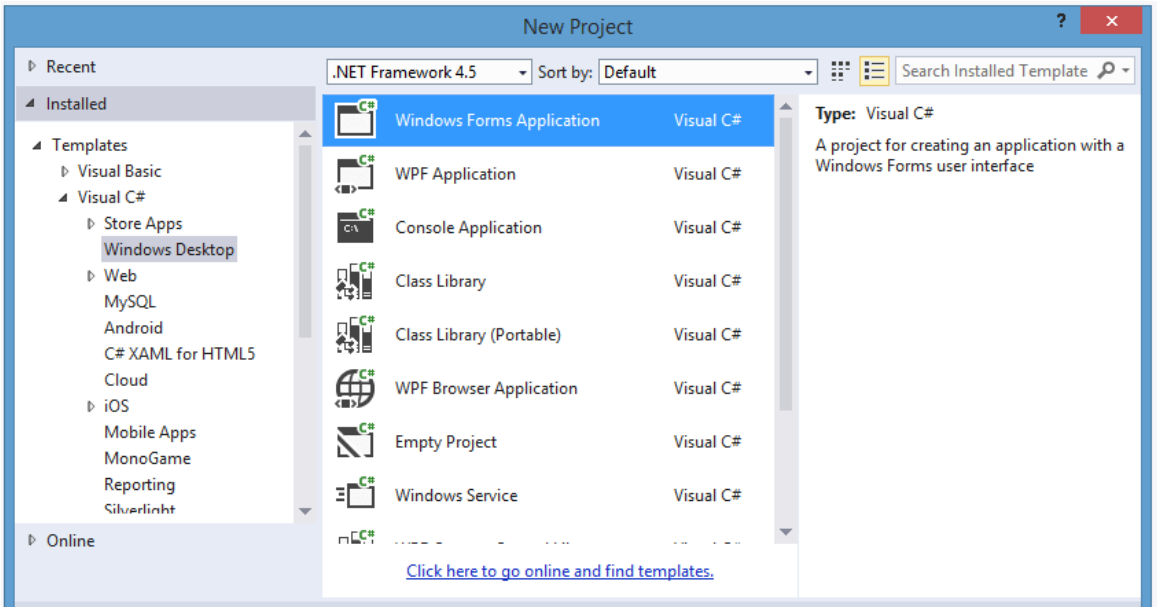

Рисунок 3.1 – Створення проекту «Додаток WINDOWS FORMS»

У лівій колонці виберемо Windows Desktop, а в центральній частині серед типів проектів - тип Windows Forms Application і дамо йому якесь ім'я в полі знизу. Наприклад, назвемо його HelloApp. Після цього натискаємо OK.

Після цього Visual Studio відкриє наш проект зі створеними за замовчуванням файлами (рис. 3.2).

Положення зазначених областей можуть відрізнятися від представлених на рис. 3.2 і виставляються користувачем на власний розсуд.

Графічний дизайнер форми – у цій області візуально наносяться елементи і створюється призначений для користувача інтерфейс.

Структура проекту – всі елементи, доступні формі (кнопки, області тексту і т.д.).

Вікно характеристик – в цій області описуються і змінюються всі події і властивості елементів форми. Під властивостями мається на увазі назва елемента, колір і т.д., під подіями – подвійне натискання, утримування кнопки миші і т.д.

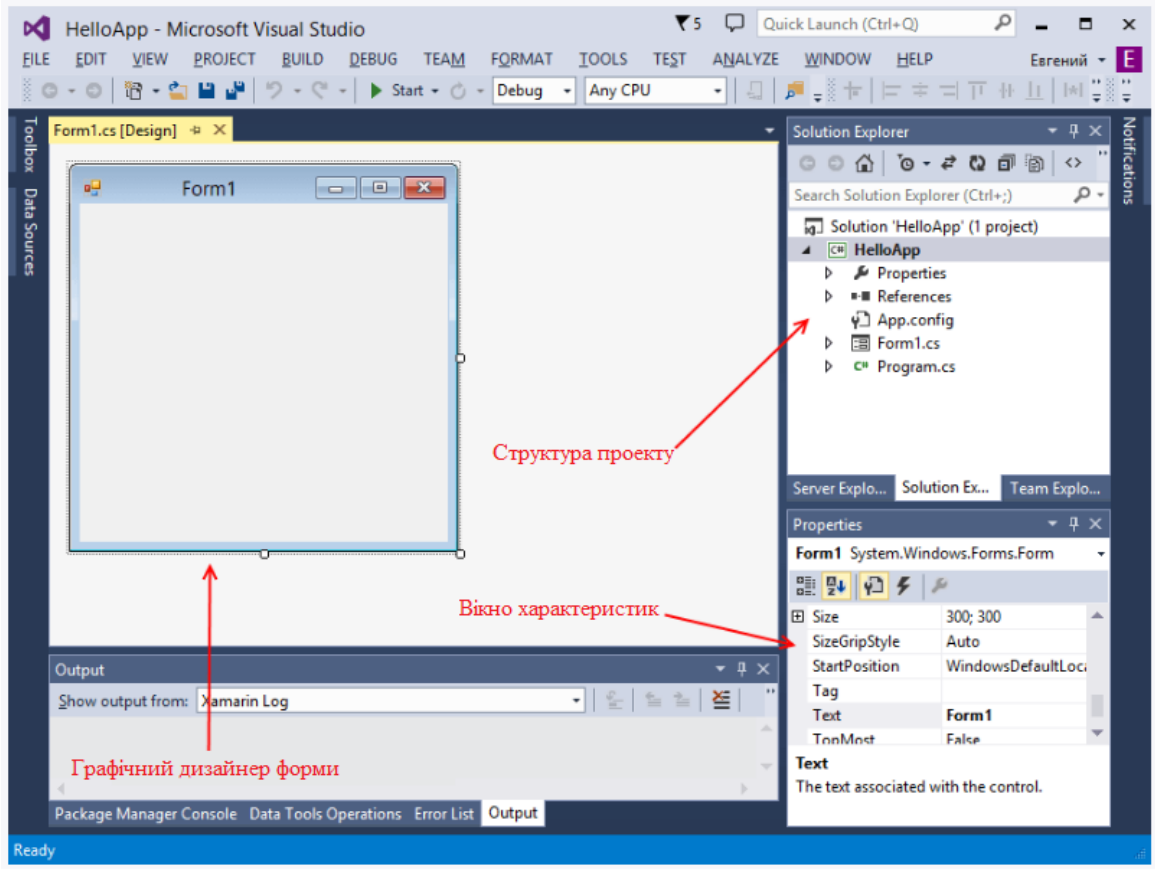

Рисунок 3.2 – Робоче середовище розробки

В ході даної лабораторної необхідно розробити генератор паролів з цифр зазначеної довжини. Для цього необхідно перетягнути необхідні елементи з області «Панель елементів» в «Конструктор». Для цього знадобляться такі елементи: Lable (для підпису поля), NumericUpDown (для вказівки довжини пароля), Button (для генерації пароля) і TextBox (для відображення пароля) (рис. 3.3).

Перейменуємо підпис і кнопку в області «Властивості», попередньо обравши необхідний елемент на формі, для роботи з ним. З кнопкою чинимо аналогічно.

Для створення обробника події для елемента форми необхідно два рази клікнути по необхідному елементу. Після кліку відкривається вікно з кодом, в якому створився метод для обробки події при натисканні на кнопку (рис. 3.4).

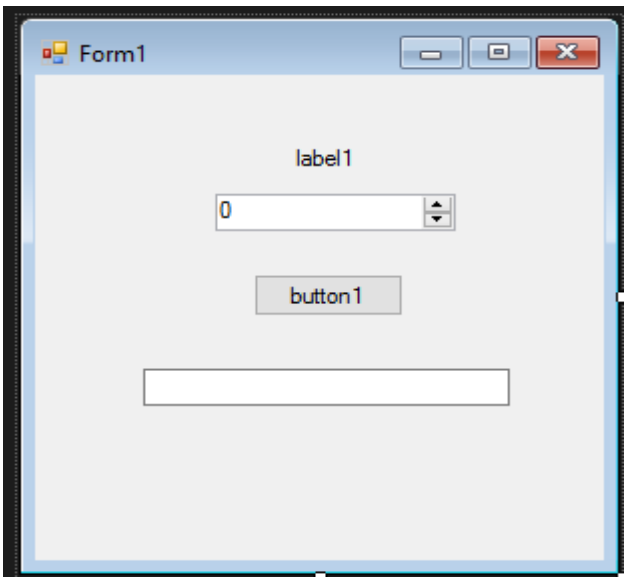

Рисунок 3.3 – Приклад форми

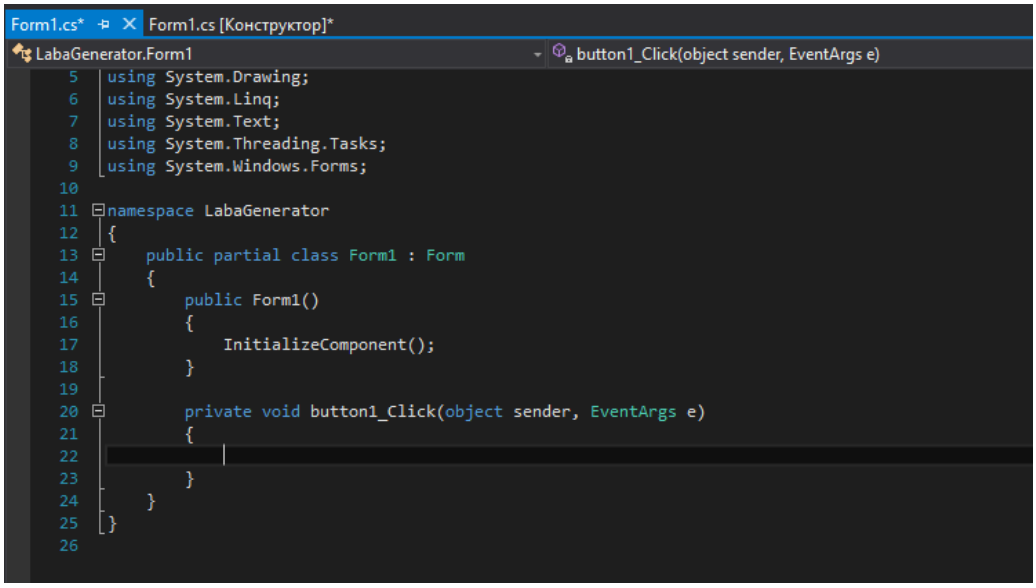

Рисунок 3.4 – Владка обробки події

Далі необхідно написати безпосередньо сам код реалізації обробника для кнопки (рис. 3.5).

```
public partial class Form1 : Form
{

    Random rnd; // Створюємо змінну rnd класу Random, для роботи з випадковими величинами

    public Form1()  // Конструктор класу форми
    {
        rnd = new Random(); //  Ініціалізуємо змінну rnd
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e) // Метод обробки події для кнопки
    {

        string password = ""; //  Створюємо "пусту" змінну рядкового типу для відображення
                              //  її в подальшому в полі textBox

        for (int i = 0; i < numericUpDown1.Value; i++) //  Створюємо цикл, в якому будуть
                        //  додаватись випадкові числа по одному до довжини, що вказана в
                        //  полі numericUpDown1
        {
            password += rnd.Next(10).ToString();   // Змінній присвоюємо випадкове
            //  згенероване значення і додаємо до наступного, що дає змогу нам
            //  "зібрати" певний пароль заданої довжини
        }
        textBox1.Text = Convert.ToString(password); //  Відображаємо пароль в текстовому полі
        MessageBox.Show("    Ваш пароль успішно згенеровано    "); // Виведення вікна про успішну генерацію
    }
}
```

Рисунок 3.5 – Описання роботи програми

Запустимо програму і переконаємося в її працездатності (рис. 3.6).

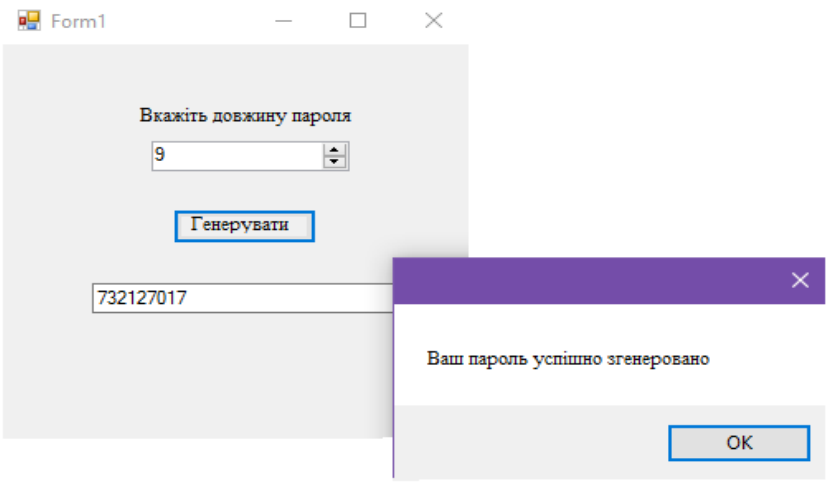

Рисунок 3.6 – Перевірка працездатності програми

Доповнимо генератор додатковими функціями, а саме вибір компонентів, з яких буде складатися пароль (рис. 3.7).

```
char[] spSymbols = new char[] { '!', ')', '(', '&' , ';', '(', '@', '#' }; // Масив спец. символів
Random rnd; // Створюємо змінну rnd класу Random, для роботи з випадковими значеннями

    public Form1()  // Конструктор класу форми
    {
        rnd = new Random(); // Ініціалізуємо змінну rnd
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e) // Метод обробки подій для кнопки
    {
        if (checkedListBox1.CheckedItems.Count == 0) // Перевірка встановлення міток
            return;  // Якщо умова виконується, то програма нічого не повертає
        string password = "";
        for (int i = 0; i < numericUpDown1.Value; i++)
        {
            int n = rnd.Next(checkedListBox1.CheckedItems.Count); // Змінна, що відповідає за
            // один символ пароля
            string s = checkedListBox1.CheckedItems[n].ToString();
            switch (s) //  Перебір вибраних міток
            {
                case "Цифри": password += rnd.Next(10).ToString();
                    break;
                case "  Великі літери  ": password += Convert.ToChar(rnd.Next(65, 88));
                    break;
                case "  Малі літери  ": password += Convert.ToChar(rnd.Next(97, 122));
                    break;
                default:
                    password += spSymbols[rnd.Next(spSymbols.Length)];
                    break;
            }
        }
        textBox1.Text = Convert.ToString(password); // Введення пароля в textbox

        if (numericUpDown1.Value != 0)
        MessageBox.Show("    Ваш пароль успішно згенеровано    "); // Виведення вікна про успішну
    }                                                               //  генерацію

    private void button2_Click(object sender, EventArgs e)
    {
        textBox1.Clear();
    }
}
```

Рисунок 3.7 – Код виконання програми

Запустимо програму і переконаємося в її працездатності (рис. 3.8).

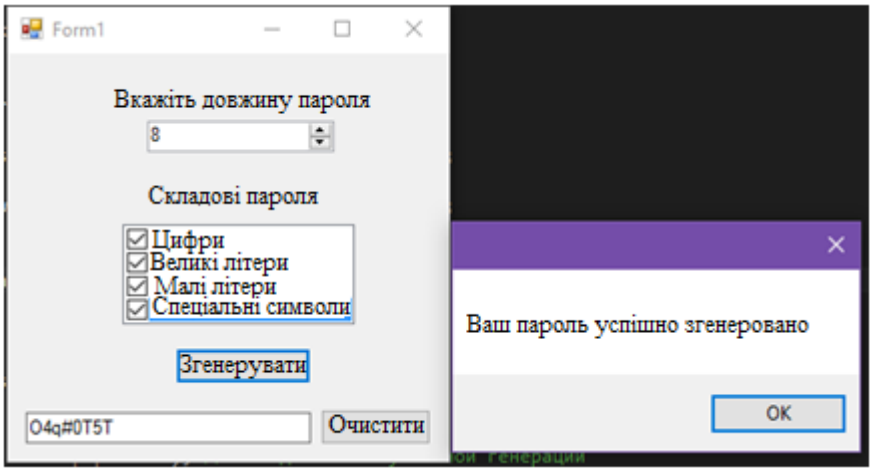

Рисунок 3.8 – Перевірка працездатності доповненої програми

Зміст звіту

1. Мета роботи.

2. Розробка рішення поставленого завдання.

3. Код та результат виконання програми.

4. Скріншоти перевірки працездатності основної та доповненої програми.

5. Висновки за результатами досліджень лабораторної роботи.

Контрольні питання

1. Які існують загрози безпеки програмних систем? Дайте їх класифікацію.

2. Які існують способи атаки зсередини системи?

3. Які існують способи атаки на систему зовні?

4. Охарактеризуйте шкідливе програмне забезпечення.

5. Які відомі методи виявлення вторгнень? Аудит і його можливості.

6. Що таке ключі шифрування?

7. Наведіть приклад несиметричного шифрування.

8. У чому відмінність аутентифікації і авторизації?

ЛІТЕРАТУРА: [1], [6], [8].

4. ЛАБОРАТОРНА РОБОТА № 4
ВІРТУАЛІЗАЦІЯ ТА ХМАРНІ ОБЧИСЛЕННЯ

Мета роботи: ознайомитися з можливістю установки операційної системи за допомогою використання середовищаVirtualBox.

Устаткування, комплектуючі та програмне забезпечення персональний комп'ютер, Oracle VM VirtualBox.

Завдання:

1. Створити віртуальну машину за допомогою Oracle VM VirtualBox (на вибір).
2. Для створеної віртуальної машини створити обкладинку робочого столу.
3. Створити 5-7 різних папок.
4. В кожну папку завантажити файли, що мають різне розширення (в кожній папці має бути по 5-10 файлів).
5. Налащтувати параметри входу в операційну систему, створити користувача та унікальний пароль для авторизації.
6. Зробити висновок.

Теоретичні відомості. Для виконання роботи необхідно завантажити та встановити з офіційного сайту дистрибутив Oracle VM VirtualBox за адресою:

https://www.virtualbox.org/wiki/Downloads/

Oracle VM VirtualBox – це програма для віртуалізації крос-платформних програм. Що це означає? По-перше, вона встановлюється на існуючі комп'ютери на основі Intel або AMD. Операційні системи Windows, Mac OS X, Linux або Oracle Solaris. По-друге, вона розширює можливості існуючого комп'ютера, так що він може запускати кілька операційних систем, всередині декількох віртуальних машин, одночасно. Наприклад, можна запускати Windows і Linux на Mac, Windows Server 2016 на сервері Linux, Linux на комп'ютері з ОС Windows і так далі поряд з існуючими програмами.

VirtualBox має єдиний практичний дисковий простір і пам'ять.

Oracle VM VirtualBox є оманливо простою, але дуже потужною. Вона може працювати всюди, від невеликих вбудованих систем або машин настільного класу, до розгортання центрів обробки даних і навіть хмарних середовищ.

Після завантаження першим кроком є запуск VirtualBox. У разі правильних налаштувань має з'явитися наступне вікно (рис. 4.1):

Рис.4.1 містить наступні елементи:

- вікно відображення вже встановлених віртуальних машин;
- кнопка створення нової віртуальної машини;
- створення віртуальної машини

Так як віртуальні машини при першому запуску не встановлені, то в панелі відображається вітальне повідомлення.

Рисунок 4.1 – Вікно з заголовком "Oracle VM VirtualBox Manager" після першого запуску Oracle VM VirtualBox

Після натискання кнопки «Створити» з’являється наступне вікно (рис. 4.2):

Рисунок 4.2 – Діалогове вікно створення віртуальної машини

У рядках:

Ім'я: вказується назва для віртуальної машини.

Тип: вибирається зі списку тип Операційної системи, якщо обраної системи немає в списку, то обирається "Other".

Версія: вибирається із запропонованого списку і повинна точно відповідати наявній дистрибутива операційної системи.

У наступному вікні пропозиція вибору розміру оперативної пам'яті, яку VirtualBox виділятиме віртуальній машині при кожному запуску. Обсяг пам'яті вказаний тут буде не доступний для хоста і виділений тільки гостьовій операційній системі.

Пам'ять, яка надається віртуальній машині, не буде доступною для хост-операційної системи під час роботи віртуальної машини, тому не вказуйте більше, ніж ви можете заощадити. Наприклад, якщо на хост-машині встановлено 1 Гб оперативної пам'яті і вводиться 512 МБ як кількість оперативної пам'яті для певної віртуальної машини, під час роботи віртуальної машини у вас залишиться лише 512 МБ. Якщо запускається дві віртуальні машини одночасно, ще більше пам'яті буде виділено для другої віртуальної машини, яка, можливо, навіть не зможе почати працювати, якщо ця пам'ять буде недоступною. З іншого боку, ви повинні вказати стільки, скільки потрібно вашій гостьовій операційній системі і вашим програмам для роботи належним чином.

Гості Windows XP вимагатимуть, щонайменше, кілька сотень МБ оперативної пам'яті, щоб працювати належним чином. Windows Vista не встановлюватиметься менше ніж на 512 МБ. Якщо ви хочете працювати з графікою програми у вашій віртуальній машині, вам може знадобитися ще більше оперативної пам'яті.

Як правило, якщо у вашому головному комп'ютері є 1 ГБ оперативної пам'яті або більше, безпечно виділити 512 МБ кожній віртуальній машині. У будь-якому випадку переконайтеся, що у вас завжди є щонайменше 256-512 МБ оперативної пам'яті, залишені для вашої операційної системи.

Як і в інших налаштуваннях, можна змінити це налаштування пізніше, після створення віртуальної машини.

У наступному вікні необхідно підключити віртуальний жорсткий диск (кількість пам'яті, що виділяється для віртуальної операційної системи). При цьому можна використовувати існуючий віртуальний жорсткий диск для раніше створеної віртуальної машини.

Рисунок 4.3 – Вікно для підключення жорсткого диска для віртуальної машини

Наступним кроком обирається «Створити новий віртуальний жорсткий диск», після вибору з'явиться вікно:

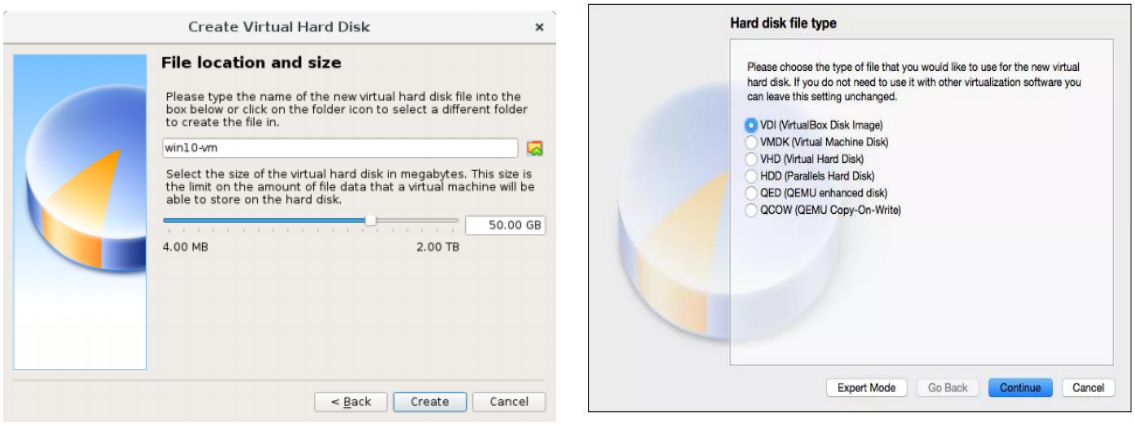

Рисунок 4.4 – Вікно створення віртуального жорсткого диска

Наступним кроком є вибір між віртуальним файлом динамічно виділеного або фіксованого розміру. Необхідно обрати "динамічно виділений", який є типовим. У посібнику користувача VirtualBox говориться, що віртуальний диск з фіксованим розміром забезпечує кращу продуктивність файлової системи, але ми не знаємо заздалегідь, скільки місця на диску нам дійсно потрібно, тому легше використовувати налаштування за замовчуванням.

Рисунок 4.5 – Вікно вибору формату зберігання

Вибираємо динамічний режим і натискаємо "Next". Далі необхідно запустити віртуальну машину: Є такі методи запуску віртуальної машини:

- двічі натиснути мишею на її найменуванні в списку вікна менеджера див. рис. 4.1;
- вибрати її в списку вікна менеджера і натиснути кнопку "Запустити".

Ця команда відкриє нове вікно з віртуальною машиною, в якому з'явиться вікно "Виберіть завантажувальний диск" і віртуальна машина почне завантаження операційної системи, яку обрано для неї.

Після встановлення операційної системи отримуємо наступне вікно (рис. 4.6).

Рисунок 4.6 – Вікно з заголовком "Oracle VM VirtualBox Manager" після встановлення віртуальної машини

Тепер ми бачимо, що у VirtualBox є віртуальна машина Debian 8, яка знаходиться в стані Off Off.

Натискаємо кнопку «Налаштувати».

У Вкладці "Загальні. Додатково":

«Загальний буфер обміну» і "Drag'n'Drop" можуть приймати чотири значення: «вимкнено», «тільки з гостьової ОС в основну», «тільки з основної ОС в гостьову», «двонаправлений», які визначають, як буде працювати буфер обміну між вашою host-системою і віртуальною машиною.

У вкладці "Загальні папки" на панелі можна підключити папки хоста з регульованими параметрами доступу.

Зміст звіту

1. Мета роботи.
2. Розробка рішення поставленого завдання.
3. Покрокове створення віртуальної операційної системи.
4. Скріншоти перевірки завантаження створеної віртуальної операційної системи.
5. Висновки за результатами досліджень лабораторної роботи.

Контрольні питання

1. Які існують технології ефективної віртуалізації? Навести приклади віртуальних пристроїв.
2. Описати, вимоги, що застосовуються до віртуалізації.
3. Як відбувається віртуалізація пам'яті і вводу-виводу?
4. В чому полягає клонування віртуальних машин?
5. Перерахувати модулі драйверів VirtualBox Oracle VM.
6. Навести приклад інсталяції віртуальної машини без нагляду?
7. Дати характеристику менеджеру віртуальних медіа.
8. Які спеціальні режими запису зображень існують при створенні віртуальних машин?

ЛІТЕРАТУРА: [1], [9], [10].

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Таненбаум, Э. Современные операционные системы / Э. Таненбаум, Х. Бос. – 4-е изд. – СПб. : Питер, 2015. – 1120 с. – («Классика computer science»).
2. Потоки и процессы [Электронный ресурс] : гл.13. – Режим доступа : http://www.4stud.info/.
3. Вандервуд, Д. Шаблоны С++. Справочник разработчика / Д. Вандервуд, Н. Джосаттис и др. – 2-е изд. – СПб. : Альфа-книга, 2018. – 848 с.
4. Шеховцов, В.А. Операційні системи : підручник / В.А. Шеховцов. – К. : Видавнича група BHV, 2005. – 576 с.: іл.
5. Партыка, Т. Операционные системы, среды и оболочки : учеб. пособие / Т. Партыка, И. Попов. – М. : ФОРУМ ; ИНФРА-М, 2009. – 400 с. – («Профессиональное образование»).
6. Гордеев, А. Операционные системы / А. Гордеев. – 2-е изд. – СПб. : Питер, 2009. – 416 с.
7. Операційні системи та системи програмування : навч. посіб. / В.П. Харченко, Є.А. Знаковська, В.А. Бородін. – К. : Вид-во Нац. авіац. ун-ту «НАУ-друк», 2012.– 360 с.
8. Практикум по операционным системам / Э.С. Спиридонов, М.Д. Рукин, Н.П. Григорьев и др. – М. : ЛИБРОКОМ, 2015. – 350 с.
9. Download VirtualBox [Електронний ресурс]. – Режим доступу : https://www.virtualbox.org/wiki/Downloads/.
10. Олифер, В.Г. Сетевые операционные системы / В.Г. Олифер, Н.А. Олифер. – 2-е изд. – СПб. : Питер, 2011. – 440 с.
11. CulOnline [Електронний ресурс]. – Режим доступу : http://www.culonline.com.ua/index.php.

ДОДАТОК А
ЗРАЗОК ОФОРМЛЕННЯ ТИТУЛЬНОЇ СТОРІНКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
КАФЕДРА АВТОМАТИКИ ТА ТЕЛЕКОМУНІКАЦІЙ

ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ № __

з навчальної дисципліни

«Операційне забезпечення систем автоматизації» («Операційне забезпечення телекомунікаційних систем»)

на тему: «_____»

Виконав:

студент(ка) групи _____

_____

Перевірив:

_____

_____

Покровськ 2019

ДОДАТОК Б
ДОДАТКОВІ КОМАНДИ РОБОТИ З ФАЙЛАМИ В ОПЕРАЦІЙНІЙ СИСТЕМІ WINDOWS. КОМАНДИ NET

1. Команда виділення ресурсів в спільне використання *NET SHARE*.

Синтаксис команди:

Net share ім'я_ресурсу = диск: шлях [/ USERS: число | / UNLIMITED] [/ REMARK: "текст"]

Видалення загального ресурсу (локально нічого не видаляється):

Net share {ім'я_ресурсу | ім'я_пристрою | диск: шлях} / DELETE

де *ім'я_ресурсу* - ім'я, присвоєне загальному ресурсу;

диск: шлях - місце розташування локального ресурсу, який необхідно виділити в спільне використання;

[/ USERS: число | / UNLIMITED] - необов'язковий параметр, що задає число користувачів, які можуть одночасно звернутися до загального ресурсу;

[/ REMARK: "текст"] - необов'язковий параметр, що містить деякий коментар.

2. Команда створення мережного диску *NET USE*.

Синтаксис команди:

*NET USE ім'я_пристрою] [\\ ім'я_комп'ютера \ ім'я_ресурсу [\ том] [пароль | *]] [/ DELETE]*

де *ім'я_пристрою* - ім'я мережного диску;

\\ ім'я_комп'ютера \ ім'я_ресурсу - ім'я мережного ресурсу, що підключається в якості диска на ваш комп'ютер;

/ DELETE - відключення мережного диску.

3. Перегляд ресурсів, що розділяються для деякого комп'ютера - команда *NET VIEW*.

Синтаксис команди:

Net view \\ ім'я_комп'ютера

4.Команди роботи з розширеннями файлів.

Перегляд і зміна зіставлень файлів, команда *ASSOC*:

ASSOC [.рсш [= [типФайла]]]

де *.рсш* - розширення імені файлу, зіставлене типу файлів;

типФайла - тип файлів, що зіставляються розширенню імені файлів.

Примітки:

Команда *ASSOC* без параметрів виводить список зіставлень типів файлів.

Якщо вказано тільки розширення імені файлу, виводиться зіставлений тип файлів для розширення.

Якщо після знака рівності не вказано тип файлів, команда видалить поточний зіставлення для цього розширення.

5. Перегляд і зміна типів файлів, зіставлених з розширенням імен файлів *FTYPE.*

Синтаксис команди:

FTYPE [типФайлів [= [командниЙРядокВідкриття]]]

де *типФайлів* - тип файлів для перегляду або зміни;

командниЙРядокВідриття - команда відкриття для використання при запуску файлів зазначеного типу.

Примітки:

Команда *FTYPE* без параметрів виводить поточний список типів файлів, для яких визначені командні рядки відкриття.

Якщо вказано тільки тип файлу, *FTYPE* виводить командний рядок відкриття для файлів цього типу.

Якщо після знака рівності не зазначено рядок відкриття, *FTYPE* видалить поточне зіставлення для зазначеного типу файлів.