**Лабораторная работа 1**

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

**Требования к программе, общие для всех вариантов**

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.

Сравнить 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.

**При работе в среде VisualStudio необходимо уметь**

* средствами Solution Explorer добавить в проект новый класс;
* с помощью диаграммы классов (Class Diagram) добавить в класс методы, поля и свойства;
* с помощью диаграммы классов добавить в класс перегруженную (override) версию виртуального метода.

**Вариант 2.**

**Требования к программе**

Определить тип **Frequency -** перечисление(enum) со значениями Weekly,  Monthly, Yearly.

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

* ∙ свойство типа Person, в котором хранятся данные автора статьи; ∙ свойство типа string для названия статьи;
* ∙ свойство типа double для рейтинга статьи.

В классе **Article** определить:

* ∙ конструктор c параметрами типа Person, string, double для инициализации  всех свойств класса;
* 10
* ∙ 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 **Magazine**, который имеет

* ∙ закрытое поле типа string c названием журнала;
* ∙ закрытое поле типа Frequency с информацией о периодичности выхода  журнала;
* ∙ закрытое поле типа DateTime c датой выхода журнала;
* ∙ закрытое поле типа int с тиражом журнала;
* ∙ закрытое поле типа Article\*+ со списком статей в журнале.

В классе **Magazine** определить конструкторы:

* ∙ конструктор с параметрами типа string, Frequency, DateTime, int для  инициализации соответствующих полей класса;
* ∙ конструктор без параметров, инициализирующий поля класса значениями  по умолчанию.

В классе **Magazine** определить свойства c методами get и set: ∙ свойство типа string для доступа к полю с названием журнала;  ∙ свойство типа Frequency для доступа к полю с информацией о  периодичности выхода журнала;

∙ свойство типа DateTime для доступа к полю c датой выхода журнала;

∙ свойство типа int для доступа к полю с тиражом журнала;

∙ свойство типа Article\*+ для доступа к полю со списком статей.

В классе **Magazine** определить

∙ свойство типа double ( только с методом get), в котором вычисляется  среднее значение рейтинга в списке статей;

∙ индексатор булевского типа (только с методом get) с одним параметром  типа Frequency; значение индексатора равно true, если значение поля типа  Frequency совпадает со значением индекса, и false в противном случае; ∙ метод void AddArticles (params Article[]) для добавления элементов в список статей в журнале;

* ∙перегруженную версию виртуального метода string ToString() для  формирования строки со значениями всех полей класса, включая список  статей;
* ∙ виртуальный метод string ToShortString(), который формирует строку со значениями всех полей класса без списка статей, но со значением среднего  рейтинга статей.

В методе **Main()**

1. Создать один объект типа Magazine, преобразовать данные в текстовый  вид с помощью метода ToShortString() и вывести данные.

2. Вывести значения индексатора для значений индекса Frequency.Weekly,  Frequency.Monthly и Frequency.Yearly.

3. Присвоить значения всем определенным в типе Magazine свойствам,  преобразовать данные в текстовый вид с помощью метода ToString() и  вывести данные.

4. C помощью метода AddArticles( params Article[] ) добавить элементы в  список статей и вывести данные объекта Magazine, используя метод  ToString().

5. Сравнить время выполнения операций с элементами одномерного,  двумерного прямоугольного и двумерного ступенчатого массивов с  одинаковым числом элементов типа Article.

**Код программы:**

1. Начало программы

class Program

{

const string SEPARATOR = "\n-----------------------------------------------------------------\n";

static void TestArraysRunTime<T>(T item, int nrow, int ncolumn)

{

/\* Одномерный массив \*/

T[] OneDimArray = new T[nrow \* ncolumn];

/\* Двумерный массив \*/

T[,] TwoDimArray = new T[nrow, ncolumn];

/\* Ступенчатый массив \*/

T[][] StepArray = new T[nrow][];

/\* Заполняем ступенчатый массив другими массивами \*/

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

StepArray[i] = new T[ncolumn];

int OneDimTimeStart = Environment.TickCount;

/\* Заполняем одномерный массив \*/

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

OneDimArray[i] = item;

int OneDimTimeEnd = Environment.TickCount;

Console.WriteLine($"Одномерный массив: {OneDimTimeEnd - OneDimTimeStart}");

int TwoDimTimeStart = Environment.TickCount;

/\* Заполняем двумерный массив \*/

for (int i = 0; i < TwoDimArray.GetLength(0); i++)

for (int j = 0; j < TwoDimArray.GetLength(1); j++)

TwoDimArray[i, j] = item;

int TwoDimTimeEnd = Environment.TickCount;

Console.WriteLine($"Двумерный массив: {TwoDimTimeEnd - TwoDimTimeStart}");

int StepTimeStart = Environment.TickCount;

/\* Заполняем ступенчатый массив \*/

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

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

StepArray[i][j] = item;

int StepTimeEnd = Environment.TickCount;

Console.WriteLine($"Ступенчатый массив: {StepTimeEnd - StepTimeStart}");

}

static void StartLab\_1(int testNrow = 100, int testNcolumn = 100)

{

var test = new Magazine();

/\*

\* 1) Создать один объект типа Magazine, преобразовать данные в текстовый

\* вид с помощью метода ToShortString() и вывести данные.

\*/

Console.WriteLine(test.ToShortString());

Console.WriteLine(SEPARATOR);

/\*

\* 2) Вывести значения индексатора для значений индекса Frequency.Weekly,

\* Frequency.Monthly и Frequency.Yearly.

\*/

Console.WriteLine(test[EnumFrequency.Weekly]);

Console.WriteLine(test[EnumFrequency.Monthly]);

Console.WriteLine(test[EnumFrequency.Yearly]);

Console.WriteLine(SEPARATOR);

/\*

\* 3) Присвоить значения всем определенным в типе Magazine свойствам,

\* преобразовать данные в текстовый вид с помощью метода ToString() и

\* вывести данные.

\*/

var articles = new Article[] { new Article(new Person(), "Тестовый журнал", 5) };

var test2 = new Magazine("Тестовый журнал", EnumFrequency.Weekly, new DateTime(), 200, articles);

Console.WriteLine(test2);

Console.WriteLine(SEPARATOR);

/\*

\* 4) C помощью метода AddArticles( params Article\*+ ) добавить элементы в

\* список статей и вывести данные объекта Magazine, используя метод

\* ToString().

\*/

test2.AddArticles(new Article(), new Article());

Console.WriteLine(test2);

Console.WriteLine(SEPARATOR);

/\*

\* 5) Сравнить время выполнения операций с элементами одномерного,

\* двумерного прямоугольного и двумерного ступенчатого массивов с

\* одинаковым числом элементов типа Article.

\*/

TestArraysRunTime(test2, testNrow, testNcolumn);

}

static void Main(string[] args)

{

StartLab\_1(10000, 10000);

Console.ReadLine();

}

1. Класс Person – необходим для описания человека

class Person

{

/\* Имя \*/

private string name;

/\* Фамилия \*/

private string secondName;

/\* Дата рождения \*/

private DateTime dateBirth;

/\*\* Имя \*/

public string Name

{

get { return name; }

}

/\* Фамилия \*/

public string SecondName

{

get { return secondName; }

}

/\* Дата рождения \*/

public DateTime DateBirth

{

get { return dateBirth; }

}

/\* Год рождения \*/

public int DateBirthYear

{

get { return dateBirth.Year; }

set { dateBirth.AddYears(value - dateBirth.Year); }

}

public Person(string name, string secondName, DateTime dateBirth)

{

this.name = name;

this.secondName = secondName;

this.dateBirth = dateBirth;

}

public Person()

{

name = "Егор";

secondName = "Дмитриев";

dateBirth = new DateTime(1999, 1, 1);

}

public override string ToString()

{

return $"Имя: {Name}\nФамилия: {SecondName}\nДата рождения: {DateBirth}";

}

public virtual string ToShortString()

{

return $"Имя: {Name}\nФамилия: {SecondName}";

}

}

1. Класс Article – необходим для описания статьи

class Article

{

/\* Автор статьи \*/

public Person Person { get; set; }

/\* Название статьи \*/

public string Name { get; set; }

/\* Рейтинг статьи \*/

public double Rating { get; set; }

public Article(Person person, string name, double rating)

{

Person = person;

Name = name;

Rating = rating;

}

public Article()

{

Person = new Person();

Name = "Тестовая статья";

Rating = 2.1;

}

public override string ToString()

{

return $"Название статьи: {Name}\nРейтинг статьи: {Rating}\nАвтор статьи: {Person}";

}

}

1. Класс Magazine – необходим для описания журнала

class Magazine

{

/\* Название журнала \*/

private string name;

/\* Переодичность выхода журнала \*/

private EnumFrequency qutputFrequency;

/\* Дата выхода журнала \*/

private DateTime releaseDate;

/\* Тираж журнала \*/

private int circulation;

/\* Статьи в журнале \*/

private Article[] articles;

/\* Название журнала \*/

public string Name

{

get { return name; }

set { name = value; }

}

/\* Переодичность выхода журнала \*/

public EnumFrequency QutputFrequency

{

get { return qutputFrequency; }

set { qutputFrequency = value; }

}

/\* Дата выхода журнала \*/

public DateTime ReleaseDate

{

get { return releaseDate; }

set { releaseDate = value; }

}

/\* Тираж журнала \*/

public int Circulation

{

get { return circulation; }

set { circulation = value; }

}

/\* Статьи в журнале \*/

public Article[] Articles

{

get { return articles; }

set { articles = value; }

}

/\* Cреднее значение рейтинга в списке статей \*/

public double AvgRatingOfArticles

{

get

{

return Articles.Sum(a => a.Rating) / Articles.Length;

}

}

public bool this[EnumFrequency frequency]

{

get => QutputFrequency == frequency;

}

/\* Получаем строковое представление частоты выхода журнала \*/

private string QutputFrequencyString

{

get

{

switch (qutputFrequency)

{

case EnumFrequency.Weekly:

return "Ежедневно";

case EnumFrequency.Monthly:

return "Ежемесячно";

case EnumFrequency.Yearly:

return "Ежегодно";

default:

return "";

}

}

}

/\* Получаем строку с информацией о статьях \*/

private string ArticlesInfoString

{

get { return articles.Select((article) => article.ToString()).Aggregate((current, next) => current + "\n\n" + next); }

}

public Magazine(string name, EnumFrequency qutputFrequency, DateTime releaseDate, int circulation, Article[] articles)

{

this.name = name;

this.qutputFrequency = qutputFrequency;

this.releaseDate = releaseDate;

this.circulation = circulation;

this.articles = articles;

}

public Magazine()

{

name = "Тестовый журнал";

qutputFrequency = EnumFrequency.Monthly;

releaseDate = DateTime.MinValue;

circulation = 0;

articles = new Article[2] { new Article(), new Article() };

}

public void AddArticles(params Article[] newArticles)

{

articles = articles.Concat(newArticles).ToArray();

}

public override string ToString()

{

return $"Название журнала: {name}\n" +

$"Периодичность выхода журнала: {QutputFrequencyString}\n" +

$"Дата выхода журнала: {releaseDate}\n" +

$"Тираж: {circulation}\n\n" +

$"Статьи:\n" +

$"{ArticlesInfoString}";

}

public virtual string ToShortString()

{

return $"Название журнала: {name}\n" +

$"Периодичность выхода журнала: {QutputFrequencyString}\n" +

$"Дата выхода журнала: {releaseDate}\n" +

$"Тираж: {circulation}\n" +

$"Cреднее значение рейтинга статей: {AvgRatingOfArticles}";

}

}

1. Перечисление EnumFrequency – периодичность выхода журнала

/\* Периодичность \*/

enum EnumFrequency

{

Weekly = 0,

Monthly = 1,

Yearly = 2,

}

Консольный вывод программы:

