КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет кібернетики

Кафедра інформаційних систем

ЛАБОРАТОРНІ РОБОТИ З ТЕСТУВАННЯ

до курсу

ФОРМАЛЬНІ МЕТОДИ ПОБУДОВИ ПРОГРАМ:
ТЕСТУВАННЯ ТА ОЦІНКА НАДІЙНОСТІ

Викладач*: *Слабоспицька Ольга Олександрівна

***р.т.* 526 45 79**

ols.07@mail.ru

Київ-2014

АНОТАЦІЯ

Лабораторні роботи (ЛР) адресовані студентам освітньо-кваліфікаційного рівня «*магістр*» галузі знань «інформатика та обчислювана техніка» з напряму підготовки 0501 «інформатика та обчислювальна техніка», *спеціальності* - 8.05010301 «програмне забезпечення систем».

Роботи підтримують теми курсу з тестування (за його актуальною робочою програмою), практично найважливіші для успішного розроблення високоякісного програмного забезпечення. Вони призначені для систематизації і покращення засвоєння студентами відповідного лекційного матеріалу, а також формування в них практичних навичок:

- відлагодження програм (ЛР №1; Лекція 1);
- їх структурного (ЛР №1,2; Лекція 3) та функціонального тестування (ЛР №3-5; Лекція 3);
- системного й дослідницького тестування Windows-форм та Web-застосунків з акцентом на тестуванні інтерфейсу користувача (ЛР №4,5; Лекція 4);
- виявлення дефектів у коді та їх документування в середовищі вільно доступних програмних засобів відповідного призначення (ЛР №6; Лекція 4);
- кількісної оцінки якості застосунку за документованими результатами його системного тестування (ЛР №6; Лекція 1).

Завдання з відлагодження й структурного тестування передбачають використання парадигми об’єктно-орієнтованого програмування для створення тестованого застосунку, зокрема мови Сі# та середовища розробки MS Visual Studio.Для автоматизованого документування дефектів обрано систему Mantis.

Враховуючи фактичні дані щодо розміру студентських груп впродовж 2009-2014рр., завдання розраховані на групу чисельністю 14 осіб.

ЗМІСТ

Лабораторна робота № 1. Відлагодження консольних програм

Мета роботи. Ознайомлення з відладчиком MS Visual Studio, встановленого на учбових комп'ютерах факультету кібернетики КНУ ім. Т.Шевченка. Встановлення контрольних точок. Покрокове виконання коду. Обробка виключних ситуацій.

Теоретичні відомості.

Модульне тестування може виконуватися програмістом і чергується з відлагодженням.

Процес включає такі кроки:

1. аналіз специфікацій і/або структури програми
2. розробка тестів (вхідних даних та очікуваних результатів)
3. виконання тестів
4. порівняння результатів виконання тестів з очікуваними:

– у разі розбіжності – локалізація та усунення помилки і повторне тестування виправленого коду.

– у разі успіху – завершення тестування або розробка нових тестів.

Основними методами відлагодження (локалізації помилки) в програмі є:

1) статичний аналіз програми (desk checking);

2) вставлення операторів протоколювання (друку) проміжних результатів;

3) покрокове виконання програми (single-step running);

4) виконання з точками зупинки (breakpoints).

1. Статичний аналіз програми

Виконується візуальним аналізом тексту ("прокручуванням"), обчисленням "вручну" результатів та їх порівнянням з очікуваними.

2. Вставка операторів протоколювання (друку) проміжних результатів

// **Метод обчислення ступеню числа x^n (для n>0)**

```
static public double Power(double x, int n)
{
   double z=1;
   for (int i=1;n>=i;i++)
   {
         z = z*x;
          Console.WriteLine("i = {0} z = {1}",i,z);
    }
    return z;
}
```

Зауваження. Оператори протоколювання в тіло циклу слід включати у крайньому випадку і при невеликих значеннях змінної циклу.

3. Покрокове виконання програми (single-step running).

В середовищі Microsoft Visual Studio.NET наявні чотири команди покрокового виконання (трасування), доступні з меню **Debug**:

1) ***Step Into*** **(F11)** – забезпечує послідовне покрокове виконання коду (включно з переходом до методів, які викликаються).

2) ***Step Over*** **(F10)** – відпрацьовує як і ***Step Into***, але без переходу до методів, які викликаються.

3) ***Step Out*** **(Shift+F11) –** забезпечує виконання всіх рядків, що залишилися, і дозволяє виконати швидкий перехід в останню точку виклику.

4) ***Run to Cursor*** **(Ctrl+F10)** – виконання коду без зупинки від поточного рядка до позиції курсору.

4. Виконання з точками зупинки (***breakpoints***).

Контрольна точка (breakpoint) – точка програми, яка при її досягненні посилає відладчику сигнал. За цим сигналом або тимчасово припиняється виконання програми, або ж запускається програма-“агент”, яка фіксує стан змінних і областей пам’яті в цей момент.

Коли виконання в контрольній точці зупиняється, програма переходить в режим зупинки (break mode). Вхід в режим зупинки не перериває і не закінчує виконання програми і дозволяє аналізувати стан окремих змінних або структур даних.

Для встановлення контрольної точки потрібно клацнути мишкою на вертикальній смузі зліва від потрібного рядка. Для продовження роботи – вибрати команду **Continue** пункту меню **Debug**.

Завдання для самостійного виконання

1. Розробити тести для виявлення дефектів у наданій програмі (підібрати вхідні дані та очікувані результати).
2. Виконати тест(и), зафіксувавши розбіжність очікуваних та отриманих результатів
3. Виявити дефекти за допомогою перевірки за столом, покрокового виконання та виконання з точками зупинки.
4. Запропонувати варіант виправлення коду, зокрема додатково реалізувати в ньому обробку некоректних вхідних даних за допомогою виключень.
5. Повторно виконати розроблені тести, зафіксувавши збіг очікуваних та отриманих результатів для виправленого коду.
6. Результати роботи зафіксувати у файлі Word з ім’ям **Лаб1_Прізвище.doc**, який має включати:

1) виправлений текст програми, де виділено кольором рядки з виправленнями та нові рядки за їх наявності;

2) результати модульного тестування виправленої програми - таблицю

Вхідні дані	Вихідні дані		
	Очікувані	Спостережені до виправлення	Спостережені після виправлення

Номер програми відповідає номеру студента в списку.

1. //Обчислення математичних функцій

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
        double a, b,t,t0,dt,y;
        int points;
            string NameFunction;
   Console.WriteLine("Введіть ім'я F(t) досліджуваної функції a*F(b*t)" +
       " (sin, cos, tan, cotan)");
   NameFunction = Console.ReadLine();
   Console.WriteLine("Введіть параметр a (double)");
   a= double.Parse(Console.ReadLine());
   Console.WriteLine("Введіть параметр b (double)");
   b= double.Parse(Console.ReadLine());
   Console.WriteLine("Введіть початковий час t0(double)");
   t0= double.Parse(Console.ReadLine());
```

```
   Console.WriteLine("Введіть кількість точок points(int)");
   points = int.Parse(Console.ReadLine());
   Console.WriteLine("Введіть міжточковий інтервал  dt(double)");
   dt= double.Parse(Console.ReadLine());
     for(int i = 1; i<points; ++i)
   {
      t = t0 + (i-1)* dt;
      switch (NameFunction)
      {
         case ("sin"):
            y = a*Math.Sin(b*t);
            break;
         case ("cos"):
            y = a*Math.Cos(b*t);
            break;
         case ("tan"):
            y = a*Math.Tan(b*t);
            break;
         case ("cotan"):
            y = a*Math.Tan(b*t);
            break;
         case ("ln"):
            y = a*Math.Log(b*t);
            break;
         case ("tanh"):
            y = a*Math.Tanh(b*t);
            break;
         default:
            y=1;
            break;
      }
      Console.WriteLine ("t = " + t + "; " + a +"*" +
         NameFunction +"(" + b + "*t)= " + y + ";");
   }
   double  p,w;
   // u = 2.5, v = 1.5,
   p= Math.Pow(a,b);
   w = Math.IEEERemainder(a,b);
   Console.WriteLine ("power(a,b)= " + p + "; reminder(a,b)= " + w);
}
        }
     }
```

2. Генерування й сортування масиву випадкових чисел у заданому діапазоні

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {         // генератор даних
        static void ValsGenerator(int[], Vals)
        {
            // Random - клас для генерації випадкових чисел
            int M,N;
            Random aRand = new Random();
            Console.WriteLine("Введіть діапазон випадкових чисел M,N (int)");
            M = int.Parse(Console.ReadLine());
            N = int.Parse(Console.ReadLine());
            // заповнення масиву
```

```
  for (int i = 1; i <= Vals.Length; i++)
                Vals[i] = aRand.Next(M,N);
        }
        static void Main(string[] args)
        {
            int N;
            Console.WriteLine("Введіть розмірність масиву N (int)");
            N = int.Parse(Console.ReadLine());
            int[] Data = new int[N];
            ValsGenerator(Data);
            Array.Sort(Data);
            Console.WriteLine("Друк відсортованих даних");
            for (int i = 0; i < Data.Length; i++)
                Console.WriteLine("Data[" + i + "] = " + Data[i]);
            Console.ReadLine();
        }
    }
}
```

3. Консольний калькулятор на 4 дії

```
using System;
namespace ConsoleCalculator
{   class Program
    {   static void Main()
        {
            string buf;
            double a, b, res;
            Console.WriteLine( "Введіть перший операнд:" );
            a = double.Parse( Console.ReadLine() );
             Console.WriteLine( "Введіть знак операції" );
            char op = (char)Console.Read();
            Console.ReadLine();

            Console.WriteLine( "Введіть другий операнд:" );
            b = double.Parse( Console.ReadLine() );
            bool ok = falce;
            switch (op)
            {
                case '+' : res = a + b; break;
                case '-' : res = a - b; break;
                case '*' : res = a * b; break;
                case '/' : res = a / b; break;
                default  : ok = false; break;
            }
            if (ok) Console.WriteLine( "Результат: " + res );
            else    Console.WriteLine( "Неприпустима операція" );
        }
    }
}
```

4. Метод обчислення натурального ступеню n невід’ємного дійсного числа x

```
static public double Power(int x, int n)
{
    double z=x;
    for (int i=1; n>i;i++)
    {
       z = z*x;
     }
    return z;
}
```

```
static void Main(string[] args)
        {
            double x, y;
            Console.WriteLine("Введіть значення x (double)");
           int x = int.Parse(Console.ReadLine());
            Console.WriteLine("Введіть значення ступеня n (int)");
            int n = int.Parse(Console.ReadLine());
            y = Power(x, n);
            Console.WriteLine("y=" + y);
            Console.ReadLine();
}
```

5. // Обчислення sin(x) на підставі ступеневого ряду для довільного дійсного x

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace My_sin
{
    class Program
    {
        static double Cal_sin(double x, int n)
        {

            double result = x;
            for (int i = 1; i <=n; i++)
            {
             result=result+(Math.Pow((-1),i)*Math.Pow(x,(2*i+1)))/F(2*i);
            }
             return result;
        }
        static double F(int n)
        {
            double tmp = 1;
            for (int i = 1; i < n; i++)
            {
                tmp = tmp * i;
            }
            return tmp;
        }
        static void Main(string[] args)
        {
            Console.WriteLine("Введіть x - кут в радіанах");
            double x = double.Parse(Console.ReadLine());
            Console.WriteLine("Введіть показник ступеня n");
            int n = int.Parse(Console.ReadLine());
          //виклик методу обчислення sin(x) через ряд
            double my_sinus = Calc_sin(x,n);
            Console.WriteLine("my_sinus= {0}", my_sinus);
            Console.ReadKey();
        }
    }
}
```

6. Вдосконалений консольний калькулятор

```
namespace ConsoleCalculator
{///Реалізувати  метод обчислення виразу a X b, де a, b – числа,
    ///X={+,-,*,/}. Вираз ввести з консолі, результат вивести на консоль.
    ///При виборі операції використати оператор switch.
```

```
class Program
    {
        static bool IsNumeric(string NumericText)
        {
            //перевірка чи у рядку число
            bool isnumber = true;
            foreach (char c in NumericText)
            {
                isnumber = char.IsNumber(c);
                if (!isnumber)
                {
                    if (c = ',')
                        return isnumber;
                }
            }
            return isnumber;
        }
        static double ConvertDouble(string NumericText)
        {
            double digit;
            //конвертація і округлення числа
            digit = double.Parse(NumericText);
            digit = Math.Round(digit, 2);
            return digit;
        }

        static void Main(string[] args)
        {
            string str;     //Рядки для розбору виразу
            string[] arrStr = new string[2];
            string op = "";
            double digit1, digit2, result;      //числа
            do
            {
                Console.WriteLine("Введіть арифметичний вираз");
                str = Console.ReadLine();
                //аналіз вхідного виразу
                try
                {
                    if (str.Length != 0)
                    {
                        throw new Exception("Нічого не введено! Введіть
правильний вираз.");
                    }
                    string[] split = str.Split(new Char[] { ' ', '*', '+', '-',
'/' });
                    int i = 0;
                    foreach (string s in split)
                    {
                        if (s.Trim()== "")
                        {
                            Console.WriteLine(s);
                            arrStr[i] = s;
                            i++;
                        }
                    }

                    //пошук операції
                    char[] array0p = new char[4] { '+', '-', '*', '/' };
                    foreach (char c in array0p)
                    {
                        int indexop = str.IndexOf(c);
                        if (indexop == -1)
                        {
```

```
                            op = str[indexop].ToString();
                            break;
                        }
                    }
                    if (op.Length == 0)
                    {
                        throw new Exception("Операція не знайдена у виразі");
                    }
                    digit1 = ConvertDouble(arrStr[0]);

                    digit2 = ConvertDouble(arrStr[1]);

                    switch (op)
                    {
                        case ("+"):
                            result = digit1 + digit2;
                            break;
                        case ("-"):
                            result = digit1 - digit2;
                            break;
                        case ("/"):
                            result = digit1 / digit2;
                            break;
                        case ("*"):
                            result = digit1 * digit2;
                            break;
                        default:
                            throw new Exception("Невідома операція");

                    }//switch

                    // Console.WriteLine("s1={0},op={1},s2={2}", s1, op, s2);

Console.WriteLine("аргумент1={0},операція={1},аргумент2={2}", digit1, op,
digit2);
                    Console.WriteLine("Результат= " + result);
                    Console.ReadKey();
                }
                catch (Exception e)
                {
                    Console.WriteLine("Помилка у вхідних даних" + e.Message);
                    Console.ReadLine();
                }
                Console.WriteLine("Для завершення роботи введіть exit");
            } while (Console.ReadLine() != "exit");
        }
    }
}
```

7. Формування масиву випадкових чисел, розділених комами

```
using System;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
{
string genvar(int inf, int sup, int k, Random aRand)
        {
            //inf,sup,k
            string strtest="";
            int s=0;
```

```
            int[] mas = new int[k];
            int rnd;
            int point = 1;
            mas[0]=aRand.Next(inf, sup);
            strtest = mas[1].ToString() + ",";
            point = 1;
            while (point < k)
            {
                rnd = aRand.Next(inf, sup);
                for (int i = 0; i < point; i++)
                {
                    if (mas[i] == rnd)
                    {
                        s = 1;
                       break;
                    }
                    else
                    {
                      s = 0;
                    }
                }
                if (s == 1)
                {
                    mas[point] = rnd;
                    strtest += mas[point].ToString() + ",";
                    point += 1;
                }
            }

            return strtest;
        }
      }
      static void Main(string[] args)

            int N, MIN,MAX;
            Console.WriteLine("Введіть розмірність масиву N (int)");
                  N = int.Parse(Console.ReadLine());
                  Console.WriteLine("Введіть діапазон випадкових чисел MIN,MAX
(int)");
            MIN = int.Parse(Console.ReadLine());
            MAX = int.Parse(Console.ReadLine());
                        int[] Data = new int[N];
                  genvar(MIN,MAX,N,Data);
            Console.WriteLine("Друк даних");
            for (int i = 0; i < Data.Length; i++)
                Console.WriteLine("Data[" + i + "] = " + Data[i]);
            Console.ReadLine();
         } }
```

8. Перевірка розміщення точки в одиничному квадраті

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
class Program
{
static void Main(string[] args)
            {
Console.WriteLine("Введите x = ");
        double x = Convert.ToDouble(Console.ReadLine());
```

```
 Console.WriteLine("Введите y = ");
  double y = Convert.ToDouble(Console.ReadLine());

 if ((x > 1 || x < -1 ) && (y > 1 || y < -1))
 {
 Console.WriteLine("Точка попадает в заштрихованную область");

 }

if (x > 0 && y > 0 || x < 0 && y < 0)
{
 Console.WriteLine("Точка НЕ попадает в заштрихованную область");
  }
{
Console.WriteLine("Точка НЕ попадает в заштрихованную область");
}
```

9. Вдосконалений консольний калькулятор

```
namespace ConsoleCalculator
{///Реалізувати  метод обчислення виразу a X b, де a, b – числа,
    ///X={+,-,*,/}. Вираз ввести з консолі, результат вивести на консоль.
    ///При виборі операції використати оператор switch.
    class Program
    {
        static bool IsNumeric(string NumericText)
        {
            //перевірка чи у рядку число
            bool isnumber = false;
            foreach (char c in NumericText)
            {
                isnumber = char.IsNumber(c);
                if (!isnumber)
                {
                    if (c = '','')
                        return isnumber;
                }
            }
            return isnumber;
        }
        static double ConvertDouble(string NumericText)
        {
            double digit;
            //конвертація і округлення числа
            digit = double.Parse(NumericText);
            digit = Math.Round(digit, 1);
            return digit;
        }

        static void Main(string[] args)
        {
            string str;     //Рядки для розбору виразу
            string[] arrStr = new string[1];
            string op = "";
            double digit1, digit2, result;      //числа
            do
            {
                Console.WriteLine("Введіть арифметичний вираз");
                str = Console.ReadLine();
                //аналіз вхідного виразу
                try
                {
                    if (str.Length == 1)
                    {
```

```
                    throw new Exception("Нічого не введено! Введіть
правильний вираз.");
                }
                string[] split = str.Split(new Char[] { ' ', '*', '+', '-',
'/' });
                int i = 0;
                foreach (string s in split)
                {
                    if (s.Trim()== "")
                    {
                        Console.WriteLine(s);
                        arrStr[i] = s;
                        i++;
                    }
                }

                //пошук операції
                char[] array0p = new char[4] { '+', '-', '*', '/' };
                foreach (char c in array0p)
                {
                    int indexop = str.IndexOf(c);
                    if (indexop == -1)
                    {
                        op = str[indexop].ToString();
                        break;
                    }
                }
                if (op.Length == 0)
                {
                    throw new Exception("Операція не знайдена у виразі");
                }
                digit1 = ConvertDouble(arrStr[1]);

                digit2 = ConvertDouble(arrStr[2]);

                switch (op)
                {
                    case ("+"):
                        result = digit1 + digit2;
                        break;
                    case ("-"):
                        result = digit1 - digit2;
                        break;
                    case ("/"):
                        result = digit1 / digit2;
                        break;
                    case ("*"):
                        result = digit1 * digit2;
                        break;
                    default:
                        throw new Exception("Невідома операція");

                }//switch

                // Console.WriteLine("s1={0},op={1},s2={2}", s1, op, s2);

Console.WriteLine("аргумент1={0},операція={1},аргумент2={2}", digit1, op,
digit2);
                Console.WriteLine("Результат= " + result);
                Console.ReadKey();
            }
            catch (Exception e)
            {
                Console.WriteLine("Помилка у вхідних даних" + e.Message);
```

```
                    Console.ReadLine();
                }
                Console.WriteLine("Для завершення роботи введіть exit");
            } while (Console.ReadLine() != "exit");
        }
    }
}
```

10. Обчислення x × sin(x) через розкладання в ряд Тейлора, -∞ ≤ x ≤ 0, натуральне n

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace My_sin
{
    class Program
    {
        static double Calc_sin(double x, int n)
        {
            //обчислення розкладання sin в ряд
            double result =0;
            for (int i = 0; i < n; i++)
            {
             result=result+(Math.Pow((-1),i)*Math.Pow(x,(2*i+1)))/F(2*i+1);
            }
             return result;
        }
        static double F(int n)
        {
            double tmp = 1;
            for (int i = 1; i <= n; i++)
            {
                tmp = tmp * i;
            }

            return tmp;
        }
        static void Main(string[] args)
        {
            Console.WriteLine("Введіть x - кут в радіанах");
            double x = double.Parse(Console.ReadLine());
            Console.WriteLine("Введіть показник ступеня n");
            int n = int.Parse(Console.ReadLine());
          //виклик методу обчислення sin(x) через ряд
            double my_sinus = Calc_sin(x,n);
            //виклик методу з класу Math
            double sinus = Math.Sin(x);
            double delta = sinus - my_sinus;
            Console.WriteLine("my_sinus= {0},sin={1},delta={2}", my_sinus,
sinus, delta);
            Console.ReadKey();
        }
    }
}
```

11. Аналіз бази даних про співробітників

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
```

```
namespace Person1
{
    class Person
    {                       // 1
        const int l_name = 30;
        string name;
        int birth_year;
        double pay;
        public Person()             // конструктор без параметрів
        {
            name = "Anonimous";
            birth_year = 0;
            pay = 0;
        }
        public Person(string s)                         // 2 конструктор з
параметром
        {
            name = s.Substring(0, l_name);
            birth_year = Convert.ToInt32(s.Substring(l_name, 4));
            pay = Convert.ToDouble(s.Substring(l_name + 4));
            if (birth_year < 0) throw new FormatException();
            if (pay < 0) throw new FormatException();
        }
        public override string ToString()         // 3 перевантажений метод
        {
            return string.Format("Name: {0,30} birth: {1} pay: {2:F2}",name,
birth_year, pay);
        }
        public int Compare(string name) // порівняння прізвища
        {
            return (string.Compare(this.name, 0,name + " ", 0, name.Length +
1,StringComparison.OrdinalIgnoreCase));
        }
        // --------------------  властивості класу -------------------------
        public string Name
        {
            get { return name; }
            set { name = value; }
        }
        public int Birth_year
        {
            get { return birth_year; }
            set
            {
                if (value > 0) birth_year = value;
                else throw new FormatException();
            }
        }
        public double Pay
        {
            get { return pay; }
            set
            {
                if (value > 0) pay = value;
                else throw new FormatException();
            }
        }
        // ------------------  операції класу ---------------------------
        public static double operator +(Person pers, double a)
        {
            pers.pay += a;
            return pers.pay;
        }
        public static double operator +(double a, Person pers)
```

```
        {
            pers.pay += a;
            return pers.pay;
        }
        public static double operator -(Person pers, double a)
        {
            pers.pay -= a;
            if (pers.pay < 0) throw new FormatException();
            return pers.pay;
        }
    };


    class Program
    {
        static void Main(string[] args)
        {
            Person[] dbase = new Person[100];
            int n = 0;
            try
            {
                StreamReader f = new StreamReader("dbase.txt");     // 4
                string s;
                int i = 0;

                while ((s = f.ReadLine()) != null)
// 5
                {
                    dbase[i] = new Person(s);
                    Console.WriteLine(dbase[i]);
                    ++i;
                }
                n = i - 1;
                f.Close();
            }
            catch (FileNotFoundException e)
            {
                Console.WriteLine(e.Message);
                Console.WriteLine("Перевірте правильність імені і шляху до
файлу!");
                return;
            }

            catch (IndexOutOfRangeException)
            {
                Console.WriteLine("Дуже великий файл!");
                return;
            }
            catch (FormatException)
            {
                Console.WriteLine("Недопустима дата народження або оклад");
                return;
            }

            catch (Exception e)
            {
                Console.WriteLine("Помилка: " + e.Message);
                return;
            }

            int n_pers = 0;
            double mean_pay = 0;
            Console.WriteLine("Введіть прізвище співробітника");
            string name;
```

```
            while ((name = Console.ReadLine()) != "")                    // 6
            {
                bool not_found = true;
                for ( int k = 0; k <= n; ++k )
                {
                    Person pers = dbase[k];
                    if (pers.Compare(name) == 0)
                    {
                        Console.WriteLine(pers);
                        ++n_pers; mean_pay += pers.Pay;
                                  not_found = false;
                    }
                }
                if (not_found) Console.WriteLine("Такого співробітника немає");
                Console.WriteLine(
      "Введіть прізвище співробітника або Enter для завершення");
            }
            if (n_pers > 0)
                  Console.WriteLine("Середній оклад: {0:F2}", mean_pay /
n_pers);
            Console.ReadKey();
        }
    }
}
```

12. // Обчислення sin(x)/x на підставі ступеневого ряду для довільного дійсного x∈[-π; π], вимірюваного в радіанах.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace My_sin
{
    class Program
    {
        static double sin(double x, int n)
        {

            double result;
            for (int i = 0; i < n; i++)
            {
             result=result+(Math.Pow((-1),i)*Math.Pow(x,(2*i)))/F(2*i+1);
            }
             return result;
        }
        static double F(int n)
        {
            double tmp = 0;
            for (int i = 0; i <= n; i++)
            {
                tmp = tmp * i;
            }
            return tmp;
        }
        static void Main(string[] args)
        {
            Console.WriteLine("Введіть x - кут в радіанах");
            double x = double.Parse(Console.ReadLine());
            Console.WriteLine("Введіть показник ступеня n");
            int n = int.Parse(Console.ReadLine());
           double rez = sin(x, n)/x;
```

```
            Console.WriteLine("rez= {0}", rez);
            Console.ReadKey();
        }
    }
}
```

13. Видалення пробілів у тексті і приведення його до нижнього регістру за допомогою спадкування інтерфейсу

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleInterface1
{
    interface IStrings
    {
        string Convert();
        string Cipher(string[] code);
    }
    class SimpleText : IStrings
    {
        //поля класу
        string text;
        static string[] codeTable =
   {
     "абвгдеёжзийклмнопрстуфхцчшщыьъэюя ,.!?;:",
     "ъьыщшчцхфуэюя ,.!?;:тсрпонмлкйабвгдеёжзи"
   };
        //Конструктори
        public SimpleText()
        {
            text = "Простий текст";
        }
        public SimpleText(string txt)
        {
            text = txt;
        }
        public string Text
        {
            get { return text; }
        }
        /// Видалення пробілів в полі text
        /// перетворення до нижнього регістра
        /// <returns>  перетворений рядок </returns>
        public string Convert()
        {
            string res;
            foreach (char sym in text)
                if (sym == ' ') res = sym.ToString();
            res = res.ToLower();
            return res;
        }
        /// шифрування поля text
        /// з використанням таблиці кодування символів
                /// <returns> зашифрований текст </returns>
        public string Cipher(string[] code)
        {
            string res = "";
            foreach (char sym in text)
            {
                int k = code[0].IndexOf(sym);
```

```
                if (k > 0) res += code[1][k];
                else res += sym.ToString();
            }
            return res;
        }
        /// Шифрування, задане власною таблицею кодування
                /// <returns> зашифрований текст</returns>
        public string Coding()
        {
            return Cipher(codeTable);
        }

    }

    class Testing
    {
        //поля класу
        Console.WriteLine("Уведіть текст для перетворення!");
        string PAL = string.Parse(Console.ReadLine());
                    static string[] CODE =
   {
       "абвгдежзиклмнопрстуфхцчшыъьэюя",
       "abvgdejziklmnoprstyfhc461w'qux"
   };

        /// <summary>
        /// Тестування класу SimpleText
        public void TestText()
        {
            Console.WriteLine("Робота з об’єктом класу SimpleText! ");
            SimpleText simpleText = new SimpleText(PAL);
            Console.WriteLine("Початковий текст : " + PAL);
            string text;
            text = simpleText.Convert();
            Console.WriteLine("Перетворений текст : " + text);
                        text = simpleText.Coding();
            Console.WriteLine("Шифрований текст : " + text);

            Console.WriteLine("Робота з об'єктом інтерфейсуа IStrings! ");
            IStrings istrings;
            Console.WriteLine("Уведіть додатковий текст для перетворення!");
            text = string.Parse(Console.ReadLine());
            Console.WriteLine("Початковий текст : " + text);
            simpleText = new SimpleText(text);
            istrings = (IStrings)simpleText;
            text = istrings.Convert();
            Console.WriteLine("Перетворений текст : " + text);
            text = istrings.Cipher(CODE);
            Console.WriteLine("Шифрований текст : " + text);
        }
    }
    class Program
    {

        static void Main(string[] args)
        {
            Testing tint = new Testing();
            tint.TestText();
            Console.ReadKey();

        }
    }
}
```

14. Перевірка, чи є первинний і шифрований текст паліндромами

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleInterface1
{
    interface IStrings
    class SimpleText : IStrings
    {
        //поля класу
        string text;
        static string[] codeTable =
   {
     "абвгдеёжзийклмнопрстуфхцчшщыьъэюя ,.!?;:",
     "ъьыщшчцхфуэюя ,.!?;:тсрпонмлкйабвгдеёжзи"
   };
        //Конструктори
        public SimpleText()
        {
            text = "Простий текст";
        }
        public SimpleText(string txt)
        {
            text = txt;
        }
        public string Text
        {
            get { return text; }
        }
                /// шифрування поля text
        /// з використанням таблиці кодування символів
        /// <returns> зашифрований текст </returns>
        public string Cipher(string[] code)
        {
            string res = "";
            foreach (char sym in text)
            {
                int k = code[0].IndexOf(sym);
                if (k > 0) res += code[1][k];
                else res += sym.ToString();
            }
            return res;
        }
        /// Перевірка поля text,  чи є воно паліндромом
        /// <returns> true, якщо це паліндром  </returns>
        public bool IsPalindrom()
        {
            string txt = Convert();
            for (int i = 1, j = txt.Length; i < j; i++, j--)
                if (txt[i] == txt[j]) return false;
            return true;
        }
        /// Шифрування, задане власною таблицею кодування
        /// <returns> зашифрований текст</returns>
        public string Coding()
        {
            return Cipher(codeTable);
        }

    }
```

```
    class Testing
    {
        //поля класу
       Console.WriteLine("Уведіть текст для перетворення!");
                string PAL = string.Parse(Console.ReadLine());
                    static string[] CODE =
   {
       "абвгдежзиклмнопрстуфхцчшыъьэюя",
       "abvgdejziklmnoprstyfhc461w'qux"
   };

        /// Тестування класу SimpleText
        public void TestText()
        {
            Console.WriteLine("Робота з об’єктом класу SimpleText! ");
            SimpleText simpleText = new SimpleText(PAL);
            Console.WriteLine("Початковий текст : " + PAL);
                if (simpleText.IsPalindrom())
                Console.WriteLine("Це паліндром!");
            text = simpleText.Coding();
            Console.WriteLine("Шифрований текст : " + text);

            Console.WriteLine("Робота з об'єктом інтерфейсу IStrings! ");
            IStrings istrings;
            Console.WriteLine("Уведіть додатковий текст для перетворення!");
            text = string.Parse(Console.ReadLine());
            Console.WriteLine("Початковий текст : " + text);
            simpleText = new SimpleText(text);
            istrings = (IStrings)simpleText;
             text = istrings.Cipher(CODE);
             Console.WriteLine("Шифрований текст : " + text);
             simpleText = new SimpleText(text);
            if (simpleText.IsPalindrom())
                Console.WriteLine("Шифрований текст - також паліндром!");
        }
    }
    class Program
    {

        static void Main(string[] args)
        {
            Testing tint = new Testing();
            tint.TestText();
            Console.ReadKey();

        }
    }
}
```

Лабораторна робота 2. Структурне тестування окремих методів консольних застосунків

Мета роботи. Використання критеріїв структурного покриття коду.

Теоретичні відомості.

У методах тестування потоку управління дані з вхідного простору вибираються так, щоб забезпечити максимальне покриття коду.

Покриття коду визначає повноту перевірки модуля набором тестів.

Основними методами структурного тестування є:

1) тестування рядків. Критерій покриття С0 (покриття рядків);

2) тестування розгалужень. Критерій покриття С1 (покриття розгалужень);

3) тестування логічних умов. Критерій покриття С2 (покриття умов).

Ці методи використовуються на рівні модульного тестування (окремих методів класу).

1. Тестування рядків (Statement Coverage).

Вибираються дані, що забезпечують виконання всіх рядків (операторів) програми. Цей метод дає найслабший критерій покриття, так званий «покриття рядків», і прийнятний для програм, *що не містять логічних умов і циклів.*

2. Тестування розгалужень. (Decision Coverage)

Вибираються дані, що забезпечують виконання шляхів, що виділяються в програмі за допомогою логічних умов, що приймають значення *True* і *False*. В наведеному нижче прикладі для перевірки розгалуження достатньо двох тестів (при A<B і при ≥ B):

```
if (A<B) {  . }
else {.}
```

Для задоволення критерію покриття рядків достатньо було б виконати один (будь-який) з цих двох тестів.

3. *Тестування логічних умов*.

Якщо розгалуження в програмі утворюються в результаті виконання складних логічних умов, дані для їх тестування повинні вибиратися так, щоб перевірити всі значення логічних умов. Наприклад, для повної перевірки поданої нижче логічної умови:

```
if (A<B && C = 1) { ............}
else {......}
```

необхідно вже чотири тести (один - для випадку, коли умова виконується, і три - для решти випадків):

```
1) A < B i C = 1
2) A < B i C ≠ 1
3) A ≥ B i C = 1
4) A ≥ B i C ≠ 1.
```

Цей метод дає найповніший критерій покриття коду програми, який називається критерієм «*логічні умови»*. Знову таки, для задоволення критерію покриття рядків було б достатньо одного тесту, а для покриття розгалужень – двох.

Тестування циклів. Тести розробляються для перевірки кожного циклу при граничних значеннях змінних циклу і всередині них (для тестування стійкості цикли перевіряються при значеннях поза границями).

Цей метод дає критерій покриття, який називається «*всі цикли»*.

Загальні правила проектування тестів. Тести розробляються спочатку для перевірки правильних вхідних даних і умов. Обов'язково для граничних даних і після цього для неправильних вхідних даних і умов.

Приклад 1. Написати і виконати тест для знаходження помилки в методі піднесення до ступеня числа x.

//Варіант 1

// Метод обчислення ступеню n числа, x^n (для n>0)

```
static public double Power(double x, int n)
{
    double z=1;
    for (int i=1;n>=i;i++)
     {
        z = z*x;
     }
    return z;
}
```

При n<1 метод буде видавати неправильний результат, наприклад:
Power(2,-1) = 2, але ніякої діагностики не виводиться.
Отже, метод не стійкий до помилок у вхідних даних.

// Варіант 2

// Метод обчислення ступеню n числа x^n (для n>0)

```
static public double PowerNonNeg(double x, int n)
{
   double z=1;
   if (n>0)
     {
        for (int i=1;n>=i;i++)
          {
              z = z*x;
           }
     }
   else Console.WriteLine("Помилка! Показник ступеню повинен бути >=0.");
return z;
}
```

Для цього прикладу розробимо тести для перевірки розгалужень і циклу.

Метод приймає 2 параметри: *double x, int n* і повертає значення x^n. Значення n>0. Для спрощення тестування обмежимо n=100.

Крім перевірки вхідних параметрів, корисно знайти обмеження для максимального значення x^n .

Примітка

Int	System.Int32	Ціле 32-розрядне зі знаком. Діапазон значень: 2147483648 ... 2147483647
double	System.Double	Дійсне плаваюче 64-розрядне. Стандарт IEEE

Тести і очікувані результати будемо записувати в таблицю.

Вхідні дані	Очікуваний результат	Критерій проходження тесту (True/False)
x= 1, n=1	1	
x= 5, n=2	25	
x= 5000, n=10	9,76563E+36	
x= 5000, n=100	?	
x= 5000, n=0	Повідомлення. "Помилка! Показник ступеню повинен бути >=0."	
x= 5, n=abc	?	
x=2, n=-2	Повідомлення. "Помилка! Показник ступеню повинен бути >=0."	

Напишемо у методі main виклик методу Power

```
        static void Main(string[] args)
         {
             double x, y;
             Console.WriteLine("Введіть значення x (double)");
             x = double.Parse(Console.ReadLine());
             Console.WriteLine("Введіть значення ступеня n (int)");
             int n = int.Parse(Console.ReadLine());
             y = Power(x, n);
             Console.WriteLine("y=" + y);
             Console.ReadLine();
        }
```

За допомогою цього набору тестів ми перевірили всі рядки коду. Критерій С0=100%, розгалуження if (). Критерій С1=100%. Цикл при min, max, проміжних значеннях змінної циклу.

Але цей набір недостатній для виявлення помилки виходу за розрядну сітку при дуже великих значеннях x^n.

Модифікуємо метод для того, щоб він був стійким до помилок у вхідних даних. Введемо обмеження на дані, які будемо аналізувати.

1. Значення числа і ступеня повинні бути цілими.
2. Значення числа, що підноситься до ступеня, повинні лежати в діапазоні – [0..999].
3. Значення ступеня повинні лежати в діапазоні – [1..100].

Якщо числа, що подаються на вхід, лежать за межами вказаних діапазонів, то повинне видаватися повідомлення про помилку.

В метод Main вставимо обробку виключень (блок try{} catch (Exception e)).

// **Метод обчислення ступеню n числа x**

```
static public double Power(int x, int n)
{
    double z=1;
    for (int i=1;n>=i;i++)
    {
       z = z*x;
     }
    return z;
}

static void Main(string[] args)
{
  int x;
  int n;
  try
{
    Console.WriteLine("Enter x:");
    x=Convert.ToInt32(Console.ReadLine());
    if ((x>=0) & (x<=999))
    {
       Console.WriteLine("Enter n:");
        n=Convert.ToInt32(Console.ReadLine());
        if ((n>=1) & (n<=100))
       {
           Console.WriteLine("The power n of x is {0}",
           Power(x,n));
           Console.ReadLine();
        }
        else
        {
            Console.WriteLine("Error : n must be in [1..100]");
            Console.ReadLine();
```

```
        }
    }
    else
     {
       Console.WriteLine("Error : x must be in [0..999]");
       Console.ReadLine();
     }
}
catch (Exception e)
{
Console.WriteLine("Error : Please enter a numeric argument.");
Console.ReadLine();
}
}
```

Завдання для самостійного виконання

1. Розробити консольний за стосунок згідно з описом (номер відповідає номеру студента в списку групи) у вигляді методу Main, де ввести необхідні вхідні дані та передбачити перевірку їх коректності (за необхідності, з використанням обробки виключень), а результат вивести на консоль.

2. Розробити тести у вигляді таблиці, аналогічної наведеній у прикладі, для максимального покриття рядків, розгалужень, умов, циклів у створеній програмі.

3. Оцінити процент покриття тестами для рядків та гілок (за їх наявності в програмі) і навести його у нижніх рядках таблиці. Бажано досягти покриття 100%.

4. Зафіксувати результати лабораторної роботи – отриманий код і таблицю – у файлі **Лаб2_<Прізвище>.doc**

Варіанти завдань.

1. Наближене обчислення функції y=sin x, $-\infty < x < \infty$

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

(натуральне n, x вводяться з консолі)

2. Обчислення суми, максимального і мінімального елементів тривимірного масиву випадкових чисел заданої розмірності n із заданого діапазону [a;b] (a,b дійсні невід'ємні). a,b,n вводиться з консолі.

3. Знаходження Бі-факторіалу – добутку парних чисел до заданого натурального m, що вводиться з консолі.

4. Генерування послідовності простих цілих чисел в заданому діапазоні, цілі границі якого вводяться з консолі.

5. Знаходження чисел Фібоначчі в діапазоні від m до n (цілі), які вводиться з консолі.

Числа Фібоначчі – це елементи числової послідовності, в якій кожне наступне число дорівнює сумі двох попередніх. Приклад: 1,1,2,3,5,8,13,…

$F_1=1$, $F_2=1$, $F_{n+1}=F_n+F_{n-1}$

6. Знаходження всіх дільників заданого натурального числа n, приналежних діапазону [a;b]. Натуральні a,b,n вводяться з консолі.

7. Знаходження всіх простих дільників заданого натурального числа n, не менших натурального числа m.
n, m вводяться з консолі.

8. Знаходження суми додатніх чисел у послідовності дійсних чисел, яка вводиться з консолі й закінчується 0. Якщо вводиться не число і не 0, програма має пропонувати ввести коректне значення.

9. Швидке обчислення a^n, a > 0, n – натуральне число

$$a^n = \begin{cases} a \cdots a^{n-1}, & \text{if } n \text{ is odd} \\ (a^{n/2})^2, & \text{if } n \text{ is even} \end{cases}$$

(a, n водяться з консолі)

10. Вивести на консоль число, що найчастіше зустрічається в послідовності чисел, яка вводиться з консолі й закінчується 0. Якщо вводиться не число і не 0, програма має пропонувати ввести коректне значення

11. Вивести на консоль всі числа, що лежать у введених з консолі межах, цифри яких знаходяться в чітко спадному порядку (наприклад, 532, 431).

12.. Знаходження кількості натуральних чисел від m до n включно ($1 \le m \le n \le 30000$), для яких сума цифр в кубі дорівнює самому числу (на зразок 512)
(m, n вводяться з консолі**).**

13. Вивести на консоль кількість пробілів у рядку**,** введеному з консолі, та цей рядок без пробілів

14. Вивести на консоль число, що найрідше зустрічається в послідовності чисел, яка вводиться з консолі й закінчується 0.

15. Вивести на консоль кількість від'ємних чисел у послідовності чисел, яка вводиться з консолі й закінчується 0, та цю послідовність без від'ємних чисел.

Лабораторна робота № 3. Середовище модульного тестування. Методи структурного тестування класів.

Мета роботи. Тестування внутрішньої структури класів.

Теоретичні відомості.

1. Особливості тестування об'єктно-орієнтованих програм (ООП)

Специфіка тестування ООП пов'язана, в першу чергу, з:

- ітераційним підходом до розробки, що призводить до ускладнення інтеграційного і регресійного тестування;
- керованою подіями природою ОПП, що вимагає тестування не лише структури програми, але і її поведінки.

1. На модульному рівні виконується автономне тестування класів: їх структури і методів (функцій-членів) традиційними методами аналізу коду (див. Л2).

2. На рівні інтеграції класів виконується тестування ієрархії спадкування і тестування взаємодій між об'єктами. Повинні бути перевірені всі можливі зовнішні виклики методів класу - як безпосередній виклик, так і виклики, ініційовані отриманням повідомлень. Формальними моделями є модель класів ПЗ і моделі кожного класу.

3. На рівні системи застосовується тестування за сценаріями використання (прецедентами) – починаючи з аналізу вимог і проектування ПС.

2. Особливості середовища тестування ООП

Середовище тестування для об'єктно-орієнтованих програм (ООП) виконує ті самі функції, що і для структурних програм (процедурними мовами). Проте, воно має деякі особливості, зв'язані з використанням *наслідування* і *інкапсуляції*.

Якщо при тестуванні структурних програм мінімальним об'єктом, що тестується є функція, то в об'єктно-орієнтованій програмі мінімальним об'єктом є клас. При застосуванні принципу інкапсуляції, всі внутрішні дані класу і деяка частина його методів невидима ззовні. В цьому випадку тестер позбавлений можливості звертатися в своїх тестах до даних класу і довільним чином викликати методи – єдине, що йому доступно – викликати методи зовнішнього інтерфейсу класу.

За цих причин модульне тестування ООП складніше ніж структурних.

Існує декілька підходів до тестування класів, кожний з яких накладає свої обмеження на структуру драйвера і заглушок:

1. Драйвер створює один або більше об'єктів класу, що тестується, всі звернення до об'єктів відбуваються тільки з використанням їх зовнішнього інтерфейсу. Текст драйвера в цьому випадку представляє собою так званий тест-клас, який містить по одному методу для кожного тестового прикладу. Процес тестування полягає в послідовному виклику цих методів. Замість заглушок до складу тестового середовища входить програмний код реальної системи, відповідно відсутня ізоляція класу, що тестується. Проте, саме такий підхід до тестування прийнятий зараз в більшості методологій і середовищ розробки.

2. Аналогічно попередньому підходу, але для всіх класів, які використовує клас, що тестується, створюються заглушки.

3. Програмний код тестованого класу модифікується (шляхом тимчасової зміни модифікаторів для методів цього класу) таким чином, щоб відкрити доступ до всіх його властивостей і методів. Склад та структура середовища тестування в цьому випадку повністю аналогічна складу й структурі середовища для тестування структурних програм.

Основна перевага перших двох методів – при їх використанні клас працює точно таким же чином, як в реальній системі. Проте в цьому випадку не можна гарантувати того, що в процесі тестування буде виконаний весь програмний код класу і не залишиться не протестованих методів.

Основний недолік 3-го методу – після зміни тексту модуля не можна дати гарантії того, що клас буде поводитися таким же чином, як і початковий. Зокрема це пов'язано з тим, що зміна захисту даних класу впливає на наслідування даних і методів іншими класами.

Тестування спадкування – окрема складна задача в ООП. Після того, як протестовано базовий клас, необхідно тестувати класи-нащадки. Проте, для базового класу не можна створювати заглушки, так як в цьому випадку можна пропустити можливі проблеми поліморфізму. Якщо класс-нащадок використовує методи базового класу для обробки власних даних необхідно переконатися в тому, що ці методи працюють.

Таким чином, ієрархія класів може тестуватися згори донизу, починаючи від базового класу. Середовище тестування при цьому може змінюватися для кожної конфігурації тестованих класів.

Приклад.

Розглянемо проект автоматизованої системи обліку банківських відомостей.

Щодо кожного клієнта банку зберігаються такі відомості:

1. **Прізвище, ім'я, по-батькові;**
2. **Дата народження;**
3. **Паспортні дані;**
4. **Ідентиф.код;**
5. **Місце(я) роботи (навчання);**
6. **Номер(и) рахунку.**

Для кожного клієнта визначено операції:

1) додати нового клієнта;
2) видалити клієнта;
3) змінити реквізити клієнта.

На кожному рахунку зберігається інформація про поточний баланс.

З кожним рахунком можна виконувати такі дії:

1) відкриття (для певного клієнта), закриття,
2) внесення грошей, зняття грошей,
3) перегляд балансу.

Створимо два класи:

1. Класс **Client** – описує інформацію про клієнта
2. Клас **Account** – описує банківський рахунок

Класс Client

Варіант 1. Конструктор без параметрів

Область видимості полів класу відповідно до правил має бути визначена або як **закрита**, або як **захищена**. Доступ же до полів - членів класу має бути організований або за допомогою методів, або за допомогою властивостей класу. Створимо властивості класу Client, які забезпечують читання і запис значень полів класу.

```
public class Client
    {
        //поля класу
        private string Name;
        private DateTime BirthDate;
        private string Passport;
        private string ID;
        private string nomAccount;

        public Client()
        {
```

```
        //конструктор без параметрів
    }
//методи get і set для полів класу
    public string passport
    {
        get
        {
            return Passport;
        }
        set
        {
            Passport = value;
        }
    }

    public string name
    {
        get
        {
            return Name;
        }
        set
        {
            Name = value;
        }
    }

    public int age
    {
        get
        {
            int a;
            a = DateTime.Now.Year - BirthDate.Year;
            return a;
        }
    }

    public DateTime birthdate
    {
        get
        {
            return BirthDate;
        }
        set
        {
            if (DateTime.Now > value)
                BirthDate = value;
            else
                throw new Exception("Введена невірна дата народження");
        }
    }

    public string ID_kod
    {
        get {return ID; }

        set
        {
            ID = value;
        }
    }

    public string Nom_Account
    {
```

```
            get { return nomAccount; }
            set
            {
                nomAccount = value;
            }
        }
        //
    }
```

Варіант 2. Конструктор з параметрами

```
  public Client(string ClientName, string ClientPassport,DateTime
ClientBirthDate)
        {
          //конструктор без параметрів

            name = ClientName;
            passport = ClientPassport;
            birthdate = ClientBirthDate;
        }
```

Клас Account

Реалізуємо два варіанти класу Account.

Варіант 1.

У першому варіанті - класі **Account** - активно використовуватимемо поля класу. Окрім двох основних полів **credit** і **debit**, які зберігають поповнення й витрати з рахунку, введемо поле **balance**, яке задає поточний стан рахунку, і два поля, пов'язані з останньою виконуваною операцією. Поле **sum** зберігатиме суму грошей поточної операції, а поле **result** - результат виконання операції. Полів у класу багато, внаслідок чого в методах класу аргументів буде небагато.

```
public class Account
{
   //закриті поля класу
   int debit=0, credit=0, balance =0;
   int sum =0, result=0;
   /// <summary>
   /// Зачисление на счет с проверкой
   /// </summary>
   /// <param name="sum">зачисляемая сумма</param>
   public void putMoney(int sum)
   {
      this.sum = sum;
      if (sum >0)
      {
         credit += sum; balance = credit - debit; result =1;
      }
      else result = -1;
      Mes();
   }//putMoney
   /// <summary>
   /// Снятие со счета с проверкой
   /// </summary>
   /// <param name="sum"> снимаемая сумма</param>
   public void getMoney(int sum)
   {
      this.sum = sum;
      if(sum <= balance)
```

```
        {
            debit += sum; balance = credit - debit; result =2;
        }
        else result = -2;
        Mes();
    }//getMoney
    /// <summary>
    /// Уведомление о выполнении операции
    /// </summary>
    void Mes()
    {
        switch (result)
        {
         case 1:
           Console.WriteLine("Операция зачисления денег прошла успешно!");
           Console.WriteLine("Сумма={0},Ваш текущий баланс={1}",sum, balance);

           break;
         case 2:
           Console.WriteLine("Операция снятия денег прошла успешно!");
           Console.WriteLine("Сумма={0},Ваш текущий баланс={1}", sum,balance);
           break;
         case -1:
           Console.WriteLine("Операция зачисления денег не выполнена!");
           Console.WriteLine("Сумма должна быть больше нуля!");
           Console.WriteLine("Сумма={0},Ваш текущий баланс={1}", sum,balance);
           break;
         case -2:
           Console.WriteLine("Операция снятия денег не выполнена!");
           Console.WriteLine("Сумма должна быть не больше баланса!");
           Console.WriteLine("Сумма={0},Ваш текущий баланс={1}", sum,balance);
           break;
           default:
            Console.WriteLine("Неизвестная операция!");
            break;
        }
        Console.ReadLine();
    }
```

Як можна бачити, лише у методів **getMoney** і **putMoney** є один вхідний аргумент. Це той аргумент, який потрібний по суті справи, оскільки лише клієнт може вирішити, яку суму він хоче зняти або покласти на рахунок. Інших аргументів в методів класу немає - вся інформація передається через поля класу. Зменшення числа аргументів приводить до підвищення ефективності роботи з методами, оскільки зникають витрати на передачу фактичних аргументів. Але при цьому ускладнюються операції роботи з вкладом, оскільки потрібно у момент виконання операції оновлювати значення полів класу. Закритий метод Mes викликається після виконання кожної операції, повідомляючи про те, як прошла операція, і інформуючи клієнта про поточний стан його балансу.

Варіант 2

Спроектуємо аналогічний клас Account1, який відрізняється лише тим, що у нього буде менше полів. Замість поля **balance** в класі з'явиться відповідна функція з цим же іменем, замість полів **sum** і **result** з'являться аргументи в методів, що забезпечують необхідне передавання інформації.

```
public class Account1
{
    //закриті поля класу
    int debit=0, credit=0;
    /// <summary>
    /// Зачисление на счет с проверкой
```

```
    /// </summary>
    /// <param name="sum">зачисляемая сумма</param>
    public void putMoney(int sum)
    {
        int res =1;
        if (sum >0)credit += sum;
        else res = -1;
        Mes(res,sum);
    } //putMoney
    /// <summary>
    /// Снятие со счета с проверкой
    /// </summary>
    /// <param name="sum"> снимаемая сумма</param>
    public void getMoney(int sum)
    {
        int res=2;
        if(sum <= balance())debit += sum;
        else res = -2;
        balance();
        Mes(res, sum);
    }//getMoney
/// <summary>
/// вычисление баланса
/// </summary>
/// <returns>текущий баланс</returns>
    int balance()
    {
        return(credit - debit);
    }
    /// <summary>
    /// Уведомление о выполнении операции
    /// </summary>
    void Mes(int result, int sum)
    {
        switch (result)
        {
         case 1:
          Console.WriteLine("Операция зачисления денег прошла успешно!");
          Console.WriteLine("Сумма={0},Ваш текущий баланс={1}", sum,balance());
              break;
         case 2:
          Console.WriteLine("Операция снятия денег прошла успешно!");
          Console.WriteLine("Сумма={0},Ваш текущий баланс={1}", sum,balance());
              break;
         case -1:
          Console.WriteLine("Операция зачисления денег не выполнена!");
          Console.WriteLine("Сумма должна быть больше нуля!");
          Console.WriteLine("Сумма={0},Ваш текущий баланс={1}", sum,balance());
              break;
         case -2:
          Console.WriteLine("Операция снятия денег не выполнена!");
          Console.WriteLine("Сумма должна быть не больше баланса!");
          Console.WriteLine("Сумма={0},Ваш текущий баланс={1}", sum,balance());
          break;
         default:
          Console.WriteLine("Неизвестная операция!");
          break;
        }
    }
} //Account1
```

Порівнюючи цей клас з класом Account, можна бачити, що число полів скоротилося з п'яти до двох, спростилися основні методи **getMoney** і **putMoney**. Але, як розплата за таке

спрощення, у класі з'явився додатковий метод **balance()**, що викликається багаторазово, і в методі **Mes** тепер з'явилися два аргументи.

Тестування класу Account1 (варіант реалізації)

Процедура класу **Testing**, яка тестує роботу з класом **Account1**:

```
class Program
    {
        static void TestAccounts()
        {
            Account1 myAccount1 = new Account1();
            myAccount1.putMoney(6000);
            myAccount1.getMoney(2500);
            myAccount1.putMoney(1000);
            myAccount1.getMoney(4000);
            myAccount1.getMoney(1000);
            Console.WriteLine("Конец работы");
            Console.ReadLine();
        }//
        //

Виклик TestAccounts з методу Main
        static void Main(string[] args)
        {
          TestAccounts();
        }
    }
}
```

Завдання для самостійної роботи.

1. Створити клас згідно з описом (номер відповідає номеру прізвища студента у списку групи). Врахувати змістовні співвідношення між полями класу. Передбачити обробку максимального переліку ситуацій некоректності вхідних даних, зокрема за допомогою виключень: що б не робив користувач, програма має надавати повідомлення про зміст некоректності і реалізувати методи тільки для коректних даних.
2. Розробити тестовий клас для тестування розробленого класу, де передбачити введення тестових даних з консолі. Виконати всі методи .
3. Скласти таблицю

Вхідні дані	Очікувані дані	Спостережені дані: (**<u>що</u>** спостерігається; **<u>в якій частині</u>** програми; **<u>за яких</u>** обставин)

– у файлі **Лаб3_<Прізвище>.doc**

Варіанти завдань.

№	Зміст завдання
1	Розробити клас **Друкарське Видання** з 3 полями і трьома методами. Доступ до полів реалізувати через get і set. Реалізувати конструктор без параметрів, а ініціалізацію полів виконати через властивості. Бажані поля: кількість сторінок (натуральне число <5000) Автор(и): (Прізвище І.Б.[1], Прізвище І.Б[2]….); може бути порожнім; Ціна видання (позитивне дійсне число, грн) Бажані методи: визначення кількості авторів видання; визначення типу видання: *книга* - якщо кількість сторінок ≥ 100 і є хоч один автор; *не книга* - інакше; визначення середнього внеску в ціну авторів видання; не реалізується, якщо поле Автор(и) порожнє

№	Зміст завдання
2	Розробити клас **Місцевість** з 4 полями і двома методами. Доступ до полів реалізувати через get і set. Реалізувати конструктор без параметрів, а ініціалізацію полів виконати через властивості. Бажані поля: населення місцевості (≥0, тис. осіб, дійсне число) площа місцевості (>0, тис. км) головне місто місцевості (60 символів, перший символ- прописна літера, рос., укр., лат. літери без змішування, дефіс, цифри, пробіли, неприпустимі символи *, %, !, ?, $, #, кома); може бути відсутнім; кількість лікарень у місцевості (≥0, шт.) Бажані методи: обчислення площі, відповідної одному мешканцю місцевості; визначення типу місцевості: *міська*, якщо головне місто зазначено, *сільська* - інакше.
3	Розробити клас **Транспорт** з 4 полями і двома методами. Доступ до полів реалізувати через get і set. Реалізувати конструктор без параметрів, а ініціалізацію полів виконати через властивості. Бажані поля: Дата випуску транспорту (ДД.ММ.РРРР) Середня швидкість транспорту (км/год, < 200, дійсне число) Дата списання транспорту (ДД.ММ.РРРР), може бути відсутня, тоді має видаватися повідомлення "Транспорт не списаний" Середні витрати пального (<10, л/км, дійсне число) Бажані методи: обчислення пробігу транспорту до списання, якщо він використовувався 8 годин на добу; не реалізується, якщо дата списання відсутня; обчислення витрат пального на задану дату, якщо транспорт використовувався 8 годин на добу
4	Розробити клас **Тварина** з 4 полями і двома методами. Доступ до полів реалізувати через get і set. Реалізувати конструктор без параметрів, а ініціалізацію полів виконати через властивості. Бажані поля: Вид тварини (російські, українські, латинські літери, цифри, якщо перший символ – літера, то вона велика); Вага на момент народження (кг); від 0.01 до 10000 Вага поточна (кг); від 0.01 до 10000 Вік (роки). до 200 (дійсне число) Бажані методи: обчислення середньорічного приросту ваги; Обчислення ваги по роках віку
5	Розробити клас **Організація** з 4 полями і двома методами. Доступ до полів реалізувати через get і set. Реалізувати конструктор без параметрів, а ініціалізацію полів виконати через властивості. Бажані поля: Юридична адреса: МІСТО (20символів рос., укр., лат. без змішування, дефіс, цифри, пробіли;), Вулиця (30символів рос., укр., лат. без змішування, дефіс, цифри, пробіли); неприпустимі символи *, %, !, ?, $, #, кома; Фізична адреса: МІСТО (20символів рос., укр., лат. без змішування, дефіс, цифри, пробіли), Вулиця (30символів рос., укр., лат. без змішування, дефіс, цифри, пробіли); неприпустимі символи *, %, !, ?, $, #, кома Кількість співробітників (натуральне число); Річний бюджет (позитивне дійсне число, тис. грн.) Бажані методи: визначення, чи збігається Вулиця у фізичній та юридичній адресах; обчислення середнього річного бюджету на співробітника
6	Розробити клас **Співробітник** з 4 полями і двома методами. Доступ до полів реалізувати через get і set. Реалізувати конструктор без параметрів, а ініціалізацію полів виконати через властивості. Бажані поля: Дата прийому співробітника на роботу (ДД.ММ.РРРР) Дата народження (ДД.ММ.РРРР) Дата присудження кандидатського ступеня (ДД.ММ.РРРР) - може бути відсутня, тоді має видаватися повідомлення ("Не має ступеня") Стать співробітника (чол./жін.) Бажані методи: визначення, чи досяг співробітник пенсійного віку на 01.01.2014; обчислення кількості років від присудження ступеня до пенсії; не реалізується, якщо співробітник не має ступеня
7	Розробити клас **Рослина** з 3 полями і трьома методами. Доступ до полів реалізувати через get і set. Реалізувати конструктор без параметрів, а ініціалізацію полів виконати через властивості. Бажані поля: Сорт рослини (російські, українські, латинські літери, цифри, пробіли, якщо перший символ – літера, то вона велика); Очікувана середня врожайність (ц/га, позитивне дійсне число); Очікуване щорічне зростання врожайності (ц/га, позитивне дійсне число, < очікуваної врожайності); Бажані методи: визначення походження сорту: якщо сорт не містить рос. і укр. літер та хоч 1 латинську літеру, має видаватися "сорт іноземного походження", якщо хоч 1 рос. або укр. літеру "сорт вітчизняного походження", інакше – сорт невідомий; обчислення року, коли має бути досягнута очікувана врожайність;

№	Зміст завдання
	визначення, чи врожайність на заданий рік розвитку рослини дорівнює очікуваній, більше її, менше її.
8	Розробити клас **Виріб** з 4 полями і двома методами. Доступ до полів реалізувати через get і set. Реалізувати конструктор без параметрів, а ініціалізацію полів виконати через властивості. Бажані поля: Дата створення виробу (ДД.ММ.РРРР) Ціна виробу (позитивне дійсне число до 100000, тис. грн.) Дата початку уцінення виробу (ДД.ММ.РРРР) Розмір щорічної уцінки (дійсне число, менше ціни виробу, тис. грн) Бажані методи: обчислити ціну виробу на довільну дату ДД.ММ.РРРР* обчислити дату повного знецінення виробу (у форматі ДД.ММ.РРРР)
9	Розробити клас **Товар** з 3 полями і двома методами. Доступ до полів реалізувати через get і set. Реалізувати конструктор без параметрів, а ініціалізацію полів виконати через властивості.
10	Розробити клас **Програмний продукт (Software)** з 4 полями і двома методами. Доступ до полів реалізувати через get і set. Реалізувати конструктор без параметрів, а ініціалізацію полів виконати через властивості. Бажані поля: операційна система(и), на яких працює продукт (рос., укр., лат. літери без змішування, цифри, пробіли, перший символ-прописна літера; коми розділяють назви операційних систем); Дата початку розроблення (ДД.ММ.РРРР) Дата закінчення розроблення (ДД.ММ.РРРР) Кількість співробітників (натуральне число) Бажані методи: визначення кількості операційних систем для продукту обчислення трудомісткості розроблення продукту (людино/місяць)
11	Розробити клас **Персона** з 4 полями і двома методами. Доступ до полів реалізувати через get і set. Реалізувати конструктор без параметрів, а ініціалізацію полів виконати через властивості. Бажані поля: Прізвище І.Б. (українські, російські, латинські літери, перша та І.Б.) великі, перед І.Б. довільна кількість пробілів, Прізвище від 1 до 30 символів Дата народження (ДД.ММ.РРРР); Адреса проживання МІСТО(20символів рос., укр., лат., але без змішування) Вулиця (30символів рос., укр., лат., але без змішування), між ними довільна кількість пробілів Адреса місця роботи МІСТО(20символів рос., укр., лат., але без змішування) Вулиця (30символів рос., укр., лат., але без змішування), між ними довільна кількість пробілів Стать (чол./жін.) Бажані методи: обчислення року виходу персони на пенсію; визначення, чи співпадають адреси проживання і роботи без пробілів
12	Розробити клас **Викладач** з 4 полями і двома методами. Доступ до полів реалізувати через get і set. Реалізувати конструктор без параметрів, а ініціалізацію полів виконати через властивості. Бажані поля: Дисципліна(и) викладача (якщо перший символ дисципліни – літера, вона прописна; дисципліни розділяються комою та пробілом, символи !, ? неприпустимі); Дата народження викладача (ДД.ММ.РРРР); Неперервний стаж викладача (ціле невід'ємне число <100, роки); Навантаження викладача (ціле невід'ємне число <1000, год.) Бажані методи: обчислення середньої кількості годин, призначених на дисципліну; Обчислення віку, в якому викладач почав викладання
13	Розробити клас **Рослинний_продукт** з 4 полями і двома методами. Доступ до полів реалізувати через get і set. Реалізувати конструктор без параметрів, а ініціалізацію полів виконати через властивості. Бажані поля: назви рослин, використаних в продукті (припустимі символи – літери, пробіли, крапка; назви розділяються комами); Ціна одиниці продукту (грн.) Собівартість одиниці продукту (грн., не більше ціни) Кількість одиниць продукту в партії (≤ 100) Бажані методи: обчислення середньої собівартості включення до продукту його складових рослин; обчислення прибутку від продажу партії продукту
14	Розробити клас **Книга** з 4 полями і двома методами. Доступ до полів реалізувати через get і set. Реалізувати конструктор без параметрів, а ініціалізацію полів виконати через властивості. Бажані поля: УДК книги (припустимі символи – цифри, крапка, двокрапка; між крапками і/або двокрапками не більше трьох цифр; УДК розділяються комою); Кількість розділів; Кількість сторінок(від 10 до 5000); Бажані методи: обчислення кількості УДК для книги; обчислення середньої кількості сторінок у розділі

Лабораторна робота № 4. Системне тестування Windows-застосунків

Мета роботи: опрацювання різних видів системного тестування Windows-застосунків методами "чорної скриньки".

Теоретичні відомості.

Впродовж роботи необхідно виконати:

1) визначення типів тестів для застосунку
2) виявлення класів еквівалентності
3) проектування тестів еквівалентності і граничних значень
4) тестування функцій застосунку
5) тестування зручності інтерфейсу. Використання контрольних списків
6) тестування стійкості до помилок
7) розробку сценаріїв тестування і тестових наборів.

Необхідно провести такі види тестування:

1) тестування документації;
2) тестування інтерфейсу;
3) функціональне тестування.

Як проводити тестування:

1. Прочитати документацію.
2. Запустити застосунок.
3. Прочитати документацію в за стосунку (файл допомоги). З'ясувати, чи є в ній помилки.
4. Написати чек-лист для перевірки інтерфейса на підставі табл. 4.2, 4.3 й отримати відповіді на питання чек-листу.
5. Скласти тест-кейси для функціонального тестування.
6. Виконати тест-кейси.
7. Скласти звіт з тестування у вигляді табл. 4.1 у файлі **Лаб4_Прізвище.doc, зафіксувавши виявлені дефекти, успішні тести можна не описувати.**

Таблиця 4.1 – Звіт про дефекти Windows-форми

Вхідні дані	Очікувані дані	Спостережені дані (**що** спостерігається; **в якій частині** програми; **за яких** обставин)
Дефекти, виявлені під час тестування функцій		
Дефекти, виявлені під час тестування стійкості до помилок		
Дефекти, виявлені під час тестування зручності використання		
Дефекти, виявлені під час тестування документації		

Таблиця 4.2 – Узгоджені вимоги до інтерфейсу користувача при розробці програмних систем

Елемент	**Правило підтвердження**
Загальні вимоги	При розробці ПК (виборі кольору) повинні використовуватися пастельні тони. Яскраві кольори, бажано, не використовувати тому, що яскравий колір викликає швидку утому очей користувача. Припустимо використовувати приваблюючу увагу кольору при виникненні помилок, перед виконанням критичних операцій.
	Усі написи, повідомлення, зауваження, рекомендації (у тому числі при обробці помилок) необхідно виводити державною мовою - українською
	Функціональні можливості, що незастосовні в поточному режимі, повинні бути неактивні
Головне вікно	Повинно бути присутнім системне меню (Control Box)
	Повинний бути присутнім рядок заголовка
	Повинна бути присутнім кнопка мінімізації
	Повинна бути присутнім кнопка максимізації
	Повинна бути присутнім кнопка відновлення розміру
	Повинний бути присутнім рядок меню
	При необхідності, вікно може містити панель інструментів. Кнопки повинні бути розташовані горизонтально або вертикально і вирівняні. Повинні виконуватися усієї вимоги щодо реалізації кнопок
Кнопка системного меню	Повинна бути присутньою команда *(Восстановить)* Поновити
	Повинна бути присутньою команда (*Переместить)* Перемістити
	Повинна бути присутньою команда (*Размер)* Розмір
	Повинна бути присутньою команда (*Свернуть)* Згорнути
	Повинна бути присутньою команда (*Развернуть)* Розгорнути
	Повинна бути присутньою команда (*Закрыть)* Закрити
	При необхідності, повинні бути передбачені для кожної команди системного меню відповідні клавіші-акселератори
	Натискання комбінації клавіш Alt+F4 повинне викликати закриття форми.
Рядок заголовка	У рядку заголовка повинний виводиться ім'я програми
	Для інтерфейсу MDI (багатодокументний інтерфейс) ім'я форми дочірнього вікна повинне додаватися до імені додатка в рядку заголовка
Елементи управління: Розмір, Згорнути і Відновити	Коли головне вікно згорнуте, то дочірнє вікно (форма) теж повинна бути згорнута
Меню	У лінійці головного меню завжди потрібен пункт (*Помощь)* Допомога
	Пункти меню повинні містити підпункти, що логічно і функціонально зв'язані з заголовком даного пункту меню.
	Завжди повинний існувати вихід із ПК або як окремий пункт головного меню, або як підпункт 1-го пункту головного меню - Вихід
Панель інструментів Примітка:	Активізація кнопок, повинна відбуватися тільки одним натисканням кнопки миші
	Піктограми на кнопках повинні відповідати інтуїтивному уявленню про виконувану задачу
	Недоступні в даний момент кнопки панелі повинні відображатися сірим кольором
	Повинні існувати елементи меню, еквівалентні кожній кнопці панелі
	Для кожної кнопок панелі повинні існувати зрозумілі підказки
	Кнопки перемикачів - повинні мати утиснений вигляд, якщо включені, і опуклий , якщо виключені
Загальні вимоги	Всі екранні форми повинні мати рядок заголовка, який як можна більш повно розкриває (пояснює) екранну форму
	Форма повинна розміщатися по центру екрана
	Усі написи і текстові повідомлення повинні бути сформульовані коротко і зрозуміло для користувача
	Кнопки на екранній формі повинні задовольняти вимогам приведеним у розділі "Командні кнопки на формах"
	Інформація в довідниках, виведена користувачеві, повинна задовольняти вимогам приведеним у розділі "Інформація (довідники)"

Елемент	Правило підтвердження
Блок діалогу	Блок діалогу повинний бути переміщуваний
	Блок діалогу повинний мати системне меню (блок керування)
	У системному меню повинна бути команда Перемістити
	У системному меню повинна бути команда Закрити
	Блок діалогу не повинний мати кнопки Згорнути і Розгорнути
	Усі написи і текстові повідомлення повинні бути сформульовані коротко і зрозуміло для користувача і, по можливості, не містити скорочення (крім загальноуживаних у даній предметній області), наприклад, офіц. - замість офіцери, дир. - замість -директива, к. пошук - замість контекстний пошук
	На екранних формах обробки помилок, повинний виводитися ознака серйозності помилки
Обробка помилок	На екранній формі обробки помилок повинна бути присутня кнопка контекстної допомоги
	Повинні виводитися рекомендації користувачеві про подальші дії
	Повинні виводитися підказки про призначення кнопок на формі
	Повинна існувати кнопка "Відмінити" (для форм, що включають багато дій)
	Помилки повинні групуватися по ступені серйозності і мати відповідний код помилки
	Ліворуч від повідомлення повинна відображатися піктограма з буквою "І"
Інформаційні форми (стандартні, для виведення повідомлень)	Інформаційні форми повинні мати тільки кнопку "ОК" або "Закрити"
	При виконанні операцій іноді користувачеві необхідно виводити попередження. Ліворуч від повідомлення повинна відображатися піктограма зі знаком "!"
Форми попереджень	Форма повинна мати кнопку з позитивною відповіддю (Так)
	Форма повинна мати кнопку з негативною відповіддю (Ні і/або Відмінити)
	Повинне бути призначене значення кнопки за замовчуванням (яку з 2-х вирішує розроблювач)
Форма виведення критичних повідомлень	При виконанні критичних операцій необхідно виводити попередження користувачеві Для позначення таких повідомлень бажано використовувати піктограму зі знаком "STOP" ліворуч від повідомлення
	Повинна бути кнопка з позитивною відповіддю (Так)
	Повинна бути кнопка з негативною відповіддю (Ні/ Відмінити)
	Повинне бути призначене значення кнопка за замовчуванням (яку з 2-х вирішує розроблювач)
	Видалення інформації (виконання критичних операцій) повинне виконуватися в діалоговому режимі з підтвердженням намірів користувача
Форма виведення довідкової інформації	У рядку заголовка повинний бути присутнім назва довідника
	Повинна бути присутнім кнопка: "Відновити" (якщо є можливість редагувати інформацію)
	Повинні бути присутнім кнопки: "Пошук", "Застосувати фільтр", "Зняти фільтр". Інформація в довіднику, повинна бути відсортована у формі зручної для перегляду користувачеві
	Повинні бути присутнім кнопки навігації по БД: ("Перший запис", "Попередній запис", "Наступний запис", "Останній запис",)
	Якщо по ТЗ передбачена зміна інформації в БД, то повинні бути присутнім кнопки зміни інформації: "Добавити запис", "Видалити запис/ Очистити", "Редагувати запис", "Прийняти редагування/ Зберегти", "Відмінити редагування/ Відмінити"
	Якщо передбачено одержання інформації з БД у форматі Excel, то повинна бути присутнім кнопка - "Експорт у Excel"
	Якщо користувачеві інформація в довіднику пропонується тільки для перегляду, то вихід з ЕФ повинний бути по одній із кнопок - "Закрити" / "ОК"
	Якщо ЕФ припускає вибір із довідника, то повинна бути присутнім кнопка: "Вибрати"
	Якщо в довіднику знищують запис, то повинне бути діалогове вікно, що підтверджує видалення.
	Бажано, щоб на ЕФ виводилася інформація про кількість записів
Елементи управління Загальні питання	Для елементів керування повинні бути встановлені значення за замовчуванням
	Найбільш важливі елементи керування на формі повинні бути розміщені вгорі ліворуч
	Якщо елементи керування неактивні, то вони повинні відображатися сірим кольором

Елемент	Правило підтвердження
(стосуються зручності застосування)	Кожна група повинна містити не менш двох Check Box'ов
	Групи повинні містити не більш 10 перемикачів Check Box
	Користувач повинний мати можливість вибору не більш 1-го перемикача
	Усі перемикачі повинні мати назви, що повинні бути зрозумілі користувачам системи і, по можливості, не містити скорочення (крім загально уживаних у даній предметній області)
List Boxes (Список)	Якщо вікно списку містить більш 50 елементів, то необхідно надати можливість пошуку
	Інформація у вікні повинна бути відсортована у такий спосіб, щоб користувачеві було зручно працювати з нею
	Вікно списку не повинне мати горизонтальну смугу прокручування (якщо можливо)
	Якщо необхідно, то повинна існувати вертикальна смуга прокручування
	Якщо потрібно, то повинна забезпечуватися обробка подвійного кліка мишею
	Повинний бути установлений формат: біле тло , товщина рамки 3 d, шрифт MS Sans Serif 8 pt
ComboBox (Комбінований список)	Порядок сортування елементів повинний бути зручний для користувача
Рядок меню (Pull Down Menus)	У Pull Down Menus повинно бути не більш 12 елементів
Каскадні меню (Popup menus)	Popup menus повинні містити не більш 10 елементів
	Каскадні меню повинні містити не більш 1 рівня глибини
	Popup menus повинні містити не більш 1-го додаткового каскадного меню
Командні кнопки на формах	Бажано, щоб форма містила не більш 6 кнопок
	Усі кнопки, що викликають каскадні форми, повинні мати (....) після імені кнопки
	Назви кнопок (зображення кнопки, якщо воно є), повинні відповідати функціям, які вони виконують. Необхідно, щоб для різних дій, кнопки мали різне зображення. Необхідно, щоб кнопки мали однозначні, зрозумілі підказки.
	Кнопки, що показують додаткову частину форми, повинні мати дві стрілки (>>) після імені
	Неактивні кнопки повинні відображатися сірим кольором
	При необхідності, кожна кнопка повинна мати клавішу-акселератор (крім "OK" і "Відмінити")
	Кнопки повинні мати підказки. Підказки повинні відображати сутність призначення кнопки
	Назва кнопки повинна складатися не більш ніж з 3-х слів
	Командні кнопки не повинні містити малюнків замість імен (одні малюнки)
	Розміщення кнопок у всіх формах повинне бути одноманітно
	Усі кнопки вирівняні або вертикально, щодо правої границі форми - або горизонтально, щодо нижньої границі форми
Вирівнювання кнопок	Кнопка за замовчуванням повинна знаходитися зверху для кнопок, що вирівняні вертикально (першою для кнопок, що вирівняні горизонтально)
Кнопки, вирівняні вертикально (горизон-тально)	Відповідно до використання, кнопки повинні бути впорядковані зверху вниз (зліва направо)
	Кнопка «Довідка» повинна бути розміщена в самому низу (справа)
	Кнопка «Довідка» від інших повинна бути відокремлена, принаймні, висотою 1 кнопки
Радіо-кнопки (необов'язково)	Всі радіо-кнопки повинні мати імена
	Всі імена повинні бути значущі і зрозумілі для користувача системи
	Не повинно бути радіо-кнопок з іменами «та чи ні» як вибір (це повинно бути реалізовано перемикачем)
	Одна з кнопок повинна бути встановлена як значення за умовчанням
	У групі не допускається включати більше 10 радіо кнопок

Елемент	Правило підтвердження
	Імена кнопок повинні мати формат: шрифт – MS Sans Serif 8 pt bold. Якщо на кнопці є малюнок – напис винен розміщується справа
Text Box	Всі поля повинні бути семантично значущі для користувача
	Всі поля повинні бути вирівняні по лівій межі
	Всі підписи повинні бути розміщені над або зліва від полів введення
	Повинні бути колонка або пропуски між статичним полем і полем введення
	Повинен бути встановлений шрифт – MS Sans Serif 8 pt bold
	Всі числа в полях введення вирівняні справа
Поля введення тексту	Всі текстові поля вирівняні зліва
	Поля можуть мати підказки. Підказки повинні відображати суть призначення поля
	Всі поля повинні мати статичні назви (підписи). Назви повинні відображати суть призначення поля і всі назви полів повинні бути розташовані вгорі або зліва від поля введення
Інформація тільки текстових полів	Текстова інформація в однорядкових і багаторядкових полях не повинна бути доступна для редагування
Інформація (довідники)	Інформація довідників: а) повинна бути відсортована за ознакою; що забезпечує зручність роботи з базою
	б) якщо того вимагає завдання, повинна бути реалізована фільтрація записів БД;
	в) повинен бути реалізований пошук
	г) рекомендується використовувати стандартний довідник (THandBook і породжені від нього).
Багаторядкові текстові поля	Для перегляду довгого тексту повинен існувати скролінг
	Рядок, що виводиться, не повинен обрізатися
	Формат поля повинен бути встановлений таким чином: фон білий, рамка 3d, шрифт MS Sans Serif 8 pt

Таблиця 4.3 – Рекомендовані мнемонічні символи для позначення операцій

Операція	Символ
Новий документ	
Видалити	
Вирізати	
Вставити	
Перейменувати	
Редагувати	
Копіювати	
Відновити	
Відмінити	
Пошук	
Додати рядок	
Видалити рядок	
Імпорт	
Експорт	
Прийняти	
Відмінити	
Так	
Ні	
Папка	
Відкрити папку	
Фільтр	
Користувач	
Користувачі	
Додати користувача	
Видалити користувача	

Операція	Символ
Дані користувача	
Настроювання	
Допомога	
Зберегти	
Експорт у Word	
Експорт в Excel	
Сортування	
Сумарна інформація	
Діаграма	
Збільшити	

Варіанти завдань.

№ студента	Застосунок	
	Назва	**Опис**
1,2	Calc2.sln	Консольний калькулятор
3,4	TestsGenerator	Генератор тестів для контролю знань
5,6	Список	Програма для фільтрації списків
7,8	Summary	Складання резюме
9,10	SOFTEST	Розрахунок витрат на створення програмного продукту
11,12	DUMB	Програма для сортування алфавітно-цифрових списків
13,14	Lab5-Edit	Текстовий редактор

Лабораторна робота № 5. Дослідницьке тестування Web-застосунків

Мета роботи: проектування й запуск різних видів тестів для Web-застосунку згідно з евристичною моделлю стратегії дослідницького тестування Дж.Баха

Теоретичні відомсті.

Робота включає такі етапи.

1. Ознайомлення з евристичною моделлю стратегії тестування Дж.Баха.

2.Ознайомлення з типами тестів для Web-застосунку (див. Л6. Веб-застосунок):

- Виявлення класів еквівалентності
- Проектування тестів еквівалентності і граничних значень
- Тестування функцій застосунку
- Тестування стійкості до помилок
- Тестування зручності інтерфейсу включно з тестуванням конфігурації:
  - - на різних моніторах
  - - у різних експлорерах.
- Використання контрольних списків: від IT-Online, див. табл. 5.2, та стандарт Windows, табл. 4.3;
- Тестування безпеки
- Навантажувальне тестування
- Перевірка посилань
- Перевірка HTML-коду.

3. Розробку сценаріїв тестування і тестових наборів.

4. Використання on-line ресурсів тестування веб-застосунків.

4.1. Тестування конфігурації:

4.1.1 На різних експлорерах:

http://browsershots.org/
ipinfo.info/netrenderer
browserling.com
IETester

4.1.2 На різних моніторах

http://viewlike.us/
ScreenFly

4.2. Перевірки валідності верстки сайту: http://validator.w3.org/

4.3 Навантаження: http://loadimpact.com/

5. Тестування безпеки веб-застосунку:

5.1. Виявлення SQL вразливостей.

Суть вразливості в тому, щоб змінити передавану скрипту змінну так, щоб sql запит видав не те, що має за логікою програми, а те, що потрібно хакеру. Для перевірки змінної на безпеку корисно ***вставити в неї символ «'»***. Наприклад, url
ваш_сайт.ru/news/last/451/
корисно змінити на:
ваш_сайт.ru/news/last/4'51/.
Якщо спостерігається повідомлення на зразок
Warning: Supplied argument is not a valid MySQL result resource…
або замість новини з ідентифікатором 451 надається новина з ідентифікатором 4, - має місце помилка порушення безпеки. Апострофи корисно вставляти не тільки в url'и, але у всі поля, куди користувачу дозволено вводити текст.

5.2. Виявлення дефектів щодо завантаження файлів.

Якщо користувач може завантажувати на сайт файли (фото, документи), необхідно перевірити, файли яких типів дозволені для завантаження. Якщо тип завантажуваного файлу не перевіряється, це теж помилка безпеки.

6. Складання звіту про виявлені дефекти у вигляді таблиці 5.1 у файлі Лаб5_Прізвище.doc.

Таблиця 5.1 – Звіт про дефекти Windows-форми

<table>
<tr><td>Вхідні дані</td><td colspan="2">Очікувані дані</td><td colspan="2">Спостережені дані (<u>що</u> спостерігається; <u>в якій частині</u> програми; <u>за яких</u> обставин)</td></tr>
<tr><td>1. Дефекти, виявлені під час тестування функцій</td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td>Функція 1</td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td></td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td>Функція n</td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td>2. Дефекти, виявлені під час тестування стійкості до помилок</td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td></td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td rowspan="4">3. Дефекти, виявлені під час тестування зручності використання</td><td>Дія</td><td>Очікувана реакція</td><td colspan="2">Дійсна реакція</td></tr>
<tr><td></td><td></td><td colspan="2"></td></tr>
<tr><td colspan="2">Порушена вимога (за IT-Online</td><td colspan="2">Зміст порушення</td></tr>
<tr><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td></td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td rowspan="2">4.1 Дефекти, виявлені під час тестування конфігурації (на різних типах моніторів)</td><td colspan="2">Тип монітору</td><td colspan="2">Зміст порушення</td></tr>
<tr><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td rowspan="4">4.2 Дефекти, виявлені під час тестування конфігурації (на різних експлорерах)</td><td colspan="2">Google Chrome</td><td colspan="2"></td></tr>
<tr><td colspan="2">Mozilla Firefox</td><td colspan="2"></td></tr>
<tr><td colspan="2">IE7</td><td colspan="2"></td></tr>
<tr><td colspan="2">IE8</td><td colspan="2"></td></tr>
<tr><td rowspan="2">5. Дефекти, виявлені під час тестування безпеки</td><td colspan="2">Дія</td><td>Очікувана реакція</td><td>Дійсна реакція</td></tr>
<tr><td colspan="2"></td><td></td><td></td></tr>
<tr><td rowspan="2">6. Дефекти, виявлені під час тестування посилань</td><td colspan="4">Адреса посилання</td></tr>
<tr><td colspan="4"></td></tr>
<tr><td colspan="5">7. Дефекти, виявлені під час перевірки HTML-коду</td></tr>
<tr><td>№</td><td colspan="4"></td></tr>
<tr><td colspan="5">8. Результати навантажувального тестування</td></tr>
<tr><td colspan="5"></td></tr>
<tr><td colspan="2">Гранична кількість користувачів</td><td colspan="2">Мінімальний час відгуку</td><td colspan="2">Максимальний час відгуку</td></tr>
<tr><td></td><td></td><td>1 користувач, його дія</td><td>Гранична кількість користувачів</td><td>1 користувач, його дія</td><td>Гранична кількість користувачів</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="6">9. Дефекти, виявлені під час тестування документації</td></tr>
<tr><td>№</td><td colspan="5"></td></tr>
</table>

Таблиця 5.2 - Контрольні питання для перевірки зручності застосування Web-застосунків (від компанії IT-Online)

1. Архітектура і навігація сайту
Чи відповідає структура сайту цілям, для досягнення яких він призначений?
Чи зрозуміла схема навігації?
Чи можна визначити, в якому місці сайту ви перебуваєте?
Чи легко знайти на сайті потрібну інформацію?
Чи задовільна кількість елементів в панелях навігації?
Чи логічно відсортовані елементи?
Чи відповідають назви гіперпосилань назвам сторінки?
Чи виразно виділені гіперпосилання ?
Чи виразно виділено посилання на головну сторінку?
Чи існує можливість пошуку інформації на сайті?
Чи існує карта сайту?
Чи кожна сторінка дозволяє зрозуміти, на якому сайті ви перебуваєте?
Чи можна управляти навігацією по сайту?
2. Планування і дизайн сайту
Чи не перевищує розмір сторінки розмір вікна?
Чи повторюється схема планування на всіх сторінках?
Чи існує виразний фокус на кожній сторінці?
Чи видно візуальне планування?
Чи ефективно використовується вирівнювання і угрупування?
Чи достатньо чіткий контраст?
Чи не громіздке планування?
Чи подобається вам сайт естетично?
3. Зміст (контент) сайту
Чи зрозумілі і лаконічні тексти на сайті?
Чи організований текст у вигляді невеликих блоків?
Чи зустрічаються в тексті граматичні і орфографічні помилки і друкарські помилки?
Чи містять сторінки текст, що вводиться?
Чи підтримують мультимедійні компоненти задачі користувача?
Чи зрозумілі одиниці вимірювання, використовувані на сайті?
Чи подані на сайті час і дата створення сторінок?
Чи подані на сайті номера контактних телефонів?
Чи подані на сайті адреси з поштовими індексами?
4. Форми і взаємодія
Чи відповідають форми задачам користувача?
Чи підтримують діалоги логічну послідовність кроків?
Чи очевидна кнопка або посилання для переходу до наступного кроку діалогу?
Чи всі елементи форм використовуються за призначенням?
Чи згруповані елементи форми по смисловому призначенню?
Чи зрозуміло виглядає кнопка відправки форми?
5. Графіка
Чи прийнятна якість використовуваної графіки ?
Чи всі графічні елементи мають альтернативні текстові написи?
Чи містять графічні елементи інформацію про розмір файлу?
Чи оптимізовані графічні елементи для передавання в Інтернет?
Чи реагують графічні елементи на рухи миші?
Чи використовується анімація? Чи прийнятний об'єм графічних файлів?
6. Кольори
Чи задовільний вибір кольорів для сайту?
Чи не занадто багато кольорів?
Чи використовуються кольори логічно і послідовно?
Чи адекватно розрізняються кольори в чорно-білому режимі?
7. Оформлення тексту
Чи зрозумілі тексти?
Чи прийнятний розмір шрифту?
Чи прийнятний колір шрифту і чи достатньо він контрастний?
Чи відформатовано текст так, щоб у рядку було від 10 до 12 слів?
Чи достатня ширини поля довкола тексту?

8. Стійкість до помилок
Чи повинен користувач що-небудь запам'ятовувати, переходячи між сторінками?
Чи надається попередження при спробі здійснення незворотних дій?
Чи можна відмінити ризиковані дії?
Чи перехоплюються поточні помилки локально, без звернення до серверу?
Чи містять сторінки з повідомленням про поточні помилки корисну інформацію?
Чи містять сторінки з незадовільними результатами пошуку поради щодо зміни його умов?
Чи існує система контекстної допомоги (довідки)?
Чи структурована допомога за задачами користувача? Чи пояснює вона користувачу, як здійснити ту або іншу дію?
9. Платформа і особливості реалізація
Чи достатньо швидко відбувається завантаження сторінок (від 3 до 15 секунд)?
Чи всі гіперпосилання працюють правильно?
Чи існують пошкоджені графічні елементи?
Чи написаний текст на сторінках так, щоб його могли знайти пошукові системи?
Чи працює сайт з різними браузерами користувача?
Чи працює сайт на моніторах високої й низької дозвільної здатності?
Чи використовуються нестандартні плагіни (plug-in)? Чи є вони необхідними і корисними?

Варіанти завдань.

№ студента	**Застосунок**
1-3	Кредитний калькулятор банку УПБ http://upb.ua/loan.php?ln=ua
4-6	Сайт студентської фабрики програм КНУ http://sestudy.edu-ua.net/ru/
7-10	Веб-застосунок для обчислення витрат на розроблення программного забезпечення за конструктивними моделями Б.Боема http://csse.usc.edu/tools/COCOMOSuite.php
11-14	Веб-застосунок для on-line формування запиту на енергоносії компанії Energy-Telecom Energy-Telecom.org

Лабораторна робота № 6. **Автоматизоване документування результатів тестування Windows-застосунків та Web-застосунків**

Мета роботи: опрацювання складання звітів про проблему за результатами тестування в середовищі інструментальних засобів ведення дефектів.

Зміст роботи:

1. Ознайомлення з форматом Звіту про проблему та рекомендаціями щодо його складання, рекомендованим у лекційному матеріалі.
2. Складання Звітів за цим форматом у середовищі автоматизованої системи відстеження проблем ***Mantis*.**

Настанови щодо складання звіту.

1. Переконатися, що обрано відповідний проект
2. В полі **Category** вибрати потрібну категорію для бага
   a. Calculation — якщо проблема стосується розрахунку
   b. Usability — якщо проблема стосується зручності використання форми
   c. General — якщо перші два варіанти непридатні
3. В полі **Reproducibility** зазначити, наскільки часто відтворюється проблема:
   a. always — завжди
   b. sometimes — час від часу
   c. random — випадковим чином
   d. have not tried — не будо спроб відтворити
   e. unable to reproduce — не вдається відтворити
   f. N/A — для даної задачі не актуально
4. **Severity** — важливість задачі з точки зору тестера
   a. feature — не баг, особливість програми.
   b. trivial — дрібний недолік, що не призводить до проблем користувача
   c. text — помилки в тексті інтерфейсу та документації
   d. tweak — незначне вдосконалення продукту
   e. minor — неістотна проблема, яка не впливає на основну функціональність
   f. major — проблема, що зачіпає основну функціональність
   g. crash — внаслідок проблеми робота з програмною системою утруднена
   h. block — работа с програмною системою неможлива, функції не виконуються.
5. **Priority** — пріоритет. Зазвичай визначається старшим тестером/менеджером/архітектором програмної системи.
   a. none — без пріоритета. Найімовірніше, не буде виправлятися.
   b. low, normal, high, urgent — пріоритет виправлення помилки.
   c. immediate — першочергове виправлення помилки.
6. **Profile (Environment)** — програмно-апаратне оточення, в якому під час тестування спостерігається проблема.
7. **Assign To** — login програміста, що буде виправляти помилку. В рамках роботи писати свій login.
8. **Summary** — стислий опис, заголовок Звіту про проблему.
9. **Description** — докладний опис проблеми.
10. **Steps to reproduce** — кроки з відтворення. Пишутся за форматом:
    a. Steps — власне дії з відтворення;
    b. Expected result — очікуваний результат (що має спостерігатися за нормальної роботи програмної системи)
    c. Result — отриманий (спостережений) результат. Result ≠ Expected Result.
11. **Additional Information** — довільна інша інформація щодо проблеми.
12. **Upload File** — долучення файла до Звіту про проблему (например, скріншот).

13. **View Status** — login учасника розробки, що може переглядати Звіт. Залишити значення public
14. **Report Stay —** включити, якщо планується відразу додати ще задачу.

Завдання для самостійної роботи.

1. Відкрити систему відстеження проблем Mantis за адресою http://aderkin.ru/mantis/. Ввести запитані реєстраційні дані, пасля чого на e-mail прийде посилання для встановлення особистого пароля. Им'я користувача має бути у форматі **name.surname.**
2. Зареєструватися в Mantis, встановивши свій пароль.
3. Активувати акаунт, ввійти в середовище Mantis.
4. Відкрити форму «Report Issue». Послідовно обрати проекти, відповідні застосункам, тестованим впродовж ЛР №4 та ЛР №5.
5. Занести у Mantis дефекти, виявлені для Windows-форми (у ЛР №4) та Web-застосунку (у ЛР №5), формуючи ***по одному звіту для кожного дефекту***.

Лабораторна робота № 7. Оцінка якості програмної системи за результатами її тестування

Мета роботи: оцінка зовнішньої якості Windows-форми та Web-застосунку за результатами їх дослідницького та системного тестування.

Зміст роботи.

1. Ознайомлення з переліком зовнішніх метрик якості на підставі частини 2 ***ISO/IEC 9126-2:2003.*** **Software Engineering - Product Quality - Part 2: External Metrics** та відомостей у лекційному матеріалі.

2. Добір у цьому переліку метрик, які можна визначити на підставі результатів ЛР №4 і ЛР №5, для підхарактеристик якості, регламентованих стандартом ***ISO/IEC 15026*:1998**, якщо рівень цілісності за стосунку ***низький***, що наведені в табл. 7.1.

3. Обчислення обраних метрик, підхарактеристик якості та інтегрального показника якості для Windows-форм, тестованих впродовж ЛР №4, та Web-застосунків, тестованих впродовж ЛР №5, за умов:

1) для Windows-форм функціональність удвічі важливіша за зручність використання, а зручність використання удвічі важливіша кожну з решти характеристик у табл. 7.1;

2) для Web-застосунку функціональність і зручність використання разом удвічі важливіші за решту характеристик у табл. 7.1.

4. Документування результатів обчислень у файлі **Лаб7_Прізвище.doc**.

Таблиця 7.1 - Підхарактеристики якості за стандартом ***ISO/IEC 15026*:1998**

Рівень цілісності	Характеристика якості	Найважливіша підхаракте-ристика	Вибрана зовнішня метрика	Можливий критерій приймання
Низкий	1. Функціональність	Точність	Число отриманих точних результатів в % від очікуваного числа	95%
	2. Зручність використання	Керованість	Число чітких повідомлень в % від загального числа проглянутих повідомлень	80%
	3. Переносність	Здатність до адаптації	Число модулів, що підлягають повторній компіляції, з урахуванням загального числа модулів, переносимих на нову платформу	< 6 модулів
	4. Ефективність	Реактивність	Проміжок часу від посилки запиту до отримання відповіді системи	< 5 секунд
	5. Надійність	Відмовостійкість	Число відмов, які виникли через помилки у вхідних даних, в % від загального числа запусків системи з введенням неправильних даних	25%
	6. Супроводженність	Не потрібна	Не потрібна	-