**Ермаков Никита Денисович**

**ДКИП-203прог**

**Лабораторная работа 4. Простейшие классы**

**Вариант 2**

Описать класс, реализующий шестнадцатеричный счетчик, который может увеличивать или уменьшать свое значение на единицу в заданном диапазоне, Предусмотреть инициализацию счетчика значениями по умолчанию и произвольными значениями. Счетчик имеет два метода: увеличения и уменьшения, - и свойство, позволяющее получить его текущее состояние, При выходе за границы диапазона выбрасываются исключения. Написать программу, демонстрирующую все разработанные элементы класса.

Код:

using System;

public class HexCounter

{

private int \_value;

private int \_minValue;

private int \_maxValue;

public HexCounter(int minValue = 0x0, int maxValue = 0xF, int initialValue = 0x0)

{

if (initialValue < minValue || initialValue > maxValue)

throw new ArgumentOutOfRangeException(nameof(initialValue), "Initial value must be within the specified range.");

\_minValue = minValue;

\_maxValue = maxValue;

\_value = initialValue;

}

public void Increment()

{

if (\_value < \_maxValue)

{

\_value++;

}

else

{

throw new InvalidOperationException("Cannot increment beyond maximum value.");

}

}

public void Decrement()

{

if (\_value > \_minValue)

{

\_value--;

}

else

{

throw new InvalidOperationException("Cannot decrement below minimum value.");

}

}

public string CurrentValue => \_value.ToString("X"); // Возвращает текущее значение в шестнадцатеричном формате

public void SetRange(int minValue, int maxValue)

{

if (minValue > maxValue)

throw new ArgumentException("Minimum value cannot be greater than maximum value.");

\_minValue = minValue;

\_maxValue = maxValue;

// Проверка текущего значения

if (\_value < \_minValue || \_value > \_maxValue)

throw new InvalidOperationException("Current value is out of new range.");

}

}

class Program

{

static void Main(string[] args)

{

try

{

HexCounter counter = new HexCounter(0x0, 0xF, 0x5);

Console.WriteLine("Текущее значение: " + counter.CurrentValue);

counter.Increment();

Console.WriteLine("После увеличения: " + counter.CurrentValue);

counter.Decrement();

Console.WriteLine("После уменьшения: " + counter.CurrentValue);

// Попытка выхода за границы

for (int i = 0; i < 5; i++)

{

counter.Increment();

}

Console.WriteLine("После увеличения до максимума: " + counter.CurrentValue);

// Попытка увеличить за пределами

try

{

counter.Increment();

}

catch (Exception ex)

{

Console.WriteLine("Ошибка: " + ex.Message);

}

// Установим новый диапазон

counter.SetRange(0x0, 0xA);

Console.WriteLine("Новый диапазон установлен. Текущее значение: " + counter.CurrentValue);

// Настроим счетчик так, чтобы он стал недопустимым

try

{

HexCounter invalidCounter = new HexCounter(0x0, 0xA, 0xB);

}

catch (Exception ex)

{

Console.WriteLine("Ошибка инициализации: " + ex.Message);

}

}

catch (Exception ex)

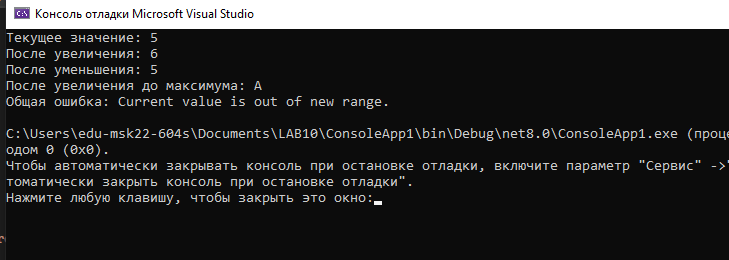
{

Console.WriteLine("Общая ошибка: " + ex.Message);

}

}

}



**Лабораторная работа 8. Классы и операции**

**Вариант 2**

Описать класс для работы с одномерным массивом строк фиксированной длины. Обеспечить следующие возможности:

задание произвольных целых границ индексов при создании объекта;

обращение к отдельной строке массива по индексу с контролем выхода за пределы массива;

выполнение операций поэлементного сцепления двух массивов с образованием нового массива;

выполнение операций слияния двух массивов с исключением повторяющихся элементов;

вывод на экран элемента массива по заданному индексу и всего массива.

Написать программу, демонстрирующую все разработанные элементы класса.

Код:

using System;

using System.Collections.Generic;

public class StringArray

{

private string[] \_array;

private int \_lowerBound;

private int \_upperBound;

public StringArray(int lowerBound, int upperBound)

{

\_lowerBound = lowerBound;

\_upperBound = upperBound;

\_array = new string[upperBound - lowerBound + 1];

}

// Индексатор для доступа к элементам массива с контролем выхода за пределы массива

public string this[int index]

{

get

{

if (index < \_lowerBound || index > \_upperBound)

throw new IndexOutOfRangeException("Индекс вне допустимого диапазона.");

return \_array[index - \_lowerBound];

}

set

{

if (index < \_lowerBound || index > \_upperBound)

throw new IndexOutOfRangeException("Индекс вне допустимого диапазона.");

\_array[index - \_lowerBound] = value;

}

}

// Метод для сцепления с другим массивом

public string[] Concatenate(StringArray otherArray)

{

string[] result = new string[this.\_array.Length + otherArray.\_array.Length];

Array.Copy(this.\_array, result, this.\_array.Length);

Array.Copy(otherArray.\_array, 0, result, this.\_array.Length, otherArray.\_array.Length);

return result;

}

// Метод для слияния с исключением повторяющихся элементов

public string[] Merge(StringArray otherArray)

{

HashSet<string> uniqueElements = new HashSet<string>();

foreach (var item in this.\_array)

{

if (!string.IsNullOrEmpty(item))

{

uniqueElements.Add(item);

}

}

foreach (var item in otherArray.\_array)

{

if (!string.IsNullOrEmpty(item))

{

uniqueElements.Add(item);

}

}

return new List<string>(uniqueElements).ToArray();

}

// Метод для вывода всего массива

public void PrintArray()

{

Console.WriteLine($"Массив:");

for (int i = \_lowerBound; i <= \_upperBound; i++)

{

Console.WriteLine($"[{i}]: {this[i]}");

}

}

}

class Program

{

static void Main(string[] args)

{

// Создаем два массива

StringArray array1 = new StringArray(0, 4);

array1[0] = "Apple";

array1[1] = "Banana";

array1[2] = "Cherry";

array1[3] = "Date";

array1[4] = "Elderberry";

StringArray array2 = new StringArray(0, 2);

array2[0] = "Fig";

array2[1] = "Grape";

array2[2] = "Apple"; // Дубликат

// Демонстрация всех возможностей

Console.WriteLine("Вывод первого массива:");

array1.PrintArray();

Console.WriteLine("\nВывод второго массива:");

array2.PrintArray();

// Сцепление двух массивов

Console.WriteLine("\nРезультат сцепления массивов:");

var concatenatedArray = array1.Concatenate(array2);

Console.WriteLine(string.Join(", ", concatenatedArray));

// Слияние двух массивов

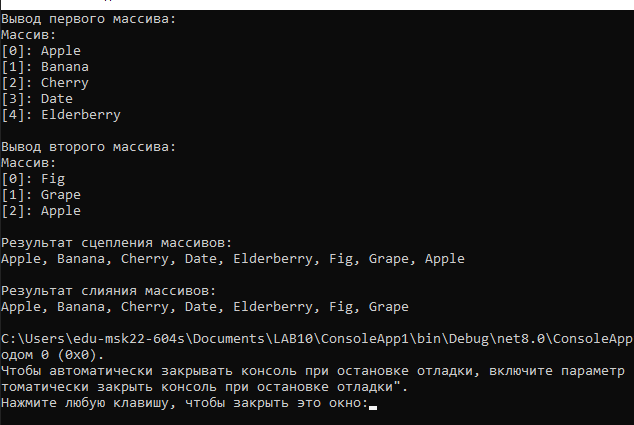
Console.WriteLine("\nРезультат слияния массивов:");

var mergedArray = array1.Merge(array2);

Console.WriteLine(string.Join(", ", mergedArray));

}

}



**Лабораторная работа 9. Наследование**

**Вариант 2**

Создать абстрактный класс Vehicle (транспортное средство). На его основе реализовать классы Plane (самолет), Car (автомобиль) и Ship (корабль). Классы должны иметь возможность задавать и получать координаты и параметры средств передвижения (цена, скорость, год выпуска и т. п.) с помощью свойств. Для самолета должна быть определена высота, для самолета и корабля - количество пассажиров, для корабля - порт приниски. Динамические характеристики задать с помощью методов.

Код:

using System;

abstract class Vehicle

{

public abstract double Price { get; set; }

public abstract double Speed { get; set; }

public abstract int YearOfManufacture { get; set; }

public abstract double[] Coordinates { get; set; }

public abstract void Move(double x, double y);

}

class Plane : Vehicle

{

public double Altitude { get; set; }

public int NumberOfPassengers { get; set; }

private double price;

public override double Price

{

get { return price; }

set { price = value; }

}

private double speed;

public override double Speed

{

get { return speed; }

set { speed = value; }

}

private int yearOfManufacture;

public override int YearOfManufacture

{

get { return yearOfManufacture; }

set { yearOfManufacture = value; }

}

private double[] coordinates = new double[2];

public override double[] Coordinates

{

get { return coordinates; }

set { coordinates = value; }

}

public override void Move(double x, double y)

{

Coordinates[0] += x;

Coordinates[1] += y;

Console.WriteLine($"Plane moved to coordinates: ({Coordinates[0]}, {Coordinates[1]}) at altitude {Altitude}m.");

}

}

class Car : Vehicle

{

public int NumberOfPassengers { get; set; }

private double price;

public override double Price

{

get { return price; }

set { price = value; }

}

private double speed;

public override double Speed

{

get { return speed; }

set { speed = value; }

}

private int yearOfManufacture;

public override int YearOfManufacture

{

get { return yearOfManufacture; }

set { yearOfManufacture = value; }

}

private double[] coordinates = new double[2];

public override double[] Coordinates

{

get { return coordinates; }

set { coordinates = value; }

}

public override void Move(double x, double y)

{

Coordinates[0] += x;

Coordinates[1] += y;

Console.WriteLine($"Car moved to coordinates: ({Coordinates[0]}, {Coordinates[1]}).");

}

}

class Ship : Vehicle

{

public string PortOfRegistry { get; set; }

public int NumberOfPassengers { get; set; }

private double price;

public override double Price

{

get { return price; }

set { price = value; }

}

private double speed;

public override double Speed

{

get { return speed; }

set { speed = value; }

}

private int yearOfManufacture;

public override int YearOfManufacture

{

get { return yearOfManufacture; }

set { yearOfManufacture = value; }

}

private double[] coordinates = new double[2];

public override double[] Coordinates

{

get { return coordinates; }

set { coordinates = value; }

}

public override void Move(double x, double y)

{

Coordinates[0] += x;

Coordinates[1] += y;

Console.WriteLine($"Ship moved to coordinates: ({Coordinates[0]}, {Coordinates[1]}) from port {PortOfRegistry}.");

}

}

class Program

{

static void Main(string[] args)

{

Plane plane = new Plane

{

Price = 500000,

Speed = 900,

YearOfManufacture = 2010,

Altitude = 10000,

NumberOfPassengers = 180

};

plane.Move(1, 2);

Car car = new Car

{

Price = 20000,

Speed = 120,

YearOfManufacture = 2019,

NumberOfPassengers = 5

};

car.Move(3, 4);

Ship ship = new Ship

{

Price = 1000000,

Speed = 50,

YearOfManufacture = 2015,

PortOfRegistry = "Helsinki",

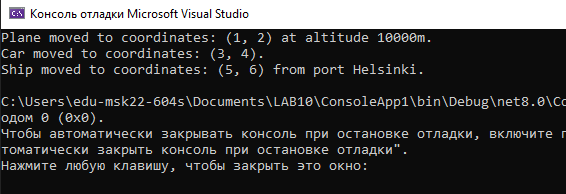
NumberOfPassengers = 200

};

ship.Move(5, 6);

}

}



**Лабораторная работа 10. Структуры**

**Вариант 2**

Описать структуру с именем STUDENT, содержащую следующие поля:

фамилия и инициалы;

номер групны;

успеваемость (массив из пяти элементов).

Написать программу, выполняющую следующие действия:

ввод с клавиатуры данных в массив, состоящий из десяти структур типа STUDENT (записи должны быть упорядочены по возрастанию среднего балла);

вывод на экран фамилий и номеров групп для всех студентов, имеющих оценки 4 и 5 (если таких студентов нет, вывести соответствующее сообщение).

Код:

using System;

using System.Linq;

public class StudentTask

{

// Определяем структуру STUDENT

struct STUDENT

{

public string фамилияИИнициалы;

public int номерГруппы;

public int[] успеваемость;

public double СреднийБалл

{

get { return успеваемость.Average(); }

}

}

public static void Main(string[] args)

{

// Создаем массив из 10 структур типа STUDENT

STUDENT[] students = new STUDENT[10];

// Ввод данных с клавиатуры

Console.WriteLine("Введите данные о студентах (10 человек):");

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

{

Console.WriteLine($"Студент {i + 1}:");

Console.Write(" Фамилия и инициалы: ");

students[i].фамилияИИнициалы = Console.ReadLine();

Console.Write(" Номер группы: ");

students[i].номерГруппы = int.Parse(Console.ReadLine());

students[i].успеваемость = new int[5];

Console.WriteLine(" Успеваемость (5 оценок через пробел):");

string[] gradesStr = Console.ReadLine().Split(' ');

for (int j = 0; j < 5; j++)

{

students[i].успеваемость[j] = int.Parse(gradesStr[j]);

}

}

// Упорядочиваем массив по возрастанию среднего балла

students = students.OrderBy(s => s.СреднийБалл).ToArray();

// Вывод фамилий и номеров групп для студентов с оценками 4 и 5

Console.WriteLine("\nСтуденты с оценками 4 и 5:");

bool found = false;

foreach (STUDENT student in students)

{

bool hasOnly4And5 = true;

foreach (int grade in student.успеваемость)

{

if (grade < 4)

{

hasOnly4And5 = false;

break;

}

}

if (hasOnly4And5)

{

Console.WriteLine($" Фамилия и инициалы: {student.фамилияИИнициалы}, Номер группы: {student.номерГруппы}");

found = true;

}

}

if (!found)

{

Console.WriteLine(" Нет студентов с оценками только 4 и 5.");

}

}