**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра програмних систем і технологій**

**О. С. Бичков, Т.В. Ковалюк**

**Методичні вказівки до лабораторних занять з дисципліни**

**«Вступ до об’єктно-орієнтованого програмування.   
Мова С# і платформа MS.Net Framework »**

для студентів спеціальності

121 «Інженерія програмного забезпечення»

освітнього рівня «бакалавр»

Київ 2021

# Розділ 1. Введення в мову програмування C#

## Лабораторна робота № 1 Методи введення, виведення, перетворення типів, оператори: умовний, вибору, циклу в мові C#

### Рейтинг лабораторної роботи №1

|  |  |  |  |
| --- | --- | --- | --- |
| № п.п | Вид діяльності студента | Рейтинговий бал | Deadline |
| 1 | Написання коду з 6 завдань | 0,5\*6 = 3 | Лютий |
| 2 | Захист роботи | 0,5 |
| 3 | Звіт з роботи | 0 |  |
| Разом за роботу | | 3,5 |  |

### Мета роботи:

1. Прості обчислення з перетворенням типів даних. Метод Parse

2. Прості арифметичні вирази

3. Клас Convert

4. Перетворення з арифметичного типу в тип string. Метод ToString

### Методичні рекомендації до виконання лабораторної роботи

1. Прочитайте лекцію.

2. Прочитайте цю лабораторну роботу та виконайте наведені в ній приклади (вони всі працездатні)

3. Зверніть увагу на те, що при введенні чисел з консолі у змінні арифметичного типу потрібно застосовувати явне приведення типу, так як неявного перетворення з типу **string** до арифметичного немає.

4. При виконанні завдань зверніть увагу на оператор **switch** (перемикач). В С# тип виразу в цьому операторі може бути не тільки цілочисельним, а і типу char або string.

### Порядок виконання роботи

1. Створити директорію Lab1, в якій буде розміщуватися проект цієї лабораторної роботи.

2. Виконати завдання свого варіанту у вигляді одного проекту

### Приклади виконання завдань

#### 1. Прості обчислення з перетворенням типів даних. Метод Parse

Розглянемо приклад створення простого консольного застосунку, який виконує прості арифметичні обчислення. Запустимо Visual Studio, виберемо тип проекту Console Application, дамо йому назву **Lab1\_1** і натиснемо кнопку **OK**.

Створимо програму, яка вводить два цілі числа, знаходить їх суму і виводить результат на консоль. Потім вводить два дійсні числа, обчислює їх частку і також виводить на консоль.

Ось таким може бути код цієї програми. Його можна просто скопіювати в VS і виконати.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Lab2\_1

{

class Program

{

static void Main(string[] args)

{

//Знаходження суми цілих чисел

Console.WriteLine("Введіть перше число");

int a1 = int.Parse(Console.ReadLine());

Console.WriteLine("Введіть друге число");

int a2 = int.Parse(Console.ReadLine());

Console.WriteLine("Сума чисел a1+a2= " + (a1 + a2));

//Знаходження частки дійсних чисел

Console.WriteLine("Введіть перше число");

double d1 = double.Parse(Console.ReadLine());

Console.WriteLine("Введіть друге число");

double d2 = double.Parse(Console.ReadLine());

Console.WriteLine("Частка чисел d1/d2= " + (d1/d2));

Console.ReadKey();

}

}

}

Дані, які вводяться з консолі мають тип string (це рядки тексту). Тому введені числа потрібно перетворити в арифметичний тип. В нашому прикладі виконується явне перетворення типів з типу string в типи **int** і **double** з використанням методу Parse і неявне - при виведенні результатів обчислень на консоль.

Розглянемо ще приклад явного перетворення типу.

**Приклад**

string str = Console.ReadLine();

int i = int.Parse(str);

float f = float.Parse(str);

double d = double.Parse(str);

#### 2. Прості арифметичні вирази

Розглянемо дещо складніший приклад обчислення арифметичного виразу. Нехай нам потрібно обчислити значення поліному р=ax3+bx2 + cx+d для будь-якого дійсного x (вводиться з консолі). Значення a,b,c,d є цілими і ініціалізуються в коді. Текст програми може бути таким:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Lab2\_2

{

class Program

{

static void Main(string[] args)

{

//обчислити значення поліному р=ax^3+bx^2+cx+d

int a = 100;

int b = 20;

int c = 25;

Console.WriteLine("Введіть x");

double x = double.Parse(Console.ReadLine());

double p = a \* x \* x \* x + b \* x \* x + c \* x + 30;

Console.WriteLine("p = " + p);

Console.ReadKey();

}

}

}

#### 3. Клас Convert

Клас Convert містить статичні методи, які також можна використовувати для перетворень типів. Розглянемо приклад використання цього класу.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Lab1\_2

{

class Program

{

/// <summary>

/// Тестування методів класу Convert

/// </summary>

static void Main(string[] args)

{

string s;

byte b;

int n;

double x;

bool flag;

char sym;

DateTime dt;

sym = '7';

s = Convert.ToString(sym);

x = Convert.ToDouble(s);

n = Convert.ToInt32(x);

b = Convert.ToByte(n);

flag = Convert.ToBoolean(b);

x = Convert.ToDouble(flag);

s = Convert.ToString(flag);

s = "300";

n = Convert.ToInt32(s);

s = "14.09";

s = "14.09.2008";

dt = Convert.ToDateTime(s);

}

}

}

#### 4. Перетворення з арифметичного типу в тип string. Метод ToString

Так як метод ToString() визначений в базовому класі object, він має свої реалізації для всіх арифметичних типів даних. Далі наведено простий приклад з лекції 2, який демонструє застосування цього методу.

В першому варіанті використовується **явне** перетворення типу з арифметичного до типу string, а у другому - **неявне**.

Варіант 1.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Lab2\_3

{

class Program

{

/// <summary>

/// Демонстрація перетворення в рядок

/// даних різного типа.

/// </summary>

static void Main(string[] args)

{

string name;

int age;

double salary;

name = "Василь Іванов";

age = 20;

salary = 2700;

string s = "Ім'я: " + name +

". Вік: " + age.ToString() +

". Зарплата: " + salary.ToString();

Console.WriteLine(s);

Console.ReadKey();

}

}

}

Варіант 2

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Lab2\_3

{

class Program

{

/// <summary>

/// Демонстрація перетворення в рядок

/// даних різного типа.

/// </summary>

static void Main(string[] args)

{

string name;

int age;

double salary;

name = "Василь Іванов";

age = 20;

salary = 2700;

string s = "Ім'я: " + name +

". Вік: " + age +

". Зарплата: " + salary;

Console.WriteLine(s);

Console.ReadKey();

}

}

}

#### 5. Створення консольних проектів з використанням операторів if

Створимо консольний проект для організації діалогу таким чином, щоб реалізувати наведений нижче алгоритм.

* Вивести на консоль запрошення для введення імені.
* Ввести своє ім'я і зберегти в текстовому рядку: string myName.
* Якщо нічого не введено, то вивести повідомлення про це і завершити роботу.
* Якщо щось введено, то вивести рядок привітання.
* Вивести рядок із запитом віку
* Якщо нічого не введено, то вивести повідомлення про це і завершити роботу.
* Якщо вік введено, привести число до цілого (int myAge).
* Якщо число myAge<15, вивести повідомлення "Ви ще не студент. "
* Якщо число myAge>40, вивести повідомлення "Вчитися ніколи не пізно!"

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleHello1

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введіть ім'я");

string myName = Console.ReadLine();

if (myName.Length == 0)

Console.WriteLine("Ви нічого не ввели, прощавайте");

else

Console.WriteLine("Здрастуйте, "+ myName);

Console.WriteLine("Скільки Вам років?");

int myAge = int.Parse(Console.ReadLine());

if (myAge == 0)

Console.WriteLine("Ви нічого не ввели, прощавайте");

else

{

if (myAge<15)

Console.WriteLine("Ви ще не студент");

else if (myAge>40)

Console.WriteLine("Вчитися ніколи не пізно!");

else

Console.WriteLine("Ваш вік" +myAge);

}

Console.ReadKey();

}

}

}

В цьому прикладі використовується декілька операторів розгалуження **if** (останні три вкладені). Зверніть увагу, що у C# в операторі **if** діють ті самі правила синтаксису, що і в С++, якщо в тілі оператора є лише один оператор блоку, то дужки не потрібні, наприклад:

if (myName.Length == 0)

Console.WriteLine("Ви нічого не ввели, прощавайте");

else

Console.WriteLine("Здрастуйте, "+ myName);

Крім цього виконується явне перетворення типів з типу **string** в тип **int** з використанням методу Parse і неявне при виведенні віку на консоль

#### 6. Створення консольних проектів з використанням операторів switch

Розробимо простий консольний калькулятор для виконання арифметичних операцій. Вхідні дані будемо вводити з консолі, результати виводити на консоль. Приклад коду з лекції

**Консольний калькулятор на 4 дії**

using System;

namespace ConsoleCalculator

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello World!");

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 = true;

switch (op)

{

case '+': res = a + b; break;

case '-': res = a - b; break;

case '\*': res = a \* b; break;

case '/': res = a / b; break;

default: res = double.NaN; ok = false; break;

}

if (ok) Console.WriteLine("Результат: " + res);

else Console.WriteLine("Неприпустима операція");

}

}

}

Зверніть увагу, що тип виразу в операторі switch є **char**. Також зверніть увагу, що перетворення з типу string до типу char повинно бути явним, тому що тип char є типом-значенням, а тип string – посилковим.

**char op = (char)Console.Read();**

Для виходу з оператора switch при виконанні умови використовується оператор переходу break.

#### 7. Створення консольних проектів з використанням операторів циклу

**7.1. Цикл з передумовою while**

Розглянемо приклад з лекції 4, в якому використовується цикл з передумовою. Потрібно написати програму, яка виводить для аргументу х, що змінюється в заданих границях із заданим кроком, таблицю значень наступної функції:



Назвемо xn – початкове значення аргументу, xk –кінцеве значення аргументу, dx – крок зміни аргументу, t – параметр. Усі величини є дійсні числа типу double. Програма повинна виводити таблицю, що складається з двох стовпців: значень аргументу і відповідних ним значень функції.Таблиця повинна мати заголовок.

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

double Xn = -2, Xk = 12, dX = 2, t = 2, y;

Console.WriteLine("| x | y |"); // заголовок таблиці

double x = Xn;

while (x <= Xk)

{

y = t;

if (x >= 0 && x < 10) y = t \* x;

if (x >= 10) y = 2 \* t;

Console.WriteLine("| {0,6} | {1,6} |", x, y);

x += dX;

}

Console.ReadKey();

}

}

}

Зверніть увагу, що в тілі циклу використовується два оператори **if**. На кожному кроці циклу обчислюється значення функції. Цикл завершиться коли умова циклу не буде виконана (тобто значення x стане більше 12).

**3.2. Цикл з пост-умовою do…while**

Цей тип циклу застосовується в тих випадках, коли тіло циклу необхідно обов'язково виконати хоч б один раз.

*Приклад 4.7. з лекції 4* демонструє використання цього циклу. В цьому прикладі на консоль виводиться текст "Будете вчитися?" до тих пір поки не буду введено"y".

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

char answer;

do

{

Console.WriteLine("Будете вчитися?");

answer = (char)Console.Read();

Console.ReadLine();

} while (answer != 'y');

}

}

}

**3.3. Цикл з параметром (for)**

Цей тип циклу використовується коли відомі границі циклу (нижня і верхня). *Приклад 4.8 з лекції 3* демонструє використання цього циклу. В цьому прикладі обчислюється сума чисел від 1 до 100. Спочатку створюється змінна цілого типу s, яка ініціалізується нулем. Потім в тілі циклу на кожному кроці до неї додається чергове число: 1,2,3,....100.

int s = 0;

for ( int i = 1; i <= 100; i++ )

s += i;

Зверніть увагу, що в тілі циклу відсутні фігурні дужки блоку. Це тому, що в циклі лише один оператор. Також зверніть увагу на використання операції інкремента:

s += i;

замість призначення: s=s+i;

Обидва оператори дозволені в C# так само як і в C++.

Розглянемо ще один приклад використання оператора циклу з параметром.

**Постановка задачі.**

Написати метод обчислення функції sin(x), використовуючи розкладання в ряд Тейлора за формулою:

\sin x = \sum^{\infin}_{n=0} \frac{(-1)^n}{(2n+1)!} x^{2n+1}

При реалізації задачі використати цикл **for**.

Число x – це значення кута в радіанах, n – кількість членів ряду. Числа **x, n** ввести з консолі. Обчислення факторіала виконати в окремому методі. Порівняти отримане значення із стандартним методом обчислення sin(x).

В цьому прикладі для обчислення ступеня використовується метод Math.Pow(). Статичний клас Math містить методи реалізації математичних функцій.

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();

}

}

}

### Варіанти завдань для лабораторної роботи № 1.

**Номер варіанту відповідає номеру прізвища студента у списку групи**

Написати в C# консольний застосунок, що реалізує завдання згідно з варіантом.

Виконання завдання 1 вимагає використання методів класу **Console.**

Виконання завдання 2 вимагає використання методів перетворення типів.

Виконання завдання 3 вимагає використання методів класу **Math.**

Виконання завдання 4 вимагає використання оператору **if**

Виконання завдання 5 вимагає використання оператору **switch**

Виконання завдання 6 вимагає використання оператору циклу

|  |  |
| --- | --- |
| **№ варіанту** | **Зміст завдання** |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Обчислення значення поліному *p = a\*x^5 − 1 / b\*x^4+c\*x + d*. Значення *a, b, c, d* – дійсні числа, які слід ініціалізувати у функції, значення змінної *x* увести з консолі. Результат вивести на консоль. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. Обчислити значення функції в точці *х*, значення якої ввести з консолі, Якщо введено не число, вивести повідомлення «Ввести число».      1. Написати функцію, яка в залежності від порядкового номера місяця (1,2,...12) виводить на екран його назву (січень,...грудень). 2. Дано натуральне число n. Обчислити добуток перших n членів ряду. |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Обчислення значення поліному *p = a\*x^4 − b\*x^3 + c\*x + d*. Значення *a, b, c, d* – дійсні числа, які слід ініціалізувати у функції, значення змінної *x* увести з консолі. Результат вивести на консоль. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. Обчислити значення функції в точці *х*, значення якої ввести з консолі:      1. Написати функцію, яка в залежності від порядкового номера дня тижня (1,2,...7) виводить на екран його назву (понеділок,...) 2. Обчислити суму чисел в заданому у функції діапазоні. Значення чисел вводити з консолі. Якщо значення, що вводиться, виходить за межі заданого діапазону, вивести повідомлення про помилку і повторити введення. Не застосовувати масиви. |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Обчислення значення поліному p=23,5\*x^5 + 30\*x^4+10\*x+87,3. Значення *a, b, c, d* – дійсні числа, які слід ініціалізувати у функції, значення змінної *x* увести з консолі. Результат вивести на консоль. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. Обчислити значення функції в точці *х*, значення якої ввести з консолі:      1. Написати функцію, яка в залежності від порядкового номера місяця (1,2,...12) виводить на екран пору року (зима, весна....) 2. Дано натуральне число n. Обчислити   , |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Обчислення значення поліному p = 3,5\*x^4 + 3\*x^3+10\*x^2+8,3. Значення *a, b, c, d* – дійсні числа, які слід ініціалізувати у функції, значення змінної *x* увести з консолі. Результат вивести на консоль. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. Обчислити значення функції в точці *х*, значення якої ввести з консолі:      1. Написати функцію, яка в залежності від одиниці виміру часу (1,2,...24) виводить на екран частини доби (ранок, день,....) 2. Задати з консолі оцінки студента з 10 дисциплін. Обчислити загальну суму балів, найгіршу і найкращу оцінку. Не використовувати масиви. |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Обчислити площу прямокутника за заданими сторонами. Результат вивести на консоль. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними *a, b, x,* значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від порядкового номера кольору у спектрі (1,2,...7) виводить його назву (червоний, помаранчевий, жовтий, зелений, блакитний, синій, фіолетовий) і код RGB. 2. N суддів поставили різні оцінки одному спортсмену. Обчислити середній бал спортсмена, видаливши найменшу та найвищу суддівські оцінки. Оцінки вводити з клавіатури, Не використовуючи масиви, обраховувати суму введених значень, найменше та найбільші значення. |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Знайти довжини всіх медіан і бісектрис трикутника, якщо значення сторін трикутника введені з клавіатури. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. Обчислити значення функції в точці *х*, значення якої ввести з консолі:      1. Написати функцію, яка виводить назву навчальної дисципліни в залежності від уподобань студента, що задаються пріоритетом 1,2,...5 дисципліни. 2. Дано натуральне число n і дійсне число x >0. Обчислити суму для заданого х: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Обчислити відстань від точки (x0, y0) до точки (х1, y1), значення координат яких введені з консолі. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. Обчислити значення функції в точці *х*, значення якої ввести з консолі:      1. Написати функцію, яка виводить на консоль назву навчального закладу в залежності від середнього балу ЗНО та пріоритетів (1,2,…,5), заданих абітурієнтом. 2. Обчислити  - число Фібоначчі з номером *n*, де: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Знайти об'єм циліндра, якщо значення радіусу його основи та висоти введені з консолі. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. Обчислити значення функції в точці *х*, значення якої ввести з консолі:      1. Написати функцію, яка виводить на консоль назву університету в залежності від його консолідованим рейтингом (1,2,…,5) за 2020, що визначається інформаційним ресурсом "Освіта.ua". 2. Дано натуральне число n. Обчислити суму перших 2n членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Обчислити відстань від точки (x0, y0) до прямої *ax + by + c*= 0. Значення координат точки та коефіцієнтів a, b, c прямої ввести з консолі. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними *a, b, x,* значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка виводить на консоль назву країни в залежності від рейтингу (1,2,…,5) щастя її населення за 2020, що визначається Social Progress Index (<https://minfin.com.ua/ua/2020/10/11/53857422/> ) 2. Дано натуральне число n і дійсне число x >0. Обчислити суму перших 2n членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Знайти довжини всіх висот трикутника, якщо значення сторін a, b, c трикутника введені з консолі 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними *a, b, x,* значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка виводить на консоль прізвище студента за його семестровим рейтингом (від 60 до 100) за результатами сесії. 2. Дано натуральне число n і дійсне число x >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Знайти об'єм конуса, якщо значення радіусу його основи та висоти ведені з консолі. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними *a, b, x,* значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від назви кольору та сигналів світлофора виводить на консоль назву дії, яку має виконати водій автомобіля (їхати, чекати, зупинитися, повертати, зменшити швидкість тощо). 2. Дано натуральне число n і дійсне число x >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Обчислити площу поверхні () сфери за значенням з консолі радіусу **r**. Результат вивести на консоль. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними *a, b, x,* значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від номеру ІТ-спеціальності (121, 121, …, 126) виводить на консоль її назву. 2. Дано натуральне число n і дійсне число x >0. Обчислити добуток членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Тіло починає рухатися без початкової швидкості з прискоренням *a*. Обчислити відстань, яку тіло пройде за час *t* від початку руху. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними *a, b, с, x,* значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від назви місяця (січень, лютий, …) виводить на консоль його порядковий номер. 2. Дано натуральне число n і дійсне число x >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Висота ромба, проведена з вершини тупого кута, ділить сторону навпіл. Знайдіть меншу діагональ, якщо значення периметра ромба введене з клавіатури. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними *a, x,* значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від назви кольору у спектрі (червоний, помаранчевий, жовтий, зелений, блакитний, синій, фіолетовий) виводить його порядковий номер і код RGB. 2. Дано натуральне число n і дійсне число x >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Написати функцію, яка визначає, чи пройде куля радіуса *r* в квадратний отвір зі стороною *a*. Дійсні значення r, a увести з консолі. Вивести на консоль відповідне повідомлення. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними *x,* значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від номеру року виводить на консоль назву країни, співак якої став переможцем конкурсу Євробачення, наприклад, у 2016 році перемогла Україна, 2017 рік - Португалія, і т. д. 2. Дано натуральне число n і дійсне число x >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою **меню**, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати в одному проекті **Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Написати функцію, яка визначає, чи пройде куб з ребром *a* в круглий отвір радіуса *r*. Дійсні значення *а,* *r* увести з консолі. Вивести на консоль відповідне повідомлення. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними x, значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від назви телеканалу виводить на консоль назву холдингу (власника), наприклад, канал «Рада», власник «Верховна Рада України», канал «1+1», холдинг «1+1 Media» і т. д. 2. Дано натуральне число *n* і дійсне число *x* >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою **меню**, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати в одному проекті **Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Написати функцію, яка визначає, з якою швидкістю спортсмен увійде у воду, стрибаючи з *n* метрової вежі, якщо спортсмен падає з прискоренням a=9,81м/с^2, початкова швидкість v0 = 0? Значення *n* увести з консолі. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними x, значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від номеру пальця на руці виводить його назву (великий, вказівний, середній, безіменний, мізинець). 2. Дано натуральне число *n* і дійсне число *x* >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою **меню**, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати в одному проекті **Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Написати функцію, яка визначає, подібність двох трикутників за трьома сторонами, значення яких введені з консолі. Вивести на консоль відповідне повідомлення. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними x, значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від назви країни виводить її рейтинг та індекс рівня освіти (<https://gtmarket.ru/ratings/education-index> ). 2. Дано натуральне число *n* і дійсне число *x* >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою **меню**, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати в одному проекті **Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Написати функцію, яка визначає, подібність двох трикутників за двома сторонами та кутом між ними. Значення двох сторін та кутів трикутників увести з консолі. Вивести на консоль відповідне повідомлення. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними x, значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від назви країни виводить її рейтинг шкільної освітньої грамотності за версією PISA (<https://factsmaps.com/pisa-2018-worldwide-ranking-average-score-of-mathematics-science-reading/> ). 2. Дано натуральне число *n* і дійсне число *x* >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою **меню**, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати в одному проекті **Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e=mail. 2. Написати функцію, яка визначає приналежність точки з координатами (*x,y*) колу з радіусом *r* та центром на початку координат. Значення x,y,r ввести з консолі. 3. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними x, значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від назви країни виводить її рейтинг освітньої грамотності за версією PISA ((<https://factsmaps.com/pisa-2018-worldwide-ranking-average-score-of-mathematics-science-reading/>). 2. Дано натуральне число *n* і дійсне число *x* >0. Обчислити суму членів ряду: |

### Література

1. О.С.Бичков, Є.В.Іванов Об’єктно-орієнтоване програмування мовою C#/

2.C# 2005 и платформа .Net 3.0 для профессионалов. Нейгел К., Ивьен Б. и др. –М.: ООО “И.Д. Вильямс", 2008. –1376 с.

## Лабораторна робота № 2 Масиви, матриці, рядки. Методи класів Console, Convert, Random, Math в C#

### Рейтинг лабораторної роботи №2

|  |  |  |  |
| --- | --- | --- | --- |
| № п.п | Вид діяльності студента | Рейтинговий бал | Deadline |
| 1 | Написання коду з 9 завдань | 0,5\*9=4,5 | Лютий |
| 2 | Захист роботи | 0,5 |
| 3 | Звіт з роботи | 0 |  |
| Разом за роботу | | 5 |  |

### Мета роботи:

1. Робота з масивами в C#

2. Оператор foreach

3. Генерація випадкових чисел

4. Робота з математичними функціями

5. Лінійний пошук

6. Двійковий пошук

### Методичні рекомендації до виконання лабораторної роботи

1. Прочитайте лекцію.

2. Прочитайте цю лабораторну роботу та виконайте наведені в ній приклади (вони всі працездатні)

3. При виконанні завдань зверніть увагу на приклад застосування до масивів циклу **foreach**

4. Також зверніть увагу на опис і реалізацію методу бісекцій (ділення навпіл) знаходження коренів нелінійних рівнянь. Те також задача на одновимірні масиви і цикли

5. Для поглибленого вивчення цього матеріалу прочитайте [1, 2].

### Порядок виконання роботи

1. Створити директорію Lab2, в якій будуть розміщуватися проекти цієї лабораторної роботи.

2. Виконати завдання свого варіанту у вигляді одного консольного проекту

3. Зафіксувати результати у Контрольній роботі 1

### Приклади виконання завдань

#### 1. Робота з масивами в C#

Розглянемо приклад програми з лекції, яка визначає суму і кількість від'ємних елементів, а також максимальний елемент масиву, що складається з 6 цілочисельних елементів.

Тут елементи масиву **а** ініціалізуються при створенні масиву. Далі в циклі з параметром елементи масиву виводяться на консоль. Зверніть увагу, що для виводу використовується метод Console.Write а не Console.WriteLine, тому всі елементи виводяться в один рядок. Символ табуляциії "\t" в методі Console.Write() розділяє елементи при виводі.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Console\_Lab4

{

class Program

{

static void Main(string[] args)

{

const int n = 6;

int[] a = new int[n] { 3, 12, 5, -9, 8, -4 };

Console.WriteLine( "Початковий масив:" );

for ( int i = 0; i < n; ++i )

Console.Write( "\t" + a[i] );

Console.WriteLine();

long sum = 0; // сума від'ємних елементів

int num = 0; // кількість від'ємних елементів

for ( int i = 0; i < n; ++i )

if ( a[i] < 0 )

{

sum += a[i];

++num;

}

Console.WriteLine( "Сума від'ємних = " + sum );

Console.WriteLine( "Кількість від'ємних = " + num );

int max = a[0]; // максимальний елемент

for ( int i = 1; i < n; ++i )

if ( a[i] > max ) max = a[i];

Console.WriteLine( "Максимальний елемент = " + max );

Console.ReadKey();

}

}

}

Розглянемо ще одну "класичну" задачу – генерація чисел Фібоначчі: послідовності чисел, яка задовольняє умовам

F1 = 1; F2 = 1; Fk = Fk-1 + Fk-2 для k>2.

Цю задачу можна реалізувати з використанням масивів. В наступному прикладі кількість чисел вводиться з консолі, потім створюється одновимірний масив вказаної розмірності, який заповнюється згенерованими числами.

##### Генерація чисел Фібоначчі

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Console\_Lab4

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введіть кількість чисел послідовності");

int n = int.Parse(Console.ReadLine());

int[] fibonachi = new int[n];

fibonachi[0] = 1;

fibonachi[1] = 1;

for (int i = 2; i < n; i++)

{

fibonachi[i] = fibonachi[i - 2] + fibonachi[i - 1];

}

for (int i = 0; i < n; i++)

Console.WriteLine("fibonachi[" + i + "]=" + fibonachi[i]);

Console.ReadKey();

}

} }

##### Генерація простих чисел. Алгоритм Ератосфена

Простим є число, яке ділиться тільки на 1 і на себе. Алгоритм пошуку послідовності простих чисел запропонував древньогрецький математик Ератосфен, і він отримав назву **Решето Ератосфена.**

**Ідея і загальний опис алгоритму.**

Є розташована в ряд за збільшенням послідовність цілих чисел. Спочатку в ній викреслюються усі числа кратні 2, окрім її самої, і так до N. Далі із списку, що вийшов, береться число, що йде за двійкою, - трійка, викреслюються усі кратні 3 числа, окрім її самої. У такому вигляді алгоритм триває для частини послідовності, що залишилася, і у результаті отримаємо усі прості числа у вказаному діапазоні.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 22 | 33 | 44 | 55 | 66 | 77 | 88 | 99 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 220 |
| 221 | 222 | 223 | 224 | 225 | 226 | 227 | 228 | 229 | 330 | 331 | 332 | 333 | 334 | 335 | 336 | 337 | 338 | 339 | 440 |
| 441 | 442 | 443 | 444 | 445 | 446 | 447 | 448 | 449 | 550 | 551 | 552 | 553 | 554 | 555 | 556 | 557 | 558 | 559 | 660 |
| 661 | 662 | 663 | 664 | 665 | 666 | 667 | 668 | 669 | 770 | 771 | 772 | 773 | 774 | 775 | 776 | 777 | 778 | 779 | 880 |
| 881 | 882 | 883 | 884 | 885 | 886 | 887 | 888 | 889 | 990 | 991 | 992 | 993 | 994 | 995 | 996 | 997 | 998 | 999 | 1100 |

У таблиці наведені усі цілі числа від 2 до 100. Червоним помічені ті, які були видалені в процесі виконання алгоритму Решето Ератосфену.

Тепер розглянемо алгоритм детальніше, розбивши його на декілька частин. Отже, для пошуку простих чисел методом Решета Ератосфену треба:

1. Організувати список з чисел від 2 до N, а також логічний масив розмірністю N;
2. У вільну змінну R записати число 2;
3. Виключити усі числа кратні R, починаючи з R\*2;
4. Записати в R наступне за R не закреслене число;

Повторювати дії, описані в двох попередніх кроках.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace SimpleNumber

{

class Program

{

//Нахождение простых чисел. Алгоритм "Решето Эратосфена"

//Запишем подряд все числа от 2 до N.

//Дальше вычеркнем из этого списка все числа кратные 2,

// исключая саму двойку, потом вычеркнем все числа кратные 3,

//исключая само число 3, число 4 уже вычеркнуто, вычеркиваем числа

// кратные 5 и т.д.

//Продолжаем этот процесс, пока квадрат очередного числа не превысит N.

static void Main(string[] args)

{

bool[] table = new bool[100];

int i, j;

// Отмечаем все числа как простые

for (i = 0; i < table.Length; i++)

table[i] = true;

// Вычеркиваем лишнее

for (i = 2; i \* i < table.Length; i++)

if (table[i])

for (j = 2 \* i; j < table.Length; j += i)

table[j] = false;

// Выводим найденное

for (i = 2; i < table.Length; i++)

{

if (table[i])

Console.WriteLine(i);

}

Console.ReadKey();

}

}

}

#### 2. Оператор foreach

Оператор **foreach** використовується для перебору елементів в масивах та інших колекціях і є новим типом циклу, реалізованому в C#. В цьому прикладі створюється *ступінчастий масив*, який заповнюється даними. За допомогою вкладених операторів циклу елементи масиву виводяться на консоль у вигляді матриці.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Console\_Lab4

{

class Program

{

static void Main(string[] args)

{

//оператор foreach і ступінчасті масиви

int[][] а = new int[3][];

а[0] = new int[5] { 24, 50, 18, 3, 16 };

а[1] = new int[3] { 7, 9, -1 };

а[2] = new int[4] { 6, 15, 3, 1 };

Console.WriteLine("Початковий масив:");

foreach (int[] mas1 in а)

{

foreach (int x in mas1)

Console.Write("\t" + x);

Console.WriteLine();

}

Console.ReadKey();

}

}

}

#### 3. Генерація випадкових чисел

Потреба в генерації випадкових чисел часто виникає в програмуванні. В C# є *нестатичний* клас Random, методи якого дозволяють генерувати різні послідовності випадкових чисел. Нижче наведено приклад з лекції 4 в якому в методі ValsGenerator створюється об'єкт **aRand**. Далі в циклі викликається метод **aRand.Next(100),** який при кожному виклику генерує одне випадкове число від 1 до100 і призначає його черговому елементу масиву. Зверніть увагу, що метод **ValsGenerator** є статичним.

В методі **Main** створюється масив **Data,** викликається метод **ValsGenerator** для його заповнення, потім викликається метод Array.Sort(Data) для його сортування. Після чого елементи масиву виводяться в циклі на консоль.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Console\_Lab4

{

class Program

{

// генератор даних

static void ValsGenerator(int[] Vals)

{

// Random - клас для генерації випадкових чисел

Random aRand = new Random();

// заповнення масиву

for (int i = 0; i < Vals.Length; i++)

Vals[i] = aRand.Next(100);

}

static void Main(string[] args)

{

const int N = 10;

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();

}

}

}

Зверніть увагу, що при виводі масиву

Console.WriteLine("Data[" + i + "] = " + Data[i]);

виконується неявне приведення з типу int до типу string.

#### 4. Робота з математичними функціями

Клас Math містить методи для роботи з математичними функціями.

Розглянемо ще один приклад – використання масивів і математичних функцій в задачах обчислювальної математики.

**Постановка задачі**

Знайти дійсні корені рівняння 6x4-3x3+8x2-5=0 за методом бісекції (ділення навпіл) на відрізках [0, 1], [-1, 0].

**Алгоритм методу:**

Нехай [а,b] відрізок, на якому шукають корені. Припустимо, що функція f(x) неперервна на [а,b] і на кінцях приймає значення різних знаків *Алгоритм* методу полягає в побудові послідовності вкладених відрізків, на кінцях яких функція приймає значення різних знаків. Кожний наступний відрізок отримують діленням навпіл попереднього. Опишемо один крок ітераційного методу. Нехай на *к-*ому кроці знайдено відрізок такий, що . Знайдемо середину відрізку . Якщо, то - корінь і задача вирішена. Якщо ні, то з двох половин відрізку вибираємо той, на кінцях якого функція має протилежні знаки:



, , якщо



, , якщо



*Критерій закінчення ітераційного процесу*: якщо довжина відрізку знаходження кореня менше 2, то ітерації припиняють і за значення кореня із заданою точністю приймають середину відрізку.



Код програми може бути таким:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace bicection

{

class Program

{ //Знайти корені нелінійного рівняння

//6x4-3x3+8x2-5=0

//x1=0.74213

//x2=-0.6365

static double f(double x)

{

//рівняння, для якого шукаємо корені

double y=6\*Math.Pow(x,4)-3\*Math.Pow(x, 3)+8\*Math.Pow(x,2)-5;

return y;

}

// --------------------------------------------------------------------

static double bicect(double left, double right)

{

//метод бісекцій

double eps = 0.00001;

double center = 0;

while (right - left > eps \* 2)

{

center = (right - left) / 2 + left;

if (f(center) \* f(left) > 0)

left = center;

else

right = center;

}

return center;

}

// ----------------------------------------------------

static void Main(string[] args)

{

//метод бісекцій

double x1 = bicect(0, 1);

double x2 = bicect(-1, 0);

Console.WriteLine("Метод бісекцій");

Console.WriteLine("x1={0},x2= {1}", x1, x2);

Console.ReadKey();

}

}

}

В цій програмі є два методи: static double f(double x), в якому визначається рівняння, і метод static double bicect(double left, double right). Для обчислення ступеня в методі **f** використовується метод Math.Pow( ).

Другий метод static double bicect(double left, double right) реалізує ітераційний процес. В методі Main два рази викликається метод бісекцій для знаходження коренів на вказаних відрізках. Результати виводляться на консоль.

#### 5. Лінійний пошук в масиві

**Лінійний, послідовний пошук** — алгоритм знаходження заданого значення довільної функції на деякому відрізку.

**Формальний запис алгоритму:**

1. Визначити елемент, який шукаємо ***key;***

2. Встановити границі області пошуку L, R для масиву A [a];

3. Якщо аi = key – елемент знайдено;

4. Інакше – переходимо до наступного елементу.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Search

{

class Program

{

static void Main(string[] args)

{

int i, n, key, nom;

bool x = false;

Console.WriteLine("Розмір масиву");

n = int.Parse(Console.ReadLine());

int[] mas = new int[n]; //визначення масиву

Console.WriteLine("Шуканий елемент");

key = int.Parse(Console.ReadLine());

// Random - клас для генерації випадкових чисел

Random aRand = new Random();

for (i = 0; i < n; i++)

{

//формування масиву, заповнення його випадковими числами

mas[i] = aRand.Next(n);

Console.Write(mas[i] + " ");

}

for (i = 0; i < n; i++)

{ //якщо цей елемент дорівнює шуканому

if (mas[i] == key) //то x призначаємо true

{ x = true; nom = i; break; } //і виходимо з циклу

}

if (x == true)

Console.WriteLine("Елемент знайдено");

else Console.WriteLine("\nЕлемент не знайдено");

Console.ReadKey();

}

}

}

#### 6. Двійковий (бінарний) пошук

**Двійковий (бінарний) пошук** (також відомий як метод ділення навпіл і дихотомія) — класичний алгоритм пошуку елемента у відсортованому масиві.

**Формальний запис алгоритму:**

1. Масив ділитися на дві рівні частини, шляхом визначення першого (a), останнього (b) і середнього (c) елементів;
2. Середній елемент порівнюється з шуканим (s):

якщо **s<c.** Останньому елементу призначається значення середнього, тим самим ділянка пошуку зменшується удвічі: b=c;

якщо **s>c**. Першому елементу призначається значення середнього, і ділянка пошуку зменшується удвічі: a=c;

якщо **s=c**. Елемент знайдений, і робота алгоритму завершується**.**

1. Якщо для перевірки не залишився жодного елементу, то алгоритм завершується, інакше виконується перехід до пункту 2.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace BinarySearch

{

class Program

{

static void Main(string[] args)

{

int i, n, key, begin, end, c;

bool x = false;

Console.WriteLine("Розмір масиву");

n = int.Parse(Console.ReadLine());

int[] mas = new int[n]; //визначення масиву

Console.WriteLine("Шуканий елемент");

key = int.Parse(Console.ReadLine());

for (i = 0; i < n; i++)

{

//формування масиву, заповнення його числами кратними n (10,20,....)

mas[i] = n \* i;

Console.Write(mas[i] + " ");

}

begin = 0; end = n; //ліва і права границі масиву

while (begin < end)

{

// власне реалізація алгоритму

c = begin + (end - begin) / 2;

if (key < mas[c]) end = c;

else if (key > mas[c]) begin = c + 1;

else { x = true; break; }

}

if (x == true)

Console.WriteLine("Елемент знайдено");

else Console.WriteLine("\nЕлемент не знайдено");

Console.ReadKey();

}

}

}

### Варіанти завдань для лабораторної роботи № 2

Завдання передбачає реалізацію 9 функцій. Функції 1 – 5 передбачають роботу з одновимірними масивами, функції 6 – 7 передбачають опрацювання матриць, функція 8 – розв’язання нелінійного рівняння, функція 9 – робота з рядком символів типу string.

|  |  |
| --- | --- |
| **№ варі-анту** | **Зміст завдання** |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість з консолі. Відсортувати згенерований масив за спаданням значень його елементів алгоритмом швидкого сортування. Вивести на консоль масив до та після сортування. 2. У згенерованому масиві визначити прості числа із заданого з консолі діапазону, використавши алгоритм Ератосфена, та вивести їх на консоль. 3. Визначити кількість повторень кожного елементу масиву, застосувавши метод лінійного пошуку. 4. Вивести на консоль значення мінімального та максимального елементів масиву та їх індекси, застосувавши метод лінійного пошуку. 5. Вивести на консоль значення елемента, заданого з консолі, застосувавши метод бінарного пошуку. У разі його відсутності в масиві ви вести відповідне повідомлення. Модифікувати функцію двійкового пошуку у масиві, використавши метод BinarySearch класу Array. 6. Згенерувати елементи матриці, задавши її вимірність з консолі. Вивести матрицю на консоль у вигляді таблиці. Визначити суму елементів заданого з консолі номеру рядка матриці та суму елементів заданого з консолі номеру стовпчика матриці. Вивести на консоль номери рядка та стовпчика матриці та обчислені суми. 7. Вивести на консоль значення мінімального та максимального елементів матриці та їх індекси, застосувавши методи класу Math. 8. Знайти корені нелінійного рівняння 6x4−3x3+8x2−25=0, застосувавши метод половинного ділення (метод бісекції). Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести з консолі рядок символів (тип string). Виконати операції редагування рядка, здійснивши пошук та заміну підрядка на заданий з консолі підрядок, вставку підрядка, видалення заданого підрядка. Вивести на консоль рядок після редагування. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом бульбашкового сортування. Вивести масив до та після сортування на консоль. 2. Побудувати новий одновимірний масив простих чисел, використавши алгоритм Ератосфена, та вивести їх на консоль. 3. Об’єднати згенерований раніше масив з масивом простих чисел. Обчислити суму елементів масиву, середнє арифметичне елементів, min і max елементів масиву, використавши методі класу Math. Результати вивести на консоль. 4. Вивести на консоль індекси усіх повторень шуканого елемента масиву, застосувавши метод лінійного пошуку. Значення шуканого елемента ввести з консолі 5. Вивести на консоль індекс елемента, значення якого задане з консолі, застосувавши метод бінарного пошуку. У разі його відсутності в масиві ви вести відповідне повідомлення. Модифікувати функцію двійкового пошуку у масиві, використавши метод BinarySearch класу Array 6. Згенерувати матрицю, i–й рядок якої визначає номер співробітника, j-й стовпчик якої визначає номер місяця року. Кількість співробітників задати з консолі. Значення на перетині i-го рядка та j-го стовпця визначає зарплату i-го співробітника у j-му місяці. Вивести матрицю на консоль у вигляді таблиці. Визначити загальний бюджет зарплати за рік, загальну і середню зарплату за місяць, номер якого ввести з консолі. 7. У згенерованій матриці поміняти місцями рядки, що містять максимальний та мінімальний елементи. Якщо шукані елементи знаходяться в одному рядку, то поміняти місцями стовпчики. Вивести на консоль матрицю до та після переміни місцями рядків чи стовпчиків. 8. Знайти корені нелінійного рівняння (x^2−6\*x)^2−2(x−3)^2=81, застосувавши метод Ньютона. Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести з консолі рядок символів (тип string). Виконати операції редагування рядка, здійснивши заміну усіх алфавітних символів на їх ASCII коди та видаливши усі цифрові символи. Вивести рядок після заміни на консоль. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом сортування вибором. Вивести на консоль масив до та після сортування. 2. Вивести значення індексів елементів масиву, що не змінилися після його сортування, та їх кількість. 3. Створити новий масив натуральних чисел, значеннями якого є прості числа в діапазоні від 1 до N. Значення N задати з консолі. Використати алгоритм Ератосфена для визначення простих чисел. Вивести прості числа на консоль. Якщо в згенерованому раніше масиві відсутні знайдені за алгоритмом Ератосфена прості числа, то вставити їх в згенерований масив, не змінивши його упорядкованість за зростанням. Вивести отриманий масив на консоль. 4. У сформованому масиві визначити кількість парних елементів і елементів з парними індексами, застосувавши алгоритм лінійного пошуку. 5. Визначити кількість і значення елементів масиву, що кратні заданому з консолі числу, застосувавши метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Модифікувати функцію двійкового пошуку у масиві, використавши метод BinarySearch класу Array 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень з консолі. На перетині i-го рядка та j-го стовчика матриці записаний прибуток за j-й місяць від i-го проекту, який виконується в компанії. Вивести матрицю на консоль. Визначити загальний прибуток від кожного проекту, загальний дохід компанії від усіх проектів за усі місяці, індекс проекту з найбільшим прибутком. Вивести результати на консоль. 7. У створеній матриці визначити індекси та значення її сідлових точок. Елемент матриці називається сідловою точкою, якщо він є найменшим у своєму рядку та найбільшим у своєму стовпчику. Вивести на консоль значення та індекси її сідлових точок або повідомлення про відсутність таких елементів. 8. Знайти корені нелінійного рівняння (x^2+x+1)(x^2+x+1)−12=0, застосувавши метод половинного ділення (метод бісекції). Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести з консолі рядок символів (тип string). Порахувати кількість слів, що починаються з великої літери. Вивести текст, складений з останніх букв усіх слів. Результати вивести на консоль. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math**   1. Згенерувати додатні та від’ємні значення елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом сортування включенням. Вивести на консоль масив до та після сортування. 2. Визначити прості числа серед додатних елементів масиву, використавши алгоритм Ератосфена. Вивести прості числа на консоль. Якщо в згенерованому масиві відсутні прості числа, то вивести на консоль відповідне повідомлення. 3. Переставити елементи масиву так, щоб спочатку були розташовані всі від’ємні елементи, потім усі додатні елементи, потім усі нульові. Порядок серед додатних і від’ємних елементів має зберегтися. Надрукувати масив після переставлення елементів. 4. Знайти найбільший серед від’ємних та найменший серед додатних елементів масиву, застосувавши алгоритм лінійного пошуку. Вивести значення знайдених елементів та їх індекси. 5. Визначити кількість і значення елементів масиву, що кратні заданому з консолі числу, застосувавши метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch класу Array 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень з консолі. На перетині i-го рядка та j-го стовчика матриці записаний прибуток за j-й місяць від продажу i-го товару в магазині. Вивести матрицю на консоль. Визначити загальний прибуток від кожного товару, загальний прибуток магазину від продажу усіх товарів за усі місяці, індекс товару, який приносить прибуток. Вивести результати на консоль. 7. Вивести на консоль значення мінімального та максимального елементів матриці та їх індекси, застосувавши методи класу Math. 8. Знайти корені нелінійного рівняння (x^2−5\*x+7)^2−(x−2)(x−3)=0, застосувавши метод половинного ділення (метод бісекції). Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести з консолі рядок символів (тип string), що містить круглі, квадратні та фігурні дужки. Визначити чи є послідовність дужок правильною, тобто кількість дужок, що відкривається, дорівнює кількості дужок, що закриваються. Результати вивести на консоль. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math**   1. Згенерувати цілочислові додатні та від’ємні елементи одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом сортування Шелла. Вивести на консоль масив до та після сортування. 2. У згенерованому масиві визначити прості числа із заданого з консолі діапазону додатних цілих значень, використавши алгоритм Ератосфена, та вивести їх на консоль. Якщо в згенерованому масиві відсутні прості числа, то вивести на консоль відповідне повідомлення 3. Переставити елементи масиву так, щоб їх значення чергувалися в порядку: від’ємний, додатний, нульовий, від’ємний, додатний, нульовий і т.д, Надрукувати масив після переставлення елементів. 4. Знайти найбільший серед від’ємних та найменший серед додатних елементів масиву, застосувавши алгоритм лінійного пошуку. Вивести значення знайдених елементів та їх індекси. 5. Визначити кількість і значення елементів масиву, що належать заданому з консолі діапазону, застосувавши метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch класу Array 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень з консолі. На перетині i-го рядка та j-го стовпчика матриці записаний прибуток за j-й місяць від продажу i-го товару в магазині. Вивести матрицю на консоль. Визначити загальний прибуток від кожного товару, загальний прибуток магазину від продажу усіх товарів за усі місяці, індекс товару, який приносить найбільший прибуток. Вивести результати на консоль. 7. У створеній матриці визначити та вивести на консоль значення та індекси мінімального елемента. Видалити рядок та стовпчик з мінімальним елементом матриці. Вивести на консоль отриману після перетворення матрицю. 8. Знайти корені нелінійного рівняння (x^2−5\*x+7)^2−(x−2)(x−3)=0, застосувавши метод половинного ділення (метод бісекції). Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести з консолі рядок символів (тип string), що містить круглі, квадратні та фігурні дужки. Визначити, чи є послідовність дужок правильною, тобто кількість дужок, що відкривається, дорівнює кількості дужок, що закриваються. Результати вивести на консоль. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати методи класу **Math**   1. Згенерувати натуральні значення елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом бульбашкового сортування. Вивести на консоль масив до та після сортування. 2. Перевірити, чи елементи згенерованого масиву, які мають непарні індекси, є простими числами. Для визначення простих чисел використати алгоритм Ератосфена, Якщо такі числа знайдені, вивести їх на консоль, інакше вивести відповідне повідомлення. 3. Переставити елементи масиву так, щоб парні і непарні елементи чергувалися. Перед кожним простим числом вставити нульове значення. Вивести на консоль масив після переставлення елементів. 4. Знайти у відсортованому масиві елементи, значення яких є трикутними числами, що визначаються за формулою , де n – порядковий номер числа в послідовності трикутних чисел (1, 3, 6, 10,…). Застосувати алгоритм лінійного пошуку. Результати вивести на консоль. 5. Визначити кількість і значення елементів масиву, що належать заданому з консолі діапазону, застосувавши метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch класу Array 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі, включаючи нульове значення. На перетині i-го рядка та j-го стовпчика матриці записана кількість вакцинованих людей проти Covid-19 в *i*-й країні в *j*-му місяці. Якщо значення дорівнює нулю, це означає, що в *i*-й країні в *j*-му місяці людей не вакцинували. Вивести матрицю на консоль. Визначити загальну кількість людей, що вакциновані в усіх країнах за весь період, країну, в якій найменша кількість вакцинованих за весь період, місяць з найбільшою кількістю вакцинованих. Результати вивести на консоль. 7. Із зенерованої матриці видалити країні, які за весь період не вакцинували жодної людини, переставити стовпці матриці відповідно до зростання сум їх елементів. Перетворену матрицю вивести на консоль. 8. Знайти корені нелінійного рівняння 2x^4−x^3−9x^2+13x−5=0, застосувавши метод бісекції. Результати вивести на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння 9. Увести рядок алфавітних та цифрових символів. Видалити з рядка символи, що стоять на парних позиціях. Подвоїти усі алфавітні символи, що введені у верхньому регістрі. Цифрові символи замінити на номер студентської групи. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати методи класу **Math**   1. Згенерувати натуральні значення елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом сортування вставками. Вивести на консоль масив до та після сортування. 2. Перевірити, чи елементи згенерованого масиву з парними індексами є простими числами. Для визначення простих чисел використати алгоритм Ератосфена, Якщо такі числа знайдені, вивести їх на консоль, інакше вивести відповідне повідомлення. 3. Переставити елементи масиву так, щоб спочатку розташовувалися прості числа, потім складені, кратні 3, потім складені кратні 2 і 3, потім усі інші. Вивести на консоль масив після переставлення елементів. 4. Знайти елементи масиву, що є квадратами цілих чисел, застосувавши алгоритм лінійного пошук. Вивести знайдені значення та їх індекси на консоль. 5. Визначити кількість і значення елементів масиву, що належать заданому з консолі діапазону, застосувавши метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch класу Array 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. На перетині i-го рядка та j-го стовпчика матриці записана кількість студентів, що проходять практику в j-му місяці в i-й компанії. Вивести матрицю на консоль. Визначити загальну кількість студентів, що проходили практику у всіх компаніях в заданому з консолі місяці, кількість компаній, в яких проходили практику задана з консолі кількість студентів, компанію, в якій практикувалась найбільша кількість студентів. Результати вивести на консоль. 7. Відсортувати кожний рядок згенерованої матриці за зростанням значень її елементів, визначити суму елементів рядків та переставити їх відповідно до спадання сум елементів по рядках. Вивести перетворену матрицю на консоль. 8. Знайти корені нелінійного рівняння (x^2−5\*x+7)^2−(x−2)(x−3)=0, застосувавши метод половинного ділення (метод бісекції). Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести з клавіатури рядок алфавітних, цифрових і символів знаків арифметичних дій. З рядка видалити всі алфавітні символи. Перевести цифрові символи у числові типи та вивести числа на екран. Над отриманими числами виконати арифметичні дії відповідно до заданих в рядку операцій. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати методи класу **Math**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом сортування вибором. Вивести на консоль масив до та після сортування. 2. Перевірити, чи непарні елементи згенерованого масиву є простими числами. Для визначення простих чисел використати алгоритм Ератосфена, Якщо такі числа знайдені, вивести їх на консоль, інакше вивести відповідне повідомлення. 3. Переставити елементи масиву так, щоб спочатку розташовувалися прості числа, потім складені, які діляться на 2, але не діляться на 3, потім числа, що діляться на 2 та на 3, потім числа, що діляться тільки на 3, потім усі інші. Вивести на консоль масив після перестановки елементів. 4. Знайти в масиві та вивести на консоль елементи та їх індекси, які є п’ятикутними числами, застосувавши алгоритм лінійного пошуку. Формула для обчислення *n*-го п’ятикутного числа  (1, 5, 12, 22,…). 5. Визначити усі входження в масив заданого з консолі елемента, застосувавши метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Вивести на консоль індекси шуканого елемента. Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch класу Array 6. Згенерувати елементи матриці, задавши її вимірність з консолі. Матриця означує дисципліни та компетентності, які вони формують, і містить значення 0 або 1. На перетині i‑го рядка та j-го стовпчика матриці записаний 0, якщо дисципліна не забезпечує формування компетентності студента, 1 – якщо між дисципліною і компетентністю є зв’язок. Визначити, кількість дисциплін, які не забезпечують формування жодної компетентності студентів, дисципліни, які формують найбільшу кількість компетентностей, компетентності, які не забезпечені дисциплінами. Результати вивести на консоль. 7. В згенерованій матриці визначити суми елементів в стовпчиках, записати їх в одновимірний масив, відсортувати його, переставити стовчики матриці відповідно до зростання їх сум. 8. Знайти корені нелінійного рівняння x^3+6x+4x^2+3=0, застосувавши метод Ньютона. Результати вивести на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння 9. Увести з клавіатури рядок символів, включаючи розділові символи (пробіли, коми та крапки) та круглі дужки. Знайти слова, які мають парну довжину, та підрахувати їх кількість. Видалити символи, що розміщені в круглих дужках. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати методи класу **Math**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом сортування Шелла. Вивести на консоль масив до та після сортування. 2. Знайти в згенерованому масиві прості числа, застосувавши рещето Сундарама. Якщо такі числа знайдені, вивести їх на консоль, інакше вивести відповідне повідомлення. Посилання на алгоритм решета Сундарама: <https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%88%D0%B5%D1%82%D0%BE_%D0%A1%D1%83%D0%BD%D0%B4%D0%B0%D1%80%D0%B0%D0%BC%D0%B0> 3. Переставити елементи масиву так, щоб на парних позиціях розмістилися елементи з лівої частини відсортованого масиву з індексами від 0 до серединного елемента-1, на непарних позиціях розмістилися елементи з правої частини відсортованого масиву з індексами від кінцевого елемента до серединного. Вивести на консоль масив після перестановки елементів. 4. Знайти в масиві та вивести на консоль елементи та їх індекси, які є пірамідальними числами, застосувавши алгоритм лінійного пошуку. Формула для обчислення n-го пірамідального числа (1, 4, 10, 35,…): . 5. Визначити усі входження в масив заданого з консолі елемента, застосувавши метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Вивести на консоль індекси шуканого елемента. Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch класу Array 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. На перетині i-го рядка та j-го стовпчика матриці записана кількість кредитів, що має i-та дисципліна в j-му навчальному році. Якщо значення елемента матриці дорівнює нулю, це означає, що i-та дисципліна не вивчається в j-му навчальному році Вивести матрицю на консоль. Визначити загальну кількість кредитів по всіх дисциплінах за усі навчальні роки, дисципліну та навчальний рік, в якому є дисципліна з найбільшою кількістю кредитів, середню кількість кредитів в кожному навчальному році. Результати вивести на консоль. 7. В згенерованій матриці знайти рядок або стовпчик з найбільшою кількістю нульових елементів, видалити відповідний рядок або стовпчик. Вивести на консоль перетворену матрицю. 8. Знайти корені нелінійного рівняння (x−1)^3+(2x+3)^3=27x^3+8, застосувавши метод половинного ділення (метод бісекції). Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести з консолі рядок символів, включаючи розділові (пробіли, коми та крапки). Визначити в рядку та вивести на консоль слова і підрахувати їх кількість. Слова – це лексеми, як не містять всередині розділових символів. Кожне друге слово розбити на склади. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом швидкого сортування. Вивести на консоль масив до та після сортування. 2. Згенерувати другий масив простих чисел від 1 до n, задавши значення n з консолі. Для отримання простих чисел використати решето Сундарама. Об’єднати масив простих чисел із згенерованим раніше масивом, не порушуючи його впорядкованість за зростанням. Надрукувати прості числа та об’єднаний масив. Посилання на алгоритм (<https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%88%D0%B5%D1%82%D0%BE_%D0%A1%D1%83%D0%BD%D0%B4%D0%B0%D1%80%D0%B0%D0%BC%D0%B0>). 3. Переставити елементи відсортованого масиву так, щоб елементи з парними та непарними індексами помінялися місцями. Вивести на консоль масив після перестановки елементів. 4. Знайти в масиві та вивести на консоль елементи та їх індекси, які є шестикутними числами, застосувавши алгоритм лінійного пошуку. Шестикутні числа складають послідовність (1, 6, 15, 28,…). Формула для обчислення *n*-го шестикутного числа: 5. Знайти в масиві та вивести на консоль елементи та їх індекси, які є шестикутними числами, застосувавши алгоритм бінарного пошуку. Шестикутні числа складають послідовність (1, 6, 15, 28,…). Формула для обчислення *n*-го шестикутного числа: . Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch() класу Array 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. На перетині *i*-го рядка та *j*-го стовпчика матриці записана кількість студентів, які працюють в *i*-ій компанії в *j*-му навчальному році. Якщо значення елемента матриці нулю, це означає, що в *i-*ій компанії не працює жодного студента в *j*-му навчальному році Вивести матрицю на консоль. Визначити компанію і рік, коли працюючих студентів найбільше, середню кількість працюючих студентів за усі роки в кожній компанії та компанії, в яких не працює жодного студента. 7. Із згенерованої матриці видалити компанії, в яких не працює жодного студента, відсортувати рядки матриці за зростанням, здійснити пошук в матриці введеного з консолі значення. Результати вивести на консоль. 8. Знайти корені нелінійного рівняння x4−13x2+36=0, застосувавши метод половинного ділення (метод бісекції). Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести рядок символів, включаючи розділові (пробіли, коми, крапки). Слова є послідовністю символів без пробілів. Розбити рядок на слова, визначити їх кількість, знайти слово, що має найбільшу кількість голосних. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом бульбашкового сортування. Вивести на консоль масив до та після сортування. 2. Знайти в згенерованому масиві прості числа, застосувавши рещето Ератосфена. Якщо такі числа знайдені, вивести їх на консоль, інакше вивести відповідне повідомлення. 3. В масиві знайти мінімальний та максимальні елементи. Поміняти їх місцями. Вивести на консоль масив після перестановки. 4. Знайти в масиві та вивести на консоль елементи та їх індекси, які є семикутними числами, застосувавши алгоритм лінійного пошуку. Семикутні числа складають послідовність (1, 7, 18, 34,…). Формула для обчислення *n*-го семикутного числа: 5. Визначити усі входження в масив заданого з консолі елемента, застосувавши метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Вивести на консоль індекси шуканого елемента. Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch класу Array 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. На перетині *i*-го рядка та *j*-го стовпчика матриці записана кількість студентів, які вибрані до складу студради *i*-го факультету в *j*-му навчальному році. Вивести матрицю на консоль. Визначити факультет і навчальний рік, в якому кількість студентів в студраді є найбільшою, сумарну кількість студентів в студраді за усі роки навчання на заданому з консолі факультеті, навчальний рік, в якому сумарна кількість студентів в студрадах усіх факультетів найменша. Результати вивести на консоль. 7. Із згенерованої матриці видалити факультети, на яких в заданому з консолі навчальному році студрада мала найменшу кількість, , відсортувати рядки матриці за зростанням, здійснити пошук в матриці введеного з консолі значення. Результати вивести на консоль. 8. Знайти корені нелінійного рівняння , застосувавши метод половинного ділення (метод бісекції). Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести з клавіатури рядок символів, який містить слова та розділові символи. Видалити з рядка однолітерні слова та зайві пропуски, коми, крапки. Вивести перетворений рядок на консоль. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом сортування включенням. Вивести на консоль масив до та після сортування. 2. Знайти в згенерованому масиві прості числа, застосувавши рещето Сундарама. Якщо такі числа знайдені, вивести їх на консоль, інакше вивести відповідне повідомлення. Посилання на алгоритм решета Сундарама: <https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%88%D0%B5%D1%82%D0%BE_%D0%A1%D1%83%D0%BD%D0%B4%D0%B0%D1%80%D0%B0%D0%BC%D0%B0> 3. В масиві знайти мінімальний та максимальні елементи. Елементи, що розташовані між мінімальним та максимальним записати в інвертованому порядку. Вивести результати на консоль. 4. Знайти в масиві та вивести на консоль елементи та їх індекси, які є семикутними числами, застосувавши алгоритм лінійного пошуку. восьмикутні числа складають послідовність (1, 8, 21, 40,…). Формула для обчислення *n*-го восьмикутного числа: 5. Визначити усі входження в масив восьмикутних чисел, застосувавши метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Вивести на консоль індекси шуканого елемента. Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch класу Array 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. На перетині *i*-го рядка та *j*-го стовпчика матриці записана кількість заяв абітурієнтів, які подали заяви до вступу на *i*-й факультет та *j*-ту спеціальність. Вивести матрицю на консоль. Визначити факультет і спеціальність, на яку подана найбільші кількість заяв абітурієнтів, сумарну кількість заяв на усі спеціальності заданого з консолі факультету, сумарну кількість заяв на задану з консолі спеціальність на всіх факультетах. Результати вивести на консоль. 7. Із згенерованої матриці видалити факультети, на яких сумарна кількість заяв на всі спеціальності найменша. Переставити стовпці матриці в порядку спадання сумарної кількості заяв на спеціальність. Здійснити пошук в матриці введеного з консолі значення. Результати вивести на консоль. 8. Знайти корені нелінійного рівняння , застосувавши метод половинного ділення (метод бісекції). Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести рядок символів, який складається із круглих дужок і знаків запитання. Передбачити вкладеність дужок, наприклад, "(???(???(?)??)". замінити знаки запитання круглими дужками так, щоб отримати правильний дужковий вираз, в якому кількість правих дужок дорівнює кількості лівих дужок, та порахувати кількість способів таких замін. Вивести перетворений рядок на консоль. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом сортування вибором. Вивести на консоль масив до та після сортування. 2. Знайти прості числа в заданому з консолі діапазоні, використавши решето Ератосфена. Записати прості числа в одновимірний масив. Об’єднати масив простих чисел із згенерованим раніше, виключивши повторення елементів і зберігаючи їх упорядкованість. Вивести масив на консоль. 3. У створеному раніше масиві поміняти місцями ліву та праву частини. В якості розділового елемента вибрати серединний елемент. Вивести результати на консоль 4. Знайти в масиві та вивести на консоль елементи та їх індекси, які є семикутними числами, застосувавши алгоритм лінійного пошуку. десятикутні числа складають послідовність (1, 10, 27, 52,…). Формула для обчислення *n*-го семикутного числа: 5. Визначити усі входження в масив семикутних чисел, застосувавши метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Вивести на консоль індекси шуканого елемента. Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch класу Array 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. На перетині *i*-го рядка та *j*-го стовпчика матриці записана кількість опадів, що випали в *i*-му місяці *j*-то року. Вивести матрицю на консоль. Визначити рік та місяць, в якому випала найбільша кількість опадів, сумарну кількість опадів, що випали у зимовий період за усі роки, кількість опадів, що випала в заданий з консолі рік. Результати вивести на консоль. 7. Із згенерованої матриці видалити роки із найменшою кількістю опадів за усі місяці. Переставити рядки матриці в порядку спадання сумарної кількості опадів за усі роки. Здійснити пошук в матриці введеного з консолі значення. Результати вивести на консоль. 8. Знайти корені нелінійного рівняння , застосувавши метод Ньютона Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести з консолі рядок символів, який складається зі слів та пропусків між ними. Визначити кількість повторів кожного слова та саме слово, що зустрічається найчастіше в тексті. Результати вивести на консоль. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом сортування Шелла. Вивести на консоль масив до та після сортування. 2. Знайти прості числа, задавши верхню границю чисел і використавши решета Сундарама. Посилання на опис алгоритму <https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%88%D0%B5%D1%82%D0%BE_%D0%A1%D1%83%D0%BD%D0%B4%D0%B0%D1%80%D0%B0%D0%BC%D0%B0>  Записати знайдені прості числа у одновимірний масив та вивести його на консоль. 3. У створеному раніше масиві перший і останній, максимальний і мінімальний елементи місцями. Результати вивести на консоль. 4. Знайти в масиві та вивести на консоль елементи та їх індекси, які є центрованими k‑кутними числами, застосувавши алгоритм лінійного пошуку. Формула для обчислення *n*-го центрованого k‑кутного числа: . Значення k увести з консолі. 5. Визначити усі входження в масив центрованих k‑кутних чисел, застосувавши метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Вивести на консоль індекси шуканого елемента. Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch класу Array. 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. На перетині *i*-го рядка та *j*-го стовпчика матриці записана оцінка *i*-го студента по *j*-й дисципліні. Вивести матрицю на консоль. Визначити студентів, які за підсумками успішності за всіма дисциплінами підпадають під відрахування, рейтинг кожного студента як середнє арифметичне по всіх дисциплінах, дисципліну, середній бал якої найбільший. Результати вивести на консоль. 7. Із згенерованої матриці видалити дисципліну, середній бал якої найменший по всіх студентах. Переставити стовпці матриці в порядку спадання середнього балу за дисциплінами. Здійснити пошук в матриці введеного з консолі значення. Результати вивести на консоль. 8. Знайти корені нелінійного рівняння x^3−4x+6=0, застосувавши метод половинного ділення (метод бісекції). Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести з консолі рядок символів, який складається зі слів та пропусків між ними. Створити новий рядок, в якому слова відсортовані за спаданням довжини слів. Відсортувати рядок за алфавітом першої літери слів. Результати вивести на консоль. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом швидкого сортування. Вивести на консоль масив до та після сортування. 2. У згенерованому масиві визначити прості числа, застосувавши решето Ератосфена, і надрукувати їх. Якщо прості числа не знайдені, вивести відповідне повідомлення. 3. У створеному масиві переставити елементи так, щоб в середині масиву знаходився мінімальний елемент, ліва частина, рахуючи від середини масиву, упорядкована за спаданням, права частина упорядкована за зростанням. Результати перетворення масиву вивести на консоль. 4. Знайти в масиві та вивести на консоль елементи та їх індекси, які є центрованими квадратними числами, застосувавши алгоритм лінійного пошуку. Формула для обчислення *n*-го центрованого квадратного числа: . 5. Визначити усі входження в масив центрованих квадратних чисел, застосувавши метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Вивести на консоль індекси шуканого елемента. Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch класу Array. 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. На перетині *i*-го рядка та *j*-го стовпчика матриці записана кількість підручників, які *i*-й студент прочитав по *j*-й дисципліні. Вивести матрицю на консоль. Визначити студентів, які прочитали найбільшу кількість підручників за всіма дисциплінами, дисципліну, по якій рекомендована найменша кількість підручників, сумарну кількість підручників, які прочитали усі студенти по всіх дисциплінах. Результати вивести на консоль. 7. Із згенерованої матриці видалити студента, який прочитав найменшу кількість підручників за всіма дисциплінами. Переставити стовпці матриці в порядку спадання сумарної кількості підручників, які прочитали всі по кожній дисципліні. Результати вивести на консоль. 8. Знайти корені нелінійного рівняння , застосувавши метод половинного ділення (метод бісекції). Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести з консолі рядок символів. Підрахувати кількість повторів кожного символу, видалити символи, що повторюються. Результати вивести на консоль. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом бульбашкового сортування. Вивести на консоль масив до та після сортування. 2. У згенерованому масиві визначити прості числа, застосувавши решето Сундарама. Посилання на опис алгоритму <https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%88%D0%B5%D1%82%D0%BE_%D0%A1%D1%83%D0%BD%D0%B4%D0%B0%D1%80%D0%B0%D0%BC%D0%B0>.  Вивести прості числа на консоль. Якщо прості числа не знайдені, вивести відповідне повідомлення. 3. У створеному масиві визначити максимальний елемент серед елементів з парними індексами, і мінімальний елемент серед елементів з непарними індексами. Вивести знайдені елементи та їх індекси на консоль. Поміняти місцями знайдені елементи. Використати методи класу Math. Результати перетворення масиву вивести на консоль. 4. Знайти у створеному масиві та вивести на консоль елементи та їх індекси, які є числами Фібоначчі, що визначаються за формулою F1 = 1; F2 = 1; Fk = Fk-1 + Fk-2 для k>2. 5. Визначити усі входження у створений масив центрованих трикутних чисел, застосувавши метод бінарного пошуку. Формула для обчислення n-го центрованого трикутного числа: . У разі їх відсутності вивести відповідне повідомлення. Вивести на консоль індекси шуканих елементів. Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch класу Array. 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів (0…15) з консолі. На перетині *i*-го рядка та *j*-го стовпчика матриці записана кількість лабораторних робіт, виконаних *i*-м студент з *j*-ї дисципліни. Вивести матрицю на консоль. Визначити студентів, які виконали найбільшу кількість лабораторних робіт за всіма дисциплінами, дисципліну, по якій усі студенти виконали найменшу кількість робіт, студентів, які хоча б з одної дисциплінами не виконали жодної (в матриці записаний 0) лабораторної роботи. Результати вивести на консоль. 7. Із згенерованої матриці видалити студентів (рядки), які хоча б з одної дисциплінами не виконали жодної (в матриці записаний 0) лабораторної роботи. Вивести перетворену матрицю на консоль. Якщо таких студентів немає, вивести відповідне повідомлення. 8. Знайти корені нелінійного рівняння , застосувавши метод Ньютона. Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести з консолі рядок символів. Видалити слова, які складаються з одної літери. Підрахувати кількість слів, перша і остання літера яких однакові. Якщо таких слів нема, вивести відповідне повідомлення. Результати вивести на консоль. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом швидкого сортування. Вивести на консоль масив до та після сортування. 2. В згенерованому масиві визначити прості числа, застосувавши решето Ератосфена, і надрукувати їх. Якщо прості числа не знайдені, вивести відповідне повідомлення. 3. У створеному масиві знайти досконалі числа, вивести їх значення та індекси на консоль. Досконале число – це число, яке дорівнює сумі всіх своїх дільників, у тому числі одиниця, але виключаючи саме себе, наприклад число 6 – досконале, оскільки його дільники 1,2,3 в сумі складають 6. Застосувати метод лінійного пошуку. 4. Знайти в масиві та вивести на консоль елементи та їх індекси, які є числами Мерсенна, застосувавши алгоритм лінійного пошуку. Формула для обчислення *n*-го числа Мерсенна: . 5. Визначити усі входження в масив заданого з консолі елемента, застосувавши метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Вивести на консоль індекси шуканого елемента. Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch класу Array 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. На перетині *i*-го рядка та *j*-го стовпчика матриці записана кількість пропущених занять *i*-м студент з *j*-ї дисципліни. Вивести матрицю на консоль. Визначити студентів, які пропустили найбільшу кількість занять за всіма дисциплінами, дисципліну, по якій пропущена всіма студентами найменша кількість занять, середнє значення пропусків всіма студентами занять з усіх дисциплін. Результати вивести на консоль. 7. Із згенерованої раніше матриці видалити студентів (рядки), які пропустили найбільшу кількість занять. Вивести перетворену матрицю на консоль. 8. Знайти корені нелінійного рівняння, застосувавши метод половинного ділення (метод бісекції). Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести з консолі рядок символів, який містить алфавітні та розділові символи (пробіли, точки, коми). Визначити кількість слів в рядку. Знайти найдовше слово. Першу літеру слова, що записано після точки, перевести у верхній регістр. Результати вивести на консоль. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом сортування вибором. Вивести на консоль масив до та після сортування. 2. Визначити прості числа, задавши найбільшу межу чисел і застосувавши решето Ератосфена. Записати прості числа у масив і вивести його на консоль. Дописати в кінець відсортованого раніше масиву масив простих чисел. 3. Знайти в масиві та вивести на консоль елементи та їх індекси, які є числами Фібоначчі, Мерсенна () або Ферма (), застосувавши алгоритм лінійного пошуку. Якщо вказаних чисел в масиві не знайдено, вивести відповідне повідомлення. 4. У створеному масиві переставити елементи так, щоб спочатку йшли числа Фібоначчі, потім числа Мерсенна, потім числа Ферма, потім решта. Результати перетворення масиву вивести на консоль. Застосувати метод лінійного пошуку. Якщо вказаних чисел в масиві не знайдено, визначити середнє арифметичне та середнє геометричне елементів масиву. 5. Визначити усі входження в масив заданого з консолі елемента, застосувавши метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Вивести на консоль індекси шуканого елемента. Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch класу Array 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі, включаючи нульове значення. На перетині *i*-го рядка та *j*-го стовпчика матриці записана кількість резюме, посланих *i*-им студентом для працевлаштування в *j*-ту компанію. Вивести матрицю на консоль. Визначити студентів, які послали резюме у всі компанії, компанію, в яку послали найменшу кількість резюме, студента, який не послав резюме в жодну компанію. Результати вивести на консоль. 7. Із згенерованої раніше матриці видалити студента, який не послав жодного резюме в компанію. Вивести перетворену матрицю на консоль. Якщо такого студента немає, вивести відповідне повідомлення. 8. Знайти корені нелінійного рівняння , застосувавши метод половинного ділення (метод бісекції). Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести з консолі рядок символів, який містить алфавітні та розділові символи (пробіли, точки, коми). Визначити кількість слів в рядку. Упорядкувати слова так, щоб вийшов "ланцюжок" слів, коли перша буква поточного слова збігається з останньою буквою попереднього слова, а остання буква поточного слова – з першою буквою наступного слова. Якщо ланцюжок створити неможливо, вивести відповідне повідомлення. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом сортування Шелла. Вивести на консоль масив до та після сортування. 2. У згенерованому масиві визначити прості числа, застосувавши решето Ератосфена, і надрукувати їх. Якщо прості числа не знайдені, вивести відповідне повідомлення. 3. У створеному масиві переставити елементи так, щоб спочатку йшли парні елементи, якщо їх індекси у вхідному масиві парні, потім непарні елементи, якщо їх індекси у вхідному масиві непарні, потім елементи з нульовим значенням, потім решта. Результати перетворення масиву вивести на консоль. 4. Знайти в масиві та вивести на консоль елементи та їх індекси, які є квадратами чисел, застосувавши алгоритм лінійного пошуку. 5. Визначити усі входження в масив заданого з консолі елемента, застосувавши метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Вивести на консоль індекси шуканого елемента. Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch класу Array 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. На перетині *i*-го рядка та *j*-го стовпчика матриці записана температура повітря в *i*-ий день *j*-го місяця. Вивести матрицю на консоль. Визначити місяці, середня температура в яких була вище за задану з консолі, день, в якому температура була найнижча по всіх місяцях, середню температура взимку. Результати вивести на консоль. 7. У згенеровану раніше матрицю додати рядок, який для кожного місяця буде містити середнє значення температури повітря за місяць. Вивести перетворену матрицю на консоль. 8. Знайти корені нелінійного рівняння , застосувавши метод половинного ділення (метод бісекції). Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести з консолі рядок символів, який містить алфавітні та розділові символи (пробіли, точки, коми). Визначити кількість слів в рядку. Якщо слово починається з голосної, його першу літеру перевести у верхній регістр. Великі літери в середині слова перевести у нижній регістр. Результати вивести на консоль. |
|  | Створити консольний застосунок мовою C#. Використати методи класів **Console**, **Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch**. Для створення масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість та діапазон значень з консолі. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом сортування включенням. Вивести на консоль масив до та після сортування. 2. У згенерованому масиві визначити прості числа, застосувавши решето Ератосфена, і надрукувати їх. Якщо прості числа не знайдені, вивести відповідне повідомлення. 3. У створеному масиві для кожної пари сусідніх елементів знайти найбільший спільний дільник і записати їх в новий одновимірний масив. Отриманий масив вивести на консоль. 4. Знайти в масиві та вивести на консоль елементи та їх індекси, які є центрованими трикутними числами, застосувавши алгоритм лінійного пошуку. Формула для обчислення *n*-го центрованого трикутного числа: . Якщо центровані трикутні числа не знайдені, вивести відповідне повідомлення. 5. Визначити усі входження в масив заданого з консолі елемента, застосувавши метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Вивести на консоль індекси шуканого елемента. Модифікувати функцію бінарного пошуку у масиві, використавши метод BinarySearch класу Array 6. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі, включаючи нульове значення. *i*-й рядок означує телеканал, *j*-й стовпчик означує телепрограму. На перетині *i*-го рядка та *j*-го стовпчика матриці записана кількість телепрограм. Вивести матрицю на консоль. Визначити телеканал, який має найбільшу кількість телепрограм, телепрограму, яку транслюють усі телеканали, середню кількість телепрограм по всіх телеканалам. Результати вивести на консоль. 7. Із згенерованої раніше матриці видалити рядки, номери яких задані з консолі і свідчать про закриття телеканалів. Вивести на консоль перетворену матрицю. 8. Знайти корені нелінійного рівняння , застосувавши метод Ньютона. Вивести результати на консоль. Здійснити перевірку правильності рішення, підставивши знайдені значення коренів в нелінійне рівняння. 9. Увести з консолі рядок символів, який містить алфавітні, цифрові та розділові символи (пробіли, точки, коми). Видалити з рядка усі символи, які не є буквами. Визначити кількість слів, які починаються та закінчуються на задану з консолі букви. слів в рядку. Результати вивести на консоль. |

### Література

1. О.С.Бичков, Є.В.Іванов Об’єктно-орієнтоване програмування мовою C#/

2.C# 2005 и платформа .Net 3.0 для профессионалов. Нейгел К., Ивьен Б. и др. –М.: ООО “И.Д. Вильямс", 2008. –1376 с.

# Розділ 2. Поняття класів і об’єктів. Принципи інкапсуляції, спадкування та поліморфізму

## Лабораторна робота № 3. Конструктори та аксесори класів, покажчик this, вкладені клаcи, статичні поля та методи класу, використання текстових файлів в C#

### Рейтинг лабораторної роботи №3

|  |  |  |  |
| --- | --- | --- | --- |
| № п.п | Вид діяльності студента | Рейтинговий бал | Deadline |
| 1 | Написання коду з 10 завдань | 0,5\*10 = 5 | 15 Березня |
| 2 | Захист роботи | 0,5 |
| 3 | Звіт з роботи | 0,5 |  |
| Разом за роботу | | 6 |  |

### Мета роботи

1. Навчитися створювати власні класи та використовувати їх в програмах.
2. Конструктори класів
3. Властивості (аксесори) класів
4. Вкладення класів
5. Часткові класи і методи
6. Статичні класи
7. Текстові файли

### Методичні рекомендації до виконання лабораторної роботи

1. Прочитайте лекцію та матеріал в підручнику.

2. Прочитайте методичні вказівки до лабораторної роботи та виконайте наведені в ній приклади (вони всі працездатні).

3. При вивченні теми лекції і виконанні завдань зверніть увагу на особливості створення конструкторів класу в C#:

* конструктор не повертає значення;
* конструктор може бути з параметрами і без параметрів;
* в класі можуть бути визначені декілька конструкторів з різними списками параметрів;
* якщо конструктор відсутній, то він створюється автоматично (конструктор за замовчуванням). Такий конструктор не має параметрів;
* якщо в класі визначений хоча б один конструктор, конструктор за замовчуванням автоматично системою не додається.

4. Також зверніть увагу, що для доступу до закритих полів в C# призначені спеціальні методи-властивості get i set. Якщо відсутня частина set, властивість доступна лише для читання (read-only), якщо відсутня частина get, властивість доступна лише для запису (write-only).

5. Для поглибленого вивчення цього матеріалу прочитайте розділи книги [1].

### Порядок виконання роботи

1. Створити директорію Lab3, в якій будуть розміщуватися проекти цієї лабораторної роботи.

2. Виконати завдання свого варіанту у вигляді окремих консольних проектів в одному рішенні.

3. Для кожного класу передбачити окремий файл

### Приклади виконання завдань

#### 1. Програма розрахунку рейтингу студента. Клас Student

##### Варіант 1. Конструктор з параметрами та створення об’єкту класу

Розглянемо клас Student, який містить відкриті поля і конструктор з параметри для їх ініціалізації. Клас містить один відкритий метод public void StudentRating(int R), який виводить відповідний текст, в залежності від рейтингу студента. В методі Main() виконується тестування класу.

Зверніть увагу, що опис класу розміщується в просторі імен, а не класі Program. Опис класу рекомендується розмістити в окремому файлі – модулі класу.

using System; //файл Student.cs

namespace ex1lab3

{

/// <summary>

/// клас студент з атрибутами і методами

/// </summary>

class Student

{

public string Name; //ім'я

public int Age; // вік

public string Role; // роль

public string Facultet; //факультет

public string Group; //група

public int Course; //курс

public int Rating; //рейтинг

/// <summary>

/// конструктор з параметрами для ініціалізації полів класу

/// </summary>

/// <param name="N">ім'я студента</param>

/// <param name="A">вік</param>

/// <param name="R">роль</param>

/// <param name="F">факультет</param>

/// <param name="G">група</param>

/// <param name="C">курс</param>

public Student(string N, int A, string R, string F, string G, int C)

{

Name = N;

Age = A;

Role = R;

Facultet = F;

Group = G;

Course = C;

}

/// <summary>

/// виведення рекомендацій студенту відповідно до його рейтингу

/// </summary>

/// <param name="R">рейтинг студента</param>

public void StudentRating(int R)

{

Rating = R;

if (Rating >= 80)

Console.WriteLine("Привiт вiдмiнникам");

else

if (Rating <= 30)

Console.WriteLine("Треба вчитися краще!");

else

Console.WriteLine("Можна вчитися ще краще!");

}

}

}

//======= файл Program.cs містить точку входу в програму Main()=======

using System;

namespace ex1lab3

{

class Program

{

static void Main(string[] args)

{

//дані рейтингу

//ініціалізація полів класу виконується в конструкторі з параметрами

Student newStudent = new Student("Бака", 20, "студент", "КННІ", "K-01", 3);

Console.WriteLine("Хто ви?");

Console.WriteLine("Прiзвище = " + newStudent.Name);

Console.WriteLine("Вiк= " + newStudent.Age);

Console.WriteLine("Роль= " + newStudent.Role);

Console.WriteLine("Факультет = " + newStudent.Facultet);

Console.WriteLine("група= " + newStudent.Group);

Console.WriteLine("курс= " + newStudent.Course);

Console.WriteLine("Вкажiть Ваш рейтинг?");

string r = Console.ReadLine();

newStudent.Rating = int.Parse(r);

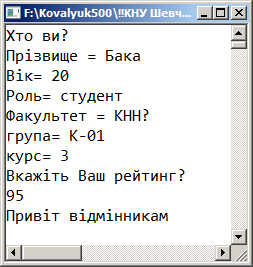
newStudent.StudentRating(newStudent.Rating);

Console.ReadLine();

}

}

}



Змінимо клас, зробимо поля Name i Age закритими, а доступ до них реалізуємо через властивості get і set.

##### Варіант 2. Використання властивостей get і set замість присвоєння в конструкторі.

using System; //файл Student.cs для класу Student

using System.Collections.Generic;

using System.Text;

namespace ex2Lab3

{

class Student

{

//закриті поля класу

string Name;

int Age;

//відкриті поля класу

public string Role; // роль

public string Facultet;

public string Group;

public int Course;

public int Rating;

/// <summary>

/// конструктор класу з параметрами

/// </summary>

/// <param name="F">факультет</param>

/// <param name="G">група</param>

/// <param name="C">курс</param>

/// <param name="R">роль</param>

public Student(string F, string G, int C, string R)

{

//конструктор з параметрами

Role = R;

Facultet = F;

Group = G;

Course = C;

}

/// <summary>

/// властивість для доступу до закритого поля Name через get і set для полів класу

/// </summary>

public string name

{

get

{

return Name;

}

set

{

Name = value;

}

}

/// <summary>

/// властивість для доступу до закритого поля Age через get і set для полів класу

/// </summary>

public int age

{

get

{

return Age;

}

set

{

Age = value;

}

}

/// <summary>

/// виведення повідомлень в залежності від значення рейтингу

/// </summary>

/// <param name="Rat">рейтинг студента</param>

public void StudentRating(int Rat)

{

Rating = Rat;

if (Rating >= 80)

Console.WriteLine("Привіт відмінникам");

else

if (Rating <= 30)

Console.WriteLine("Треба вчитися краще!");

else

Console.WriteLine("Можна вчитися ще краще!");

}

}

}

//файл Program.cs для класу Program

using System;

namespace ex2Lab3

{

class Program

{

static void Main(string[] args)

{

//дані рейтингу

//ініціалізація полів класу виконується в конструкторі з параметрами

Student newStudent = new Student("КННІ", "K-01", 3,"Студент");

Console.WriteLine("Хто ви?");

//використовуємо властивість

newStudent.name = Console.ReadLine();

Console.WriteLine("Скiльки вам рокiв?");

//використовуємо властивість

newStudent.age = int.Parse(Console.ReadLine());

Console.WriteLine("Прiзвище = " + newStudent.name);

Console.WriteLine("Вiк= " + newStudent.age);

Console.WriteLine("Роль= " + newStudent.Role);

Console.WriteLine("Факультет = " + newStudent.Facultet);

Console.WriteLine("група= " + newStudent.Group);

Console.WriteLine("курс= " + newStudent.Course);

Console.WriteLine("Ваш рейтинг?");

string r = Console.ReadLine();

// привсоєння значення відкритому полю класу

newStudent.Rating = int.Parse(r);

newStudent.StudentRating(newStudent.Rating);

Console.ReadLine();

}

}

}

#### 2. Програма автоматизованого обліку банківських відомостей

Програма є автоматизованою системою обліку банківських відомостей. На кожного клієнта банку зберігаються наступні відомості:

* прізвище, ім’я, по-батькові;
* дата народження;
* паспортні дані;
* ідентифікаційний код;
* місце роботи (навчання);
* номери рахунків.

Для кожного клієнта визначимо операції:

* додати нового клієнта;
* видалити клієнта;
* змінити реквізити клієнта.

На кожному рахунку зберігається інформація про поточний баланс. З кожним рахунком можна виконувати наступні дії:

* відкриття, закриття;
* внесення грошей, зняття грошей;
* перегляд балансу.

Створимо два класи:

Класс *Client* для опису інформації про клієнта;

Клас *Account* для опису банківського рахунку

##### Клас Client. Варіант 1 - конструктор без параметрів, властивості класу

using System;

using System.Collections.Generic;

using System.Text;

namespace ex3Lab3

{

class Client

{

//поля класу

private string Name;

private DateTime BirthDate;

private string Passport;

private string ID;

private string Job;

private string nomAccount;

/// <summary>

/// конструктор без параметрів

/// </summary>

public Client()

{

}

}

}

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 закритих полів класу.

using System;

using System.Collections.Generic;

using System.Text;

namespace ex3Lab3

{

class Client

{

//поля класу

private string Name;

private DateTime BirthDate;

private string Passport;

private string ID;

private string nomAccount;

/// <summary>

/// конструктор без параметрів

/// </summary>

public Client()

{

}

/// <summary>

/// методи get і set для закритого поля Passport класу

/// </summary

public string passport

{

get

{

return Passport;

}

set

{

Passport = value;

}

}

/// <summary>

/// методи get і set для закритого поля Name

/// </summary>

public string name

{

get

{

return Name;

}

set

{

Name = value;

}

}

/// <summary>

/// ////////////////

/// </summary>

public int age

{

get

{

int a;

a = DateTime.Now.Year - BirthDate.Year;

return a;

}

}

/// <summary>

/// методи get і set для закритого поля BirthDate

/// </summary>

public DateTime birthdate

{

get

{

return BirthDate;

}

set

{

if (DateTime.Now > value)

BirthDate = value;

else

throw new Exception("Введена невірна дата народження");

}

}

/// <summary>

/// методи get і set для закритого поля ID

/// </summary>

public string ID\_kod

{

get { return ID; }

set

{

ID = value;

}

}

/// <summary>

/// методи get і set для закритого поля nomAccount

/// </summary>

public string Nom\_Account

{

get { return nomAccount; }

set

{

nomAccount = value;

}

}

}

}

Як видно з прикладу, властивість складається з методів **set і get**. При цьому властивість повинна містити хоч би один з методів. Метод *set* дозволяє змінювати значення поля класу, *get* − отримувати значення. У метод *set* передається значення параметра за допомогою змінної *value*. Обидва методи можуть містити довільну кількість операторів, що описують алгоритм виконання дій в процесі читання або запису значення в полі класу. У даному прикладі властивості *passport* і *name* дозволяють просто дістати доступ до полів класу, читаючи або встановлюючи значення відповідних змінних. Властивість *birthdate* також призначена для читання і запису значення змінної - члена класу *BirthDate*. При цьому при читанні значення (операція *get*) відбувається просто передача значення змінної *BirthDate*, при спробі ж запису нового значення в цю змінну відбувається перевірка допустимості встановлюваного значення змінної. В даному випадку перевірка зводиться до порівняння нового значення дати народження з поточною датою. Якщо встановлюване значення дати народження більше або рівно поточній даті, генерується виключення, яке не дозволяє записати нове значення в змінну, - член класу.

Властивість *age* застосовується для отримання поточного віку клієнта. Вона призначена лише для читання значення із змінної, тому містить лише метод *get*. При використанні властивості *age* відбувається обчислення поточного значення віку клієнта в роках шляхом віднімання року народження від поточного значення року.

Використання властивостей аналогічно використанню змінних. У наступному прикладі створюється об'єкт *с1* класу *Client*. Потім поля цього об'єкту заповнюються значеннями з використанням властивостей. Після цього на екран виводяться значення полів, для цього також застосовуються властивості класу:

using System;

namespace ex3Lab3

{

class Program

{

static void Main(string[] args)

{

Client c1 = new Client();

c1.name = "Вася";

c1.passport = "9002";

c1.birthdate = new DateTime(2003, 08, 03);

c1.ID\_kod = "123456789";

c1.Nom\_Account = "8097";

Console.WriteLine("Имя=" + c1.name);

Console.WriteLine("Паспорт=" + c1.passport);

Console.WriteLine("Возраст=" + c1.age);

Console.WriteLine("Иден.код=" + c1.ID\_kod);

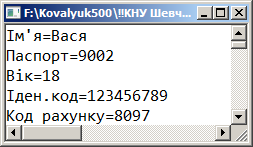
Console.WriteLine("Код счета=" + c1.Nom\_Account);

Console.ReadKey();

}

}

}



##### Клас Client. Варіант 2 - конструктор класу з параметрами

/// <summary>

/// конструктор з параметрами

/// </summary>

/// <param name="ClientName">ім'я клієнта</param>

/// <param name="ClientPassport">паспорт</param>

/// <param name="ClientBirthDate">дата народження</param>

public Client(string ClientName, string ClientPassport, DateTime  
 ClientBirthDate)

{

name = ClientName;

passport = ClientPassport;

birthdate = ClientBirthDate;

}

Видно, що конструктор має три параметри. У тілі конструктора відбувається запис переданих як параметри значень у відповідні поля класу за допомогою властивостей даного класу. У випадку з датою народження це дозволяє не дублювати процедуру перевірки введеної дати, а скористатися алгоритмом, реалізованим у властивості *birthdate*.

Створимо також метод, що дозволяє змінити значення полів об'єкту класу Client:

/// <summary>

/// редагування полів

/// </summary>

/// <param name="ClientName">ім'я клієнта</param>

/// <param name="ClientPassport">паспорт</param>

/// <param name="ClientBirthDate">дата народження</param>

public void EditClient(string ClientName, string ClientPassport,  
 DateTime ClientBirthDate)

{

Name = ClientName;

Passport = ClientPassport;

birthdate = ClientBirthDate;

}

Як видно з прикладу, код цього методу практично повністю ідентичний конструктору з параметрами з різницею в імені, а також в типі значення, яке повертається. Звичайно, в даному випадку можна було б обійтися і використанням властивостей для зміни значень полів класу, проте, інколи буває корисно, аби такого роду зміни були реалізовані в рамках одного методу, тим більше, якщо алгоритм змін є нестандартним.

Тепер, з використанням конструктора з параметрами, можна створити і відразу ж ініціалізувати об'єкт класу *Client*:

Client c1 = new Client("Вася", "9002", new DateTime(2003, 08, 03));

Змінений код з викликом редагування полів класу:

static void Main(string[] args)

{

Client c1 = new Client("Вася", "9002", new DateTime(2003, 08, 03));

c1.ID\_kod = "123456789";

c1.Nom\_Account = "8097";

Console.WriteLine("Iм'я=" + c1.name);

Console.WriteLine("Паспорт=" + c1.passport);

Console.WriteLine("Вiк=" + c1.age);

Console.WriteLine("Iден.код=" + c1.ID\_kod);

Console.WriteLine("Код рахунку=" + c1.Nom\_Account);

Console.ReadKey();

// редагування полів

c1.EditClient("Iван", "4567", new DateTime(2000, 09, 01));

//Виведення значень відредагованих полів

Console.WriteLine("Виведення значень вiдредагованих полiв");

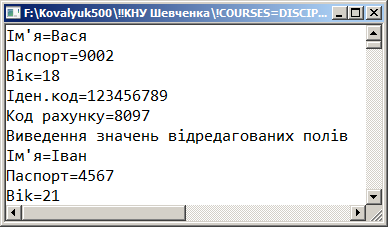
Console.WriteLine("Iм'я=" + c1.name);

Console.WriteLine("Паспорт=" + c1.passport);

Console.WriteLine("Вik=" + c1.age);

Console.ReadKey();

}



##### Клас Account. Варіант 1

У першому варіанті класу *Account* активно використовуватимемо поля класу. Окрім двох основних полів *credit* і *debit*, які зберігають надходження і витрати 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 в класу багато, і як наслідок, в методах класу аргументів буде небагато.

class Account

{

//закриті поля класу

int debit = 0, credit = 0, balance = 0;

int sum = 0, result = 0;

/// <summary>

/// Зарахування на рахунок з перевіркою

/// </summary>

/// <param name="sum">сума, що зараховується</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>

/// Зняття з рахунку з перевіркою

/// </summary>

/// <param name="sum">сума, що знімається</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>

/// Повідомлення про виконанн операції

/// </summary>

void Mes()

{

switch (result)

{

case 1:

Console.WriteLine("Операцiя зарахування грошей пройшла успiшно!");

Console.WriteLine("Cума={0},Ваш поточний баланс={1}", sum, balance);

break;

case 2:

Console.WriteLine("Операцiя зняття грошей пройшла успiшно!");

Console.WriteLine("Cума={0},Ваш поточний баланс={1}", sum, balance);

break;

case -1:

Console.WriteLine("Операцiя зарахування грошей не виконана!");

Console.WriteLine("Сума має бути бiльше за нуль!");

Console.WriteLine("Cума={0},Ваш поточний баланс={1}", sum, balance);

break;

case -2:

Console.WriteLine("Операцiя зняття грошей не виконана!");

Console.WriteLine("Сума має бути не бiльше балансу!");

Console.WriteLine("Cума={0},Ваш поточний баланс={1}", sum, balance);

break;

default:

Console.WriteLine("Невiдома операцiя!");

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 пройшла операція, і інформуючи клієнта про поточний стан його балансу.

##### Клас Account. 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 методах, що забезпечують необхідну передачу інформації. От як виглядає цей клас:

using System;

using System.Collections.Generic;

using System.Text;

namespace ex3Lab3

{

class Account1

{

//закриті поля класу

int debit = 0, credit = 0;

/// <summary>

/// Зарахування на рахунок з перевіркою

/// </summary>

/// <param name="sum">зарахована сума</param>

public void putMoney(int sum)

{

int res = 1;

if (sum > 0) credit += sum;

else res = -1;

Mes(res, sum);

} //putMoney

/// <summary>

/// Зняття з рахунку со счета з перевіркою

/// </summary>

/// <param name="sum"> сума, що знімається</param>

public void getMoney(int sum)

{

int res = 2;

if (sum <= balance()) debit += sum;

else res = -2;

balance();

Mes(res, sum);

}//getMoney

/// <summary>

/// розрахунок балансу

/// </summary>

/// <returns>поточний баланс</returns>

int balance()

{

return (credit - debit);

}

/// <summary>

/// Повідомлення про виконання операції

/// </summary>

void Mes(int result, int sum)

{

switch (result)

{

case 1:

Console.WriteLine("Операцiя зарахування грошей пройшла успiшно!");

Console.WriteLine("Cумма={0},Ваш текущий баланс={1}", sum, balance());

break;

case 2:

Console.WriteLine("Операцiя зняття грошей пройшла успiшно!");

Console.WriteLine("Cумма={0},Ваш текущий баланс={1}", sum, balance());

break;

case -1:

Console.WriteLine("Операцiя зарахування грошей не виконана!");

Console.WriteLine("Сума має бути бiльше за нуль!");

Console.WriteLine("Cума={0},Ваш поточний баланс={1}", sum, balance());

break;

case -2:

Console.WriteLine("Операцiя зняття грошей не виконана!");

Console.WriteLine("Сума має бути не бiльше балансу!");

Console.WriteLine("Cума={0},Ваш поточний баланс={1}", sum, balance());

break;

default:

Console.WriteLine("Невiдома операцiя!");

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 плата, в класі з'явився додатковий метод *balance*(), що багато разів викликається, і в методі *Mes*() тепер з'явилися два аргументи. Який клас кращий? Однозначно сказати не можна, все залежить від контексту, від пріоритетів, заданих при створенні конкретної системи.

Метод *TestAccounts*() класу Program, що тестує роботу з класами Account і Account, має викликатися у методі *Main()*:

using System;

namespace ex3Lab3

{

class Program

{

static void TestAccounts()

{

Account myAccount = new Account();

myAccount.putMoney(6000);

myAccount.getMoney(2500);

myAccount.putMoney(1000);

myAccount.getMoney(4000);

myAccount.getMoney(1000);

//Аналогічна робота з класом Account1

Console.WriteLine("Новий клас i новий рахунок!");

Account1 myAccount1 = new Account1();

myAccount1.putMoney(6000);

myAccount1.getMoney(2500);

myAccount1.putMoney(1000);

myAccount1.getMoney(4000);

myAccount1.getMoney(1000);

Console.WriteLine("Кiнець роботи");

Console.ReadLine();

}// TestAccounts

static void Main(string[] args)

{

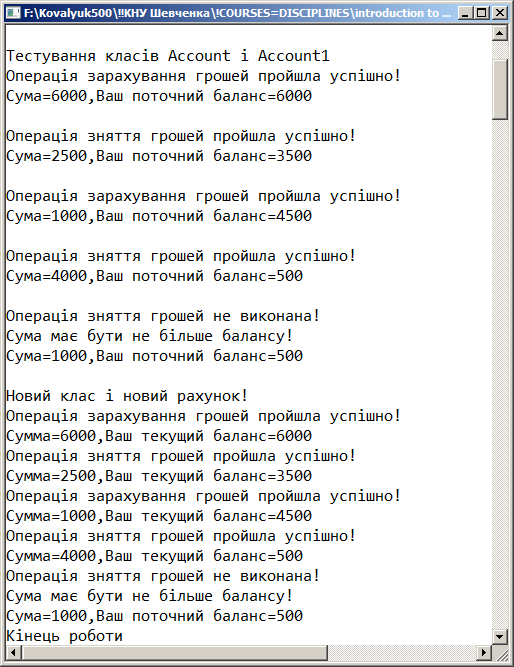
Console.WriteLine("Тестування класiв Account i Account1 ");

TestAccounts();

}

}

}



#### 3. Вбудовані (вкладені) класи

У мові C# будь-який клас у своїй реалізації може містити оголошення іншого класу. Клас, що оголошується в межах фігурних дужок іншого класу, називається вкладеним класом. Вкладені класи можуть мати модифікатори доступу *public, protected, internal, protected internal, private* або *private protected*. Об‘єкт вкладеного класу можна оголосити у випадку, якщо вкладений клас оголошений як видимий (не як *private*).

Нехай *Outer* – ім’я класу, який містить в собі оголошення іншого класу з іменем *Inner*, *Inner* – ім’я класу, який оголошується в класі *Outer*. Якщо вкладений клас оголошено як не *private*-клас, то створення екземпляру цього класу має такий вигляд: Outer.Inner objInner = new Outer.Inner();

using System;

using System.Collections.Generic;

using System.Text;

namespace ex6Lab3

{

class Outer

{

// внутрішні змінні класу Outer

public int d;

static public int sd;

// доступ до private-класу Inner1 з внутрішнього методу класу

public Inner GetInner()

{

Console.WriteLine("Метод GetInner() зовнiшнього класу");

Inner i1 = new Inner(); // створити екземпляр класу Inner1

// доступ до члена даних класу Inner1 через екземпляр класу

i1.d1 = 25;

// доступ до статичного члена вкладеного класу Inner1

Inner.sd1 = 30;

return i1;

}

// private-вкладений клас Inner

public class Inner

{

public int d1;

public static int sd1;

public string Method()

{

Console.WriteLine("Метод внутрiшнього класу ");

string str = " результат вкладеного класу Inner";

return str;

}

}

}

}

using System;

namespace ex6Lab3

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello World!");

// Використання вкладених класів Outer.Inner1, Outer.Inner2, Outer

// 1. Оголосити об'єкт класу Outer

Outer o = new Outer();

o.d = 300; // доступ до змінної через екземпляр класу Outer

Outer.sd = 230; // доступ до статичної змінної

Console.WriteLine("Поля зовнiшнього класу: "+ o.d+" "+ Outer.sd);

Console.WriteLine("Main:"+ o.GetInner());

// 2. Оголосити об'єкт public-класу Inner

Outer.Inner i1 = new Outer.Inner();

i1.d1 = 440;

Outer.Inner.sd1 = 500; // доступ до статичної змінної внутрішнього класу

Console.WriteLine("Поля внутрiшнього класу " + i1.d1 + " " + Outer.Inner.sd1);

Console.WriteLine("Main:"+i1.Method());

}

}

}

#### Часткові класи (класи, що розділяються) і часткові методи

Класи можуть бути частковими. Тобто ми можемо мати кілька файлів з визначенням одного і того самого класу, і при компіляції всі ці визначення будуть скомпільовані в одне. Наприклад, визначимо в проекті два файли з кодом.

// частина 1 класу Person

public partial class Person

{

public void Move()

{

Console.WriteLine("I am moving");

}

}

// частина 2 класу Person

public partial class Person

{

public void Eat()

{

Console.WriteLine("I am eating");

}

}

Отже, два файли в проекті містить визначення одного і того самого класу *Person*, які містять два різних методу. І обидва визначені класи є **частковими**. Для цього вони визначаються з ключовим словом **partial**.

Ключове слово *partial* вказує, що інші частини класу, структури або інтерфейсу можуть бути визначені в просторі імен. Всі частини повинні використовувати ключове слово *partial*. Для формування остаточного класу всі частини повинні бути доступні під час компіляції. Всі частини повинні мати однакові модифікатори доступу, наприклад public, private тощо.

Приклад використання методів часткових класів:

class Program

{

static void Main(string[] args)

{

Person Vadim = new Person();

Vadim.Move(); // метод першого часткового класу

Vadim.Eat(); // метод другого часткового класу

Console.ReadKey();

}

}

Часткові класи можуть містити часткові методи. Такі методи також визначаються з ключовим словом *partial*. Причому визначення часткового методу без тіла методу знаходиться в одному частковому класі, а реалізація цього самого методу – в іншому частковому класі..

public partial class Person

{

partial void DoSomething(); //оголошення часткового методу

public void OneMethod()

{

Console.WriteLine("Start");

DoSomething(); //call method of second partion class

Console.WriteLine("Finish");

}

}

public partial class Person

{

//визначення часткового методу

partial void DoSomething()

{

Console.WriteLine("I am reading a book");

}

}

Як правило, часткові методи завжди закриті (private). Виклик часткових методів здійснюється через відкриті методи

Person Ivan = new Person();

Ivan. OneMethod(); //метод викликає закритий частковий метод

#### Текстові файли

В С# для роботи з файлами і потоками використовуються класи **FileStream, StreamWriter**, **StreamReader, FileInfo** і інші класи.

Для введення і виведення даних не потоками, а рядками призначені класи StringReader, StringWriter.

Розглянемо приклад**.** В цьому прикладі спочатку створюється об'єкт **f** класу StreamWriter, який містить файл для виводу **output.txt**. Якщо не вказати повний шлях до файлу як нашому прикладі, то файл буде створений в директорії, де розміщується exe-файл програми: ...\\bin\\debug\\output.txt

Далі визначаються і ініціалізуються змінні **int i** та **string s**, які записуються у вихідний файл. Метод Close() закриває файл. В другій частині прикладу по черзі читаються рядки з файлу “input.txt", перетворюються у числові типи і виводяться на консоль.

using System;

using System.IO;

namespace ex7Lab3

{

class Program

{

static void Main(string[] args)

{

//запис у текстовий файл

Console.WriteLine("Hello World!");

StreamWriter fout = new StreamWriter("output.txt"); // 2

int i = 3;

string s1 = "Мова програмування С# - це C++++ ";

fout.WriteLine("i = " + i); // 3

fout.WriteLine("s1 = " + s1); // 4

fout.Close(); // 5

//=========читання з текстового файлу================

StreamReader fin = new StreamReader("f:\\input.txt");

string s2 = fin.ReadLine(); // читати рядок

Console.WriteLine("s = " + s2);

char c = (char)fin.Read(); // читати символ

fin.ReadLine();

Console.WriteLine("c = " + c);

string buf = fin.ReadLine(); //читати рядок і конвертувати у ціле число

int j = Convert.ToInt32(buf);

Console.WriteLine(j);

buf = fin.ReadLine(); //читати рядок і конвертувати у число double

double x = Convert.ToDouble(buf);

Console.WriteLine(x);

buf = fin.ReadLine(); //читати рядок і конвертувати у число double

double y = double.Parse(buf);

Console.WriteLine(y);

buf = fin.ReadLine(); //читати рядок і конвертувати у ціле число

decimal z = decimal.Parse(buf);

Console.WriteLine(z);

fin.Close();

}

}

}

### Варіанти завдань для лабораторної роботи № 3

Завдання передбачають такі дії :

Завдання 1, 2 передбачають розробку класу, що містить закриті поля та відкриті методи;

Завдання 3, 4 передбачають розробку конструкторів за замовчування та з параметрами;

Завдання 5 – розробка властивостей *get, set* для доступу до закритих полів класу;

Завдання 6 – розробка методів виведення на консоль значень полів класу

Завдання 7 – розробка методів за сценарієм роботи програми з використанням текстових файлів для збереження результатів роботи програми;

Завдання 8 – доповнення класів вбудованими (вкладеними класами). Рекомендується у вигляді другої версії лабораторної роботи.

Завдання 9 – доповнення попередньої версії програми частковими класами та методами. Зробити у вигляді третьої версії лабораторної роботи

Завдання 10 – доповнення попередньої версії новим статичним класом.

В методі *Main*() класу *Program* продемонструвати виклик усіх методів усіх класів. Усі значення, що розраховуються, записувати до *текстових файлів*.

**Студент має право додати додаткові поля та методи в класи завдань свого варіанту**. **Номер варіанта визначається за порядковим номером в журналі.**

|  |  |
| --- | --- |
| **№ варі-анту** | **Зміст завдання** |
|  | Розробити консольний застосунок мовою C# для обробки даних про студентів і викладачів.   1. Створити класи **Teacher** та **Student**. Рекомендується кожний клас створювати в окремому файлі командою Project → Add class 2. Визначити в класах закриті поля:  * клас **Teacher**: ім’я викладача, обсяг його навчального навантаження в годинах, назва дисципліни, яку читає викладач, кількість студентів, яким викладач читає дисципліну; * клас **Student**: FirstName, LastName, Adress, Passport, Age, Telephon, Rating  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Teacher**:   + запис у текстовий файл імені викладача, обсягу навчального навантаження, назви дисципліни та кількості студентів, яким викладач читає дисципліну;   + збільшення (зменшення) кількості студентів, які вивчають дисципліну, і відповідне збільшення (зменшення) обсягу навчального навантаження викладача; * клас **Student**:   + визначення рейтингу студента як середнє арифметичне його оцінок, які задавати з консолі;   + запис імені, прізвища студента та його рейтингу у текстовий файл.  1. Додати до класу **Student** вбудований (вкладений) клас Дипломний\_проект (**DiplomaProject**) з полями:    * тема дипломного проекту,    * кількість реалізованих алгоритмів,    * складність теми.   та методами:   * + вибір теми з переліку. Алгоритм вибору теми такий: перелік подати текстовим файлом, з консолі ввести ключові слова, читати рядки файлу в пам’ять, шукати входження ключових слів в прочитані рядки, вивести на консоль рядок, в якому знайдені задані ключові слова.   + визначення оцінки дипломного проекту в залежності від кількості реалізованих алгоритмів і складності теми.  1. Модифікувати клас **Student**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод вибору теми, в іншу (інший файл) – метод розрахунку оцінки за дипломний проект. 2. Додати до проекту новий статичний клас **СreativeWork**, включивши в нього три функції (на вибір студента) з варіанта 1 лабораторної роботи 2. |
|  | Розробити консольний застосунок мовою C# для обробки даних про кафедру і студентів.   1. Створити класи: **Department** та **Student**. Рекомендується кожний клас створювати в окремому файлі командою Project → Add class 2. Визначити в класах закриті поля:  * клас **Department**: назва кафедри, кількість викладачів, кількість студентів, назва освітньої програми, за якою навчаються студенти (наприклад «Інженерія програмного забезпечення»), оцінка з акредитації освітньої програми (A, B, E); * клас **Student**: FirstName, LastName, назва освітньої програми, рік навчання (курс), Rating,  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Department**:   + запис у текстовий файл назви кафедри, назви освітньої програми, кількості студентів та викладачів, оцінка з акредитації освітньої програми (A, B, E);   + збільшення (зменшення) кількості студентів, які навчаються за освітньою програмою в залежності від оцінки з акредитації: оцінка А – збільшення кількості студентів на 20%, оцінка В кількість студентів не змінюється, оцінка Е (повторна акредитація), кількість студентів зменшується на 10%. * клас **Student**:   + визначення рейтингу студента як середнє арифметичне 10 оцінок, які генерувати методом класу Random;   + запис імені, прізвища, курсу, назви освітньої програми, рейтингу студента у текстовий файл.  1. Додати до класу Student вбудований (вкладений) клас Конкурсна робота (**ContestWork**) з полями:    * назва конкурсу,    * назва роботи    * апробація роботи (кількість наукових статей, написаних на тему конкурсної роботи).   та методами:   * + відповідність назви конкурсної роботи тематиці конкурсу. Алгоритм перевірки відповідності назви роботи тематиці конкуру такий: перелік тематики конкурсу подати рядками в текстовому файлі, в циклі читати рядок з файлу, шукати збіги ключових слів з назви конкурсної роботи з прочитаним рядком з файлу, вивести на консоль рядки файлу, в яких знайдені збіги ключових слів з назви роботи з тематикою конкурсу.   + визначення якості конкурсної роботи за алгоритмом: 0 наукових статей – 0 балів за якість, наступні бали складають арифметичну прогресію з різницею (кроком) рівним; . Максимальна сума балів 10 за 5 наукових статей.  1. Модифікувати клас **Student**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод перевірки відповідності назви конкурсної роботи тематиці конкурсу, в іншу (інший файл) – метод розрахунку балів за конкурсну роботу. 2. Додати до проекту новий статичний клас **СreativeWork**, включивши в нього три функції (на вибір студента) з варіанта 2 лабораторної роботи 2. |
|  | Розробити консольний застосунок мовою C# для обробки даних про кафедру і викладачів.   1. Створити класи: **Department** та **Teacher**. Рекомендується кожний клас створювати в окремому файлі командою Project → Add class 2. Визначити в класах закриті поля:  * клас **Department**: назва кафедри, кількість викладачів, кількість студентів, кількість дисциплін, закріплених за кафедрою, обсяг навчального навантаження кафедри (загальна кількість годин на усі дисципліни) * клас **Teacher**: FirstName, LastName, кількість наукових праць, назва дисципліни, яку читає викладач, кількість годин, що відводиться на дисципліну, сумарний обсяг педагогічного навантаження (сумарна кількість годин викладача по всім дисципліна, які він викладає).  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Department**:   + запис у текстовий файл значень полів класу (назва кафедри, кількість викладачів, кількість студентів, кількість дисциплін, загальна кількість годин);   + збільшення (зменшення) кількості викладачів в залежності від загального обсягу навантаження кафедри та кількості студентів: співвідношення кількості студентів і викладачів 10:1, максимальне навантаження викладача за рік не перевищує 600 годин, мінімальний обсяг одної дисципліни складає 90 годин. * клас **Teacher**:   + визначення рейтингу викладача за алгоритмом: за кожні 100 годин навантаження нараховується 10 рейтингових балів, за кожну наукову статтю нараховується 10 рейтингових балів;   + запис імені, прізвища, назви дисципліни, рейтингу викладача у текстовий файл.  1. Додати до класу **Teacher** вбудований (вкладений) клас Керівництво конкурсними роботами (**ContestWorkManage**r) з полями:    * назва конкурсу,    * назва роботи    * призове місце, отримане конкурсною роботою на конкурсі (1,2,3). Значення 0 свідчить про те, що робота не попала в номінацію призових.   та методами:   * + Відповідність назви конкурсної роботи тематиці конкурсу. Алгоритм перевірки відповідності назви роботи тематиці конкуру такий: перелік тематики конкурсу подати рядками в текстовому файлі, в циклі читати рядок з файлу, шукати збіги ключових слів з назви конкурсної роботи з прочитаним рядком з файлу, вивести на консоль рядки файлу, в яких знайдені збіги ключових слів з назви роботи з тематикою конкурсу.   + Розрахунок рейтингу конкурсної роботи за алгоритмом: згенерувати методами класу Random 3 цілих числа в діапазоні від 0 до 10 (числа свідчать про наукову новизну, практичну цінність, якість оформлення), Сума балів від 25 до 30 свідчить про перше призове місце, сума балів від 20 до 24 – друге призове місце, сума балів від 15 до 19 – третє призове місце, сума балів менше за 15 – робота в номінацію не попала .  1. Модифікувати клас **Teacher**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод перевірки відповідності назви конкурсної роботи тематиці конкурсу, в іншу (інший файл) – метод розрахунку рейтингу конкурсної роботи. 2. Додати до проекту новий статичний клас **InterProject** (міжнародний проект), включивши в нього три функції (на вибір студента) з варіанта 3 лабораторної роботи 2. |
|  | Розробити консольний застосунок мовою C# для обробки даних про факультет і кафедру.   1. Створити класи: **Faculty** та **Department**. Рекомендується кожний клас створювати в окремому файлі командою Project → Add class 2. Визначити в класах закриті поля:  * клас **Faculty**: назва факультету, кількість кафедр, кількість спеціальностей, загальна кількість студентів за всіма спеціальностями; * клас **Department**: назва кафедри, кількість викладачів, кількість студентів, кількість дисциплін, закріплених за кафедрою, обсяг навчального навантаження кафедри (загальна кількість годин на усі дисципліни).  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі та виведення на консоль значень ініціалізованих полів класу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Faculty**:   + запис у текстовий файл значень полів класу (назва факультету, кількість кафедр, кількість спеціальностей);   + збільшення (зменшення) кількості кафедр в залежності від збільшення (зменшення) кількості спеціальностей (з розрахунку одна спеціальність відповідає одній кафедри), та кількості студентів (на спеціальності має навчатися не менше, ніж 200 студентів). * клас **Department**:   + запис у текстовий файл значень полів класу (назва кафедри, кількість викладачів, кількість студентів, кількість дисциплін, загальна кількість годин;   + розрахунок навчального навантаження одного викладача (співвідношення кількості студентів і викладачів 10:1, максимальне навантаження викладача за рік не перевищує 600 годин, мінімальний обсяг одної дисципліни складає 90 годин).  1. Додати до класу **Department** вбудований (вкладений) клас Філія\_кафедри (**BranchChair**) в ІТ-компанії з полями:    * Назва ІТ-компанії, в якій відкрита філія кафедри;    * Рейтинг ІТ-компанії в ІТ-галузі;    * Обсяг фінансів, які ІТ-компанія вкладає в розвиток кафедри.   та методами:   * + Покращення якості підготовки студентів. Показник якості підготовки студентів (індекс KPI) розраховувати як  Індекс KPI = ((Факт – База) / (Норма – База)) \* 100%, де Факт – фактичні результати роботи; База – допустиме мінімальне значення показника. Нижче базового рівня – відсутність результату; Норма – плановий рівень, те, що студент повинен виконувати обов'язково, нижче – студент не впорався зі своїми обов'язками; Ціль – значення, до якого потрібно прагнути, наднормативний показник, що дозволяє поліпшити результати.   + Оптимізація витрат ІТ-компанії за рахунок зменшення коштів на перенавчення молодих спеціалістів, зменшення витрат на рекламу вакансій, зменшення витрат на обладнання, зменшення витрат на навчальні курси, тренінги, семінари тощо. Числові дані генерувати методами класу Random.  1. Модифікувати клас **Department**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод покращення якості підготовки студентів, в іншу (інший файл) – метод розрахунку навчального навантаження одного викладача. 2. Додати до проекту новий статичний клас **CompanyProject** (проект ІТ-компанії), включивши в нього три функції (на вибір студента) з варіанта 4 лабораторної роботи 2. |
|  | Розробити консольний застосунок мовою C# для обробки даних про університет і факультет.   1. Створити класи: **University** та **Faculty**. Рекомендується кожний клас створювати в окремому файлі командою Project → Add class 2. Визначити в класах закриті поля:  * клас **University**: назва університету, адреса, кількість факультетів, кількість спеціальностей, загальна кількість студентів за всіма факультетами, рейтинг університету; * клас **Faculty**: назва факультету, кількість кафедр, кількість спеціальностей, загальна кількість студентів факультету за всіма спеціальностями.  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **University**:   + розрахунок рейтингу університету за версією Osvita.ua: загальний бал університету визначається як сума балів за рейтингами «SCОPUS», «ТОП200 Україна», «Бал ЗНО на контракт». Рейтинг визначається як порядковий номер університету в масиві відсортованих за зростанням загальних балів. Бали за різні номінації отримати в результаті генерації псевдовипадкових цілих чисел в діапазоні від 0 до 200 методами класу Random.   + Розрахунок розміру річного фінансування університету відповідно до його рейтингу, розміру, місця розташування та кількості студентів, враховуючи, що підготовка одного бюджетного студента в рік коштує від $2000 до $6000. Значення показників задати самостійно. * клас **Faculty**:   + запис у текстовий файл значень полів класу (назва факультету, кількість кафедр, кількість спеціальностей, кількість студентів факультету);   + збільшення (зменшення) кількості кафедр в залежності від збільшення (зменшення) кількості спеціальностей (з розрахунку одна спеціальність відповідає одній кафедри), та кількості студентів (на спеціальності має навчатися не менше, ніж 200 студентів).  1. Додати до класу **Faculty** вбудований (вкладений) клас Стартап\_Інкубатор факультету (**StartupIncubator**) з полями:    * Кількість стартап проектів, які розробляються в інкубаторе;    * Кількість студентів, ідеї яких реалізуються в інкубаторе;    * Обсяг інвестицій, які Стартап\_ Інкубатор отримає для впровадження своїх проектів.   та методами:   * + Вибір найкращого стартап проекту. Алгоритм метода такий. Вважатимемо, що є 10 проектів і 5 експертів, кожний з яких ставить проекту оцінку. В якості оцінок проекту виступають псевдовипадкові числа, що отримані за допомогою класу Random в діапазоні від 1 до 10. В результаті отримаємо матрицю вимірністю 5\*10 (кількість експертів\*кількість проектів). Для кожного проекту знаходимо суму балів (сума елементів по стовпчиках), потім розраховуємо середнє арифметичний бал для проекту, поділивши сумарний бал проекту на кількість експертів. Сортуємо середні бали за зростанням. Проект, який має найменший середній бал, вважатимемо найкращим.   + Рейтинг результативності студентів в стартап Інкубаторе. Алгоритм метода такий. Для кожного студента визначаємо долю (значення від 0 до 1) його участі в кожному проекті за допомогою генератору псевдовипадкових чисел. Отриману долю участі множимо на обсяг інвестицій, сортуємо отримані значення за спаданням. Студент з найвищим показником є найрезультативнішим.  1. Модифікувати клас **University**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод розрахунок рейтингу університету, в іншу (інший файл) – метод розрахунку розміру річного фінансування університету. 2. Додати до проекту новий статичний клас **StartupProject** (стартап проект), включивши в нього три функції (на вибір студента) з варіанта 5 лабораторної роботи 2. |
|  | Розробити консольний застосунок мовою C# для обробки даних про кафедру та ІТ компанію, з якою кафедра взаємодіє.   1. Створити класи: **Department** та **Company**. Рекомендується кожний клас створювати в окремому файлі командою Project → Add class 2. Визначити в класах закриті поля:  * клас **Department**: назва кафедри, кількість викладачів, кількість студентів, освітня програма (кількість дисциплін, закріплених за кафедрою), обсяг навчального навантаження кафедри (загальна кількість годин на усі дисципліни), кількість нагород студентів на конкурсах, тощо; * клас **Company**: назва компанії, рейтинг компанії в ІТ-галузі, кількість працівників компанії, величина щорічного прибутку компанії.  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Department**:   + Покращення якості освітньої програми за рахунок залучення для викладання фахівців ІТ-компанії (збільшення кількості викладачів та дисциплін) та виконання реальних проектів (збільшення кількості креативних студентів);   + Розрахунок розміру щорічних інвестицій в кафедру з боку ІТ-компанії із фонду прибутку ІТ-компанії. * клас **Company**:   + запис у текстовий файл значень полів класу (назва ІТ-компанії, рейтинг компанії в ІТ-галузі, кількість працівників компанії, величина щорічного прибутку компанії);   + збільшення кількості співробітників і прибутку ІТ-компанії за рахунок підготовки фахівців із складу випускників кафедри (10% випускників кафедри прийняті на роботу в ІТ-компанію)  1. Додати до класу **Company** вбудований (вкладений) клас ІТ акселератор (**ITAccelerator**) з полями:    * Кількість проектів, участь в яких беруть викладачі і студенти кафедри;    * Кількість студентів, які паралельно навчаються в акселераторі;    * Обсяг інвестицій, які акселератор передає на кафедру.   та методами:   * + Конкурс на вакантні посади в ІТ-компанію. Для цього розраховується рейтинг успішності студента, до якого додаються рейтинг участі в ІТ-проектах. Студенти з високим рейтингом збільшують кількість співробітників ІТ-компанії.   + Розрахунок обсягу інвестицій: витрати = (зарплата + вартість матеріалів + вартість обладнання + вартість електроенергії + інші витрати) \* відсоток прибутку.  1. Модифікувати клас **Department**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод розрахунку розміру щорічних інвестицій в кафедру, в іншу (інший файл) – метод визначення якості освітньої програми. 2. Додати до проекту новий статичний клас **StartupProject** (стартап проект), включивши в нього три функції (на вибір студента) з варіанта 6 лабораторної роботи 2. |
|  | Розробити консольний застосунок мовою C# для обробки даних про студента та його індивідуальну освітню траєкторію.   1. Створити класи: **Student** та **Curiculum** (індивідуальний навчальний план). Рекомендується кожний клас створювати в окремому файлі командою Project → Add class 2. Визначити в класах закриті поля:  * клас **Student**: FirstName, LastName, рік навчання (курс), група, рейтинг, стипендія; * клас **Curiculum**: назва освітньої програми (ОП), обсяг в годинах освітньої програми, кількість дисциплін ОП, .  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Student**:   + Визначення рейтингу студента як середній бал в сесії за 10 дисциплінами. Оцінки студента генерувати псевдовипадкоми цілими числами в діапазоні від 40 до 100.   + Розрахунок розміру стипендії, якщо рейтинг студента більше 80 балів. Розмір стипендії підвищується для відмінників. * клас **Curiculum**:   + запис у текстовий файл значень полів класу (назва освітньої програми (ОП), обсяг в годинах освітньої програми, кількість дисциплін ОП);   + Визначення якості освітньої програми за критеріями НАЗЯВО:  1. Додати до класу **Student** вбудований (вкладений) клас **WorkInCompany** (Робота в компанії) з полями:    * Назва компанії    * Кількість проектів, які виконує студент на роботі;    * Обсяг освітньої програми, яка корелюється (збігається за змістом) з профілем роботи студента.   та методами:   * + Розрахунок обсягу ОП, який можна зарахувати студенту як результат його ироботи в компанії в процесі виконання виробничих проектів;   + Формування рекомендацій щодо удосконалення індивідуального навчального плану класу **Curiculum** для наближення його до профіля роботи студента.  1. Модифікувати клас **WorkInCompany**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод розрахунок обсягу ОП, в іншу (інший файл) – метод Формування рекомендацій до зміни змісту ОП. 2. Додати до проекту новий статичний клас **IndividWork** (індивідуальна робота), включивши в нього три функції (на вибір студента) з варіанта 7 лабораторної роботи 2. |
|  | Розробити консольний застосунок мовою C# для обробки даних про клієнта та банкомат, в якому клієнт отримує готівку.   1. Створити класи: **Client** та **ATM** (банкомат). Рекомендується кожний клас створювати в окремому файлі командою Project → Add class. 2. Визначити в класах закриті поля:  * клас **Client**: FirstName, LastName, пін-код, номер і термін дії картки, * клас **ATM**: назва банку, якому належить банкомат, ідентифікаційний номер банкомату, місце його розташування, кількість завантажених грошей.  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.   6Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу.   1. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 2. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Client**:   + зміна пін-коду   + перегляд балансу, зняття готівки, поповнення рахунку. * клас **ATM**:   + блокування банківської картки через неправильний пін-код після третьої спроби;   + виведення меню команд банкомату, виведення повідомлень у разі невірного пін-коду або відсутності грошей в банкоматі,  1. Додати до класу **ATM** вбудований (вкладений) клас банку (**Bank**) з полями:    * Назва банку, адреса    * Ідентифікатори клієнта    * Кількість грошей на рахунку клієнта   та методами:   * + Збільшення, зменшення грошей на рахунку клієнта;   + Переказ грошей, оплата комунальних платежів з картки клієнта.  1. Модифікувати клас **ATM**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод блокування банківської картки, в іншу (інший файл) – методи виведення меню команд банкомату, виведення повідомлень у разі невірного пін-коду або відсутності грошей в банкоматі. 2. Додати до проекту новий статичний клас **BankCard** (банківська картка), включивши в нього дві функції з варіанта 8 лабораторної роботи 2, та дописавши метод, який розраховує відсотки з депозиту клієнту та скидає їх на картку. Для цього з консолі задати розмір депозиту та відсоткову ставку. |
|  | Розробити консольний застосунок мовою C# для обробки даних про студентів та студентський парламент університету, з яким взаємодіє студент.   1. Створити класи: **Student** та **Student**P**arliament** (студентський парламент). Рекомендується кожний клас створювати в окремому файлі командою Project → Add class 2. Визначити в класах закриті поля:  * клас **Student**: FirstName, LastName, курс, група, рейтинг, стипендія; * клас **Student**P**arliament**: назва факультету, голова студентського парламенту, контакти,  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Student**:   + Визначення рейтингу студента як середній бал в сесії за 10 дисциплінами. Оцінки студента генерувати псевдовипадковими цілими числами в діапазоні від 40 до 100.   + Розрахунок розміру стипендії, якщо рейтинг студента більше 80 балів. Розмір стипендії підвищити для відмінників. * клас **Student**P**arliament**:   + Формування рекомендацій щодо заохочення або відрахування студентів за неуспішність (за результатами рейтингу, який менше 40 до моменту перездач)   + Призначення стипендій студентам. Для цього визначити рейтинги студентів, визначити обсяг стипендіального фонду, визначити студентів, яким слід призначити соціальну стипендію, та студентів, яким слід призначити стипендію за результати навчання (рейтинг не менше 80 балів).  1. Додати до класу **StudentParliament** вбудований (вкладений) клас **StudCouncilCampus** (Студентська Рада студмістечка ) з полями:    * Ім’я голови студ ради, його контакти    * Кількість заходів, які здійснює студ. рада   та методами:   * + Формування списків студентів на поселення у гуртожиток на основі аналізу потреб студентів та можливостей кампусу.   + Контроль умов проживання студентів, дотримання студентами правил внутрішнього розпорядку, реагування на їх порушення (виселення, догана тощо).  1. Модифікувати клас **Student**P**arliament**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод розрахунок стипендії студентам, в іншу (інший файл) – метод Формування списків на відрахування студентів за неуспішність. 2. Додати до проекту новий статичний клас **IndividWork** (індивідуальна робота), включивши в нього дві функції (на вибір студента) з варіанта 9 лабораторної роботи 2 та дописавши метод, що визначає мотивації голови студентського парламенту. |
|  | Розробити консольний застосунок мовою C# для обробки даних про водія та автомобіль.   1. Створити класи **Driver** (водій) та **Car** (автомобіль). Рекомендується кожний клас створювати в окремому файлі командою Project → Add class 2. Визначити в класах закриті поля:  * клас **Driver**: FirstName, LastName, адреса проживання; телефон, номер автомобіля * клас **Car**: модель автомобіля, номер автомобіля, рік випуску, габарити машини, витрати палива, ціна  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Driver**:   + Реєстрація та технічний контроль автомобіля.   + Порушення ПДР водієм, як наслідок поломка авто і його заміна * клас **Car:**   + Поломка авто в результаті зіткнення або іншої дорожньої пригоди   + Розрахунок вартості ремонту автомобіля в результаті дорожньої аварії. Для цього задати з клавіатури назву складової частини авто, що потребує ремонту, вартість нормогодини роботи працівника ремонтної служби, згенерувати вартість запчастин авто в діапазоні від 500 до 5000, кількіть годин роботи ремонтника.  1. Додати до класу **Car** вбудований (вкладений) клас **CarEngine** (двигун) з полями:    * Потужність двигуна (в к.с.), витрати пального (в літрах на 100 км)    * Швидкість розгону (в км/год)   та методами:   * + Розрахунок витрат пального на задану з консолі довжину шляху, якість дороги, кліматичні умови тощо.   + Розрахунок тривалості маршруту при заданій швидкості руху, обсягу пального, параметрах двигуна.  1. Модифікувати клас **Car**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод Поломка авто, в іншу (інший файл) – метод Розрахунок вартості ремонту автомобіля. 2. Додати до проекту новий статичний клас **Passenger** (пасажир), включивши в нього дві функції:    * допомога водію в якості штурмана з прокладання маршруту руху;    * ігри, наприклад, «Міста». Для цього ввести рядок символів, значенням якого є назва міста, задати наступний рядок, який має починатися на останню літеру попереднього слова, повторювати дії, поки пасажир не закінчить гру. |
|  | Розробити консольний застосунок мовою C# для обробки даних про конференцію і автора, який подає на конференцію статтю.   1. Створити класи **Conference** (Конференція) та **Author** (Автор). Рекомендується кожний клас створювати в окремому файлі командою Project → Add class 2. Визначити в класах закриті поля:  * клас **Conference**: назва конференції, перелік тематики конференції, дата, місце проведення, вартість участі; * клас **Author**: ім’я, прізвище, вартість оплати участі, назва статті, дата подачі статті.  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Conference**:   + Реєстрація та рецензування статей, у тому числі відмова у прийняття статті за відсутності наукової та практичної новизни. Реалізувати реєстрацію статей із записом назв статей, імен авторів, назви тематики конференції, дати подання статті у текстовий файл.   + Реєстрація допущених учасників, розробка програми проведення конференції, інформування про дату та час дистанційної режиму проведення конференції та організація її дистанційного проведення. Реалізувати реєстрацію допущений учасників із записом імені учасника, назви його статті, дати та часу його виступу на конференції у текстовий файл. * клас **Author:**   + Реєстрація на конференцію із записом у текстовий файл прізвища автора, назви його статті та тематики конференції.   + Розрахунок вартості участі в конференції з урахуванням вартості проїзду, бронювання житла, організаційний збір, оплати збірника конференції.  1. Додати до класу **Author** вбудований (вкладений) клас **Paper** (стаття) з полями:    * Назва статті    * Обсяг статті у кількості сторінок та кількості посилань на релевантні джерела.   та методами:   * + Перевірка відповідності оформлення статті вимогам конференції. (в діалогі запитання про наявність розділів: мета, постановка проблеми, обгрунтування результатів, висновки, рекомендації, список посилань). Якщо вказані розділи присутні у статті, то вивести повідомлення про відповідність статті вимогам, інакше вивести повідомлення про невідповідніть статті вимогам.   + Розрахунок вартості переведення на англійську мову (кожні 1800 знаків коштують 105 грн, + верстка 50 грн за 1 сторінку + подвійний тариф за терміновість). Мінімальна кількість сторінок статті =10.  1. Модифікувати клас **Author**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод Реєстрація на конференцію, в іншу (інший файл) – метод Розрахунок вартості участі в конференції. 2. Додати до проекту новий статичний клас **Subject** (предмет статті), включивши в нього дві функції (на вибір студента) з варіанта 11 лабораторної роботи 2 та дописавши метод, що визначає наукову новизну предмету дослідження. |
|  | Розробити консольний застосунок мовою C# для обробки даних про студента і біржу практик.   1. Створити класи **Student** (Студент) та **PracticeExchange** (Біржа практик). Рекомендується кожний клас створювати в окремому файлі командою Project → Add class 2. Визначити в класах закриті поля:  * клас **Student**: прізвище, ім’я, курс, назва спеціальності, назва університету, рейтинг; * клас **PracticeExchange**: назва біржи, сайт для пошуку компаній, кількість компаній, в яких проводяться практики, список ІТ-компаній та їх спеціалізацій..  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Student**:   + Розрахунок рейтингу студенту як середній бал в сесії за 10 дисциплінами. Оцінки студента генерувати псевдовипадковими цілими числами в діапазоні від 40 до 100. Включити до рейтингового балу бали за професійні здобутки студента (кількість виконаних проектів – 5 балів за кожний проект, кількість наукових статей – 4 бали за кожну статтю, участь в хакатонах та бізнес-кейсах – по 3 бали за кожну подію тощо)   + Визначення IQ здібностей студента через проведення його тестування. Розробляти тестову систему не потрібно, потрібно здійснити імітацію визначення IQ студента із записом даних у текстовий файл * клас **PracticeExchange:**   + Реєстрація студентів на біржі практик із записом даних про студентів у текстовий файл.   + Пошук ІТ-компаній, які погоджуються проводити практику та відповідають вимогам студентів. Застосовувати метод бінарного пошуку в масиві рядків, якими є назви компаній та їх спеціалізації.  1. Додати до класу **PracticeExchange** вбудований (вкладений) клас **ІТCompany** (ІТ-компанія, з якою біржа має договір про співпрацю), з полями:    * Назва компанії, кількість і профіль вакансій    * Кількість студентів, які проходять практику і претендують на вакансії.   та методами:   * + Розрахунок витрат ІТ-компанії або прибутку від проведення практики студентів. Алгоритм розрахунку витрат або прибутку розробити самостійно.   + Формування списку компетентностей, які студент здобуватиме на практиці (перелік знань, умінь, досвід)  1. Модифікувати клас **PracticeExchange**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод Пошук ІТ-компаній, в іншу (інший файл) – метод Реєстрація студентів на біржі. 2. Додати до проекту новий статичний клас **Rezume** (резюме), включивши в нього дві функції:    * Створення резюме та запис його в текстовий файл;   Розрахунок бажаної зарплати студента відповідно до його рейтингу та професійних здобутків. |
|  | Розробити консольний застосунок мовою C# для обробки даних про клієнта та агентство нерухомості з продажу та покупки квартир.   1. Створити класи **RealEstateAgency** (Агентство нерухомості) та **Client** (клієнт)Рекомендується кожний клас створювати в окремому файлі командою Project → Add class 2. Визначити в класах закриті поля:  * клас **Client**: FirstName, LastName, контакти, сума, за яку клієнт хоче купити квартиру, банк клієнта * клас **RealEstateAgency**: назва Агентства, кількість успішних операцій, вартість послуг.  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (*get, set*) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Client**:   + Пошук агентства нерухомості методом лінійного пошуку в списку агенцій, поданих у текстовому файлі.   + Оформлення замовлення (вимоги до кількості кімнат, місця розташування будинку, вартості квартири, первинний-вторинний ринок, тип (елітне, бізнес-клас, економ-клас)). * клас **RealEstateAgency**:   + Пошук квартир відповідно до вимог клієнта з каталогу, поданому у вигляді текстового файлу.   + Оцінка нерухомості: розрахунок вартості квартири з урахуванням місця розташування, поверху, метражу, вартості квадратного метра тощо.  1. Додати до класу **RealEstateAgency** вбудований (вкладений) клас **Rieltor** (ріелтор) з полями:    * Назва банку, адреса    * Ідентифікатори клієнта    * Кількість грошей на рахунку клієнта   та методами:   * + Збільшення, зменшення грошей на рахунку клієнта;   + Переказ грошей, оплата комунальних платежів з картки клієнта.  1. Модифікувати клас **RealEstateAgency**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод пошук квартир, в іншу (інший файл) – методи Оцінка нерухомості. 2. Додати до проекту новий статичний клас **Apartment** (квартира), включивши в нього дві функції з варіанта 13 лабораторної роботи 2, та дописавши метод, який розраховує відсотки ріелтору за супровід угоди на покупку квартири. |
| **14.** | Розробити консольний застосунок мовою C# для обробки даних про академічну мобільність студентів.   1. Створити класи **Student** (Студент) та **AcademicMobility** (відділ академічної мобільності). Рекомендується кожний клас створювати в окремому файлі командою Project → Add class 2. Визначити в класах закриті поля:  * клас **Student**: прізвище, ім’я, курс, назва спеціальності, назва університету, рейтинг успішності, наукові здобутки; * клас **AcademicMobility**: міжнародні програми, університети-партнери, контакти  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Student**:   + Розрахунок рейтингу студенту як середній бал в сесії за 10 дисциплінами. Оцінки студента генерувати псевдовипадковими цілими числами в діапазоні від 40 до 100. Включити до рейтингового балу бали за наукову діяльність студента (кількість статей – 5 балів за кожну статтю, участь в конференціях і конкурсах – по 4 бали за кожну подію тощо) та бали за знання іноземної мови.   + Визначення IQ здібностей студента через проведення його тестування. Розробляти тестову систему не потрібно, потрібно здійснити імітацію визначення IQ студента із записом даних у текстовий файл. * клас **AcademicMobility:**   + Реєстрація студентів на програму міжнародної мобільності із записом даних про студентів у текстовий файл.   + Пошук університету, який відповідає бажанням студента. Застосовувати метод бінарного пошуку в масиві рядків, якими є назви міжнародних програм та університетів.  1. Додати до класу **AcademicMobility** вбудований (вкладений) клас **InterProgram** (міжнародні програми) з полями:    * Назва програми (подвійного диплому, DAAD, Fulbright тощо);    * Кількість студентів, які беруть участь у міжнародних програмах.   та методами:   * + Презентація міжнародних і стипендіальних програм в різних країнах;   + Проведення конкурсу серед претендентів на участь у міжнародних програмах (сортування студентів по рейтингах і вибір найкращих студентів).  1. Модифікувати клас **AcademicMobility**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод Презентація міжнародних і стипендіальних програм, в іншу (інший файл) – метод Проведення конкурсу. 2. Додати до проекту новий статичний клас **ResearchWork** (наукова робота), включивши в нього функції 2, 6, 9 з варіанта 3лабораторної роботи 2 |
| **15.** | Розробити консольний застосунок мовою C# для обробки даних про обслуговування клієнтів службою таксі.   1. Створити класи **Client** (Клієнт) та **TaxiService** (служба такси) Рекомендується кожний клас створювати в окремому файлі командою Project → Add class 2. Визначити в класах закриті поля:  * клас **Client**: LastName, адреса проживання, під’їзд, телефон, дисконтна карта; * клас **TaxiService**: назва служби такси, телефон диспетчера, тип сервісу (бізнес-клас, економ-клас), тариф за км, вартість мінімального замовлення, вартість простою.  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Client**:   + замовлення таксі телефоном або online;   + оплата замовлення або відмова від оплати у випадку незадоволеності клієнта. * клас **TaxiService:**   + розрахунок вартості виконання замовлення (вартість проїзду = відстань від адреси клієнта до пункту призначення\*тариф, якщо вартість проїзду менша за мінімальний замовлення, приймається значення мінімального замовлення);   + розрахунок оптимального маршруту руху з мінімальної кількістю заторів (можна використати готові програми визначення оптимального маршруту).  1. Додати до класу **TaxiService** вбудований (вкладений) клас **Car** (автомобіль) з полями:    * марка, номер;    * тип машини (бізнес-клас, економ-клас тощо).   та методами:   * + розрахунок витрат пального (кількість літрів палива в баку / кількість км шляху × 100 = витрата палива в л / 100 км);   + розрахунок вартості палива на 1 км шляху (вартість за 1км = середнє споживання палива × ціна палива / 100).  1. Модифікувати клас **TaxiService**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод розрахунок вартості виконання замовлення, в іншу (інший файл) – метод розрахунок оптимального маршруту руху. 2. Додати до проекту новий статичний клас **Сity** (місто), включивши в нього дві функції з варіанта 15 лабораторної роботи 2 і дописавши метод розрахунку необхідної кількості автомобілів-таксі для міста на рік (кількість автомобілів= (кількість населення міста\*кількість поїздок одного мешканця в рік\*відсоток мешканців, що користуються таксі) / (365 днів\*кількість пасажирів в день\*100)) |
| **16.** | Розробити консольний застосунок мовою C# для обробки даних про обслуговування клієнтів в банку.   1. Створити класи **Client** (Клієнт) та **Bank** (Банк). Рекомендується кожний клас створювати в окремому файлі командою Project → Add class 2. Визначити в класах закриті поля:  * клас **Client**: FirstName, LastName, паспорт, адреса прописки, ідентифікаційний код, телефон; * клас **Bank**: назва банку, МФО банку, SWIFT код, сайт банку.  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Client**:   + зменшення суми на рахунку (оплата комунальних платежів, оплата покупок);   + поповнення рахунку (зарплата, пенсія, перекази тощо) * клас **Bank:**   + видача кредитів під відсотки з погашення відсотків та поверненням коштів по кредиту.   + прийом вкладів з розрахунком прибутку по депозиту за алгоритмом: сума прибутку = розмір депозиту \* (1+(відсоткова ставка/12)\*термін зберігання вкладу/100);   + відкриття та ведення рахунків (за відкриття та ведення зарплатних та інших рахунків банк отримує кошти від установ та клієнтів)  1. Додати до класу **Bank** вбудований (вкладений) клас **Website** (сайт банку) з полями:    * URL сайта;    * Назва Інтернет-банку (наприклад, «Приват24»)   та методами:   * + контроль залишків по поточних рахунках,   + оформлення депозиту,   + оплата комунальних послуг, переказ коштів на рахунки  1. Модифікувати клас **Bank**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод видачі кредитів під відсотки, в іншу (інший файл) – метод прийому вкладів з розрахунком прибутку по депозиту. 2. Додати до проекту новий статичний клас **NBU** (Національний банк України), включивши в нього два методи:    * регулювання курсу грошової одиниці України по відношенню до іноземних валют шляхом купівлі-продажу іноземної валюти на фінансових ринках;    * емісія грошей в обіг (друкування банкнот і збільшення інфляції). |
| **17.** | Розробити консольний застосунок мовою C# для обробки даних про пацієнтів і лікарів.   1. Створити класи **Patient** (пацієнт) **Doctor** (Лікар). Рекомендується кожний клас створювати в окремому файлі командою Project → Add class. 2. Визначити в класах закриті поля:  * клас **Patient**: FirstName, LastName, паспорт, адреса прописки, телефон; * клас **Doctor**: FirstName, LastName, спеціальність, кваліфікація,  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Patient**:   + запис до лікаря, візит до лікаря, опис симптомів хвороби   + лікування (розрахунок вартості ліків та лікування) * клас **Doctor:**   + обстеження пацієнта та визначення діагнозу хвороби (пошук назви діагнозу хвороби по симптомах здійснити у текстовому файлі, задавши назви симптомів);   + оформлення історії хвороби, направлення на аналізи та госпіталізацію, призначення ліків тощо. Результати записати до текстового файлу.  1. Додати до класу **Patient** вбудований (вкладений) клас **MedicalCard** (сайт банку) з полями:    * FirstName та LastName пацієнта;    * назва діагнозу, перелік призначених аналізів та ліків;   та методами:   * + корегування діагнозу, переліку призначених аналізів та ліків,   + опис процесу лікування, дані обстежень, виписний епікриз.  1. Модифікувати клас **Doctor**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод визначення діагнозу хвороби, в іншу (інший файл) – метод оформлення історії хвороби. 2. Додати до проекту новий статичний клас **Illness** (хвороба), включивши в нього завдання 2 , 6, 9 з варіанта 17 лабораторної роботи 2. |
| **18.** | Розробити консольний застосунок мовою C# для обробки даних про кафедру та ІТ компанію, з якою кафедра взаємодіє.   1. Створити класи **SocialNetwork** (соціальна мережа) та **People** (люди). Рекомендується кожний клас створювати в окремому файлі командою Project → Add class. 2. Визначити в класах закриті поля:  * клас **SocialNetwork**: назва мережі, сайт, спрямованість мережі (для науковців, для розваг, для пошуку роботи тощо) * клас **People**: FirstName, LastName, дата народження, місце роботи  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **SocialNetwork**:   + Розміщення та обмін тестів, зображень, файлів, реклам тощо   + Взаємодія користувачів, пошук колег і співпраця з ними. * клас **People:**   + Створення індивідуальних профілів;   + Обмін інформацією,  1. Додати до класу **SocialNetwork** вбудований (вкладений) клас **Friends** (друзі) з полями:    * FirstName та LastName, e-mail друга;   та методами:   * + Обмін повідомленнями,   + Обмін відео, фото, музичними та іншими файлами.  1. Модифікувати клас **SocialNetwork**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод взаємодії користувачів, в іншу (інший файл) – метод розміщення та обміну інформацією . 2. Додати до проекту новий статичний клас **Internet**, включивши в нього завдання 2 , 6, 9 з варіанта 18 лабораторної роботи 2. |
| **19.** | Розробити консольний застосунок мовою C# для обробки даних про транспорт у місті.   1. Створити класи  **City** ( місто) **Тransport** (транспорт). Рекомендується кожний клас створювати в окремому файлі командою Project → Add class 2. Визначити в класах закриті поля:  * клас **City**: назва міста, площа міста, кількість населення, щільеість населення; * клас **Тransport**: тип транспорту (автобус, тролейбус, легковий тощо), величина пасажиропотоку.  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **City**:   + розрахунок перспективної чисельності населення міста (перспективна чисельність = кількість працюючих на промисловості\*100% / (100 – частка обслуговуючого населення + частка непрацючого населення)). Результати записати у текстовий файл:   + розрахунок площ функціональних зон міста (площа міського центру= чисельність населення міста\*площа на одну людину /100000;   площа жилих районів міста=кількість населення міста/ щільність населення території )   * клас **Тransport**:   + планування міського бюджету, що виділяється на транспорт;   + визначення кількості шляхопровідів та доріг (км), що ремонтуватимуться. Адреси вулиць записати у текстовий файл.  1. Додати до класу **City** вбудований (вкладений) клас **Infrastructure**  (інфраструктура) з полями:    * Довжина транспортної мережі;   та методами:   * + розрахунок кількості маршрутів міського пасажирського транспорту із записом в тектовий файл (кількість маршрутів=щільність транспортної мережі / середню довжину поїздки;   щільність транспортної мережі  довжина транспортної мережі / площа міста;  середня довжина поїздки  )   * + розрахунок кількості машин на маршруті із записом в тектовий файл (кількість машин на маршруті = час оберту машини\*коефіцієнт змінності/інтервал руху;   інтервал руху=60\*маасимальні місткість машини/величина пасажиропотоку)   1. Модифікувати клас **City**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод розрахунок перспективної чисельності населення, в інший файл – метод розрахунок площ функціональних зон міста. 2. Додати до проекту новий статичний клас **Weather** (погода), включивши в нього в якості методів функції 2 та 6 варіанта 19 лабораторної роботи 2, та додавши метод прогнозу аварійності на транспорті через снігопад. |
| **20.** | Розробити консольний застосунок мовою C# для обробки даних про обслуговування читачів в бвбліотец.   1. Створити класи **Reader** (читач) та **Library** (бібліотека). Рекомендується кожний клас створювати в окремому файлі командою Project → Add class. 2. Визначити в класах закриті поля:  * клас **Reader**: FirstName, LastName, паспорт, адреса прописки, телефон; * клас **Library**: назва, адреса, обсяг книжкого фонду  1. Визначити в класах конструктори без параметрів для ініціалізації полів класів нульовими та пустими (для типу string) значеннями. 2. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями. 3. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів. 4. Визначити в класах відкриті методи для введення з консолі, виведення на консоль значень ініціалізованих полів класу та запис значень до текстового файлу. 5. Визначити в класах відкриті методи для роботи з об’єктами:  * клас **Reader**:   + запис в бібліотеку та оформлення читацького абонемента із записом інформації про читачів у текстовий файл;   + вибір книжок відповідно до вимог читача (пошук потрібної книжки здійснювати у текстовому файлі за алгоритмом бінарного пошуку) * клас **Library**:   + додавання книжок до бібліотечного фонду (перелік книжок записаний у текстовий файл; додавання книжок реалізується додаванням записів у текстовий файл).   + видача книжок читачам з відміткою в текстовому файлі про читача, який взяв книжку.  1. Додати до класу **Library** вбудований (вкладений) клас **Book** (книжка) з полями:    * Автор книги;    * Назва книги;    * Рік видання книги.   та методами:   * + Збільшення тривалості життя від читання книг (якщо в тиждень витрачати 4 годин на читання, на 25% змешується ризик захворювання мозку. Порахувати, наскільки збільшиться тривалість життя людини, що систематично читає книжки)   + Підвищення рівня IQ (інтелекту людини) в залежності від систематичного читання  1. Модифікувати клас **Library**, подавши його як частковий, тобто поділивши його на дві частини: в одну частину (окремий файл) включити метод видачі внижок читачам, в інший файл – метод збільшення кількості книжок в бібліотеці. 2. Додати до проекту новий статичний клас **Content** (зміст книги), включивши в нього в якості методів три функції варіанта 20 лабораторної роботи 2. |

### Література

1. О.С.Бичков, Є.В.Іванов Об’єктно-орієнтоване програмування мовою C#/

2.C# 2005 и платформа .Net 3.0 для профессионалов. Нейгел К., Ивьен Б. и др. –М.: ООО “И.Д. Вильямс", 2008. –1376 с.

## Лабораторна робота №4 Успадкування класів, конструктори базових та похідних класів, інтерфейси, абстрактні класи в мові програмування C#

### Рейтинг лабораторної роботи №4

|  |  |  |  |
| --- | --- | --- | --- |
| № п.п | Вид діяльності студента | Рейтинговий бал | Deadline |
| 1 | Написання коду з 6 завдань | 0,5\*9 = 5,5 | 25 Березня |
| 2 | Захист роботи | 0,5 |
| 3 | Звіт з роботи | 0,5 |  |
| Разом за роботу | | 5,5 |  |

### Мета роботи

1. Вивчення спадкування класів
2. Конструктори базових та похідних класів
3. Ієрархія класів
4. Оголошення та реалізація інтерфейсів. Стандартні інтерфейси.
5. Абстрактні класи

### Методичні рекомендації до виконання лабораторної роботи

1. Уважно прочитайте лекцію та методичні вказівки до цієї лабораторної роботи.

2. Виконайте і проаналізуйте всі наведені в лекції приклади.

3. Зверніть увагу, що в С# заборонено множинне спадкування класів, тобто кожний похідний клас може мати тільки одного предка (базовий клас).

4. Якщо в базовому класі є конструктор з параметрами, то для його використання в похідному класі необхідно використовувати конструкцію ...(...):base(...){...}, яка розташовується в оголошенні конструктора похідного класу між заголовком конструктора і тілом.

5. Для поглибленого вивчення цього матеріалу прочитайте розділи 3, 4 книги [2] та розділ 11 книги [3].

### Порядок виконання роботи

1. Створити директорію Lab4, в якій будуть розміщуватися проекти цієї лабораторної роботи.

2. Виконати завдання свого варіанту у вигляді одного консольного проекту

### Приклади виконання завдань

Спадкування — один з трьох фундаментальних принципів об'єктно-орієнтованого програмування, оскільки саме завдяки йому можливо створення ієрархічних класифікацій. Використовуючи спадкування, можна створити загальний клас, який визначає характеристики, властиві безлічі пов'язаних елементів. Цей клас потім може бути успадкований іншими, вузькоспеціалізованими класами з додаванням до кожного з них своїх унікальних особливостей. В мові С# клас, який успадковується, називається базовим. Клас, який успадковує базовий клас, називається похідним. Отже, похідний клас — це спеціалізована версія базового класу. У похідний клас, що успадковує всі змінні, методи, властивості, оператори і індексатори, визначені в базовому класі, можуть бути додані унікальні елементи.

#### Оголошення базових і похідних класів

Нехай є клас **Person**, який описує людину;

class Person

{

private string name;

public string Name

{

get { return name; }

set { name = value; }

}

public void Display()

{

Console.WriteLine(name);

}

}

Але раптом нам став потрібен клас, що описує співробітника підприємства, наприклад, клас **Employee**. Оскільки цей клас буде реалізовувати той самий функціонал, що і клас Person, так як співробітник – це також і людина, то було б раціонально зробити клас **Employee** похідним (або спадкоємцем, або підкласом) від класу **Person**, який, в свою чергу, називається базовим класом або батьком (або суперкласом). Для класу **Employee** базовим є **Person**, і тому клас **Employee** успадковує всі ті самі властивості, методи, поля, які є в класі **Person**. Єдине, що не передається при спадкуванні, це конструктори базового класу.

class Employee : Person

{

public void Display()

{

Console.WriteLine(\_name);

}

}

Таким чином, спадкування реалізує ставлення ***is-a*** (є), об'єкт класу Employee також є об'єктом класу **Person**:

static void Main(string[] args)

{

Person p = new Person { Name = "Nikita" };

p.Display();

p = new Employee { Name = "Kolya" };

p.Display();

Console.Read();

}

За замовчуванням всі класи успадковуються від базового класу **Object**, навіть якщо ми явно не встановлюємо успадкування. Тому вище означені класи **Person** і **Employee** крім своїх власних методів, також матимуть і методи класу O**bject: ToString(), Equals(), GetHashCode() і GetType)**.

Усі класи за замовчуванням можуть успадковуватися. Однак є ряд обмежень:

* не підтримується множинне спадкування, клас може успадковуватися тільки від одного класу;
* при створенні похідного класу треба враховувати тип доступу до базового класу. Тип доступу до похідному класу повинен бути таким самим, як і у базового класу, або більш строгим. Тобто, якщо базовий клас має тип доступу **internal**, то похідний клас може мати тип доступу *internal* або *private*, але **не public;**
* однак слід також враховувати, що якщо базовий і похідний клас знаходяться в різних збірках (проектах), то в цьому випадку похідний клас може успадковувати тільки від класу, який має модифікатор **public;**
* якщо клас оголошений з модифікатором **sealed**, то від цього класу не можна успадковувати і створювати похідні класи.

#### 2. Доступ до членів базового класу з класу-спадкоємця

Похідний клас може мати доступ тільки до тих членів базового класу, які визначені з модифікаторами **private** **protected** (якщо базовий і похідний клас знаходяться в одній збірці), **public**, **internal** (якщо базовий і похідний клас знаходяться в одній збірці), **protected** і **protected** **internal**.

class Employee : Person

{

public void Display()

{

Console.WriteLine(Name);

}

}

#### 3. Конструктори, ключове слово base

Тепер додамо в класи конструктори. Клас **Person** має конструктор, який встановлює властивість **Name**. Оскільки клас **Employee** успадковує і встановлює ту саму властивість **Name**, то логічно було б не писати по сто разів код установки, а якось викликати відповідний код класу **Person**. До того ж властивостей, які треба встановити в конструкторі базового класу, і параметрів може бути набагато більше. За допомогою ключового слова **base** можна звернутися до базового класу. У нашому випадку в конструкторі класу **Employee** нам треба встановити ім'я і компанію. Але ім'я ми передаємо на установку в конструктор базового класу, тобто в конструктор класу **Person**, за допомогою виразу **base** (**name**).

class Person

{

public string Name { get; set; }

public Person(string str) //конструктор базового класу

{

Name = str;

}

public void Display()

{

Console.WriteLine(Name);

}

}

class Employee : Person

{

public string Company { get; set; }

//конструктор похідного класу

public Employee(string str, string company): base(str)

{

Company = company;

}

}

static void Main(string[] args)

{

Person p = new Person("Nikita");

p.Display();

Employee emp = new Employee("Kolya", "Microsoft");

emp.Display();

Console.Read();

}

Конструктори не передаються похідному класу при спадкуванні. І якщо в базовому класі не визначений конструктор за замовчуванням без параметрів, а тільки конструктори з параметрами (як у випадку з базовим класом **Person**), то в похідному класі ми обов'язково повинні викликати один з цих конструкторів через ключове слово **base**.

#### 4.Порядок виклику конструкторів

При виклику конструктора класу спочатку відпрацьовують конструктори базових класів і тільки потім конструктори похідних. Наприклад, візьмемо такі класи:

class Person

{

string name;

int age;

public Person(string name)

{

this.name = name;

Console.WriteLine("Person(string name)");

}

public Person(string name, int age) : this(name)

{

this.age = age;

Console.WriteLine("Person(string name, int age)");

}

}

class Employee : Person

{

string company;

public Employee(string name, int age, string company) : base(name, age)

{

this.company = company;

Console.WriteLine("Employee(string name, int age, string company)");

}

}

При створенні об’єктуа Employee:

Employee worker = new Employee("Kolya", 22, "Microsoft");

отримаємо наступний консольний вивід:

Person(string name)

Person(string name, int age)

Employee(string name, int age, string company)

У підсумку ми отримуємо наступний ланцюг виконань.

Спочатку викликається конструктор **Employee (string name, int age, string company)**. Він делегує виконання конструктору **Person (string name, int age)**

Викликається конструктор **Person (string name, int age)**, який сам поки не виконується і передає виконання конструктору **Person (string name)**

Викликається конструктор **Person (string name)**, який передає виконання конструктору класу **System.Object**, так як це базовий за замовчуванням клас для **Person**.

Виконується конструктор **System.Object.Object()**, потім виконання повертається конструктору **Person (string name)**

Виконується тіло конструктора **Person (string name)**, потім виконання повертається конструктору **Person (string name, int age)**

Виконується тіло конструктора **Person(string name, int age)**, потім виконання повертається конструктору **Employee(string name, int age, string company)**

Виконується тіло конструктора **Employee (string name, int age, string company)**. В результаті створюється об'єкт **Employee**

#### Інтерфейси

Інтерфейс являє посилальний тип, який може оголошувати деякий функціонал, тобто набір методів і властивостей без їх реалізації. Потім цей функціонал реалізують класи і структури, які застосовують дані інтерфейси.

##### Оголошення інтерфейсу

Для оголошення інтерфейсу використовується ключове слово **interface**. Як правило, назви інтерфейсів в C# починаються з великої літери **I**, наприклад, **IComparable**, **IEnumerable** (так звана угорська нотація), однак це не обов'язкова вимога, а більше стиль програмування.

Що може визначати інтерфейс? В цілому інтерфейси можуть визначати такі сутності:

* методи;
* властивості;
* індексатори;
* події;
* статичні поля і константи (починаючи з версії C# 8.0).

Однак інтерфейси **не можуть** визначати нестатичні змінні. Наприклад, найпростіший інтерфейс, який визначає всі ці компоненти:

interface IMovable

{

const int minSpeed = 0; // константа мінімальна швидкість

// статична змінні

static int maxSpeed = 60; // максимальна швидкість

void Move(); // метод руху

string Name { get; set; } // властивість назва

delegate void MoveHandler(string message); //делегат для події

event MoveHandler MoveEvent; // подія руху

}

В даному випадку визначено інтерфейс **IMovable**, який представляє певний об'єкт, що рухається. Методи і властивості інтерфейсу можуть не мати реалізації, в цьому вони зближуються з абстрактними методами і властивостями абстрактних класів. В даному випадку інтерфейс оголошує метод **Move**, який буде представляти деяке пересування. Він не має реалізації. Те саме стосується властивості **Name**. Якщо його члени – методи і властивості не мають модифікаторів доступу, то за замовчуванням цей доступ є **public**, так як мета інтерфейсу - оголошення функціоналу для реалізації його класом. Це стосується також і констант і статичних змінних, які в класах і структурах за замовчуванням мають модифікатор **private**. В інтерфейсах вони мають за замовчуванням модифікатор **public**.

interface IMovable

{

public const int minSpeed = 0; // мінімальна швидкість

private static int maxSpeed = 60; // максимальна швидкість

public void Move(); // метод руху

protected internal string Name { get; set; } // властивість назва

public delegate void MoveHandler(string message); // делегат для події

public event MoveHandler MoveEvent; // подія руху

}

##### Застосування інтерфейсів

Не можна створювати об'єкти інтерфейсу безпосередньо за допомогою конструктора, як наприклад, в класах: Інтерфейс призначений для реалізації в класах і структурах. Наприклад, маючи інтерфейс **IMovable**, який-небудь клас або структура можуть застосувати даний інтерфейс:

using System;

namespace HelloApp

{

interface IMovable

{

void Move();

}

class Person : IMovable

{

public void Move()

{

Console.WriteLine("Людина рухається");

}

}

struct Car : IMovable

{

public void Move()

{

Console.WriteLine("Машина їде");

}

}

class Program

{

static void Action(IMovable movable)

{

movable.Move();

}

static void Main(string[] args)

{

Person person = new Person();

Car car = new Car();

Action(person);

Action(car);

Console.Read();

}

}

}

У даній програмі визначено метод **Action**(), який в якості параметра приймає об'єкт інтерфейсу **IMovable**. На момент написання коду ми можемо не знати, що це буде за об'єкт - якийсь клас або структура. Єдине, в чому ми можемо бути впевнені, що цей об'єкт обов'язково реалізує метод **Move**() і ми можемо викликати цей метод. Іншими словами, інтерфейс – це контракт, що якийсь певний тип обов'язково реалізує деякий функціонал.

Інтерфейси мають ще одну важливу функцію: C# не підтримує множинне успадкування, тобто можна успадкувати клас тільки від одного класу. Інтерфейси дозволяють частково обійти це обмеження, оскільки в C# клас може реалізувати відразу кілька інтерфейсів. Усі реалізовані інтерфейси вказуються через кому:

using System;

namespace HelloApp

{

interface IAccount

{

int CurrentSum { get; } // Поточна сума на рахунку

void Put(int suma); // Покласти гроші на рахунок

void Withdraw(int suma); // зняти гроші з рахунку

}

interface IClient

{

string Name { get; set; }

}

class Client : IAccount, IClient //клас реалізує два інтерфейси

{

int sum; // сума грошей

public string Name { get; set; }

public Client(string name, int suma)

{

Name = name;

sum = sum;

}

public int CurrentSum { get { return sum; } }

public void Put(int sum) { sum += suma; }

public void Withdraw(int sum)

{

if (sum >= suma)

{

sum -= suma;

}

}

}

class Program

{

static void Main(string[] args)

{

Client client = new Client("Igor", 200);

client.Put(30);

Console.WriteLine(client.CurrentSum); //230

client.Withdraw(100);

Console.WriteLine(client.CurrentSum); //130

Console.Read();

}

}

}

#### Стандартні інтерфейси NET

|  |  |  |
| --- | --- | --- |
| **Інтерфейс** | **Методи** | **Опис** |
| IComparable | CompareTo() | Задає метод порівняння об'єктів за принципом більше або менше. Метод повертає результат порівняння двох об'єктів - поточного і переданого йому в якості параметра. |
| IComparer | Compare() | Метод повертає результат порівняння двох об'єктів, переданих йому в якості параметрів. |
| ICloneable | MemberwiseClone()  Clone() | Клонування – створення копії об'єкта.. Якщо необхідно скопіювати в іншу область пам'яті поля об'єкта, можна скористатися методом MemberwiseClone (), який будь-який об'єкт успадковує від класу Оbject. Це називається поверхневим клонуванням. Для створення повністю незалежних об'єктів необхідно глибоке копіювання, коли в пам'яті створюється дублікат всього дерева об'єктів, тобто об'єктів, на які посилаються поля об'єкта, поля полів тощо. Об'єкт, що має власні алгоритми клонування, повинен оголошуватися як похідний інтерфейсу ICloneable і перевизначати його єдиний метод Clone (). |
| IEnumerable | GetEnumerator() | Задає послідовність перебору елементів об'єкта, повертає об'єкт типу IEnumerator |
| IEnumerator | GetEnumerator(), Current, MoveNext(), Reset(). | Властивість Current повертає поточний елемент об'єкта, метод MoveNext () просуває нумератор на наступний елемент об'єкта, метод Reset () встановлює нумератор в початок перегляду. |

#### Абстрактні класи

Абстрактний клас схожий на звичайний клас. Він також може мати змінні, методи, конструктори, властивості. Єдине, що при визначенні абстрактних класів використовується ключове слово **abstract**. Головна відмінність полягає в тому, що не можна використовувати конструктор абстрактного класу для створення його об'єкта.

Навіщо потрібні абстрактні класи? Припустимо, в нашій програмі для банківського сектора ми можемо визначити дві основні сутності: клієнт банку і співробітник банку. Кожна з цих сутностей має свою специфіку, наприклад, для співробітника треба визначити його посаду, а для клієнта – суму на рахунку. Відповідно клієнт і співробітник становитимуть окремі класи **Client** та **Employee**. У той самий час обидві ці сутності можуть мати щось спільне, наприклад, ім'я та прізвище, якусь іншу загальну функціональність. І цю загальну функціональність краще винести в якийсь окремий клас, наприклад, **Person**, який описує людини. Тобто класи **Employee** (співробітник) і **Client** (клієнт банку) будуть похідними від класу **Person**. І так як всі об'єкти в нашій системі представлятимуть або співробітника банку, або клієнта, то безпосередньо ми від класу **Person** створювати об'єкти не будемо. Тому має сенс зробити його абстрактним:

**abstract** class Person

{

public string Name { get; set; }

public Person(string name)

{

Name = name;

}

public void Display()

{

Console.WriteLine(Name);

}

}

class Client : Person

{

public int Sum { get; set; } // сума на рахунку

public Client(string name, int sum) : base(name)

{

Sum = sum;

}

}

class Employee : Person

{

public string Position { get; set; } // посада

public Employee(string name, string position): base(name)

{

Position = position;

}

}

Потім ми зможемо використовувати ці класи:

Client client = new Client("Oleg", 500);

Employee employee = new Employee("Boris", "Apple");

client.Display();

employee.Display();

### Варіанти завдань для лабораторної роботи № 4

Завдання передбачають такі дії :

Завдання 1 передбачає розробку базових і похідних класів з закритими полями та відкритими властивостями.

Завдання 2 передбачає додавання до базового класу конструкторів з параметрами та без параметрів.

Завдання 3 передбачає додавання до похідних класів конструкторів з параметрами.

Завдання 4 – розробка властивостей *get, set* для доступу до закритих полів класу;

Завдання 5 – розробка методів виведення на консоль значень полів класу

Завдання 6 – розробка методів за сценарієм роботи програми;

Завдання 7 – друга версія консольного застосунку, що реалізує ієрархію класів у вигляді інтерфейсу та похідних від інтерфейсу класів, що реалізують інтерфейс

Завдання 8 – третя версія консольного застосунку, що реалізує ієрархію класів у вигляді абстрактного базового та похідних класів

Завдання 9 – четверта версія консольного застосунку, що реалізує стандартні інтерфейси

В методі *Main*() класу *Program* продемонструвати виклик усіх методів усіх класів. Усі значення, що розраховуються, записувати до *текстових файлів*.

**Студент має право додати додаткові поля та методи в класи завдань свого варіанту**.

**Номер варіанта визначається за порядковим номером в журналі.**

|  |  |
| --- | --- |
| **№ варі-анту** | **Зміст завдання** |
|  | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Тварина*, та похідні класи: *Кішка, Собака*. Задати закриті загальні поля у базовому класі (порода, кличка тварини, характеристики породи) і закриті специфічні поля в похідних класах, реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами )для ініціалізації полів класу Тварина. 3. Конструктори похідних класів (з параметрами) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів на консоль. 6. Методи, що демонструють *протистояння* кішки та собаки або їх *дружбу*, *дресирування* кішки та собаки тощо. Алгоритми методів та їх кількість студент придумує самостійно. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді інтерфейсу, від якого успадковується похідний клас *Тварина*, від якого успадковуються класи *Кішка* та *Собака*. Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді абстрактного базового класу *Тварина*, від якого успадковуються класи *Кішка* та *Собака*. Протестувати відмінності застосування інтерфейсів та абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу Тварина (або масив об’єктів похідних класів). В класі реалізувати:  * інтерфейс *IComparable* для порівняння тварин за вагою в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння тварин не тільки за вагою, але і за зростом; * інтерфейс *IEnumerable* для виведення на консоль списку тварин, впорядкованих за вагою. |
|  | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Людина* та похідні класи: *Робітник, Інженер*. Задати закриті загальні поля у базовому класі (ім’я, посада, вік, стаж роботи, зарплата тощо) і закриті специфічні поля в похідних класах (трудові досягнення, професійні навички, особисті якості), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами)для ініціалізації полів класу *Людина*. 3. Конструктори похідних класів (з параметрами)для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів на консоль. 6. Методи, що демонструють творчу проектну роботу інженера та технічну наладку обладнання робітником. Алгоритми методів та їх кількість студент придумує самостійно. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді інтерфейсу, від якого успадковується похідний клас *Людина*, від якого успадковуються класи *Робітник, Інженер*. Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді абстрактного базового класу та похідного класу *Людина*, від якого успадковуються класи *Робітник, Інженер*. Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Робітник*. В класі реалізувати:  * інтерфейс *IComparable* для порівняння робітників за віком в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння робітників не тільки за віком, але і за зарплатою; * інтерфейс *IEnumerable* для виведення на консоль списку робітників, впорядкованих за зарплатою. |
|  | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Державний\_службовець* та похідні класи *Державний\_секретар, Керівник\_апарату\_суду.* Задати закриті загальні поля у базовому класі (ім’я, вік, посада, заплата) і закриті специфічні поля в похідних класах (особисті якості, стаж роботи, місце роботи тощо), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами)для ініціалізації полів класу *Державний\_службовець*. 3. Конструктори похідних класів (з параметрами)для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів на консоль. 6. Методи, що демонструють підготовку *плану* роботи Міністерства та звіту про його виконання *Державним\_секретарем* та розроблення *кошторису* видатків на утримання суду *Керівником\_апарату\_суду.* Алгоритми методів та їх кількість студент придумує самостійно. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді інтерфейсу, від якого успадковується похідний клас *Державний\_службовець*, від якого успадковуються класи *Державний\_секретар, Керівник\_апарату\_суду.* Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді абстрактного базового класу *Державний\_службовець*, від якого успадковуються класи *Державний\_секретар, Керівник\_апарату\_суду.* Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Державний\_службовець*, В класі реалізувати:  * інтерфейс *IComparable* для порівняння службовців за віком в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння службовців не тільки за віком, але і за стажем роботи; * інтерфейс *IEnumerable* для виведення на консоль списку службовців, впорядкованих за стажем роботи. |
|  | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Виріб* та похідні класи *Вузол, Деталь.* Задати закриті загальні поля у базовому класі (найменування, габаритні розміри (ширина, висота, довжина), призначення) і закриті специфічні поля в похідних класах (вага, ціна, матеріал, з якого зроблений, сертифікат якості тощо), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Виріб*. 3. Конструктори похідних класів (з параметрами) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують *вагу* вузла як суму ваг усіх деталей, вагу виробу як суму ваг усіх вузлів, *вартість* виробу як суму вартості усіх деталей. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді інтерфейсу, від якого успадковується похідний клас *Виріб*, від якого успадковуються класи *Вузол, Деталь.* Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді абстрактного базового класу *Виріб*, від якого успадковуються класи *Вузол, Деталь.* Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Виріб.* В класі реалізувати:  * інтерфейс *IComparable* для порівняння виробів за вагою в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння виробів за ціною і за якістю (створити свою шкалу якості); * інтерфейс *IEnumerable* для виведення на консоль списку виробів, впорядкований за ціною. |
| **5.** | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Підприємство* та похідні класи *Страхова\_Компанія*, *Нафтогазова\_Компанія, Завод*. Задати закриті загальні поля у базовому класі (назва, місцезнаходження, сфера діяльності: торгівля, виробництво, переробна галузь, рейтинг успішності підприємства тощо) і закриті специфічні поля в похідних класах (вид продукції, обсяг виробництва, прибуток, кількість співробітників), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Підприємство*. 3. Конструктори похідних класів (з параметрами)для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують *прибуток* підприємства як відсоток (задати з клавіатури) від вартості продукції, та кількість працівників (кількість працівників = фонд зарплати/середня зарплата одного працівника). Фонд зарплати згенерувати в заданому діапазоні. Середню зарплату задати з консолі. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді інтерфейсу, від якого успадковується похідний клас *Підприємство*, від якого успадковуються класи *Страхова\_Компанія, Нафтогазова\_Компанія, Завод.* Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді абстрактного базового класу *Підприємство*, від якого успадковуються класи *Страхова\_Компанія, Нафтогазова\_Компанія, Завод.* Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Підприємство.* В класі реалізувати:  * інтерфейс IComparable для порівняння підприємств за кількістю співробітників в методі CompareTo(); * інтерфейс IComparer для порівняння підприємств за кількістю співробітників і за рейтингом успішності; * інтерфейс IEnumerable для виведення на консоль списку підприємств, впорядкований за їх рейтингом. |
| **6.** | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Друкарське\_Видання* та похідні класи *Журнал*, *Книга*. Задати закриті загальні поля у базовому класі (назва, вартість, мова, цільове призначення: наукове, навчальне, суспільно-політичне, художнє тощо) і закриті специфічні поля в похідних класах (автор, кількість сторінок, номер журналу, рік випуску, наклад в кількості примірниках, рейтинг популярності), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Друкарське\_Видання*. 3. Конструктори похідних класів (з параметрами) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують *рейтинг* популярності видання як результат сумарних продажів, *вартість* видання як сума вартостей: паперу, поліграфічних робіт, податки тощо. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді інтерфейсу, від якого успадковується похідний клас *Друкарське\_Видання*, від якого успадковуються класи *Журнал, Книга.* Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді абстрактного базового класу *Друкарське\_Видання*, від якого успадковуються класи *Журнал, Книга.* Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Журнал.* В класі реалізувати:  * інтерфейс *IComparable* для порівняння журналів за ціною в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння журналів за кількістю сторінок і за рейтингом продажів (створити свою шкалу 10 балів); * інтерфейс *IEnumerable* для виведення на консоль списку журналів, впорядкований за їх рейтингом продажів. |
| **7** | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Рослина* та похідні класи *Дерево, Квіти.* Задати закриті загальні поля у базовому класі (назва виду рослини, назва родини, назва сорту, тощо) і закриті специфічні поля в похідних класах (висота, тривалість життя, використання, лікувальні властивості, вартість тощо), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Рослина*. 3. Конструктори похідних класів (з параметрами) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують *вартість* букетів квітів у святкові (вартість підвищена) та звичайні дні, *вартість* дерев в залежності від їх віку, породи тощо, та кількість квіток в букеті та кількість дерев. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді інтерфейсу, від якого успадковується похідний клас *Рослина*, від якого успадковуються класи *Дерево, Квіти.* Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді абстрактного базового класу *Рослина*, від якого успадковуються класи *Дерево, Квіти* Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Дерево.* В класі реалізувати:  * інтерфейс *IComparable* для порівняння порід дерев за ціною в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння дерев за висотою і за ціною; * інтерфейс *IEnumerable* для виведення на консоль списку порід дерев, впорядкований за ціною. |
| **8** | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Місцевість* та похідні класи *Село, Місто.* Задати закриті загальні поля у базовому класі (розмір території, кількість населення, місце розташування тощо) і закриті специфічні поля в похідних класах (назва, бюджет, географічні особливості тощо), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Місцевість*. 3. Конструктори похідних класів (з параметрами) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують економічний потенціал *Села, Міста* у вигляді прироста доходів від промисловості, трудового потенціалу від населення, інвестиції тощо. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді інтерфейсу, від якого успадковується похідний клас *Місцевість*, від якого успадковуються класи *Село, Місто.* Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді абстрактного базового класу *Місцевість*, від якого успадковуються класи *Село, Місто.* Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Місто.* В класі реалізувати:  * інтерфейс *IComparable* для порівняння міст за розміром території в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння міст за розміром території і за кількістю населення; * інтерфейс *IEnumerable* для виведення на консоль списку міст, впорядкований за кількістю населення. |
| **9** | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Товар* та похідні класи *Іграшка, Продукти.* Задати закриті загальні поля у базовому класі (торгова марка, ціна, розмір, вага, якість тощо) і закриті специфічні поля в похідних класах (вікова група для іграшок, матеріал, строк користування, види продуктів, корисність продуктів, режими та терміни зберігання тощо), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Товар*. 3. Конструктори похідних класів (з параметрами) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують кількість іграшок і продуктів, які вдається продати за певний час та за певною ціною. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді **інтерфейсу**, від якого успадковується похідний клас *Товар*, від якого успадковуються класи *Іграшка, Продукти.* Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді **абстрактного** базового класу *Товар*, від якого успадковуються класи *Іграшка, Продукти.* Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Товар.* В класі реалізувати:  * інтерфейс *IComparable* для порівняння товарів за ціною в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння товарів за ціною і за розмірами; * інтерфейс *IEnumerable* для виведення на консоль списку товарів, впорядкований за ціною. |
| **10** | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Документ* та похідні класи *Квитанція, Накладна.* Задати закриті загальні поля у базовому класі (назва, тип (акт, наказ, заява, звіт тощо), призначення (плановий, довідковий, обліковий, фінансовий), обсяг документа в кількості сторінок тощо) і закриті специфічні поля в похідних класах (вид квитанцій (касовий чек, багажна, розрахункова тощо), вид накладних (товарно-транспортна, витратно-прибуткова накладна, податкова тощо), дата, ), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Документ*. 3. Конструктори похідних класів (з параметрами) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують загальну суму витрат та кількість товару за витратно-прибутковою накладною, кількість багажу, його вагу та вартість за квитанцією. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді **інтерфейсу**, від якого успадковується похідний клас *Документ*, від якого успадковуються класи *Квитанція, Накладна.* Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді **абстрактного** базового класу *Документ*, від якого успадковуються класи *Квитанція, Накладна.* Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Документ.* В класі реалізувати:  * інтерфейс *IComparable* для порівняння документів за кількістю сторінок в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння документів за кількістю сторінок і за таємністю (наприклад, 1 – цілком таємно, 2 – таємно, 3 – для службового використання, 4 – для загального доступу); * інтерфейс *IEnumerable* для виведення на консоль списку документів, впорядкований за кількістю сторінок. |
| **11** | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Транспорт* та похідні класи Ав*томобіль, Потяг.* Задати закриті загальні поля у базовому класі (вид транспорту: вантажний, пасажирський, кількість перевезених пасажирів або вантажів, пасажирообіг=кількість пасажирів×відстань перевезень) і закриті специфічні поля в похідних класах (для автомобіля: тип, марка, ціна, потужність, кілометраж, для потягу: тип потягу (експрес, швидкий, пасажирський), вартість квитків, вантажообіг, пасажирообіг), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Транспорт*. 3. Конструктори похідних класів (з параметрами) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують *вартість* квитка на потяг в залежності від типу потягу, типу вагону (купейний, люкс тощо) та відстані проїзду, *вирати* пального для автомобіля (лінійна норма витрат пального на 100 км×пробіг автомобіля в км×поправочний коефіцієнт: надбавка або знижка×0.01), вартість утримання автомобіля в місяць. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді **інтерфейсу**, від якого успадковується похідний клас *Транспорт*, від якого успадковуються класи Ав*томобіль, Потяг.* Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді **абстрактного** базового класу *Транспорт*, від якого успадковуються класи Ав*томобіль, Потяг.* Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу Ав*томобіль.* В класі реалізувати:  * інтерфейс *IComparable* для порівняння автомобілів за ціною в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння автомобілів за ціною і за потужністю;   інтерфейс *IEnumerable* для виведення на консоль списку автомобілів, впорядкованого за ціною і потужністю. |
| **12** | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Викладач* та похідні класи *Професор, Доцент.* Задати закриті загальні поля у базовому класі (ім’я, вчене звання (професор, доцент), заплата, стаж роботи) і закриті специфічні поля в похідних класах (для професора: кількість аспірантів, наукових проектів, яким він керує; для доцента: обсяг навантаження в годинах, кількість дипломників, яким доцент керує), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Викладач*. 3. Конструктори похідних класів (з параметрами) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують посадовий *оклад* в залежності від стажу, *надбавку* в залежності від вартості наукового проекту та кількості дипломників. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді **інтерфейсу**, від якого успадковується похідний клас *Викладач*, від якого успадковуються класи *Професор, Доцент.* Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді **абстрактного** базового класу *Викладач*, від якого успадковуються класи *Професор, Доцент.* Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Викладач.* В класі реалізувати:  * інтерфейс *IComparable* для порівняння викладачів за зарплатою в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння кладачів за зарплатою і за стажем роботи; * інтерфейс *IEnumerable* для виведення на консоль списку икладачів, впорядкований за зарплатою і за стажем роботи. |
| **13** | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Рослина* та похідні класи *Овочі, Фрукти*,*.* Задати закриті загальні поля у базовому класі (назва, розмір, життєвий цикл) і закриті специфічні поля в похідних класах (споживчі властивості, енергетична цінність, безпека плодів, сорт, вартість, кількість, виробник), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Рослина*. 3. Конструктори похідних класів (з параметрами) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують *шкідливість* плодів для людини через екологічне забруднення та внаслідок отруєння їх нітратами; *енергетичну цінність* плодів, вартість плодів в залежності від сорту, пори року, кількості, виробника. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді **інтерфейсу**, від якого успадковується похідний клас *Рослина*, від якого успадковуються класи *Овочі, Фрукти.* Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді **абстрактного** базового класу *Рослина*, від якого успадковуються класи *Овочі, Фрукти.* Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Овочі.* В класі реалізувати:  * інтерфейс *IComparable* для порівняння овочів за ціною в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння овочів за ціною і за енергетичною цінністю; * інтерфейс *IEnumerable* для виведення на консоль списку овочів, впорядкований за ціною і за енергетичною цінністю. |
| **14** | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Книга* та похідні класи *Підручник, Конспект.* Задати закриті загальні поля у базовому класі (назва, вартість) і закриті специфічні поля в похідних класах (автор книги чи ім’я студента для конспекту, кількість сторінок, рейтинг популярності, мова, вільний доступ), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Книга*. 3. Конструктори похідних класів (з параметрами) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують *рейтинг* популярності книги (статус бестселера) в залежності від обсягу продажів за тиждень, місяць; *складність* розрахункових завдань і тестів в залежності від кількості студентів, що їх розв’язують, *ефективність* ведення конспекту студентом в залежності від кількості інформації, що запам’ятовується та усвідомлюється під час конспектування. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді **інтерфейсу**, від якого успадковується похідний клас *Книга*, від якого успадковуються класи *Підручник, Конспект.* Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді **абстрактного** базового класу *Книга*, від якого успадковуються класи *Підручник, Конспект.* Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Книга.* В класі реалізувати:  * інтерфейс *IComparable* для порівняння книг за ціною в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння книг за ціною і за кількістю сторінок; * інтерфейс *IEnumerable* для виведення на консоль списку книг, впорядкований за ціною і за кількістю сторінок. |
| **15** | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Меблі* та похідні класи *Корпусні меблі, М'які меблі, Кухні.* Задати закриті загальні поля у базовому класі (вартість, споживчі властивості і показники якості меблів) і закриті специфічні поля в похідних класах (матеріал, асортимент, габаритні розміри), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Меблі*. 3. Конструктори похідних класів (з параметрами) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують *вартість* меблів в залежності від вартості матеріалів, комплектуючих виробів і фурнітури, показників *якості* меблів, розмірів; здійснити оцінку якості меблів в залежності від наявності в матеріалах шкідливих речовин (формальдегід, фенол тощо), суб’єктивного враження (подобається, не подобається), надійності кріплення тощо. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді **інтерфейсу**, від якого успадковується похідний клас *Меблі*, від якого успадковуються класи *Корпусні меблі, М'які меблі, Кухні.* Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді **абстрактного** базового класу *Меблі*, від якого успадковуються класи *Корпусні меблі, М'які меблі, Кухні.* Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Меблі.* В класі реалізувати:  * інтерфейс *IComparable* для порівняння меблів за ціною в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння меблів за ціною і за габаритами; * інтерфейс *IEnumerable* для виведення на консоль списку меблів, впорядкований за ціною. |
| **16** | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Комп’ютерна\_техніка* та похідні класи *Комп’ютер, Монітор, Планшет*  Задати закриті загальні поля у базовому класі (назва моделі, вартість, виробник, гарантія) і закриті специфічні поля в похідних класах (характеристики: обсяг RAM, обсяг вінчестера тощо; розмір екрана, рейтинг популярності тощо), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Комп’ютерна\_техніка*. 3. Конструктори похідних класів (з параметрами) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують *вартість* комп’ютерної техніки в залежності від бренда, габаритів, терміну випуску, комплектації тощо; методи, що визначають *рейтинг* популярності за кількістю позитивних відгуків. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді **інтерфейсу**, від якого успадковується похідний клас *Комп’ютерна\_техніка*, від якого успадковуються класи *Комп’ютер, Монітор, Планшет.* Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді **абстрактного** базового класу *Комп’ютерна\_техніка*, від якого успадковуються класи *Комп’ютер, Монітор, Планшет.* Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Комп’ютер.* В класі реалізувати:  * інтерфейс *IComparable* для порівняння комп’ютерів за ціною в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння меблів за комп’ютерів і за габаритами; * інтерфейс *IEnumerable* для виведення на консоль списку комп’ютерів, впорядкований за ціною. |
| **17** | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Геометрична\_фігура* та похідні класи *Коло, Трикутник, Многокутник*  Задати закриті загальні поля у базовому класі (назва фігури, вид фігури, координати фігури, ) і закриті специфічні поля в похідних класах (для кола: координати центра та радіус; для трикутника, координати його вершин; для правильного многокутника: кількість вершин, довжина одної сторони), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Геометрична\_фігура.* 3. Конструктори похідних класів (з параметрами) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують площу фігури, радіус вписаного та описаного кіл, суму кутів фігури. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді **інтерфейсу**, від якого успадковується похідний клас *Геометрична\_фігура*, від якого успадковуються класи *Коло, Трикутник, Многокутник.* Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді **абстрактного** базового класу *Геометрична\_фігура*, від якого успадковуються класи *Коло, Трикутник, Многокутник.* Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Геометрична\_фігура.* В класі реалізувати:  * інтерфейс *IComparable* для порівняння геометричних фігур за їх площами в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння геометричних фігур за площами та кількістю кутів; * інтерфейс *IEnumerable* для виведення на консоль списку геометричних фігур, впорядкований за площами. |
| **18** | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Колекції* та похідні класи *Вектор, Матриця,* Задати закриті загальні поля у базовому класі (назва колекції, кількість елементів колекції) і закриті специфічні поля в похідних класах (для вектору: значення елементів одновимірного масиву, для матриці – кількість вимірів, кількість рядків/стовпчиків, значення елементів матриці), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Колекції .* 3. Конструктори похідних класів (з параметраи) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують скалярний добуток елементів вектору, транспонування матриці, максимальний елемент вектору, максимальний елемент матриці 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді **інтерфейсу**, від якого успадковується похідний клас *Колекції* від якого успадковуються класи *Вектор, Матриця.* Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді **абстрактного** базового класу *Колекції*, від якого успадковуються класи *Вектор, Матриця* Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Вектор.* В класі реалізувати:  * інтерфейс *IComparable* для порівняння одновимірних масивів за скалярним добутком їх елементів в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння одновимірних масивів за скалярним добутком їх елементів та кількістю елементів * інтерфейс *IEnumerable* для виведення на консоль список одновимірних масивів, впорядкований за кількістю елементів. |
| **19** | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Навчальний\_процес* та похідні класи *Лабораторні\_роботи, Модульний\_контроль, Екзамен*. Задати закриті загальні поля у базовому класі (ім’я студента, курс, група) і закриті специфічні поля в похідних класах (назва дисципліни, кількість лабораторних робіт, кількість балів за кожну лабораторну роботу, загальна сума балів за лабораторні роботи; бали за модульний контроль, кількість завдань на іспиті, кількість балів за кожне завдання, загальна оцінка за екзамен), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Навчальний\_процес.* 3. Конструктори похідних класів (з параметраи) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують *загальну суму балів* за лабораторні роботи за певною дисципліною, загальний рейтинг студента перед виходом на екзамен, загальний *рейтинг* з урахуванням екзамену за певною дисципліною. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді **інтерфейсу**, від якого успадковується похідний клас *Навчальний\_процес* від якого успадковуються класи *Лабораторні\_роботи, Модульний\_контроль, Екзамен*.Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді **абстрактного** базового класу *Навчальний\_процес*, від якого успадковуються класи *Лабораторні\_роботи, Модульний\_контроль, Екзамен*. Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Лабораторні\_роботи..* В класі реалізувати:  * інтерфейс *IComparable* для порівняння лабораторних робіт за їх балами в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння лабораторних робіт за їх балами та кількістю завдань в кожній * інтерфейс *IEnumerable* для виведення на консоль список лабораторних робіт, упорядкований за їх балами та кількістю завдань в кожній |
| **20** | 1. Створити консольний застосунок мовою C# для роботи з ієрархією класів, що включає базовий клас *Стартап\_проєкт* та похідні класи *Команда\_проєкту, Наукові\_дослідження*. Задати закриті загальні поля у базовому класі (назва проєкту, вартість проєкту) і закриті специфічні поля в похідних класах (кількість членів команди, рейтинг кожного члена команди, тема дослідження, тривалість дослідження, вартість і складність наукових досліджень, кількість статей, написаних за матеріалами досліджень), реалізувати такі відкриті методи класів: 2. Конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу *Стартап-проєкт.* 3. Конструктори похідних класів (з параметраи) для ініціалізації полів похідних класів. 4. Методи –аксесори (властивості) для доступу до закритих полів класів. 5. Методи виведення значень полів класів на консоль. 6. Методи, що розраховують вартість проекту в залежності від тривалості проекту, кількості людей в команді і середньої зарплати кожного члена команди, та рейтинг команди в залежності від кількості статей, складності та вартості наукових досліджень. 7. Розробити другу версію консольного застосунку, що реалізує ієрархію класів у вигляді **інтерфейсу**, від якого успадковується похідний клас *Стартап-проєкт* від якого успадковуються класи *Команда\_проєкту, Наукові\_дослідження*.Продемонструвати доступ до методів, що реалізовані в похідних класах через посилання на інтерфейс. 8. Розробити третю версію консольного застосунку, що реалізує ієрархію класів у вигляді **абстрактного** базового класу *Стартап-проєкт*, від якого успадковуються класи *Команда\_проєкту, Наукові\_дослідження*. Протестувати відмінності застосування інтерфейсів і абстрактних класів. 9. Розробити нову версію консольного застосунку, в якій створити масив об'єктів класу *Стартап-проєкт*,*.* В класі реалізувати:  * інтерфейс *IComparable* для порівняння проєктів за кількістю членів команд в методі *CompareTo*(); * інтерфейс *IComparer* для порівняння проєктів за кількістю членів команд та вартістю проєктів * інтерфейс *IEnumerable* для виведення на консоль список проектів, упорядкований за кількістю членів команд |

### Література

1. О.С.Бичков, Є.В.Іванов Об’єктно-орієнтоване програмування мовою C#/
2. Шилдт Г. С# 4.0: полное руководство. : Пер. с англ. — М. : ООО "И.Д. Вильямс", 2011. — 1056 с.
3. C# 2005 и платформа .Net 3.0 для профессионалов. Нейгел К., Ивьен Б. и др. –М.: ООО “И.Д. Вильямс", 2008. –1376 с.

## Лабораторна робота №5 Поліморфізм методів і операцій, віртуальні функції, перевантаження операторів, індексатори в мові програмування C#

### Рейтинг лабораторної роботи №5

|  |  |  |  |
| --- | --- | --- | --- |
| № п.п | Вид діяльності студента | Рейтинговий бал | Deadline |
| 1 | Написання коду з 5 завдань | 0,5\*7=3,5 | 10 квітня |
| 2 | Захист роботи | 0,5 |
| 3 | Звіт з роботи | 0,5 |  |
| Разом за роботу | | 4,5 |  |

### Мета роботи:

1. Використання перевантажених методів і операцій (поліморфізм методів і операцій). Віртуальні функції

2. Індексатори

### Методичні рекомендації до виконання лабораторної роботи

1. Уважно прочитайте лекцію та цю лабораторну роботу.

2. Реалізуйте та проаналізуйте всі приклади з лекції.

3. При роботі над лекцією і виконанні завдань зверніть увагу, що в C# є **дві** різні реалізації перевантаження методів. В першому випадку реалізації, який і розглядається в цій роботі, в одному класі може бути кілька методів з **однаковими іменами**. При цьому вони повинні мати різні типи і/або кількість параметрів.

В другому випадку перевантаження методів використовується при спадкуванні класів – віртуальні методи.

Крім перевантаження методів в C# можна створювати методи зі змінною кількістю параметрів. При цьому останнім параметром повинен бути масив.

4. Для поглибленого вивчення цього матеріалу прочитайте [2].

### Порядок виконання роботи

1. Створити директорію Lab5, в якій будуть розміщуватися проекти цієї лабораторної роботи.

2. Виконати завдання свого варіанту у вигляді окремих консольних проектів.

### Приклади виконання завдань

Під час виконання об'єкти похідного класу можуть оброблятися як об'єкти базового класу в таких місцях, як параметри методу і колекції або масиви. Коли виникає поліморфізм, оголошений тип об'єкта перестає відповідати своєму типу під час виконання.

#### 1. Перевантаження методів

Використання декількох методів з одним і тим самим **іменем**, але різними типами **параметрів** називається *перевантаженням методів.*

Компілятор визначає, який саме метод потрібно викликати за типом фактичних параметрів. Наприклад, нижче наведено декілька реалізацій метода max з лекції 8, який повертає найбільше значення для різних типів і кількості параметрів.

// Повертає найбільше з двох цілих:

int max( int а, int b )

// Повертає найбільше з трьох цілих:

int max( int а, int b, int з )

// Повертає найбільше першого параметра і довжини другого:

int max ( int а, string b )

// Повертає найбільше другого параметра і довжини першого:

int max ( string b, int а )

...

Console.WriteLine( max( 1, 2 ));

Console.WriteLine( max( 1, 2, 3 ));

Console.WriteLine( max( 1, "2" ));

Console.WriteLine( max( "1", 2 ));

Багато методів, які реалізовані в мові C# перевантажені.

#### 2. Віртуальні члени класу

Базові класи можуть визначати і реалізовувати *віртуальні методи*, а похідні класи - перевизначати їх, тобто надавати своє власне визначення і реалізацію. Під час виконання, коли клієнт викликає метод, CLR виконує пошук типу об'єкта під час виконання і викликає перезапис віртуального методу. У вихідному коді можна викликати метод в базовому класі і забезпечити виконання версії методу, що відноситься до похідному класу.

Похідний член повинен використовувати ключове слово *override*, яке вказує, що метод призначений для участі у віртуальному виклику. Віртуальні методи і властивості дозволяють похідним класам розширювати базовий клас без необхідності використовувати реалізацію методу базового класу.

public class BaseClass

{

public virtual void DoWork() { }

public virtual int WorkProperty

{

get { return 0; }

}

}

public class DerivedClass : BaseClass

{

public override void DoWork() { }

public override int WorkProperty

{

get { return 0; }

}

}

#### Приховування членів базового класу новими членами

Якщо ви хочете, щоб похідний клас мав член з тим же ім'ям, що і член в базовому класі, можна використовувати ключове слово *new*, щоб приховати член базового класу. Ключове слово *new* вставляється перед типом значення, що повертається, члена класу, що заміщає.

public class BaseClass

{

public void DoWork() { WorkField++; }

public int WorkField;

public int WorkProperty

{

get { return 0; }

}

}

public class DerivedClass : BaseClass

{

public new void DoWork() { WorkField++; }

public new int WorkField;

public new int WorkProperty

{

get { return 0; }

}

}

Доступ до прихованих членам базового класу можна здійснювати з клієнтського коду приведенням примірника похідного класу до примірника базового класу

DerivedClass B = new DerivedClass();

B.DoWork(); // Calls the new method.

BaseClass A = (BaseClass)B;

A.DoWork(); // Calls the old method.

#### Захист віртуальних членів від перевизначення похідними класами

Похідний клас може зупинити віртуальне успадкування, оголосивши перевизначення як *запечатаний*. Для зупинки спадкування в оголошення члена класу потрібно вставити ключове слово *sealed* перед ключовим словом *override*. Прикладом є наступний код:

public class A

{

public **virtual** void DoWork() { }

}

public class B : A

{

public **override** void DoWork() { }

}

public class C : B

{

public **sealed** override void DoWork() { }

}

Запечатані методи можна замінити похідними класами за допомогою ключового слова new, як показано в наступному прикладі:

public class D : C

{

public new void DoWork() { }

}

#### Доступ до віртуальних членів базового класу з похідних класів

Похідний клас, який замінив або перевизначив метод або властивість, може отримати доступ до методу або властивості базового класу за допомогою ключового слова *base*.

public class Base

{

public virtual void DoWork() {/\*...\*/ }

}

public class Derived : Base

{

public override void DoWork()

{

//Perform Derived's work here

//...

// Call DoWork on base class

base.DoWork();

}

}

#### Перевантаження операторів

Поряд з методами ми можемо також перевантажувати оператори. Перевантаження операторів тісно пов'язана з перевантаженням методів. Для перевантаження оператора служить ключове слово *operator*, що визначає операційний метод, який, в свою чергу, визначає дію оператора щодо свого класу. Існують дві форми операторних методів (operator): одна - для унарних операторів, інша - для бінарних. Нижче наведена загальна форма для кожного різновиду цих методів:

// Загальна форма перевантаження унарного оператора.

*public static возвращаемый\_тип* **operator** *op(тип\_параметра операнд)*

*{*

*// операції*

*}*

*// Загальна форма перевантаження бінарного оператора.*

*public static возвращаемый\_тип* **operator** *op(тип\_параметра1 операнд1,*

*тип\_параметра2 операнд2)*

*{*

*// операції*

*}*

##### Перевантаження бінарних операторів

Для бінарних операторів (+, -, \*, /, %, ==, !=, >, < тощо) тип значення , що повертається співпадає з класом, в якому перевантажується оператор, в якості параметрів використовуються об’єкти того самого класу, що й тип значення, що повертається. Зверніть увагу на те, що операційні методи повинні мати обидва специфікатори типу *public і static,* оскільки застосовуються до усіх об’єктів *.*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class MyArr

{

// Координати точки в тривимірному просторі

public int x, y, z;

public MyArr(int x = 0, int y = 0, int z = 0)

{

this.x = x;

this.y = y;

this.z = z;

}

// Перевантажуємо бінарний оператор +

public static MyArr operator +(MyArr obj1, MyArr obj2)

{

MyArr arr = new MyArr();

arr.x = obj1.x + obj2.x;

arr.y = obj1.y + obj2.y;

arr.z = obj1.z + obj2.z;

return arr;

}

}

class Program

{

static void Main(string[] args)

{

MyArr Point1 = new MyArr(1, 12, -4);

MyArr Point2 = new MyArr(0, -3, 18);

Console.WriteLine("Координати першої точки: " +

Point1.x + " " + Point1.y + " " + Point1.z);

Console.WriteLine("Координати другої точки: " +

Point2.x + " " + Point2.y + " " + Point2.z + "\n");

MyArr Point3 = Point1 + Point2;

Console.WriteLine("\nPoint1 + Point2 = "

+ Point3.x + " " + Point3.y + " " + Point3.z);

Console.ReadLine();

}

}

}

##### Перевантаження унарних операторів

Унарні оператори (++, --, -) перевантажуються так само як і бінарні. Головна відмінність полягає в тому, що у них є лише один операнд. Для унарних операторів параметр позначає операнд, що передається.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class MyArr

{

// Координати точки в тривимірному просторі

public int x, y, z;

public MyArr(int x = 0, int y = 0, int z = 0)

{

this.x = x;

this.y = y;

this.z = z;

}

// Перевантажуємо унарний оператор -

public static MyArr operator -(MyArr obj1)

{

MyArr arr = new MyArr();

arr.x = -obj1.x;

arr.y = -obj1.y;

arr.z = -obj1.z;

return arr;

}

// Перевантажуємо унарний оператор ++

public static MyArr operator ++(MyArr obj1)

{

obj1.x += 1;

obj1.y += 1;

obj1.z +=1;

return obj1;

}

}

class Program

{

static void Main(string[] args)

{

MyArr Point1 = new MyArr(1, 12, -4);

Console.WriteLine("Координати точки: " +

Point1.x + " " + Point1.y + " " + Point1.z);

MyArr Point2 = -Point1;

Console.WriteLine("-Point1 = "

+ Point2.x + " " + Point2.y + " " + Point2.z);

Point2++;

Console.WriteLine("Point2++ = "

+ Point2.x + " " + Point2.y + " " + Point2.z);

Console.ReadLine();

}

}

#### Індексатори

Індексатори дозволяють *індексувати об'єкти* і звертатися до даних за індексом. Фактично за допомогою індексаторів ми можемо працювати з об'єктами як з масивами. За формою вони нагадують властивості зі стандартними блоками get і set, які повертають і привласнюють значення. Формальне визначення індексатора:

Тип\_що\_повератється this [Тип параметр1, ...]

{

    get { ... }

    set { ... }

}

На відміну від властивостей індексатор не має назви. Замість нього вказується ключове слово *this*, після якого в квадратних дужках йдуть параметри. Індексатор повинен мати як мінімум *один параметр*.

class Person

{

    public string Name { get; set; }

}

class People

{

    Person[] data;

    public People()

    {

        data = new Person[5];

    }

    // індексатор

    public Person this[int index]

    {

        get

        {

            return data[index];

        }

        set

        {

            data[index] = value;

        }

    }

}

Після цього ми можемо працювати з об'єктом People як з набором об'єктів Person:

class Program

    {

    static void Main(string[] args)

    {

        People people = new People();

        people[0] = new Person { Name = "Tom" };

        people[1] = new Person { Name = "Bob" };

        Person tom = people[0];

        Console.WriteLine(tom?.Name);

        Console.ReadKey();

    }

}

Слід враховувати, що індексатор *не може бути статичним* і застосовується тільки до *примірника класу*. Але при цьому індексатори можуть бути *віртуальними* й *абстрактними* і можуть перевизначатися в похідних класах.

### Варіанти завдань для лабораторної роботи № 5

Завдання передбачають такі дії :

Завдання 1 передбачає додавання похідного класу до ієрархії класів лабораторної роботи №4 із закритими полями та відкритими властивостями.

Завдання 2 передбачає перевантаження конструкторів похідних класів. Потрібно перевантажити по 3 конструктори (без параметрів, та 2 конструктори з параметрами).

Завдання 3 і 4 – перевантажити методи в похідних класах через віртуальні функції у базовому класі та *override* функції в похідних.

Завдання 5 – перевантаження бінарних операторів (+, -, ==, != , >, < ).;

Завдання 6 – унарних операторів (+, -, -- ).

Завдання 7 – використання індексаторів для рооти з масивами об’єктів

В методі *Main*() класу *Program* продемонструвати виклик усіх методів класів. Усі значення, що розраховуються, записувати до *текстових файлів*.

**Студент має право додати додаткові поля та методи в класи завдань свого варіанту**.

**Номер варіанта визначається за порядковим номером в журналі.**

|  |  |
| --- | --- |
| **Номер варіанта** | **Зміст завдання** |
|  | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Тварина* (базовий клас), *Кішка, Собака* (похідні класи) додати похідний клас *Папуга*. Задати закриті специфічні поля в похідних класах та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити метод, який визначають умови, за яких тварина щаслива (їсть, спить, гуляє, грається тощо), використавши віртуальні **функції** в базовому класі та *override* функції в похідних. 4. У кожному похідному класі перевантажити методи, який визначають характер тварини відповідно до характеру її господаря, використавши віртуальну функцію в базовому класі та *override* функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, != , >, < ), наприклад, порівняння характерів домашніх тварин, збільшення (зменшення) раціону харчування в залежності від темпераменту, розміру, віку тварини. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) раціону харчування в залежності від темпераменту тварини. 7. Створити масив об’єктів класу *Кішка* (або *Собака, або Папуга*), використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
|  | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Людина* (базовий клас), *Робітник, Інженер* (похідні класи) додати похідний клас *Науковець*. Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які задають обсяг виконаних робіт та оплату за її виконання (робітник налагодив обладнання, інженер здав замовнику проект, науковець захистив патент на винахід), використавши віртуальні **функції** в базовому класі та *override* функції в похідних. 4. У кожному похідному класі перевантажити методи, які визначають рівень інтелекту (розумових здібностей) робітника, інженера та науковця в залежності від спадковості, впливу сім’ї, оточення людини, дохід сім’ї, економічний розвиток країни, здоров’я тощо (див. статтю «Коефіцієнт інтелекту» у Вікіпедія), використавши віртуальну фукцію в базовому класі та *override* функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, != , >, <), наприклад, порівняння трудомісткості праці, зарплат, коефіцієнтів інтелекту робітника, інженера та науковця, збільшення (зменшення) заробітної плати в залежності від обсягу виконаної роботи. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) обсягів робіт, заборгованість по зарплаті (зміна знаку «-» ). 7. Створити масив об’єктів класу *Робітник, (*або *Інженер,* або *Науковець)*, використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
|  | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Державний\_службовець* (базовий клас), *Державний\_секретар, Керівник\_апарату\_суду* (похідні класи) додати похідний клас *Спеціаліст*. Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які задають обсяг виконаних робіт та оплату за її виконання (наприклад, *Державний\_секретар* виконує аналітичну роботу,  *Керівник\_апарату\_суду* складає плани роботи суду та контролює їх виконання, *Спеціаліст* виконує підготовку документів та обробку інформації), використавши віртуальні функції в базовому класі та *override* функції в похідних. 4. У кожному похідному класі перевантажити методи, які визначають кримінальну відповідальність державних службовців, наприклад, за «зловживання владою або службовим становищем», за «службову недбалість», за «одержання хабаря», використавши віртуальну функцію в базовому класі та override функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, != , >, < ), наприклад, порівняння окладів державних службовців, збільшення (зменшення) заробітної плати в залежності від рангу (ієрархії) державного службовця, наприклад, преміщення на вищу посаду . 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) премії, розтрата бюджетних коштів (зміна знаку «-» ). 7. Створити масив об’єктів класу *Державний\_секретар, (або Керівник\_апарату\_суду, вбо Спеціаліст)*, використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
|  | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Виріб* (базовий клас), *Вузол, Деталь* (похідні класи) додати похідний клас *Агрегат*. Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які задають вартість *Вузла, Деталі* та *Агрегату* (кількість матеріалу, з якого виконаний об’єкт,× вартість одиниці матеріалу +вартість роботи), використавши віртуальні **функції** в базовому класі та *override* функції в похідних. 4. У кожному похідному класі перевантажити методи, які визначають кількість об’єктів, які можна виробити із заданої кількості сировини (кількість об’єктів = кількість матеріалу/норми витрат матеріалів на одиницю об’єкту ), використавши віртуальну фукцію в базовому класі та override функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, != , >, < ), наприклад, порівняння вартості об’єктів, збільшення (зменшення) вартості об’єктів в залежності від вартості матеріалу та оплати праці робітників. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) ваги об’єктів, виконання бракованих об’єктів (зміна знаку «-» ). 7. Створити масив об’єктів класу *Вузол, (*або *Деталь,* або *Агрегат)*, використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
| **5** | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Підприємство* (базовий клас), *Страхова\_Компанія, Нафтогазова\_Компанія, Завод,* (похідні класи) додати похідний клас *Університет*. Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково достворених раніше методів, перевантажити методи, які задають прибуток компаній, використавши віртуальні **функції** в базовому класі та *override* функції в похідних. Прибуток університету визначається як дохід від продажу наукових розробок, проведення експертиз, розробка технологій тощо. 4. У кожному похідному класі перевантажити методи, які визначають кількість працівників в залежності від фонду заробітної плати (кількість працівників=фонд заробітної плати/середню річну зарплату працівників), використавши віртуальну функцію в базовому класі та *override* функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, !=, >, < ), наприклад, порівняння прибутків компаній та університету, збільшення (зменшення) кількісті працівників компаній в залежності від суми прибутків. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) прибутків компаній, отримання штрафних санкцій за порушення антимонопольного законодавства = від’ємний прибуток (операція зміна знаку). 7. Створити масив об’єктів класу *Підприємство*, (або *Страхова\_Компанія, або Завод, або Університет)*, використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
| **6** | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Друковане\_Видання* (базовий клас), *Журнал, Книга* (похідні класи) додати похідний клас *Буклет*. Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які визначають вартість виготовлення певного видання (кількість примірників видання×кількість сторінок видання × вартість одної сторінки видання × коефіцієнт, що визначає тверду або м’яку обкладинку × коефіцієнт формату друку × коефіцієнт кольорового/чорно-білого друку), використавши віртуальні функції в базовому класі та *override* функції в похідних. 4. У кожному похідному класі перевантажити методи, які визначають кількість друкованих видань в залежності від загальної суми та вартості одного примірника видання, використавши віртуальну фукцію в базовому класі та *override* функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, !=, >, < ), наприклад, порівняння вартості одного примірника кожного з видань, збільшення (зменшення) кількісті друкованих примірників в залежності від виділеної загальної суми. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) ва друкування одного примірника видання при збільшенні вартості паперу, нарахування знижок вартості виготовлення видань (операція зміна знаку). 7. Створити масив об’єктів класу *Журнал* (або *Книга*, або *Буклет* ), використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
| **7** | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Рослина* (базовий клас), *Дерево, Квіти.* (похідні класи) додати похідний клас В*одорості*. Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які визначають користь від кожного виду рослин. Наприклад, дерева сприяють зменшенню шкідливих викидів, квіти мають терапевтичний ефект і позитивно впливають на здоров’я (заспокійлива дія, очищення повітря, полегшення дихання тощо), водорості мають харчову цінність, антиканцерогенну дію тощо. Для перевантаження використати віртуальні функції в базовому класі та *override* функції в похідних. 4. У кожному похідному класі перевантажити методи, які визначають економічний збиток від знищення кожного з видів рослин. Наприклад, вирубка лісів призводить до зміни клімату, вимірання тварин, знищення рідкісних квітів призводить до порушення екологічного законодавства, знищення волоростей призводить до забруднення водойм. Для перевантаження використати віртуальні **функції** в базовому класі та *override* функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, !=, >, < ), наприклад, порівняння вартості різних рослин, збільшення (зменшення) кількісті рослин на певній території. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення штрафів за незаконну торгівлю рослинами, визначення шкоди від знищення рослин як противага користі від них (операція зміна знаку). 7. Створити масив об’єктів класу *Дерево* (або *Квіти*, або *Водорості*), використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
| **8** | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Місцевість* (базовий клас), *Село, Місто.* (похідні класи) додати похідний клас *Селище\_Міського\_Типу*. Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які визначають розвиток *Села*, *Міста*, С*елища\_Міського\_Типу* у зв’язку із залученням інвестицій. Наприклад, у місті побудували нове підприємство, як наслідок нові прибутки в казну міста, в селі відкрили ферму, в селищі міськогог типу відкрити курорт. Для перевантаження використати віртуальні функції в базовому класі та override функції в похідних. 4. У кожному похідному класі перевантажити методи, які визначають міграційні процеси в різних типах місцевості. Наприклад, збільшується кількість населення у містах, зменшується кількість населення в селах та в селищах міського типу Для перевантаження використати віртуальні **функції** в базовому класі та *override* функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, !=, >, < ), наприклад, порівняння вартості життя в містах та селах, збільшення (зменшення) кількості населення в містах та селах. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, об’єднання декількох сіл, зменшення кількості районів у містах, збільшення рейтингу привабливості місцевості. 7. Створити масив об’єктів класу *Села*, (*Міста*, С*елища\_Міського\_Типу)*, використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
| **9** | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Товар* (базовий клас), *Іграшка, Продукти* (похідні класи) додати похідний клас *Житлово\_Комунальні\_Послуги* Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які визначають популярність різних товарів та їх користь для людини. Наприклад, іграшки корисні для розвитку інтелекту, продукти мають харчову цінність (калорійність, енергія), житлово-комунальні послуги задовольняють фізичні потреби людини. Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 4. У кожному похідному класі перевантажити методи, які розраховують вартість споживчого кошика людини відповідно до наборів продуктів харчування, непродовольчих товарів та послуг, що забезпечують продуктовий мінімум. (https://www.buhoblik.org.ua/rizni/spozhivkoshik/postanova-780-zmist.html). Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, !=, >, < ), наприклад, порівняння вартості споживчого кошику для різних соціальних груп населення (діти, працездатне та непрацездатне населення), збільшення (зменшення) вартості споживчого кошику чи окремих товарів. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) кількості товарів, збільшення (зменшення) корисності товарів. 7. Створити масив об’єктів класу *Іграшка, (*або *Продукти,* або *Житлово\_Комунальні\_Послуги),* використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
| **10** | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Документ* (базовий клас), *Квитанція, Накладна* (похідні класи) додати похідний клас *Чек.* Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які визначають рух грошових коштів та звітність про облік доходів та витрат. Наприклад, квитанція свідчить про покупку, тобто про витрати коштів, накладна свідчить про вартість та специфікацію товарів, що підлягають доставці та оплати, чек свідчить про придбаний товар або зобов’язання банку видати готівку. Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 4. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які розраховують штрафні санкції, якщо відсутні фінансові документи у вишляді квитанцій, накладних та чеків. Наприклад, відсутність чеку унеможливлює повернення неякісного товару, відсутність квитанцій унеможливлює підтвердження оплати (прийдеться повторити оплату), відсутність накладної унеможливлює перевірити якість, специфікацію та вартість поставленого товару. Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, !=, >, < ), наприклад, порівняння суми грошей, що записані в квитанції, накладній та чеку, збільшення (зменшення) записаних у документах сум в залежності від кількості товарів та якості послуг. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) кількості товарів, записаних у документах. 7. Створити масив об’єктів класу *Квитанція,* (або *Накладна,* або *Чек),* використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
| **11** | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Транспорт* (базовий клас), Ав*томобіль, Потяг* (похідні класи) додати похідний клас *Літак.* Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які розраховують вартість поїздки одної людини на кожному з видів транспорту в залежності від відстані, класу квитка (бізнес-клас, економ-клас, купейний, люкс тощо), сезону (на свята та в літній період квитки дорожче). Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. Порівняти вартість поїдки в різних видах транспорту за тим самим маршрутом. 4. У кожному похідному класі перевантажити методи, які розраховують та порівнюють тривалість поїздки різними видами транспорту за тим самим маршрутом з урахуванням відстані, швидкості транспорту, можливі затримки через непередбачувані ситуації (наприклад, нельотна погода для літаків, аварія автомобіля, ремонт дороги або залізничних шляхів, заборона зупинки потягу через коронавірус) тощо. Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, !=, >, < ), наприклад, порівняння вартості поїздки, збільшення (зменшення) вартості квитків та бензину. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) кількості куплених квитків на різні види транспорту. 7. Створити масив об’єктів класу Ав*томобіль,* (або *Потяг,* або *Літак),* використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
| **12** | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Викладач* (базовий клас), *Професор, Доцент* (похідні класи) додати похідний клас *Асистент.* Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які розраховують рейтинг викладача в залежності від кількості наукових праць, кількості годин лекційного навантаження, кількості студентів, які виконують під керівництвом викладача наукову роботу, популярності серед студентів. Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 4. У кожному похідному класі перевантажити методи, які розраховують розмір премії викладача в залежності від рейтингу викладача, рейтингу кафедри, розміру преміального фонду. Врахувати, що мінімальний розмір премії 1000 грн. Розмір преміального фонду задати з консолі. Рейтинг кафедри генерувати прсевдовипадковими числами в діапазоні від 1 до 100. Найкращий рейтинг дорівнює 100, найгірший рейтинг дорівнює 1. Кожні 10 балів рейтингу кафедр збільшують преміальний фонд кафедри на 1000 грн. Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, !=, >, < ), наприклад, порівняння рейтингу викладачів та суми премії, збільшення (зменшення) розміру премії. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) кількості викладачів, рейтинг яких в заданому користувачем діапазоні. 7. Створити масив об’єктів класу *Професор,* (або *Доцент,* або *Асистент),* використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
| **13** | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Рослина* (базовий клас), *Овочі, Фрукти* (похідні класи) додати похідний клас *Ягоди.* Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які розраховують вартість овочів, фруктів та ягод в залежності від ціни за 1 кг, сезону, постачальника та магазину. Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 4. У кожному похідному класі перевантажити методи, які розраховують зміну калорійності овочів, фруктів та ягод при їх приготуванні (наприклад, під час варіння фруктів та ягід їх калорійність зменшується на 5-10%, варіння овочів –картопля, морква, буряк, збільшує їх калорійність на 30%). Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, !=, >, < ), наприклад, порівняння вартості овочів, фруктів та ягід в залежності від пори року, постачальника та магазину-продавця, збільшення (зменшення) вартості овочів, фруктів та ягід в залежності від сезону. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) кількості овочів, фруктів та ягід, які можна купити за визначеною в п.5 вартістю на задану користувачем суму. Передбачити нестачу грошей для оплати (операція зміна знаку). 7. Створити масив об’єктів класу *Овочі,* (або *Фрукти,* або *Ягоди),* використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
| **14** | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Книга* (базовий клас), *Підручник, Конспект* (похідні класи) додати похідний клас *Електронна\_книга.* Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які визначають доступність матеріалу та статус підручника, конспекту та електронної книжки. Статус підручника означає надання грифу «рекомендовано» чи «затверджено», статус конспекту – відсутній, статус електронної книги: «опубліковано», «завантажено», «в обробці», «знята з публікації». Доступність матеріалу кодувати цифрами, наприклад, «легка для розуміння», «важка для розуміння», «проста», «чіткість» тощо. Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 4. У кожному похідному класі перевантажити методи, які визначають рейтинг підручника, конспекта та електронної книжки з погляду студента. Врахувати такі показники, як актуальність і сучасність, доступність матеріалу для опанування, наявність практичних вправ і задач з відповідями, розвиток логічного мислення, відгуки в Інтернет. Алгоритм визначення рейтингу визначити самостійно. Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, !=, >, < ), наприклад, порівняння рейтингів підручників, конспектів, електронної книжки, збільшення (зменшення) вартості книги в залежності від її рейтингу. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) рейтингу книги в залежності від кількості скачувань для електронної книжки, кільксті позитивних/негативних відгуків для підручників і конспектів. Передбачити від’ємний рейтинг (операція зміна знаку). 7. Створити масив об’єктів класу *Підручник,* (або *Конспект,* або *Електронна\_книга.),* використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
| **15** | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Меблі* (базовий клас), *Корпусні\_меблі, М'які\_меблі,* (похідні класи) додати похідний клас *Фурнітура.* Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які імітують онлайн голосування для визначення найкращих меблів. Згенеровані числа (голоси) підсумовуються для кожного виду меблів і визначається той вид меблів, який набере найбільшу кількість голосів. Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 4. У кожному похідному класі перевантажити методи, які імітують проведення акцій, наприклад, з нагоди дня народження клієнта, річниці магазину, чорної п’ятниці тощо. Акційні знижки задавати з консолі. Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, !=, >, < ), наприклад, порівняння вартості меблів, порівняння якості меблів, зменшення вартості меблів з нагоди проведення акцій. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) рейтингу меблів в залежності від результатів онлайн голосування клієнтів. 7. Створити масив об’єктів класу *Корпусні\_меблі, (*або *М'які\_меблі,* або *Фурнітура),* використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
| **16** | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Комп’ютерна\_техніка* (базовий клас), *Комп’ютер, Монітор, Планшет* (похідні класи) додати похідний клас *Принтер.* Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які визначають *вартість експлуатації* комп’ютерної техніки з урахуванням періоду експлуатації (кількість років/місяців), вартості ремонту або модернізації, вартості електроенергії, вартості Інтернет тощо. Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 4. У кожному похідному класі перевантажити методи, які визначають користь і шкоду від комп’ютерної техніки. Наприклад, показники користі: швидкість пошуку інформації = економія часу, засоби комунікації = економія часу, розширення контактів =підвищення ефективності роботи. Показники шкоди: погіршення здоров’я (погіршення зору, проблеми з хребтом, гіподинамія, вплив на психіку)= витрати на оздоровлення.. Для перевантаження використати віртуальні функціїв базовому класі та override функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, !=, >, < ), наприклад, порівняння вартості комп’ютерної техніки, порівняння рейтингів кожного виду комп’ютерної техніки, збільшення (зменшення) вартості комп’ютерної техніки в результаті збільшення (зменшення) рейтингу. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) рейтингу комп’ютерної техніки в залежності від дати її випуску (сучасна або застаріла техніка). 7. Створити масив об’єктів класу *Комп’ютер, (*або *Монітор,* або *Планшет, або Принтер),* використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
| **17** | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Геометрична\_фігура* (базовий клас), *Коло, Трикутник, Многокутник* (похідні класи) додати похідний клас *Точка.* Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які визначають розмір сторони правильного -кутника, рівностороннього трикутника за радіусами вписаного та описаного кіл, відстань між двома точками для класу *Точка* .Для перевантаження використати *віртуальні* функції в базовому класі та *override* функції в похідних. 4. У кожному похідному класі перевантажити методи, які визначають опуклість многокутника, перевірити теорему Піфагора для трикутника, довжину хорди кола через радіус та центральний кут. Для перевантаження використати віртуальні функціїв базовому класі та override функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, !=, >, < ), наприклад, порівняння площ геометричних фігур, збільшення (зменшення) периметру фігури в залежності від збільшення (зменшення) довжин їх сторін. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, зміна значень розрахунків в залежності від збільшення (зменшення) сторін геометричних фігур. Для точки збільшення (зменшення) її координат. 7. *Створити масив об’єктів класу Коло (*або *Трикутник,* або *Многокутник),* використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
| **18** | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Колекції* (базовий клас), *Вектор, Матриця* (похідні класи) додати похідний клас *Список.* Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які здійснюють пошук значення елемента, що задане з консолі, та його порядкового номеру. Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 4. У кожному похідному класі перевантажити методи, які визначають мінімальний елемент колекції та його порядковий номер. Для перевантаження використати віртуальні функціїв базовому класі та override функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, !=, >, < ), наприклад, порівняння кількості елементів колекції, збільшення (зменшення) суми елементів колекції в результаті збільшення (зменшення) кількості її елементів. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) кількості елементів колекції, збільшення (зменшення) суми елементів колекції в результаті збільшення (зменшення) кількості її елементів. 7. Створити масив об’єктів класу *Вектор, (*або *Матриця,* або Список*),* використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
| **19** | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Стартап-проєкт* (базовий клас), *Команда\_проєкту, Наукові\_дослідження*. (похідні класи) додати похідний клас *Менеджер\_проекта.* Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які визначають його мотивацію виконувати лабораторні та контрольні роботи, брати участь в олімпіаді та здавати іспити. Алгоритм вимірювання мотивації визначити самостійно. Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 4. У кожному похідному класі перевантажити методи, які визначають залежність успіхів в професійній кар’єрі випускника від загального рейтингу студента. Для перевантаження використати віртуальні функціїв базовому класі та override функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, !=, >, < ), наприклад, порівняння балів за лабораторні та модульні контрольні роботи, збільшення (зменшення) балів в залежності від кількості виконаних лабораторних робіт. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) кількості лабораторних робіт та балів за їх виконання, зарахування штрафних санкцій (операція зміни знаку). 7. Створити масив об’єктів класу *Лабораторні\_роботи, (*або *Екзамен,* або *Олімпіада),* використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |
| **20** | Створити новий консольний застосунок мовою C# шляхом модифікації програми лабораторної роботи №4 в такий спосіб:   1. В ієрархію класів *Стартап-проєкт* (базовий клас), *Команда\_проєкту, Наукові\_дослідження* (похідні класи) додати похідний клас *Менеджер\_проекту.* Задати закриті специфічні поля в новому похідному класі та реалізувати такі відкриті методи класів: 2. Перевантажити конструктори у всіх класах, передбачивши по 3 конструктори: за замовчуванням (без параметрів з ініціалізацією значень полів за замовчуванням), конструктор з параметрами для ініціалізації полів класів значеннями аргументів, конструктор для введення значень полів класу з клавіатури або копіювання об’єктів. 3. У кожному похідному класі, додатково до створених раніше методів, перевантажити методи, які визначають тривалість роботи команди над проектом, тривалість виконання наукових досліджень, тривалість роботи менеджера проекту в залежності від кількості задач проекту, складності задач та кількості членів команди. Для перевантаження використати *віртуальні* функціїв базовому класі та *override* функції в похідних. 4. У кожному похідному класі перевантажити методи, які визначають оцінку ефективності роботи команди, менеджера проекту, наукових досліджень Алгоритм визначення ефективності визначити самостійно. Для перевантаження використати віртуальні функціїв базовому класі та override функції в похідних. 5. У похідних класах створити функцію для перевантаження бінарних операторів (+, -, ==, !=, >, < ), наприклад, порівняння тривалості роботи команд, тривалості виконання робіт над проектом, збільшення (зменшення) фінансування проекту в залежності від кількості виконаних робіт проекту. 6. У похідних класах створити функцію для перевантаження унарних операторів (++, -, -- ), наприклад, збільшення (зменшення) кількості робіт проекту та фінансування проекту. 7. Створити масив об’єктів класу *Наукові\_дослідження, (*або *Менеджер\_проекту),* використавши індексатори для ініціалізації масиву об’єктів, виведення значень полів об’єктів на консоль та доступу до заданих за індексом об’єктів. |

### Література

1. О.С.Бичков, Є.В.Іванов Об’єктно-орієнтоване програмування мовою C#/
2. Шилдт Г. С# 4.0: полное руководство. : Пер. с англ. — М. : ООО "И.Д. Вильямс", 2011. — 1056 с.
3. Нейгел К., Ивьен Б. и др. C# 2005 и платформа .Net 3.0 для профессионалов.–М.: ООО “И.Д. Вильямс", 2008. –1376 с.

## Лабораторна робота №6 Взаємодії класів, делегати та обробка виняткових ситуацій

### Рейтинг лабораторної роботи №6

|  |  |  |  |
| --- | --- | --- | --- |
| № п.п | Вид діяльності студента | Рейтинговий бал | Deadline |
| 1 | Написання коду з 5 завдань | 0,5\*5=2,5 | 25 квітня |
| 2 | Захист роботи | 0,5 |
| 3 | Звіт з роботи | 0,5 |  |
| Разом за роботу | | 3,5 |  |

### Мета роботи:

Виконанати лабораторну роботу, підсумувавши знання з усіх попередніх лабораторних робіт.

Реалізувати взаємодії класів у вигляді асоціації, агрегації, композиції.

Навчитись програмувати події та їх обробників.

Нгавчитись визначати та обробляти виняткові ситуації

### Методичні рекомендації до виконання лабораторної роботи

1. Уважно прочитайте лекції та методичні вказівки до лабораторної роботи.

2. Реалізуйте та проаналізуйте всі приклади з лекцій.

### Порядок виконання роботи

1. Створити директорію Lab6, в якій будуть розміщуватися проект цієї лабораторної роботи.

2. Виконати завдання свого варіанту у вигляді **одного** консольного проекту.

3. Для кожного класу передбачити окремий файл.

### Приклади виконання завдань

#### 1. Програма обробки виключних ситуацій

При створенні програм із застосуванням структурованої обробки виключень передбачається використовувати такі чотири взаємозалежних елементи:

* тип класу, який надає детальну інформацію про виниклу виняткову ситуацію;
* член, який генерує, або надсилає (**throw**) екземпляр класу, що відповідає винятковій ситуації, стороні, що викликає;
* блок програмного коду, в якому був викликаний член, що генерує виняток;
* блок програмного коду сторони, що викликає, в якому виконується обробка, або захоплення (**catch**) даного виключення.

У мові програмування C# пропонуються чотири ключових слова (**try, catch, throw і finally**), за допомогою яких генеруються і обробляються виключення. Тип, що представляє відповідну проблему, є класом, похідним від System.Exception (або його нащадком).

З урахуванням цього давайте з'ясуємо роль зазначеного базового класу. Створимо новий консольний додаток, в якому визначатимуться два типи класу Car (автомобіль) і Radio (радіо), пов'язані відношенням агрегації ("has-a"). Тип Radio визначає один метод, що включає і вимикає радіо.\

using System;

using System.Collections.Generic;

using System.Text;

namespace except\_example

{

public class Radio

{

/// <summary>

/// метод, що вмикає та вимикає радіо

/// </summary>

/// <param name="on"></param>

public void TurnOn(bool on)

{

if (on)

Console.WriteLine("Радіозавади…");

else

Console.WriteLine("….І тиша…");

}

}

}

Клас Car (автомобіль) визначає таку поведінку. Якщо користувач об'єкта Car перевищить межу для швидкості (ця межа задається значенням члена-константи), то «двигун вибухає» і об'єкт Car стає непридатним для використання, що виражається змінами значення поля типу bool з ім'ям carIsDead (автомобіль зруйнований). Крім того, тип Car має кілька членів-змінних, що представляють поточну швидкість і ім'я, дане автомобілю користувачем, а також кілька конструкторів. Ось повне визначення цього типу (з відповідними коментарями).

using System;

using System.Collections.Generic;

using System.Text;

namespace except\_example

{

public class Car

{

public const int maxSpeed = 100;// Константа для максимума швидкості.

private int currSpeed; // поточна швидкість автомобіля

private string carName; //назва автомобіля

private bool carIsDead;// Працюєт автомобиль?

private Radio theMusicBox = new Radio();// автомобіль має радіо

//відношення агрегації

/// <summary>

/// Конструктор////////////

/// </summary>

public Car()

{

currSpeed = 0;

carName = "";

}

/// <summary>

/// ////////

/// </summary>

/// <param name="name"></param>

/// <param name="currSp"></param>

public Car(string name, int currSp)

{

currSpeed = currSp;

carName = name;

}

/// <summary>

/// Запрос для внутрішнього об’єкта/

/// </summary>

/// <param name="state"></param>

public void CrankTunes(bool state)

{

theMusicBox.TurnOn(state);

}

/// <summary>

/// Чи перегрівся автомобіль?/////////

/// </summary>

/// <param name="delta"></param>

public void Accelerate(int delta)

{

if (carIsDead)

Console.WriteLine("не працює "+ carName);

else

{

currSpeed += delta;

if (currSpeed >= maxSpeed)

{

carIsDead = true;

currSpeed = 0;

//створення об’єкта виключення

CarException ex = new CarException(string.Format("{0} перегрівся!", carName));

ex.Data.Add("Дата та час", string.Format("Автомобіль поломався {0}", DateTime.Now));

ex.Data.Add("Причина", " перевищення швидкості");

// використовується оператор throw для генерації виключення

throw ex;

}

else

Console.WriteLine("=› currSpeed = "+ currSpeed);

}

}

}

}

Припустимо, що ми хочемо створити для користувача виняток (з ім'ям CarException), що представляє помилку перевищення швидкості приреченого автомобіля. Насамперед тут має бути створення нового класу з System.ApplicationException (за згодою, класи виключень мають суфікс "Exception", що в перекладі означає "виключення"). Як і в разі будь-якого іншого класу, можна визначити для користувача члени, які будуть потім використовуватися в блоці catch в рамках програмної логіки викликів. Точно так само можна перевизначити будь-які віртуальні члени, визначені батьківськими класами. Наприклад, можна реалізувати CarException, перевизначивши віртуальну властивість Messagе.

Якщо ви хочете побудувати точний призначений для користувача клас виключення, то створений вами тип повинен відповідати кращим зразкам, які використовують виключення .NET. Зокрема, ваше призначене для користувача виняток мав би підпорядковуватися наступним вимогам:

* + бути похідним від Exception / ApplicationException;
  + визначати конструктор, який використовується за умовчанням;
  + визначати конструктор, який встановлює успадковане властивість Message;
  + визначати конструктор, що обробляє "внутрішні виключення".

using System;

using System.Collections.Generic;

using System.Text;

namespace except\_example

{

/// <summary>

/// клас користувацького виключення

/// </summary>

public class CarException: ApplicationException

{

private string messageDetails;

public CarException()

{

messageDetails = "";

}

public CarException(string message):base(message)

{

}

/// <summary>

/// Перевизначення властивості Exception.Message

/// </summary>

public override string Message

{

get

{

return string.Format("Повiдомлення про помилку Car: {0}", messageDetails);

}

}

}

}

Метод Accelerate() може генерувати виключення, яке сторона, що викликає, повинна бути готова обробити такий виняток. При виклику методу, здатного генерувати виняток, ви повинні використовувати блок **try / catch**. Прийнявши виняток, ви можете викликати члени типу **System.Exception** і прочитати детальну інформацію про проблему. Що ви будете робити з отриманими даними, залежить, в основному, від вас. Ви можете помістити відповідну інформацію в файл звіту, записати її в журнал реєстрації подій Windows, відправити її електронною поштою адміністратору системи або показати повідомлення з описом проблеми кінцевому користувачеві. Тут ми просто виводимо інформацію у вікно консолі.

В рамках блоку **try / catch** можна також визначити необов'язковий блок **finally**. Завдання блоку **finally** – забезпечити безумовне виконання деякого набору операторів програмного коду, незалежно від наявності або відсутності виключення (будь-якого типу). Для прикладу припустимо, що ви хочете завжди вимикати радіо автомобіля перед виходом з Main(), незалежно від винятків.

using System;

namespace except\_example

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("\*\*\* Створення та тестування автомобiля \*\*\*");

Car myCar = new Car("Zippy", 20);

myCar.CrankTunes(true);

try //спроба виконати дію, яка генерує виключення

{

for (int i = 0; i <= 10; i++)

myCar.Accelerate(10);

}

catch (CarException e) //перехоплення виключення та створення

//об’єкту виключення х відповідними вдастивостями

{

Console.WriteLine("\n\*\*\* Помилка1 \*\*\*");

Console.WriteLine("Метод: {0}", e.TargetSite);

Console.WriteLine("Повiдомлення : {0}", e.Message);

Console.WriteLine("Джерело помилки: {0}", e.Source);

}

catch (ArgumentOutOfRangeException ex) //перехоплення

//іншого виключення

{

Console.WriteLine("\n\*\*\* Помилка2 \*\*\*");

Console.WriteLine(ex.Message);

}

finally // виконується завжди, незалежно від виключень

{

myCar.CrankTunes(false); //вимикання радіо

}

// помилка оброблена, виконується насупнй оператор.

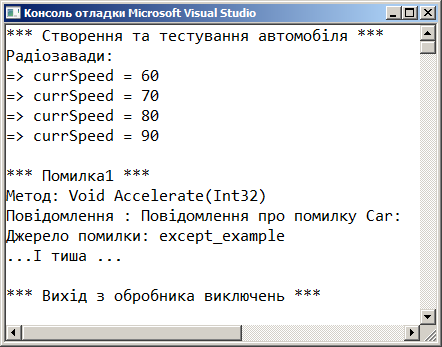
Console.WriteLine("\n\*\*\* Вихiд з обробника виключення \*\*\*");

Console.ReadLine();

}

}

}



Результат роботи програми оброки виключення

#### 2. Програма використання делегатів

Делегат – це тип, який представляє посилання на методи з певним списком параметрів і типом значення, що повертається, тобто делегат описує сигнатуру методу. При створенні екземпляра делегата цей екземпляр можна пов'язати з будь-яким методом з сумісною сигнатурою і типом значення, що повертається. Метод можна викликати (активувати) за допомогою екземпляра делегата.

Делегати використовуються для передачі методів в якості аргументів до інших методів. Обробники подій – це ніщо інше, як методи, що викликаються за допомогою делегатів. При створенні користувальницького методу клас (наприклад, елемент керування Windows) може викликати цей метод при появі певної події.

У наступному прикладі показано оголошення делегата:

public delegate тип\_значення\_що\_повертається ім’я\_делегата(список параметрів);

Делегату можна призначити будь-який метод з будь-якого доступного класу або структури, що відповідає типу делегата. Цей метод повинен бути **статичним методом** або методом екземпляра. Така гнучкість дозволяє програмно змінювати виклики методу, а також включати новий код в існуючі класи.

##### Властивості делегата

* Делегати подібні покажчикам на функції в C++, але є повністю об'єктно-орієнтованими і, на відміну від покажчиків C++ на функції-члени, інкапсулюють екземпляр об'єкта разом з методом.
* Делегати допускають передачу методів в якості параметрів.
* Делегати можна використовувати для визначення методів зворотного виклику.
* Делегати можна пов'язувати між собою; наприклад, при появі одної події можна викликати кілька методів.
* Точна відповідність методів типу делегата не потрібна.

##### Приклад коду з використанням делегата

Необхідно змінити рядок символів, видаливши пробіли або замінивши пробіли на інший символ. Операції модифікації рядка символів подати функціями, яким передається вхідний рядок для модифікації. Функції повертають рядок, що змінений відповідно до умови задачі. Для виклику функцій використати делегат, який повертає тип string, і приймає в якості параметра рядок..

using System;

using System.Collections.Generic;

using System.Text;

namespace examplDelegat

{

public class TestDelegat

{

//делегат strmodify для модифікації рядку

public delegate string strmodify(string str);

/// <summary>

/// функція, яку викликатиме делегат для заміни пробілів символом'-'

/// </summary>

/// <param name="s"></param>

/// <returns></returns>

public static string replaceSpace(string s)

{

return s.Replace(' ', '-');

}

/// <summary>

/// функція, яку викликатиме делегат для видалення пробілів

/// </summary>

/// <param name="s"></param>

/// <returns></returns>

public static string removeSpace(string s)

{

string temp = " ";

for (int i = 0; i < s.Length; i++)

if (s[i] != ' ') temp += s[i];

return temp;

}

}

}

Код головної функції, яка використовуватиме делегат, працює так:

1. Створити об’єкт делегат відповідно до його оголошення. В якості параметра використати ім’я функції, яку викликатиме делегат. Використовуються складені імена:

**ім’я класу.ім’я делегата** для об’єкта делегата

**ім’я класу.ім’я функції** для параметра делегата

1. Викликати об’єкт делегата, передавши йому в якості аргумента вхідний рядок
2. Створити новий екземпляр делегата відповідно до його оголошення.
3. Викликати об’єкт делегата, передавши йому в якості аргумента новий рядок
4. Вивести на консоль результати виконання функцій, яким делегат переда управління.

using System;

namespace examplDelegat

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Using delegates");

// створення об'єкту objDelegate типу делегат

//в якості аргументу ім'я функції, яку викликатиме делегат

TestDelegat.strmodify objDelegate = new TestDelegat.strmodify(TestDelegat.replaceSpace);

string stroka = "t e s t 1"; //рядок, що модифікується

Console.WriteLine("вхiдний рядок до виклику делегату = " + stroka);

//виклик функції replaceSpace класу TestDelegat дл заміни пробілів

string stroka1 = objDelegate("t e s t 1");

Console.WriteLine("результат виклику делегата replaceSpace = " + stroka1);

// створення нового екземпляру objDelegate типу делегат

//в якості аргументу ім'я функції, яку викликатиме делегат

objDelegate = new TestDelegat.strmodify(TestDelegat.removeSpace);

//виклик функції removeSpace класу TestDelegat дл видалення пробілів

string stroka2 = objDelegate("t e s t 2");

Console.WriteLine("результат виклику делегата removeSpace = " + stroka2);

}

}

}

#### Програма використання подій і делегатів для їх підтримки

Подія являє собою автоматичне повідомлення про те, що відбулася деяка дія. Події діють за таким принципом: об'єкт, що виявляє інтерес до події, реєструє обробник цієї події. Коли ж подія відбувається, викликаються всі зареєстровані обробники цієї події. Обробники подій зазвичай представлені делегатами.

Події є членами класу і оголошуються за допомогою ключового слова **event**. Найчастіше для цієї мети використовується наступна форма:

**event делегат\_події ім’я\_події;**

Як і делегати, події підтримують групову адресацію. Це дає можливість декільком об'єктам реагувати на повідомлення про подію.

Методи екземпляра і статичні методи можуть бути використані в якості обробників подій, але між ними є одна істотна відмінність. Коли статичний метод використовується в якості обробника, повідомлення про подію поширюється на весь клас. А коли в якості обробника використовується метод екземпляра, то події адресуються конкретним екземплярів об'єктів. Отже, кожен об'єкт певного класу, якому потрібно отримати повідомлення про подію, повинен бути зареєстрований окремо. На практиці більшість обробників подій є методи екземпляра.

using System;

namespace evenApp1

{

class Program

{

static void Main(string[] args)

{

MyEvent evt = new MyEvent();

UserInfo user1 = new UserInfo(Name: "Alex", Family: "Erohin", Age: 18);

Console.WriteLine("Name:" + user1.Name+" Family="+ user1.Family +" Age="+ user1.Age);

// Додати обробник події

evt.UserEvent += user1.UserInfoHandler;

// Запустити подію

evt.OnUserEvent();

Console.ReadLine();

}

}

}

Клас події на базі делегата

using System;

using System.Collections.Generic;

using System.Text;

namespace evenApp1

{

delegate void UI();

class MyEvent

{

// оголошення події

public event UI UserEvent;

// використання методу для запуску події

public void OnUserEvent()

{

UserEvent();

}

}

}

Клас обробника події

using System;

using System.Collections.Generic;

using System.Text;

namespace evenApp1

{

class UserInfo

{

string name, family;

int age;

public UserInfo(string Name, string Family, int Age)

{

name = Name;

family = Family;

age = Age;

}

public string Name

{

set { name = value; }

get { return name; }

}

public string Family

{

set { family = value; }

get { return family; }

}

public int Age

{

set { age = value; }

get { return age; }

}

// Обробник події

public void UserInfoHandler()

{

Console.WriteLine("Подiя викликана!\n");

Name = " Serg";

Family = "Khmelnyk";

Console.WriteLine("Iм'я: {0}\nПрiзвище: {1}\nВiк: {2}", Name, Family, Age);

}

}

}

### Варіанти завдань для лабораторної роботи № 6

**Студент має право додати додаткові класи, поля та методи в класи завдань свого варіанту**.

**Номер варіанта визначається за порядковим номером в журналі.**

|  |  |
| --- | --- |
| **№ варі-анту** | **Зміст завдання** |
| **1** | Створити проект для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проект класи для моделювання роботи «*розумного холодильника*» в якому зберігається масив продуктів. Холодильник складається (<https://www.moyo.ua/news/kakie_byvayut_holodilniki_5_raznovidnosteyi_holodilnyh_ustanovok.html>) з окремих елементів, зокрема ізотермічний шкаф (корпус, двери, теплоізоляція), електричне обладнання (електродвигун, компресор, испаритель, конденсатор, терморегулятор, хладагент, прибори автоматики), вбудований мікропроцесор для «розумного холодильника». Кожний пристрій холодильника і масив продуктів описати окремими класами в окремих файлах. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи холодильника. Придумати методи, які описують сценарій роботи «розумного холодильника», наприклад, автоматичне регулювання температури, виконання голосових команд користувача, сигналізація про зіпсовані продукти тощо. (мобілізувати студентську фантазію). 3. Розробити похідний від «розумного холодильника» клас «розумний кондиціонер» та промоделювати його роботу (<https://www.moyo.ua/news/kak_sdelat_kondicioner_iz_starogo_holodilnika_rukovodstvo_po_sborke.html> 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних, запису та читання файлів, виходу масиву за його межі, із семантикою предметної області у вигляді обробки сбоїв в роботі холодильника (наприклад, моделювання виходу з ладу окремих пристроїв). 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи розумного холодильника, придумати самостійно. |
| **2** | Створити проект для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проект класи для моделювання роботи «*розумного автомобіля*», який перевозить масив людей. Автомобіль складається (<https://monolith.in.ua/structure-avto/osnovnye-elementy-legkovogo-avto/>) з окремих елементів, зокрема кузова, двигуна, шассі, яке включає ходову частину, рульове управління, трансмісію, тормозну систему, smart систему. Кожний пристрій автомобіля і масив людей описати окремими класами в окремих файлах. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи автомобіля. **Придумати методи, які описують сценарій роботи «розумного автомобіля», наприклад, автоматичне регулювання температури, вологості, виконання голосових команд користувача, сигналізація про можливу аварійну ситуацію, блокування роботи у випадку водія, що в неадекватному стані тощо** (мобілізувати студентську фантазію). 3. Розробити похідний від «розумного автомобіля» клас «розумний мобільний туристичний душ» та промоделювати його роботу (<https://www.kolesa.ru/article/avtomojka-dlya-voditelya-testiruem-pohodnyj-dush>). Можна запропонувати інший похідний від автомобіля клас 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних, запису та читання файлів, виходу масиву за його межі, із семантикою предметної області у вигляді обробки сбоїв в роботі автомобіля (наприклад, моделювання виходу з ладу окремих пристроїв автомобіля). 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи розумного автомобіля, придумати самостійно. |
| **3** | Створити проект для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проект класи для моделювання роботи «*розумного годинника*», який виконує функції (<https://secunda.com.ua/uk/news/funkcii-smart-chasov.html>) смартфона, GPS-трекер (пошуковика) потрібної людини, фітнес-браслет, (контроль пульсу, тиску, крокоміру тощо), виконання голосових команд користувача, передавача та приймача сигналів, які надсилаютьсяє іншим користувачам для комунікації ощо. Складові «розумного годинника» придумати самостійно. Кожний елемент розумного годиника і масив людей, які можуть комунікувати за допомогою розумного одинника, описати окремими класами в окремих файлах. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи годинника. Придумати методи, які описують сценарій роботи «розумного годинника», відповідно до п.1 завдання, наприклад, автоматичне відстеженн повідомлень про будь-які ситуації, що трапляються з людиною, яка носить розумний годиник (мобілізувати студентську фантазію). 3. Розробити похідний від «розумного годинника» клас «розумний голосовий помічник» або інший розумний елемент годинника та промоделювати його роботу (<https://pingvin.pro/gadgets/article-gadget/rozumnyj-godynnyk-dlya-chogo-potriben.html>). 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних, запису та читання файлів, виходу масиву за його межі, із семантикою предметної області у вигляді обробки сбоїв в роботі годинника (наприклад, моделювання закінчення заряду батареї розумного годиника, потрапляння людини у критичну ситуацію (раптовий інфаркт, інсульт, втрата свідомості, висока температура тіла через хворобу тощо). 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи розумного годинника, придумати самостійно. |
| **4** | Створити проект для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проект класи для моделювання роботи «*розумного дзеркала*» (<https://ceramamarket.com/santehnica/dzerkala/rozyumne-dzerkalo-smart-mirror-800h600-touch-screen-23-6-weight-skin-detector>), яке виконує функції фотоапарата, передавача та приймача фото від друзів, відеозв’язку, підтримки Wi-Fi, bluetooth, детектора стану людини за її відображенням в дзеркалі (підвищенна температури, тискк, пульсу, поганий настрій, радість, сум людини), моделювання зачиски, визначення індексу маси тіла, ваги, співідношення ширової та м’язової тканин тіла тощо. Складові «розумного дзеркала» придумати самостійно. Кожний елемент розумного дзеркала і масив людей, які можуть комунікувати за допомогою розумного одинника, описати окремими класами в окремих файлах. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи «розумного дзеркала» (придумати самостійно відповідно до п.1 завдання), наприклад, автоматичне прогнозування самопочуття людини за її відображенням тарозпізнавання емоцій людин (мобілізувати студентську фантазію). 3. Розробити похідний від «розумного дзеркала» клас «розумний голосовий помічник» або інший розумний елемент дзеркала наприклад, доповнена реальність) та промоделювати його роботу. 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних, запису та читання файлів, виходу масиву за його межі, із семантикою предметної області у вигляді обробки сбоїв в роботі дзеркала (наприклад, моделювання закінчення заряду батареї розумного дзеркала, потрапляння людини у критичну ситуацію (раптовий інфаркт, інсульт, втрата свідомості, висока температура тіла через хворобу тощо). 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи розумного дзеркала, придумати самостійно (наприкл, критичні ситуації п. 4 завлання). |
| **5** | Створити проект для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проект класи для моделювання роботи «*розумних окулярів*» (<https://gogps.me/uk/blog/umnye-ochki-s-avtomaticheskim-fokusom-i-dopolnennoj-realnostju-/>), які виконують функції забезпечення високої чіткості та яскравості зображення, віртуальної та доповненої реальності, адаптації до погіршення зору користувача, розпізнавання осіб (корисно для поліції), зв’язок зі сматфорном для зображення повідомлень від нього тощо. «Розумні окуляри» складаються з лазера для передачі зображення на сітківку ока, відеокамери, мікрофона та динаміків для слабобачущих людей, віртуального дисплею, мікрокомп’ютера для інсталяції програмних додатків, компасу, акселерометру для визначення об’єктів в просторі та інших пристроїв. Кожний елемент розумних окулярів і масив людей, які можуть комунікувати за їх допомогою, описати окремими класами в окремих файлах. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи «розумних окулярів» (придумати самостійно відповідно до п.1 завдання), наприклад, автоматична ідентифікація та розпізнавання людей, прогнозування самопочуття людини за її станом сітківки тощо (мобілізувати студентську фантазію). 3. Розробити похідний від «розумних окулярів» клас «розумний голосовий помічник» або інший розумний елемент окулярів (наприклад, доповнена реальність) та промоделювати його роботу. 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних, запису та читання файлів, виходу масиву за його межі, із семантикою предметної області у вигляді обробки сбоїв в роботі розумних окулярів (наприклад, моделювання закінчення заряду батареї розумних окулярів, потрапляння людини у критичну ситуацію у зв’язку відмовою окремих елементів розумних окулярів тощо. 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи розумних окулярів, придумати самостійно (наприклад, критичні ситуації п. 4 завлання). |
| **6** | Створити проект для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проект класи для моделювання роботи «*розумного телевізору*» (<https://pershyj.com/p-yak-vibrati-rozumnii-televizor-poradi-fahivtsiv-41828>), який дозволяє користуватися інтернет-ресурсами, «серфити» в соціальних мережах, дивитися фотографії і відеоролики, спілкуватися через месенджери, робити покупки в Інтернет-магазинах, записувати контент, дублювати фільми, грати в комп’ютерні ігри, управляти через сматрфон та приймати голосові команди тощо. «Розумний телевізор» складається з пристроїв, що входять до складу звичайного телевізору: пристрій відображення візуальної інформації (рідкокристалічної або плазмової панелі); шасі (плата, яка містить основні електронні блоки телевізора (телетюнери, декодер з підсилювачем аудіо- і відеосигналів тощо), корпусу (який містить роз'єми, кнопки управління і гучномовці). Окрім того, смарт ТВ має процесор, звуковий адаптер, постійну та оперативну пам'ять, мережеві інтерфейси (Bluetooth, Wi-Fi і LAN). Є можливість підключення акустичної системи та інших мультимедійних пристроїв. Елементи розумного телевізору потрібно описати окремими класами в окремих файлах. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи «розумного телевізору» (придумати самостійно відповідно до п.1 завдання), наприклад, (<http://remoo.ru/bytovaya-tekhnika/chto-takoe-smart-tv-v-televizore>) доступ до розважального контенту, ігор та інших застосунків. 3. Розробити похідний від «розумного телевізору» клас «розумний візуальний помічник» або інший розумний пристрій (наприклад, віртуальна або доповнена реальність) та промоделювати його роботу. (включити фантазію студента) 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних, запису та читання файлів, виходу масиву за його межі, із семантикою предметної області у вигляді обробки сбоїв в роботі розумного ТВ (наприклад, моделювання втрати файлів під час дистанцйного виключення телевізору, відсутність зв’язку під час трансляції з Інтернету контенту тощо. 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи розумних окулярів, придумати самостійно (наприкл, критичні ситуації п. 4 завлання). |
| **7** | Створити проект для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проект класи для моделювання роботи «*інтелектуального адаптивного підручника*» (<https://techtoday.in.ua/news/shtuchnij-intelekt-u-pidruchnikax-32792.html>), який надає можливість адаптуватися до рівня знань читача, розробляти додаткові пояснення, підбирати контрольні та лабораторні завдання, які цікаві студентам та відповідають їх можливостям, автоматично створювати реферати та огляди матеріалу, надавати питання для практики та відображати споріднені теми для ліпшого розуміння матеріалу, задавати запитання, проводити контрольні, заохочувати ігровими навчальними технологіями тощо. «І*нтелектуальний адаптивний підручник*» включає програмне забезпечення, комп’ютер, пристрої, що реалізують доповнену та віртуальну реальність, підтримують Wi-Fi для доступу до Internet, Bluetooth для мобільних платформ, можуть транслювати зображення на розумний телевізор тощо. Елементи *інтелектуального адаптивного підручника* потрібно додумати самостійно, описати окремими класами в окремих файлах. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи «*інтелектуального адаптивного підручника*» (придумати самостійно відповідно до п.1 завдання). 3. Розробити похідний від «*інтелектуального адаптивного підручника*» клас «розумний візуальний та голосовий тьютор (репетитор)» та промоделювати його роботу. (включити фантазію студента) 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних, запису та читання файлів, виходу масиву за його межі, із семантикою предметної області у вигляді обробки критичних ситуацій, пов’язаних з відмовою підручника адаптуватися під потреби студента через виходу з ладу ПЗ або комп’ютера, відсутність Інтернету тощо. 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи *інтелектуального адаптивного підручника*, придумати самостійно (наприкл, критичні ситуації п. 4 завлання). |
| **8** | Створити проект для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проект класи для моделювання роботи «*автомобіля-трансформера*», який перевозить масив людей. Автомобіль складається (<https://monolith.in.ua/structure-avto/osnovnye-elementy-legkovogo-avto/>) з окремих елементів, зокрема кузова, двигуна, шассі, яке включає ходову частину, рульове управління, трансмісію, тормозну систему, smart систему, систему перетворення наземного автомобіля у літальний апарат або підводний човен. Кожний пристрій автомобіля і масив людей описати окремими класами в окремих файлах. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи автомобіля. Придумати методи, які описують сценарій роботи «автомобіля-трасформера», наприклад, автоматичне перетворення у літак у випадку заторів на дорогах, перетворення у підводний човен у випадку заторів на мостах або відсутності мостів через річки, виконання голосових команд користувача, сигналізація про зможливу аварійну ситуацію, блокування роботи у випадку водія, що в неадекватному стані тощо. (мобілізувати студентську фантазію). 3. Розробити похідний від «автомобіля-трасформера» клас «літальне крісло» або інший пристрій (вілповідно до фантазії студента) та промоделювати його роботу. 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних, запису та читання файлів, виходу масиву за його межі, із семантикою предметної області у вигляді обробки сбоїв в роботі автомобіля (наприклад, моделювання виходу з ладу окремих пристроїв автомобіля). 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи розумного автомобіля, придумати самостійно. |
| **9** | Створити проект для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проект класи для моделювання роботи «розумного *обруча на голову для зчитування електричних сигналів мозку* (<https://www.iotforall.com/brain-sensing-technology-muse-headband>), який дозволяє покращити зосередженість користувача на проблемному питанні, керувати комп’ютером за допомогою думки, здійснювати обмін думками (зчитувати та передавати думки) з іншими носіями обруча, збільшувати продуктивність розумової праці, зняти стрес, керувати розумними пристрояти подумки (від одної думки зачиняти та відкривати двері домівки, включати світло, комп’ютер, ТВ тощо. «Розумний обруч» складається з датчиків чола, датчиків вух, еталонних датчиків, зарядних портів живлення, LED ліхтаря, пристроїв для зв’язку з комп’ютером, мікропроцесора (система на кристалі) тощо. Кожний пристрій «Розумного обруча» описати окремими класами в окремих файлах. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи «Розумного обруча». Придумати методи, які описують сценарій роботи «Розумного обруча», відповідно до п.1 завдання. (мобілізувати студентську фантазію). 3. Розробити похідний від «Розумного обруча» клас «Чип для вживлення людину» або інший пристрій (відповідно до фантазії студента) та промоделювати його роботу. 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних, запису та читання файлів, виходу масиву за його межі, із семантикою предметної області у вигляді обробки сбоїв або відхиленн від стандартного сценарію в роботі «Розумного обруча» (наприклад, обруч почав генерувати несанкціоновані команди). 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи розумного автомобіля, придумати самостійно. |
| **10** | Створити проект для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проект класи для моделювання роботи «розумного *стола з вбудованим штучним інтелектом»* (<https://g-pulse.com/smart-tables>, <https://gtracer.com.ua/ru/smartdesk-3--stol-s-iskusstvennym-intellektom-ot-autonomous>), який дозволяє регулювати висоту для сидіння або стояння за столом за допомогою клавіатури, підігрівати час, каву, нагадувати про першочергові завдання, обліковувати час роботи, нагадувати про перерви в роботі, покращувати зосередженість користувача на проблемному питанні через підказки та допомогу в організації роботи тощо. «Розумний стіл» складається з комп’ютера, який вбудований в стільницю, електродвигуна для зміни висоти та положення столу, портів для підключення будь-яких пристроїв, мультисенсорного дисплею, вбудованого програмного забезпечення для відтворення та редагування різного типу мультимедійний вміст на екрані (екран може міститися в стільниці), додаток для виконання функцій органайзера (нагадування про перелік завдань на поточний день, моніторинг часу, проведенного за столом), нагрівач для напоїв (чай, кава), бездротової док-станції для зарядки смартфона, USB-концентратор, динамік Bluetooth та інші (можлива уява студента). Кожний пристрій «Розумного столу» описати окремими класами в окремих файлах. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи «Розумного столу». Придумати методи, які описують сценарій роботи «Розумного столу», відповідно до п.1 завдання. (мобілізувати студентську фантазію). 3. Розробити похідний від «Розумного столу» клас «Ігровий гаджет» або інший пристрій (відповідно до фантазії студента) та промоделювати його роботу. 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних, запису та читання файлів, виходу масиву за його межі, із семантикою предметної області у вигляді обробки сбоїв або відхиленн від стандартного сценарію в роботі «Розумного столу» (наприклад, стіл почав генерувати несанкціоновані команди). 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи розумного автомобіля, придумати самостійно. |
| **11** | Створити проект для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проект класи для моделювання роботи «*розумного* *кухонний стіл з вбудованим штучним інтелектом»* (<https://g-pulse.com/smart-tables>), який дозволяє розпізнати продукти, пропонувати рецепти приготування їжі, поради кухарю, визначення енергетичнох цінності продуктів та готової їжі, зваження інгредієнтів під час приготування блюда, визначення наявності та кількості шкідливих речовин (наприклад, нітрати), визначення якості та свіжості продуктів тощо. «Розумний кухонний стіл» складається з комп’ютера, який вбудований в стільницю, камери, проектора для скаунування продуктів та розпізнавання продуктів та їжі, вбудованої плити, вбудованого програмного забезпечення з рецептами та порадами, мультисенсорного дисплею, бездротової док-станції для зарядки смартфона, USB-концентратор, динамік Bluetooth та інші (можлива уява студента). Кожний пристрій «Розумного кухонного столу» описати окремими класами в окремих файлах. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи «Розумного кухонного столу». Придумати методи, які описують сценарій роботи «Розумного кухонного столу», відповідно до п.1 завдання. (мобілізувати студентську фантазію). 3. Розробити похідний від «Розумного кухонного столу» клас «Довідник рецептів» або інший пристрій (відповідно до фантазії студента) та промоделювати його роботу. Довідник має розпізнавати думки кухаря та пропонувати відповідний рецепт приготування їжі. 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних, запису та читання файлів, виходу масиву за його межі, із семантикою предметної області у вигляді обробки сбоїв або відхиленн від стандартного сценарію в роботі «Розумного кухонного столу» (наприклад, стіл почав генерувати несанкціоновані команди). 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи розумного автомобіля, придумати самостійно. |
| **12** | Створити проект для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проект класи для моделювання роботи «р*озумних дверей у квартиру»*, які дозволяють розпізнати особу і сигналізувати у випадку сторонньої людини, сканувати людину у повний зріз, відкриватися за голосовою командою або за думкою розпізнаної людини, пропускати тільки розпізнану людину (мешканців квартири), подавати сигнал тривоги у поліцію у випадкуспроби взлому дверей або несанкційного протикнення у квартиру, записувати та вести облік людей, що відвідують квартиру, розпізнавати емоції та настрій людини, вітати та проводжати людини голосовим супровідом тощо. «*Розумні двері*» складаються з комп’ютера, візуалізатора, проектора, сканера тіла людини, інтерактивного сенсорного дисплею, вбудованого програмного забезпечення, датчиків, камери, звукового генератора, мобільний зв’язок тощо. Кожний пристрій «Розумних дверей» описати окремими класами в окремих файлах. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи «Розумних дверей*»*. Придумати методи, які описують сценарій роботи «Розумних дверей*»*, відповідно до п.1 завдання. (мобілізувати студентську фантазію). 3. Розробити похідний від «*Розумних дверей*» клас «*Розумний охоронець*» або інший пристрій (відповідно до фантазії студента) та промоделювати його роботу. «*Розумний охоронець*» (<https://blueset.ru/product/gsm-storozh-smart>) має включити сирену, здійснювати дозвон і розсилку SMS повідомлень за списком номерів, записаному в пам'ять приладу, виконувати інші за фантазією студента дії. 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних, запису та читання файлів, виходу масиву за його межі, із семантикою предметної області у вигляді обробки сбоїв або відхиленн від стандартного сценарію в роботі «*Розумних дверей*» (наприклад, почався несанціонований сигнал сирени). 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи розумного автомобіля, придумати самостійно. |
| **13** | Створити проект для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проект класи для моделювання роботи «р*озумний сканер тіла людини»*, який дозволяє розпізнати особу і сигналізувати у випадку сторонньої людини, сканувати людину у повний зріз, визначати склад тіла − процентний вміст кісток, жиру, води і м'язів в людському тілі (<https://tanita-ua.com/page/info_text_1>), будувати 3D модель людини для конструювання індивідуальних протезів, зняття точних вимірювань і відстеження змін форми тіла в часі, для тривимірної візуалізації результатів майбутньої пластичної операції, для моделювання фасону одягу тощо. «*Розумний сканер*» складається з комп’ютера, візуалізатора, проектора, аналізатора тіла (ваги, пульт управління), інтерактивного сенсорного дисплею, вбудоване програмне забезпечення тощо. Кожний пристрій «Розумного сканера» описати окремими класами в окремих файлах. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи «*Розумного сканера»*. Придумати методи, які описують сценарій роботи «*Розумного сканера»*, відповідно до п.1 завдання. (мобілізувати студентську фантазію). 3. Розробити похідний від «*Розумного сканера*» клас «*Розумний 3D принтер*» або інший пристрій (відповідно до фантазії студента) та промоделювати його роботу. «*3D принтер*» має друкувати штучну копію людини з матеріалу, який вибирає користувач. 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних, запису та читання файлів, виходу масиву за його межі, із семантикою предметної області у вигляді обробки сбоїв або відхиленн від стандартного сценарію в роботі «Розумного *сканера*» (наприклад, сканер почав генерувати несанкціоновані команди). 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи розумного автомобіля, придумати самостійно. |

### Література

1. О.С.Бичков, Є.В.Іванов Об’єктно-орієнтоване програмування мовою C#/

2.C# 2005 и платформа .Net 3.0 для профессионалов. Нейгел К., Ивьен Б. и др. –М.: ООО “И.Д. Вильямс", 2008. –1376 с.

## Рейтингові бали за усі лабораторні роботи з дисципліни «Вступ до ООП»

|  |  |  |  |
| --- | --- | --- | --- |
| № п.п | Номер лабораторної роботи | Рейтинговий бал | Deadline |
| 1 | ЛР1 | 3,5 | 15 лютий |
| 2 | ЛР2 | 5 | 27 лютий |
| 3 | ЛР3 | 6 | 15 березня |
| 4 | ЛР4 | 5,5 | 25 березня |
| 5 | ЛР5 | 4,5 | 10 квітня |
| 6 | ЛР6 | 3,5 | 25 квітня |
| Разом за лабораторні роботи | | 28 |  |