|  |  |
| --- | --- |
| **К Г Э У** | МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ»**  (ФГБОУ ВО «КГЭУ») |

**Отчёт по лабораторной работе № 7**

**«Потоки»**

**По дисциплине «Объектно-ориентированное программирование на языке C#»**

Выполнил: Хасаншин Д.Р.

Группа: ТРП-2-20

Проверил: Низямова А.Р.

Казань, 2021

**Вариант 18 % 3 + 1 = 1**

Классы, свойства, индексаторы. Одномерные, прямоугольные и ступенчатые массивы.

Определить класс **Person**, который имеет

* закрытое поле типа string, в котором хранится имя;
* закрытое поле типа string, в котором хранится фамилия;
* закрытое поле типа System.DateTime для даты рождения.

В классе **Person** определить конструкторы:

* конструктор c тремя параметрами типа string, string, DateTime для инициализации всех полей класса;
* конструктор без параметров, инициализирующий все поля класса некоторыми значениями по умолчанию.

В классе **Person** определить свойства c методами get и set:

* свойство типа string для доступа к полю с именем;
* свойство типа string для доступа к полю с фамилией;
* свойство типа DateTime для доступа к полю с датой рождения;
* свойство типа int c методами get и set для получения информации(get) и изменения (set) года рождения в закрытом поле типа DateTime, в котором хранится дата рождения.

В классе **Person** определить

* перегруженную(override) версию виртуального метода string ToString() для формирования строки со значениями всех полей класса;
* виртуальный метод string ToShortString(), который возвращает строку, содержащую только имя и фамилию.

Cравнить время, необходимое для выполнения операций с элементами одномерного, двумерного прямоугольного и двумерного ступенчатого массивов с одинаковым числом элементов.

Для этого в методе Main() создать

* одномерный массив;
* двумерный прямоугольный массив;
* двумерный ступенчатый массив.

Тип элементов массивов зависит от варианта лабораторной работы. Массивы должны иметь одинаковое число элементов. Если число строк в двумерном прямоугольном массиве равно nrow, а число столбцов ncolumn, то одномерный массив должен содержать nrow\*ncolumn элементов, в двумерном ступенчатом массиве общее число элементов также должно быть равно nrow\*ncolumn.

Значения nrow и ncolumn вводятся в процессе работы приложения в виде одной строки с разделителями. В приглашении, которое получает пользователь, должна быть информация о том, какие символы можно использовать как разделители, число разделителей должно быть больше 1. С помощью метода Split класса System.String приложение разбирает введенную пользователем текстовую строку с информацией о числе строк и числе столбцов двумерного массива и присваивает значения переменным, которые содержат значения nrow и ncolumn. В первой лабораторной работе не требуется обрабатывать ошибки ввода, предполагается, что пользователь правильно ввел данные.

Приложение распределяет память для всех массивов и инициализирует элементы массивов. Для инициализации элементов можно использовать конструктор без параметров.

Для всех элементов массивов выполняется одна и та же операция, например, присваивается значение одному из свойств, определенных для элементов массива. В лабораторной работе требуется сравнить время выполнения этой операции для одномерного, двумерного прямоугольного и двумерного ступенчатого массивов с одинаковым числом элементов.

Для измерения времени выполнения операций можно использовать свойство Environment.TickCount. Cтатическое свойство TickCount класса Environment имеет тип int, использует информацию системного таймера и содержит время в миллисекундах, которое прошло с момента перезагрузки компьютера.

Чтобы получить время выполнения некоторого блока кода, необходимо вызвать Environment.TickCount непосредственно перед блоком и сразу же после последнего оператора блока и взять разность значений.

В блоке кода, для которого измеряется время, не должно быть операций распределения памяти для массивов, инициализации элементов массивов и операций вывода данных на консоль. Блоки кода должны содержать только операции с элементами массива.

Вычисленные значения времени выполнения операций для трех типов массивов, а также число строк nrow и столбцов ncolumn выводятся на консоль. Вывод должен быть подписан, т.е. вывод должен содержать информацию о том, какому типу массива отвечает выведенное значение.

**Индивидульное задание.**

Определить тип **Education** - перечисление(enum)со значениями Specialist, Вachelor, SecondEducation.

Определить класс **Exam**, который имеет три открытых автореализуемых свойства, доступных для чтения и записи:

* свойство типа string, в котором хранится название предмета;
* свойство типа int, в котором хранится оценка;
* свойство типа System.DateTime для даты экзамена.

В классе **Exam** определить:

* конструктор с параметрами типа string, int и DateTime для инициализации всех свойств класса;
* конструктор без параметров, инициализирующий все свойства класса некоторыми значениями по умолчанию;
* перегруженную(override) версию виртуального метода string ToString() для формирования строки со значениями всех свойств класса.

Определить класс **Student**, который имеет

* закрытое поле типа Person, в котором хранятся данные студента;
* закрытое поле типа Education для информации о форме обучения;
* закрытое поле типа int для номера группы;
* закрытое поле типа Exam [] для информации об экзаменах, которые сдал студент.

В классе **Student** определить конструкторы:

* конструктор c параметрами типа Person, Education, int для инициализации соответствующих полей класса;
* конструктор без параметров, инициализирующий поля класса значениями по умолчанию.

В классе **Student** определить свойства c методами get и set:

* свойство типа Person для доступа к полю с данными студента;
* свойство типа Education для доступа к полю с формой обучения;
* свойство типа int для доступа к полю с номером группы;
* свойство типа Exam [] для доступа к полю со списком экзаменов.

В классе **Student** определить

* свойство типа double ( только с методом get), в котором вычисляется средний балл как среднее значение оценок в списке сданных экзаменов;
* индексатор булевского типа (только с методом get) с одним параметром типа Education; значение индексатора равно true, если значение поля с формой обучения студента совпадает со значением индекса, и false в противном случае;
* метод void AddExams ( params Exam [] ) для добавления элементов в список экзаменов;
* перегруженную версию виртуального метода string ToString() для формирования строки со значениями всех полей класса, включая список экзаменов;
* виртуальный метод string ToShortString(), который формирует строку со значениями всех полей класса без списка экзаменов, но со значением среднего балла.

В методе **Main()**

1. Создать один объект типа Student, преобразовать данные в текстовый вид с помощью метода ToShortString() и вывести данные.
2. Вывести значения индексатора для значений индекса Education.Specialist, Education.Bachelor и Education.SecondEducation.
3. Присвоить значения всем определенным в типе Student свойствам, преобразовать данные в текстовый вид с помощью метода ToString() и вывести данные.
4. C помощью метода AddExams( params Exam[] ) добавить элементы в список экзаменов и вывести данные объекта Student, используя метод ToString().
5. Сравнить время выполнения операций с элементами одномерного, двумерного прямоугольного и двумерного ступенчатого массивов с одинаковым числом элементов типа Exam.

Код:

Main:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace L7

{

class Program

{

enum Education

{

Specialist,

Bachelor,

SecondEducation

}

static void Main(string[] args)

{

// Обычная инициализация

Student student\_1 = new Student(new Person("Aaron", "Lewis", new DateTime(2000,12,20)),L7.Education.Specialist,1232);

student\_1.Exam = new Exam[]

{

new Exam("History",4,new DateTime(2021, 1, 10)),

new Exam("Geography", 3, new DateTime(2021, 1, 7))

};

// Инициализация обьектов класса

Student student\_0 = new Student()

{

Person = new Person

{

FirstName = "Rere",

LastName = "Gug",

Birthday = new DateTime(2000, 11, 17)

},

Group = 121231,

Exam = new Exam[]

{

new Exam

{

Subject = "Computer Science",

DateExam = new DateTime(2021, 1, 15),

},

new Exam

{

Subject = "Mathematics",

Grade = 5,

DateExam = new DateTime(2021, 1, 15),

}

}

};

Console.WriteLine(student\_0.ToShortString());

Console.WriteLine(student\_0.ToString());

Console.WriteLine(student\_1.ToString());

// значения индексатора

Console.WriteLine(student\_0[L7.Education.SecondEducation]);

Console.WriteLine(student\_0[L7.Education.Bachelor]);

Console.WriteLine(student\_0[L7.Education.Specialist]);

// добавление экзамена

student\_0.AddExam(new Exam("Duetch", 5, new DateTime(2021, 1, 20)), new Exam("Probability Theory", 4, new DateTime(2021,1,18)),

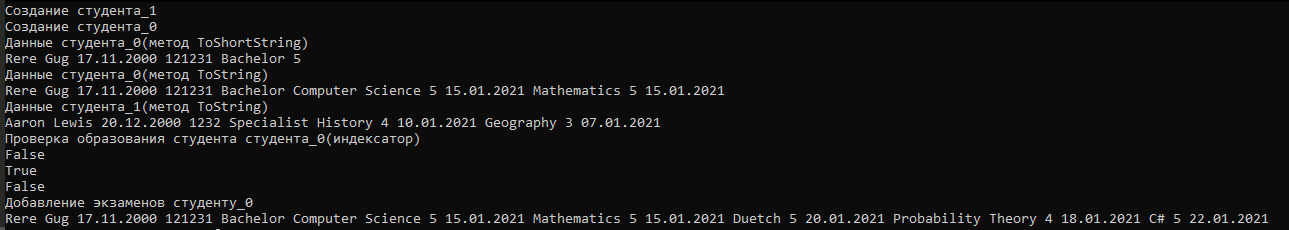
new Exam("C#",5, new DateTime(2021, 1, 22)));

Console.WriteLine(student\_0.ToString());

}

}

}



Student:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace L7

{

enum Education

{

Specialist,

Bachelor,

SecondEducation

}

class Student

{

// constructors

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

{

this.person = person;

this.education = education;

this.group = group;

}

public Student()

{

person = new Person();

education = Education.Bachelor;

group = 190222;

}

// properties

private Person person;

public Person Person

{

get { return person; }

set { person = value; }

}

private Education education;

public Education Education

{

get { return education; }

set { education = value; }

}

private int group; // №группы

public int Group

{

get { return group; }

set { group = value; }

}

private Exam[] exam;

public Exam[] Exam

{

get { return exam; }

set { exam = value; }

}

private double average\_grade;

public double AverageGrade

{

get

{

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

{

average\_grade += Exam[i].Grade;

}

return average\_grade/Exam.Length;

}

}

public bool this[Education education]

{

get

{

if (this.education == education) return true;

else return false;

}

}

//methods

public void AddExam(params Exam[] exam)

{

Exam[] newArray = new Exam[this.exam.Length + exam.Length];

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

{

newArray[i] = this.exam[i];

}

int j = this.exam.Length;

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

{

newArray[j] = exam[i];

j++;

}

Exam = newArray;

}

public override string ToString()

{

string s = $"{person.ToString()} {group} {education}";

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

{

s += $" {exam[i]}";

}

return s;

}

public virtual string ToShortString()

{

return $"{person.ToString()} {group} {education} {AverageGrade}";

}

}

}

Exam:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace L7

{

class Exam

{

// constructors

public Exam(string subject, int grade, DateTime date\_exam)

{

Subject = subject;

Grade = grade;

DateExam = date\_exam;

}

public Exam()

{

Subject = "Mathematics";

Grade = 5;

DateExam = new DateTime(2021,1,17);

}

// properties

public string Subject { get; set; }

public int Grade { get; set; }

public DateTime DateExam { get; set; }

// methods

public override string ToString()

{

return Subject + " " + Grade + " " + DateExam.ToShortDateString();

}

}

}

Person:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace L7

{

class Person

{

// Constructors

public Person(string first\_name, string last\_name, DateTime birthday)

{

\_first\_name = first\_name;

\_last\_name = last\_name;

\_birthday = birthday;

}

public Person()

{

\_first\_name = "Jacob";

\_last\_name = "Kelly";

\_birthday = new DateTime(2000,10,5);

}

// Properties

private string \_first\_name;

public string FirstName

{

get { return \_first\_name; }

set { \_first\_name = value; }

}

private string \_last\_name;

public string LastName

{

get { return \_last\_name; }

set { \_last\_name = value; }

}

private DateTime \_birthday;

public DateTime Birthday

{

get { return \_birthday; }

set { \_birthday = value; }

}

// Methods

public override string ToString()

{

string data = $"{\_first\_name} {\_last\_name} {\_birthday.ToShortDateString()}";

return data;

}

public virtual string ToShortString()

{

string data = $"{\_first\_name} {\_last\_name}";

return data;

}

}

}