Министерство образования и науки РФ

Томский государственный университет
систем управления и радиоэлектроники (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

Романенко В.В.

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Учебно-методическое пособие по выполнению
лабораторных работ

Томск – 2016

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА №1

Разработанная в результате выполнения данной лабораторной работы программа должна работать по следующему алгоритму:

1. При запуске программы считывается файл данных, с которым программа работала в предыдущем сеансе. Имя файла данных читается из пользовательских параметров приложения. По умолчанию файл данных имеет имя «Default» или «Autosave», расширение зависит от типа файла. Если требуемый файл отсутствует, то программа начинает работу с пустым набором данных.

2. Далее пользователю предлагается выбрать дальнейший вариант действий:

2.1. Добавить новую запись в набор данных. Поля записи (структуры данных) зависят от варианта задания. Набор данных представляет собой массив записей заданного типа, новая запись добавляется в конец массива. Чтобы не требовалось постоянное динамическое выделение памяти для массива, следует заранее выделить память для большого количества записей (например, 1000). Можно использовать классы, реализующие функциональность динамических списков (типа List и его универсальных аналогов).

2.2. Удалить запись из набора данных. Запись удаляется по номеру или другому ключу. При этом все записи, расположенные в наборе данных после удаляемой записи, перемещаются на одну позицию вверх (если они хранится в виде массива).

2.3. Очистить набор данных. После этой операции набор данных считается пустым.

2.4. Вывод на консоль всех записей из набора данных. Параметры вывода на экран берутся из пользовательских параметров приложения. Параметры вывода по умолчанию определить самостоятельно.

2.5. Параметры вывода записей на экран. Здесь предлагается выбрать формат вывода полей записей на консоль. Предлагается три варианта – формат по умолчанию, специальный формат или шаблон пользователя. В первом случае используется обычное преобразование поля в строку. Во втором случае пользователю предлагается ввести спецификатор формата для каждого поля записи. Соответственно, в третьем случае для каждого поля записи (кроме строковых полей) вводится шаблон пользователя. Выбранные пара-

метры сохраняются в пользовательских параметрах приложения.

2.6. Загрузить файл данных. При этом программа отображает имеющиеся в текущем каталоге (в которой расположен исполняемый файл приложения) файлы данных (при желании можно реализовать функциональность изменения текущего каталога). После загрузки файла его имя становится текущим для программы.

2.7. Сохранить файл. При этом набор данных сохраняется в текущем файле.

2.8. Сохранить файл как... При этом отображаются имеющиеся в текущем каталоге файлы данных, а пользователь вводит новое имя файла. Если файл с таким именем уже существует, выдается предупреждение. Программа должна иметь три возможных варианта сохранения данных в файл: сохранение в виде текстового файла, а также сериализация в двоичном формате и формате SOAP. В первом случае в файл в «ручном» режиме сначала сохраняется количество записей в наборе, а затем поочередно все поля всех записей. В остальных случаях используется механизм сериализации. Выбор типа файла можно организовать в виде небольшого подменю, либо определять его по введенному расширению. Расширения для файлов данных можно использовать любые, например – TXT, BIN и XML соответственно.

2.9. Выход. Обеспечивает выход из программы, во всех остальных случаях происходит возврат к выбору действия.

3. При выходе из программы имеющийся набор данных сохраняется в файл с текущим типом и именем.

Варианты 00-19

Запись содержит анкетные данные студента:

1. Фамилия, имя, отчество (строка);
2. Пол (перечисление);
3. Номер группы (строка);
4. Номер зачётной книжки (целое число);
5. Дата рождения (дата);
6. Проживание в общежитии (булево значение).

Варианты 20-39

Имеется запись справочника телефонной компании:
1. Фамилия, имя, отчество абонента (строка);
2. Тип телефона – домашний, рабочий, сотовый и т.п. (перечисление);
3. Адрес (строка);
4. Номер подключен (булево значение);
5. Дата подключения (дата);
6. Абонентская плата (число с плавающей точкой).

Варианты 40-59

Запись содержит данные о работниках из отдела кадров предприятия:
1. Фамилия, имя, отчество сотрудника (строка);
2. Отдел (строка);
3. Должность – руководящая, не руководящая и т.п. (перечисление);
4. Возраст (целое число);
5. Дата трудоустройства (дата);
6. Продолжительность рабочего дня (время).

Варианты 60-79

Запись содержит данные из библиотечной карты:
1. Название книги (строка);
2. Автор (строка);
3. Издание – город, издательство (строка);
4. Год издания (целое число);
5. Дата поступления (дата);
6. Наличие (булево значение).

Варианты 80-99

Запись содержит сведения о товаре в магазине:
1. Код товара (целое число);
2. Наименование (строка);
3. Цена (число с плавающей точкой);
4. Количество (целое число);

5. Единица измерения – килограмм, литр, упаковка, бутылка и т.п. (перечисление);

6. Срок годности (дата).

ЛАБОРАТОРНАЯ РАБОТА №2

Варианты 00-33

Составить описание класса «Вектор» для объектов-векторов, задаваемых одномерным массивом вещественных чисел типа **double**. Компоненты вектора (x_1, x_2, …, x_n) должны быть инкапсулированы в классе.

I. Предусмотреть применение конструкторов:

а) по умолчанию (создающий пустой вектор);

б) для инициализации вектора заданного размера;

в) для инициализации вектора с заданными в виде одномерного массива компонентами.

г) для копирования одного вектора в другой.

Организовать в конструкторах и деструкторе вывод на экран информационных сообщений, например, «Конструктор вектора XXX», «Деструктор вектора XXX» и т.д. Вместо «XXX» указывать некоторый уникальный идентификатор вектора.

II. С помощью методов класса обеспечить:

1) вычисление модуля (длины, или нормы) вектора ||A||;

2) нормировку вектора (получение вектора единичной длины – A/||A||);

3) поиск максимального элемента вектора;

4) поиск минимального элемента вектора.

III. С помощью перегруженных операторов класса обеспечить операции сложения, вычитания и скалярного умножения векторов (A + B, A – B и (A, B)), а также умножения и деления вектора на скаляр (A*k, k*A и A/k). Выполнению операций сложения, вычитания и скалярного умножения векторов должна предшествовать проверка возможности их выполнения над данными объектами.

IV. С помощью статических методов обеспечить:

1) вычисление синуса угла между двумя векторами;

2) вычисление косинуса угла между двумя векторами;

3) вычисление величины угла в радианах между двумя векторами.

Для определения синуса и косинуса угла между векторами можно использовать соотношение (A, B) = ||A||×||B||×cos(α), для вычисления величины угла – функцию Atan2 (tg(α) = sin(α)/cos(α), поэтому α = Atan2(sin(α), cos(α))).

V. С помощью индексатора обеспечить доступ к элементам вектора по

индексу (чтение/запись). С помощью свойства – доступ к количеству элементов (только чтение).

VI. Перегрузить метод ToString для представления вектора, заключенного в скобки (любой формы), в виде строки.

При невозможности выполнения над вектором тех или иных операций генерировать исключение (типа ArgumentException или других типов, в зависимости от операции).

Варианты 34-66

Составить описание класса «Матрица» для объектов прямоугольных матриц, задаваемых прямоугольным массивом вещественных чисел типа **double**. Компоненты матрицы должны быть инкапсулированы в классе.

I. Предусмотреть применение конструкторов:

а) по умолчанию (создающий пустую матрицу);

б) для инициализации квадратной матрицы заданного размера;

в) для инициализации прямоугольной матрицы заданных размеров;

г) для инициализации матрицы с заданными в виде прямоугольного двумерного массива компонентами.

д) для копирования одной матрицы в другую.

Организовать в конструкторах и деструкторе вывод на экран информационных сообщений, например, «Конструктор матрицы XXX», «Деструктор матрицы XXX» и т.д. Вместо «XXX» указывать некоторый уникальный идентификатор матрицы.

II. С помощью методов класса обеспечить:

1) проверку возможности умножения двух матриц;

2) проверку возможности сложения двух матриц;

3) поиск максимального элемента матрицы;

4) поиск минимального элемента матрицы.

III. С помощью перегруженных операторов класса обеспечить операции сложения, вычитания и умножения матриц, а также умножения матрицы на скаляр. Выполнению операций сложения, вычитания и умножения матриц должна предшествовать проверка возможности их выполнения над данными объектами.

IV. С помощью индексатора обеспечить доступ к элементам матрицы по индексу строки и столбца (чтение/запись). С помощью свойств – доступ к

количеству строк и столбцов (только чтение).

V. Перегрузить метод ToString для представления матрицы в построчной форме в виде строки. Использовать форматирование, чтобы элементы одного столбца матрицы располагались друг под другом.

При невозможности выполнения над матрицей тех или иных операций генерировать исключение (типа ArgumentException или других типов, в зависимости от операции).

Варианты 67-99

Составить описание класса «Полином» для объектов-полиномов, задаваемых одномерным массивом коэффициентов – вещественных чисел типа **double**. Коэффициенты полинома степени n (a_0, a_1, a_2, …, a_n) должны быть инкапсулированы в классе. Полином всегда содержит, как минимум, один коэффициент – a_0.

I. Предусмотреть применение конструкторов:

а) по умолчанию (создающий полином нулевой степени с единственным коэффициентом, равным 0);

б) для инициализации полинома заданной степени;

в) для инициализации полинома с заданными в виде одномерного массива коэффициентами.

г) для копирования одного полинома в другой.

Организовать в конструкторах и деструкторе вывод на экран информационных сообщений, например, «Конструктор полинома XXX», «Деструктор полинома XXX» и т.д. Вместо «XXX» указывать некоторый уникальный идентификатор полинома.

II. Предусмотреть свойство типа **bool**, определяющее, будут ли автоматически при совершении любых операций с полиномом отбрасываться старшие члены с нулевыми коэффициентами. Отбрасывание коэффициентов реализовать в отдельном открытом (**public**) методе.

III. С помощью перегруженных операторов класса обеспечить операции сложения, вычитания, умножения, деления и остатка от деления полиномов. Деление полиномов выполняется по алгоритму Евклида.

IV. С помощью индексатора обеспечить доступ к коэффициентам полинома по индексу (чтение/запись). С помощью свойства – доступ к степени полинома (только чтение).

V. Перегрузить метод ToString для представления полинома в виде строки в удобной форме:

$$a_n x\text{^}n \pm |a_{n-1}|x\text{^}(n-1) \pm \ldots \pm |a_2|x\text{^}2 \pm |a_1|x \pm |a_0|,$$

причем члены с нулевыми коэффициентами выводить не нужно.

ЛАБОРАТОРНАЯ РАБОТА №3

Варианты 00-33

Общее задание:

• Реализовать в классе интерфейс ICloneable.

• Реализовать в классе интерфейс IComparable. Критерий сравнения – норма вектора. Продемонстрировать применение этого интерфейса, сортируя массив векторов методом Array.Sort.

• Описать в классе «Вектор» события, сигнализирующие об изменении размеров или компонентов вектора.

Дополнительное задание по вариантам:

00) Обеспечить представление вектора в виде координат точки начала и точки конца.

01) Реализовать еще одну версию объекта типа «Вектор» в виде структуры. В комментариях пояснить, какие пришлось внести модификации в члены структуры по сравнению с членами класса.

02) Перегрузить для векторов операторы «==» и «!=», а также метод Equals. Сравнение векторов проводить поэлементно.

03) Определить оператор неявного преобразования вектора к типу **double**, результатом которого будет норма вектора, а также преобразования значения типа **double** к вектору, результатом которого будет заполнение вектора указанным значением.

04) Добиться того, чтобы оператор «&&» объединял векторы. Результатом операции «`x && y`» должен быть вектор, в котором компоненты вектора `y` располагаются правее компонентов вектора `x`.

05) Добиться того, чтобы оператор «||» объединял векторы. Результатом операции «`x || y`» должен быть вектор, в котором компоненты вектора `x` чередуются с компонентами вектора `y`.

06) Добиться того, чтобы оператор «>>» циклически сдвигал элементы вектора указанное количество раз вправо, а оператор «<<» – влево.

07) Добиться того, чтобы оператор «>>» уменьшал длину вектора на 10^i, а оператор «<<» – увеличивал ее на 10^i, где i – величина сдвига.

08) Обеспечить возможность сложения, вычитания и деления векторов с операндами типа **double** и результатом типа **double**, допустимых в том слу-

чае, если вектор состоит из единственного элемента.

09) Перегрузить для вектора операторы **true**, **false** и неявного преобразования вектора к типу **bool**. Будем считать, что вектор = «ложь», если он пуст или его длина равна нулю, и «истина» в противном случае.

10) Написать алгоритм сортировки элементов вектора методом пузырька. При этом выполнять сравнение элементов должен делегат, передаваемый в этот метод в качестве параметра. Разные делегаты должны обеспечивать разные методы сортировки, например: по возрастанию; по убыванию; сначала четные элементы, а затем нечетные или наоборот; сначала отрицательные элементы, а потом положительные и наоборот; и т.д.

11) Обеспечить поиск требуемого элемента в векторе. Критерий поиска должен задаваться в виде делегата, передаваемого в этот метод в качестве параметра. Например, поиск минимального или максимального элемента, первого или последнего отрицательного или положительного элемента и т.д.

12) Реализовать в классе интерфейс IEnumerable, позволяющий использовать вектор в качестве итератора, например, для извлечения его элементов в цикле **foreach**.

13) Реализовать в классе метод Copy, возвращающий копию вектора и виртуальный метод Assign, принимающий аргумент типа **object**. Если данный аргумент содержит ссылку на вектор, скопировать его в текущий экземпляр вектора.

14) Перегрузить в векторе операторы отношения. Сравнение векторов осуществлять на основании их длины.

15) Реализовать в классе метод, вычисляющий проекцию вектора на другой вектор, передаваемый в качестве параметра.

16) Избавиться от хранения одинаковых копий векторов. Для этого реализовать класс-регистратор, хранящий ссылки на все имеющиеся векторы. Прямой вызов конструкторов векторов запретить, вместо этого реализовать метод CreateInstance, возвращающий новый вектор, если он уникален, и ссылку на имеющийся вектор в противном случае. Экземпляр класса-регистратора создавать в статическом конструкторе вектора.

17) Используя оператор **yield**, реализовать в классе итератор для перечисления всех элементов вектора.

18) Добавить методы, позволяющие удалить из вектора отдельный элемент или вставить элемент в вектор.

19) Перегрузить для вектора унарные операторы «+» и «–» с сохранением их математического смысла.

20) Реализовать в классе интерфейс IList. Некоторые методы данного интерфейса можно сделать в виде заглушек (генерировать в них исключение NotSupportedException) – например, Insert, Remove и т.д.

21) Реализовать в классе интерфейс ICollection для возможности интерпретации вектора как коллекции.

22) Реализовать в классе интерфейс IFormattable для форматирования вывода элементов вектора на экран.

23) Добавить в класс методы Split и Join. Первый должен возвращать два вектора, являющиеся частями исходного вектора (разделяя его по элементу с указанным индексом). Второй метод должен реализовывать обратную операцию – соединять два вектора в один.

24) Перегрузить для вектора унарные операторы «++» и «—», увеличивающие и уменьшающие количество его компонентов на 1. При увеличении количества элементов новый элемент должен быть равен нулю. Остальные элементы должны оставаться без изменений.

25) Разрешить доступ к свойству, соответствующему количеству элементов в векторе, на запись. При этом, если количество элементов вектора увеличивается, то новые элементы должны быть равны нулю. Остальные элементы должны оставаться без изменений.

26) Изменить тип элементов вектора на обнуляемый тип **double**?. При выполнении всех операций с неинициализированными элементами вектора (со значением **null**) должна генерироваться исключительная ситуация типа NullReferenceException.

27) Обеспечить операторы преобразования вектора к типу **double**[], и наоборот – от типа **double**[] к вектору.

28) Написать метод Init для инициализации элементов вектора требуемыми значениями. Способ инициализации должен быть представлен делегатом, передаваемым в этот метод в качестве параметра. Написать несколько предопределенных инициализаторов (для обнуления вектора, для получения единичного вектора и т.п.).

29) Написать метод с переменным числом аргументов для сложения произвольного количества векторов с текущим вектором и помещением результата в текущий вектор, а также аналогичный метод для вычитания.

30) Добавить в конструктор вектора заданного размера возможность передачи произвольного количества элементов типа **double** для инициализации элементов вектора.

31) Создать методы расширения в отдельном классе. Первый должен заполнять одномерный массив элементами вектора, второй – элементами вектора, начиная с указанного индекса. Если размер массива больше размеров копируемой части вектора, то некоторые элементы останутся неинициализированными, а если меньше, то лишние элементы вектора должны быть отброшены.

32) Написать интерфейс IVector, определяющий свойства, возвращающие размер вектора и индексатор для доступа к элементам вектора произвольного типа. Реализовать этот интерфейс в классе вектора.

33) Написать метод Process для выполнения над элементами вектора требуемых преобразований. Вид преобразований должен быть представлен делегатом, передаваемым в этот метод в качестве параметра. Написать несколько предопределенных преобразований (для изменения знака вектора, умножения или деления на константу и т.п.).

Варианты 34-66

Общее задание:

• Реализовать в классе интерфейс ICloneable.

• Реализовать в классе интерфейс IComparable. Критерий сравнения – норма матрицы. Продемонстрировать применение этого интерфейса, сортируя массив матриц методом Array.Sort.

• Описать в классе «Матрица» события, сигнализирующие об изменении размеров или компонентов матрицы.

Дополнительное задание по вариантам:

34) Обеспечить хранение элементов матрицы в классе в виде одномерного массива. Извне класса пользователь должен иметь возможность работать с ним как с двумерным объектом.

35) Реализовать еще одну версию объекта типа «Матрица» в виде структуры. В комментариях пояснить, какие пришлось внести модификации в члены структуры по сравнению с членами класса.

36) Перегрузить для матриц операторы «==» и «!=», а также метод Equals. Сравнение матриц проводить поэлементно.

37) Добавить рекурсивный метод поиска определителя квадратной матрицы методом разложения по строке. Определить оператор неявного преобразования матрицы к типу **double**, результатом которого будет значение определителя матрицы.

38) Добиться того, чтобы оператор «`&&`» объединял матрицы. Причем операция «`x && y`» должна объединять матрицы, имеющие одинаковое количество столбцов таким образом, чтобы в результирующей матрице строки матрицы `y` располагались ниже строк матрицы `x`.

39) Добиться того, чтобы оператор «`||`» объединял матрицы. Причем операция «`x || y`» должна объединять матрицы, имеющие одинаковое количество строк таким образом, чтобы в результирующей матрице столбцы матрицы `y` располагались правее столбцов матрицы `x`.

40) Добиться того, чтобы оператор «>>» циклически сдвигал столбцы матрицы указанное количество раз вправо, а оператор «<<» – влево.

41) Добиться того, чтобы оператор «>>» циклически сдвигал строки матрицы указанное количество раз вниз, а оператор «<<» – вверх.

42) Обеспечить возможность сложения, вычитания и деления матриц с операндами типа **double** и результатом типа **double**, допустимых в том случае, если матрица состоит из единственного элемента, а также деления произвольной матрицы на операнд типа **double**.

43) Перегрузить для матрицы операторы **true**, **false** и неявного преобразования матрицы к типу **bool**. Будем считать, что матрица = «ложь», если она пуста или имеет только нулевые коэффициенты. Также перегрузить оператор явного преобразования из типа **bool** к матрице. При этом, если матрице присваивается значение **false**, она должна обнуляться, а если **true** – становиться единичной.

44) Написать алгоритм построчной сортировки элементов матрицы методом пузырька. При этом выполнять сравнение элементов должен делегат, передаваемый в этот метод в качестве параметра. Разные делегаты должны обеспечивать разные методы сортировки, например: по возрастанию; по убыванию; сначала четные элементы, а затем нечетные или наоборот; сначала отрицательные элементы, а потом положительные и наоборот; и т.д.

45) Обеспечить поиск требуемого элемента в матрице. Критерий поиска должен задаваться в виде делегата, передаваемого в этот метод в качестве параметра. Например, поиск минимального или максимального элемента,

первого или последнего отрицательного или положительного элемента и т.д.

46) Реализовать в классе интерфейс IEnumerable, позволяющий использовать матрицу в качестве итератора, например, для извлечения ее элементов в цикле **foreach**. Добавить в класс свойство типа **object**, которое, если оно не равно **null**, должно возвращаться итератором в конце каждой строки элементов. Например, это может быть "\r\n" для вывода каждой строки матрицы на отдельной строке консольного окна.

47) Реализовать в классе метод Copy, возвращающий копию матрицы и виртуальный метод Assign, принимающий аргумент типа **object**. Если данный аргумент содержит ссылку на матрицу, скопировать ее в текущий экземпляр матрицы.

48) Перегрузить в матрице операторы отношения. Сравнение матриц (в т.ч. при реализации интерфейса IComparable) осуществлять на основании количества элементов.

49) Перегрузить в матрице операторы отношения. Сравнение матриц осуществлять на основании их норм.

50) Избавиться от хранения одинаковых копий матриц. Для этого реализовать класс-регистратор, хранящий ссылки на все имеющиеся матрицы. Прямой вызов конструкторов матриц запретить, вместо этого реализовать метод CreateInstance, возвращающий новую матрицу, если она уникальна, и ссылку на имеющуюся матрицу в противном случае. Экземпляр класса-регистратора создавать в статическом конструкторе матрицы.

51) Используя оператор **yield**, реализовать в классе итератор для перечисления всех элементов матрицы. Параметр типа **object**, передаваемый в итератор, если он не равен **null**, должен возвращаться итератором в конце каждой строки элементов. Например, это может быть "\r\n" для вывода каждой строки матрицы на отдельной строке консольного окна.

52) Используя оператор **yield**, реализовать в классе итератор для перечисления элементов матрицы, удовлетворяющих требуемому условию. Условие должно передаваться в итератор в виде параметра, имеющего тип делегата. Предусмотреть методы для извлечения положительных, отрицательных, нулевых и т.п. элементов матриц.

53) Реализовать транспонирование матрицы в виде перегрузки какого-либо унарного оператора.

54) Реализовать в классе интерфейс IList. Некоторые методы данного

интерфейса можно сделать в виде заглушек (генерировать в них исключение NotSupportedException) – например, Insert, Remove и т.д.

55) Реализовать в классе интерфейс ICollection для возможности интерпретации матрицы как коллекции.

56) Реализовать в классе интерфейс IFormattable для форматирования вывода элементов матрицы на экран.

57) Добавить в класс методы Split и Join. Первый должен возвращать две матрицы, являющиеся частями исходной матрицы (разделяя ее по указанной строке или столбцу). Второй метод должен реализовывать обратную операцию – соединять две матрицы построчно или по столбцам, если их размеры соответствуют.

58) Обеспечить хранение элементов матрицы в ортогональном массиве с возможностью задавать различное количество элементов для каждой строки.

59) Добавить в класс еще один индексатор с одним целочисленным индексом, обеспечивающий доступ к строкам матрицы (на чтение и запись).

60) Изменить тип элементов матрицы на обнуляемый тип **double**?. При выполнении всех операций с неинициализированными элементами матрицы (со значением **null**) должна генерироваться исключительная ситуация типа NullReferenceException.

61) Обеспечить операторы преобразования матрицы к типу **double**[] (при этом элементы матрицы должны располагаться в результирующем массиве построчно), и наоборот – от типа **double**[] к матрице.

62) Написать метод Init для инициализации элементов матрицы требуемыми значениями. Способ инициализации должен быть представлен делегатом, передаваемым в этот метод в качестве параметра. Написать несколько предопределенных инициализаторов (для обнуления матрицы, для получения единичной матрицы и т.п.).

63) Написать метод с переменным числом аргументов для сложения произвольного количества матриц с текущей матрицей и помещением результата в текущую матрицу, а также аналогичный метод для вычитания.

64) Добавить в конструкторы квадратных и прямоугольных матриц возможность передачи произвольного количества элементов типа **double** для инициализации элементов матрицы.

65) Создать методы расширения в отдельном классе. Первый должен

заполнять прямоугольный массив элементами матрицы, второй – элементами матрицы, начиная с указанной строки и столбца. Если размер массива больше размеров копируемой части матрицы, то некоторые элементы останутся неинициализированными, а если меньше, то лишние элементы матрицы должны быть отброшены.

66) Написать интерфейс IMatrix, определяющий свойства, возвращающие размеры матрицы и индексатор для доступа к элементам матрицы произвольного типа. Реализовать этот интерфейс в классе матрицы.

Варианты 67-99

Общее задание:

• Реализовать в классе интерфейс ICloneable.

• Реализовать в классе интерфейс IComparable. Критерий сравнения – порядок полинома. Если сравниваемые полиномы имеют одинаковый порядок, сравниваются абсолютные значения их коэффициентов, начиная со старших. Продемонстрировать применение этого интерфейса, сортируя массив векторов методом Array.Sort.

• Описать в классе «Полином» события, сигнализирующие об изменении размеров или коэффициентов полинома.

Дополнительное задание по вариантам:

67) Написать метод Process для выполнения над коэффициентами полинома требуемых преобразований. Вид преобразований должен быть представлен делегатом, передаваемым в этот метод в качестве параметра. Написать несколько предопределенных преобразований (для изменения знака коэффициентов, умножения или деления на константу и т.п.).

68) Реализовать еще одну версию объекта типа «Полином» в виде структуры. В комментариях пояснить, какие пришлось внести модификации в члены структуры по сравнению с членами класса.

69) Перегрузить для полиномов операторы «==» и «!=», а также метод Equals. Сравнение полиномов проводить поэлементно.

70) Определить в классе полинома еще один индексатор с параметром типа **double**, вычисляющий значение полинома в указанной точке.

71) Добиться того, чтобы оператор «&&» объединял полиномы. Результатом операции «x && y» должен быть полином, в котором коэффициенты полинома y располагаются правее коэффициентов полинома x.

72) Обеспечить возможность сложения и вычитания полиномов с операндами типа **double** и результатом типа полинома.

73) Добавить в класс метод, позволяющий брать производную любого порядка от полинома.

74) Добиться того, чтобы оператор «>>» понижал степень полинома на i, а оператор «<<» – увеличивал ее на i, где i – величина сдвига.

75) Обеспечить возможность сложения, вычитания и деления полиномов с операндами типа **double** и результатом типа **double**, допустимых в том случае, если полином состоит из единственного коэффициента.

76) Перегрузить для полинома операторы **true**, **false** и неявного преобразования полинома к типу **bool**. Будем считать, что полином = «ложь», если все его коэффициенты равны нулю, и «истина» в противном случае.

77) Добавить в класс метод, позволяющий найти первообразную полинома, а также метод, использующий первообразную для вычисления определенного интеграла полинома на указанном отрезке [a, b].

78) Обеспечить поиск требуемого коэффициента полинома. Критерий поиска должен задаваться в виде делегата, передаваемого в этот метод в качестве параметра. Например, поиск минимального или максимального коэффициента, минимального или максимального коэффициента по абсолютному значению, первого или последнего отрицательного, положительного или нулевого коэффициента и т.д.

79) Реализовать в классе интерфейс IEnumerable, позволяющий использовать полином в качестве итератора, например, для извлечения его коэффициентов в цикле **foreach**.

80) Реализовать в классе метод Copy, возвращающий копию полинома и виртуальный метод Assign, принимающий аргумент типа **object**. Если данный аргумент содержит ссылку на полином, скопировать его в текущий экземпляр полинома.

81) Перегрузить в полиноме операторы отношения. Сравнение полиномов осуществлять по тому же критерию, что и в реализации интерфейса IComparable.

82) Реализовать в классе метод, формирующий полином степени n на основании значений n его корней.

83) Избавиться от хранения одинаковых копий полиномов. Для этого реализовать класс-регистратор, хранящий ссылки на все имеющиеся полино-

мы. Прямой вызов конструкторов полиномов запретить, вместо этого реализовать метод CreateInstance, возвращающий новый полином, если он уникален, и ссылку на имеющийся полином в противном случае. Экземпляр класса-регистратора создавать в статическом конструкторе полинома.

84) Используя оператор **yield**, реализовать в классе итератор для перечисления всех коэффициентов полинома.

85) Реализовать в классе проверку типа полинома – является ли он четным, нечетным, или ни тем, ни другим. Результат возвращать в виде константы перечисления. Результат должен быть получен на основе анализа коэффициентов полинома.

86) Перегрузить для полинома унарные операторы «+» и «–» с сохранением их математического смысла.

87) Реализовать в классе интерфейс IList. Некоторые методы данного интерфейса можно сделать в виде заглушек (генерировать в них исключение NotSupportedException) – например, Insert, Remove и т.д.

88) Реализовать в классе интерфейс ICollection для возможности интерпретации полинома как коллекции.

89) Реализовать в классе интерфейс IFormattable для форматирования вывода коэффициентов полинома на экран.

90) Добавить в класс методы Split и Join. Первый должен возвращать два полинома, являющиеся частями исходного полинома (разделяя его по коэффициенту с указанной степенью). Второй метод должен реализовывать обратную операцию – соединять два полинома в один.

91) Перегрузить для полинома унарные операторы «++» и «—», увеличивающие и уменьшающие его степень на 1. При увеличении степени младший коэффициент должен стать равным нулю. Остальные коэффициенты должны остаться без изменений.

92) Разрешить доступ к свойству, соответствующему степени полинома, на запись. При этом, если степень полинома увеличивается, то новые коэффициенты должны быть равны нулю. Остальные коэффициенты должны остаться без изменений.

93) Изменить тип коэффициентов полинома на обнуляемый тип **double**?. При выполнении всех операций с неинициализированными коэффициентами полинома (со значением **null**) должна генерироваться исключительная ситуация типа NullReferenceException.

94) Обеспечить операторы преобразования полинома к типу **double**[], и наоборот – от типа **double**[] к полиному.

95) Написать метод Init для инициализации элементов полинома требуемыми значениями. Способ инициализации должен быть представлен делегатом, передаваемым в этот метод в качестве параметра. Написать несколько предопределенных инициализаторов (для обнуления полинома, для получения коэффициентов полинома $(x + c)^n$ и т.п.).

96) Написать метод с переменным числом аргументов для сложения произвольного количества полиномов с текущим полиномом и помещением результата в текущий полином, а также аналогичный метод для вычитания.

97) Добавить в конструктор полинома заданной степени возможность передачи произвольного количества элементов типа **double** для инициализации коэффициентов полинома.

98) Создать методы расширения в отдельном классе. Первый должен заполнять одномерный массив коэффициентами полинома, второй – коэффициентами полинома, начиная с указанного индекса. Если размер массива больше размеров копируемой части полинома, то некоторые элементы останутся неинициализированными, а если меньше, то лишние коэффициенты полинома должны быть отброшены.

99) Написать интерфейс IPolynom, определяющий свойства, возвращающие степень полинома и индексатор для доступа к коэффициентам полинома произвольного типа. Реализовать этот интерфейс в классе полинома.

ЛАБОРАТОРНАЯ РАБОТА №4

Общее задание. Составить описание структуры «Рациональная дробь» для представления объектов-дробей вида

$$\frac{A}{B},$$

где A и B (числитель и знаменатель) – целые числа. По умолчанию A = 0, B = 1.

I. Предусмотреть применение конструкторов:

а) для инициализации дроби целым числом;

б) для инициализации дроби указанными значениями числителя и знаменателя;

в) для копирования одной дроби в другую.

II. Предусмотреть метод, обеспечивающий декомпозицию дроби. Он должен возвращать целую часть дроби, а сама дробь в результате его работы должна стать правильной.

III. С помощью перегруженных операторов структуры обеспечить операции сложения, вычитания, умножения и деления дробей. Также перегрузить операторы преобразования дроби к типам **int** и **double**, и значений типа **int** – в дробь.

IV. С помощью свойств обеспечить доступ для чтения значений числителя и знаменателя дроби. Также предусмотреть свойство типа **bool**, определяющее, будет ли автоматически при совершении любых операций с дробью происходить ее сокращение (делением числителя и знаменателя на НОД). Сокращение дроби реализовать в отдельном открытом (**public**) методе.

V. Перегрузить метод ToString для представления дроби в виде строки «A/B».

Далее приводится дополнительное задание по вариантам.

Варианты 00-33

Задание состоит из следующих этапов:

• Реализовать в классе универсальный интерфейс IEquatable<T>. Критерий сравнения – норма вектора. Продемонстрировать применение этого интерфейса, осуществляя поиск в списке векторов методами List<T>.Contains и List<T>.IndexOf.

• Создать универсальную версию класса «Вектор». Добиться того, чтобы элементами вектора могли быть числа с плавающей точкой (вернее, экземпляры структуры, эмулирующей тип **double**) или рациональные дроби.

• Добавить в проект класс атрибута (или несколько классов атрибутов). Функциональность классов атрибутов предлагается выбрать самостоятельно – это могут быть какие-либо данные о классе или его членах.

• Используя механизм отражения, вывести в текстовый файл полную информацию о полученном универсальном классе «Вектор» (имена и типы полей и свойств, сигнатуры методов и т.п.).

Варианты 34-66

Задание состоит из следующих этапов:

• Реализовать в классе универсальный интерфейс IEquatable<T>. Критерий сравнения – норма матрицы. Продемонстрировать применение этого интерфейса, осуществляя поиск в списке матриц методами List<T>.Contains и List<T>.IndexOf.

• Создать универсальную версию класса «Матрица». Добиться того, чтобы элементами матрицы могли быть числа с плавающей точкой (вернее, экземпляры структуры, эмулирующей тип **double**) или рациональные дроби.

• Добавить в проект класс атрибута (или несколько классов атрибутов). Функциональность классов атрибутов предлагается выбрать самостоятельно – это могут быть какие-либо данные о классе или его членах.

• Используя механизм отражения, вывести в текстовый файл полную информацию о полученном универсальном классе «Матрица» (имена и типы полей и свойств, сигнатуры методов и т.п.).

Варианты 67-99

Задание состоит из следующих этапов:

• Реализовать в классе универсальный интерфейс IEquatable<T>. Критерий сравнения – порядок и коэффициенты полинома (как в ЛР№3). Продемонстрировать применение этого интерфейса, осуществляя поиск в списке полиномов методами List<T>.Contains и List<T>.IndexOf.

• Создать универсальную версию класса «Полином». Добиться того, чтобы коэффициентами полинома могли быть числа с плавающей точкой

(вернее, экземпляры структуры, эмулирующей тип **double**) или рациональные дроби.

• Добавить в проект класс атрибута (или несколько классов атрибутов). Функциональность классов атрибутов предлагается выбрать самостоятельно – это могут быть какие-либо данные о классе или его членах.

• Используя механизм отражения, вывести в текстовый файл полную информацию о полученном универсальном классе «Полином» (имена и типы полей и свойств, сигнатуры методов и т.п.).

ЛАБОРАТОРНАЯ РАБОТА №5

Варианты 00-33

Задание состоит из следующих этапов:

• Сформировать массив векторов. Обеспечить многопоточный поиск в массиве вектора (или векторов) согласно требуемому критерию. Критерий поиска задавать в виде предиката (делегата). Программа должна корректно работать с любым количеством потоков (но не большим, чем количество элементов в массиве), по умолчанию их количество равно числу ядер процессора. Каждому потоку отводится свой диапазон элементов массива для поиска.

• Сформировать список векторов (List). Аналогичным образом обеспечить поиск по критерию, используя пул потоков. Каждый освободившийся поток берет следующий необработанный элемент из списка.

• Об окончании поиска должно сигнализировать окно сообщения (функция API Windows MessageBox).

• Реализовать хранение коэффициентов вектора в виде указателя на неуправляемый динамический массив.

• Обеспечить документирование кода проекта. Все классы и члены классов должны быть снабжены специальными комментариями для генерации XML-файла документации. По данному XML-коду сформировать документацию в любом удобном для просмотра формате.

Варианты 34-66

Задание состоит из следующих этапов:

• Сформировать массив матриц. Обеспечить многопоточный поиск в массиве матрицы (или матриц) согласно требуемому критерию. Критерий поиска задавать в виде предиката (делегата). Программа должна корректно работать с любым количеством потоков (но не большим, чем количество элементов в массиве), по умолчанию их количество равно числу ядер процессора. Каждому потоку отводится свой диапазон элементов массива для поиска.

• Сформировать список матриц (List). Аналогичным образом обеспечить поиск по критерию, используя пул потоков. Каждый освободившийся

поток берет следующий необработанный элемент из списка.

• Об окончании поиска должно сигнализировать окно сообщения (функция API Windows MessageBox).

• Реализовать хранение коэффициентов матрицы в виде указателя на неуправляемый динамический массив.

• Обеспечить документирование кода проекта. Все классы и члены классов должны быть снабжены специальными комментариями для генерации XML-файла документации. По данному XML-коду сформировать документацию в любом удобном для просмотра формате.

Варианты 67-99

Задание состоит из следующих этапов:

• Сформировать массив полиномов. Обеспечить многопоточный поиск в массиве полинома (или полиномов) согласно требуемому критерию. Критерий поиска задавать в виде предиката (делегата). Программа должна корректно работать с любым количеством потоков (но не большим, чем количество элементов в массиве), по умолчанию их количество равно числу ядер процессора. Каждому потоку отводится свой диапазон элементов массива для поиска.

• Сформировать список полиномов (List). Аналогичным образом обеспечить поиск по критерию, используя пул потоков. Каждый освободившийся поток берет следующий необработанный элемент из списка.

• Об окончании поиска должно сигнализировать окно сообщения (функция API Windows MessageBox).

• Реализовать хранение коэффициентов полинома в виде указателя на неуправляемый динамический массив.

• Обеспечить документирование кода проекта. Все классы и члены классов должны быть снабжены специальными комментариями для генерации XML-файла документации. По данному XML-коду сформировать документацию в любом удобном для просмотра формате.