Чернівецький національний університет імені Юрія Федьковича Факультет математики та інформатики

Кафедра математичного моделювання

**Лабораторна робота №1**

з навчальної дисципліни: “Платформи корпоративних інформаційних систем”

Виконав: студент 3 курсу

301 групи

спеціальності “Комп’ютерні науки”

Малованюк С.М.

Перевірив: доцент Горбатенко М.Ю.

Варіант №1

**Чернівці – 2025**

**Exam.cs:**

namespace Exams;

enum Education

{

Master,

Bachelor,

SecondEducation

}

class Exam

{

public string SubjectName { get; set; }

public int Grade { get; set; }

public DateTime ExamDate { get; set; }

public Exam(string subjectName, int grade, DateTime examDate)

{

SubjectName = subjectName;

if (grade < 0 || grade > 100)

throw new ArgumentOutOfRangeException(nameof(grade), "Grade must be between 0 and 100");

Grade = grade;

ExamDate = examDate;

}

public Exam() : this("Math", 55, new DateTime(2025, 11, 23))

{

}

public override string ToString()

{

return $"Subject: {SubjectName}, Grade: {Grade}, Exam Date: {ExamDate.ToShortDateString()}";

}

}

**Student.cs:**

using Exams;

using System.Text;

class Student

{

private Person \_person;

private Education \_education;

private int \_groupNumber;

private Exam[] \_exams;

public Student(Person person, Education education, int groupNumber)

{

PersonInfo = person;

EducationInfo = education;

GroupInfo = groupNumber;

Exams = Array.Empty<Exam>();

}

public Student() : this(new Person(), Education.Bachelor, 301)

{

}

public Person PersonInfo

{

get => \_person; init => \_person = value ?? throw

new ArgumentNullException(nameof(value));

}

public Education EducationInfo { get => \_education; init => \_education = value; }

public int GroupInfo { get => \_groupNumber; init => \_groupNumber = value; }

public Exam[] Exams { get => \_exams; init => \_exams = value ?? Array.Empty<Exam>(); }

public double AverageGrade

{

get

{

if (Exams == null || Exams.Length == 0)

return 0;

int sum = 0;

foreach (var exam in Exams)

{

sum += exam.Grade;

}

return (double)sum / Exams.Length;

}

}

public bool this[Education eduIndex] => \_education == eduIndex;

public void AddExams(params Exam[] newExams)

{

if (newExams == null || newExams.Length == 0)

return;

int currentLength = Exams?.Length ?? 0;

Exam[] newArray = new Exam[currentLength + newExams.Length];

if (Exams != null)

Array.Copy(Exams, newArray, currentLength);

Array.Copy(newExams, 0, newArray, currentLength, newExams.Length);

\_exams = newArray;

}

public override string ToString()

{

StringBuilder sb = new StringBuilder().AppendLine(

$"Student Info:\n" +

$"Person: {PersonInfo}\n" +

$"Education: {EducationInfo}\n" +

$"Group Number: {GroupInfo}\n" +

$"Exams:");

if (Exams == null || Exams.Length == 0)

{

sb.AppendLine("\n No exams");

}

else

{

foreach (var exam in Exams)

{

sb.AppendLine($"\n {exam}");

}

}

return sb.ToString();

}

public virtual string ToShortString()

{

return $"Person: {PersonInfo}, Education: {EducationInfo}, Group: {GroupInfo}, Average Grade: {AverageGrade}";

}

}

**Person.cs:**

class Person

{

private string \_name;

private string \_surname;

private DateTime \_dateOfBirth;

public Person() : this(name: "Vasya", surname: "Pupkin", dateOfBirth: new DateTime(2000, 6, 12))

{

}

public Person(string name, string surname, DateTime dateOfBirth)

{

Name = name;

Surname = surname;

DateOfBirth = dateOfBirth;

}

public string Name

{

get => \_name;

init => \_name = value;

}

public string Surname

{

get => \_surname;

init => \_surname = value;

}

public DateTime DateOfBirth

{

get => \_dateOfBirth;

set => \_dateOfBirth = value;

}

public int YearOfBirthday

{

get => DateOfBirth.Year;

set => DateOfBirth = new DateTime(value, DateOfBirth.Month, DateOfBirth.Day);

}

public override string ToString()

{

return $"Name: {Name}, Surname: {Surname}, Date of birth: {DateOfBirth.ToShortDateString()}";

}

public virtual string ToShortString() => $"{Surname} {Name}";

}

**Program.cs:**

using Exams;

using System.Diagnostics;

class Program

{

static void Main()

{

var person1 = new Person("Vanya", "Nelisiy", new DateTime(1990, 5, 15));

Console.WriteLine(person1.ToString());

Console.WriteLine(person1.ToShortString());

Person person2 = new Person();

Console.WriteLine(person2.ToString());

person1.YearOfBirthday = 1995;

Console.WriteLine(person1.ToString());

Person person3 = new Person() { Name = "Luffy", Surname = "Monkey D" };

Console.WriteLine(person3.ToShortString());

Console.WriteLine(person3.ToString());

Console.WriteLine("-------------------------------NEXT PART-------------------------------");

Exam exam1 = new Exam("Programming", 90, new DateTime(2024, 2, 15));

Console.WriteLine(exam1.ToString());

Exam exam2 = new Exam();

Console.WriteLine(exam2.ToString());

Console.WriteLine("-------------------------------STUDENT CLASS-------------------------------");

Student student = new Student();

Console.WriteLine(student.ToShortString());

Console.WriteLine($"Indexer Education.Master: {student[Education.Master]}");

Console.WriteLine($"Indexer Education.Bachelor: {student[Education.Bachelor]}");

Console.WriteLine($"Indexer Education.SecondEducation: {student[Education.SecondEducation]}");

student = new Student

{

PersonInfo = new Person("Creatine", "Monohydrate", new DateTime(1999, 6, 28)),

EducationInfo = Education.Master,

GroupInfo = 502

};

Console.WriteLine(student.ToString());

student.AddExams(

new Exam("Programming", 95, new DateTime(2025, 12, 15)),

new Exam("Math", 90, new DateTime(2025, 12, 20)),

new Exam("English", 75, new DateTime(2025, 12, 25))

);

Console.WriteLine(student.ToString());

Console.WriteLine("-------------------------------BANCHMARKS-------------------------------");

int rows, cols;

do

{

Console.WriteLine("Enter dimensions [rows, columns] use ',', ' ' or ';', (example: 3,3):");

var input = Console.ReadLine()?.Split([',', ' ', ';'], StringSplitOptions.RemoveEmptyEntries);

if (input?.Length == 2 && int.TryParse(input[0], out rows) && int.TryParse(input[1], out cols))

break;

Console.WriteLine("Invalid input! Please try again.");

} while (true);

Console.WriteLine($"\nCreating arrays: {rows}x{cols} ({rows \* cols} elements)");

int totalElem = rows \* cols;

var oneDim = new Exam[totalElem];

var twoDim = new Exam[rows, cols];

var jaggedArray = new Exam[rows][];

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

jaggedArray[i] = new Exam[cols];

int acc = 0, maxRows = 0;

while (acc < totalElem)

{

maxRows++;

acc += maxRows;

}

var jaggedVarying = new Exam[maxRows][];

for (int i = 0; i < maxRows - 1; i++)

{

jaggedVarying[i] = new Exam[i + 1];

}

jaggedVarying[maxRows - 1] = new Exam[maxRows - (acc - totalElem)];

var sw1 = Stopwatch.StartNew();

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

oneDim[i] = new Exam { Grade = 100 };

sw1.Stop();

Console.WriteLine($"1. One-dimensional array: {sw1.Elapsed.TotalMilliseconds:0.000} ms");

var sw2 = Stopwatch.StartNew();

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

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

twoDim[i, j] = new Exam { Grade = 100 };

sw2.Stop();

Console.WriteLine($"2. Rectangular 2D array: {sw2.Elapsed.TotalMilliseconds:0.000} ms");

var sw3 = Stopwatch.StartNew();

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

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

jaggedArray[i][j] = new Exam { Grade = 100 };

sw3.Stop();

Console.WriteLine($"3. Jagged array (equal): {sw3.Elapsed.TotalMilliseconds:0.000} ms");

var sw4 = Stopwatch.StartNew();

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

for (int j = 0; j < jaggedVarying[i].Length; j++)

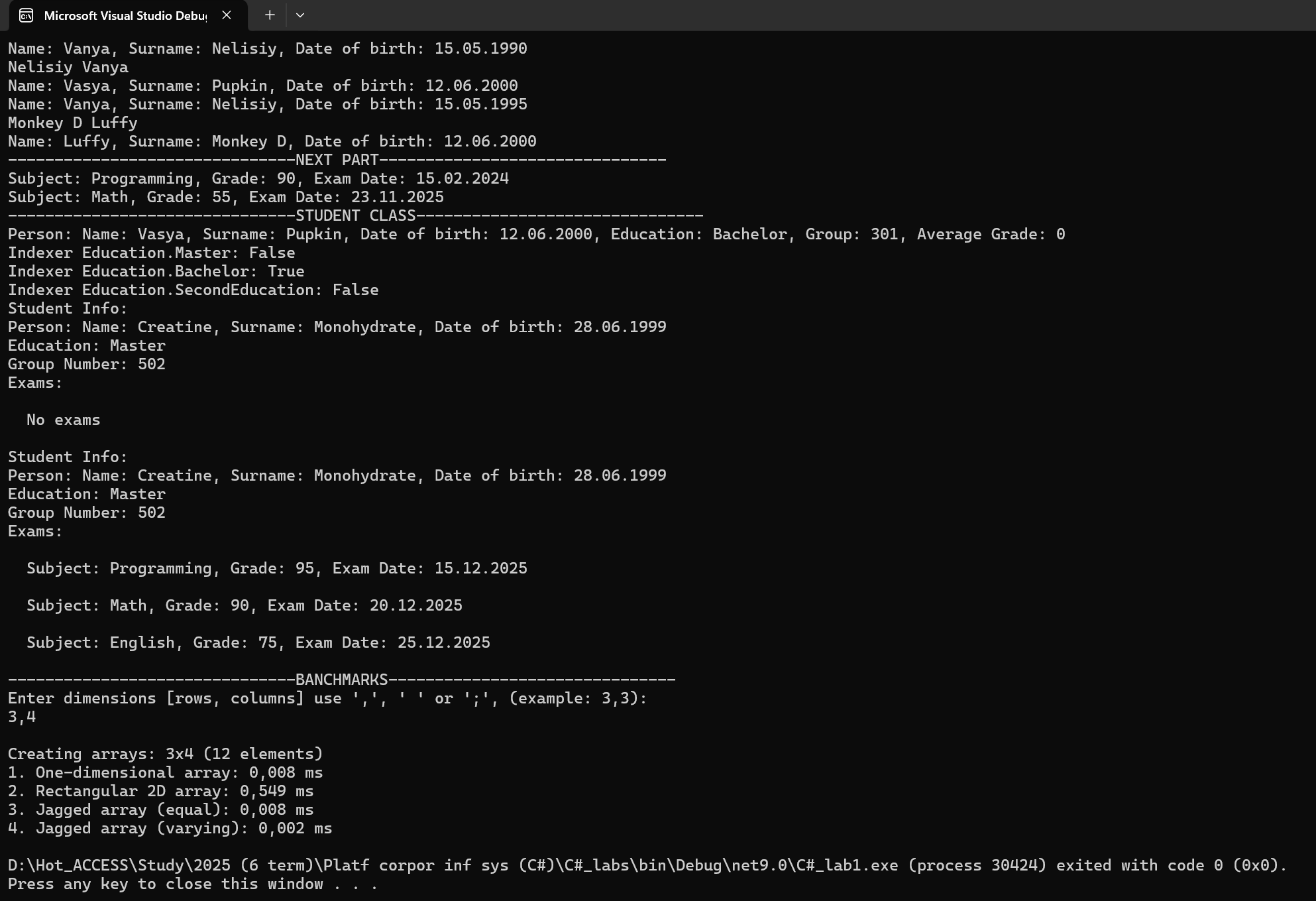
jaggedVarying[i][j] = new Exam { Grade = 100 };

sw4.Stop();

Console.WriteLine($"4. Jagged array (varying): {sw4.Elapsed.TotalMilliseconds:0.000} ms");

}

}

****