Министерство образования и науки Российской Федерации

Томский государственный университет систем управления
и радиоэлектроники (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

Романенко В.В.

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Лабораторные работы

Методические указания по выполнению лабораторных работ по дисциплине «Объектно-ориентированное программирование»
для студентов очной формы обучения специальностей
010400 – «Прикладная математика и информатика» и
230100 – «Информатика и вычислительная техника»

Томск – 2014

Содержание

Введение

Данное пособие предназначено для студентов специальностей 010400 – «Прикладная математика и информатика», а также 230100 – «Информатика и вычислительная техника» ТУСУР и содержит требования к выполнению лабораторных работ по дисциплине «Объектно-ориентированное программирование». В рамках дисциплины «Объектно-ориентированное программирование» изучаются основные принципы ООП, а также программирование на языках C++ и C#.

Формат титульного листа отчета приведен в **приложении А**.

1. Лабораторная работа №1

1.1. Цель работы

Практическое ознакомление с правилами составления протоколов описаний классов C++, получение навыков составления элементарных программ с типами данных «объект-экземпляр класса».

1.2. Задание к лабораторной работе

Составить описание класса для объектов-векторов, задаваемых координатами концов в трехмерном пространстве, считая, что компоненты векторов представлены вещественными числами типа **double**. Компоненты векторов должны быть скрыты (инкапсулированы) в объекте. Предусмотреть в классе деструктор и, как минимум, два конструктора:

а) для инициализации векторов нулевыми компонентами и

б) заданным набором компонентов.

Можно использовать параметры по умолчанию для сокращения количества конструкторов.

Организовать в конструкторах и деструкторе вывод на экран информационных сообщений, например, «*Конструктор 1*», «*Деструктор*» и т.д.

I. С помощью **функций-элементов класса** обеспечить

1) доступ к элементам вектора (чтение/запись);

2) вычисление модуля вектора;

3) копирование вектора;

4) умножение вектора на скаляр;

5) нормировку вектора (получение вектора единичной длины).

II. С помощью **внешних функций** обеспечить двуместные операции над векторами *A* и *B*:

а) с получением нового вектора *C*:

1) сложение ($C = A + B$);

2) вычитание ($C = A – B$);

3) векторное произведение ($C = A \times B$);

б) с получением скалярных величин:

1) скалярного произведения двух векторов;

2) косинуса и синуса угла между двумя векторами;

3) величины угла в градусах между векторами в пределах [–180º, 180º].

УКАЗАНИЕ: для расчета угла воспользуйтесь функцией **atan2**, подключив заголовочный файл **math.h**.

Создайте функцию-элемент класса для вывода на экран компонентов вектора в удобной форме, например, в виде строки:

x = <значение *x*>; *y* = <значение *y*>; *z* = <значение *z*>.

По возможности используйте передачу параметров и возврат значений из функций по ссылке. Там, где это возможно, используйте модификатор **const** при описании функций-элементов класса и параметров.

Исследуйте, в каких местах программы происходит автоматический вызов конструкторов и деструктора. Объясните, почему так происходит.

Математические сведения, необходимые для программирования методов векторной алгебры представлены в **Приложении Б**.

1.3. Задачи для самостоятельного решения

1. Описать класс с двумя полями X и P, инкапсулирующий число $X \cdot 10^P$. Определить методы деления, умножения и возведения в степень таких чисел.

2. Класс инкапсулирует число N, записанное в системе счисления по основанию P ($2 \le P \le 16$). Определить методы вывода числа на консоль и копирования строки такому числу.

3. Класс инкапсулирует вектор из N элементов. Определить методы сравнения векторов. В качестве критерия сравнения использовать норму векторов.

4. Класс инкапсулирует десятичное число, хранящееся в виде строки S, максимальная длина которой равна N. Определить методы сложения и копирования таких чисел.

5. Класс инкапсулирует точку на декартовой плоскости. Определить методы покоординатного сложения и вычитания точек, а также метод обращения знака.

6. Класс инкапсулирует точку на декартовой плоскости. Определить методы поворота точки вокруг центра координат на указанный угол, а также поворота на угол $\pm\pi$.

7. Класс инкапсулирует двоичное число, хранимое в виде строки S максимальной длины N. Определить методы циклического сдвига двоичного числа вправо или влево, а также инверсии этого числа.

8. Класс инкапсулирует прямоугольник со сторонами A и B. Определить метод, соединяющий два прямоугольника горизонтально, если они имеют одинаковую высоту, и метод, соединяющий два прямоугольника вертикально, если они имеют одинаковую ширину, а также метод копирования.

9. Описать класс с полем P, инкапсулирующий число e^P. Определить методы деления, умножения и возведения в степень таких чисел, а также их деления и умножения с числами типа double.

10. Класс инкапсулирует шар радиуса R. Определить метод сложения, в результате которой получается шар, объем которого равен сумме объемов исходных шаров, а также метод вычитания по схожему принципу. При получении отрицательного объема выдавать ошибку.

11. Класс инкапсулирует дату (в виде номера дня, месяца и года – D, M, Y). Определить метод сравнения дат, а также увеличения и уменьшения даты на целое количество дней.

12. Класс инкапсулирует рациональную дробь (в виде числителя A и знаменателя B). Определить методы сравнения дробей.

13. Класс инкапсулирует мнимое число. Определить методы деления, умножения и вывода на экран таких чисел.

14. Класс инкапсулирует вектор произвольной размерности. Определить методы доступа к элементам вектора.

15. Описать класс с двумя полями X и P, инкапсулирующий число X, возведенное в степень P (X^P). Определить методы деления, умножения и возведения в степень таких чисел.

2. Лабораторная работа №2

2.1. Цель работы

Освоение методов использования динамической памяти, изучение свойства полиморфизма, реализуемого перегрузкой функций и операций в классах C++.

2.2. Задание к лабораторной работе

Составить описание класса для объектов прямоугольных матриц, задаваемых массивом вещественных чисел типа **double**, располагающегося в памяти по строкам. Компоненты матрицы должны быть скрыты (инкапсулированы) в объекте.

Предусмотреть применение конструкторов:

а) по умолчанию;

б) для инициализации квадратной матрицы заданного размера с заданными компонентами;

в) для инициализации прямоугольной матрицы заданных размеров с заданными компонентами;

г) копирования.

Можно использовать параметры по умолчанию для сокращения количества конструкторов.

Конструкторы должны создавать объекты в динамической памяти (оператор **new**), а деструктор – освобождать память (оператор **delete**). Способ размещения объекта в динамической памяти (в виде одномерного или двумерного динамического массива, либо комбинированный вариант –

одномерный массив с массивом указателей на начало каждой строки матрицы) выбрать самостоятельно. Все эти способы имеют как достоинства, так и недостатки.

Организовать в конструкторах и деструкторе вывод на экран информационных сообщений, например, «*Конструктор 1*», «*Деструктор*» и т.д.

I. С помощью функций-элементов класса обеспечить:

1) проверку возможности умножения двух матриц;

2) проверку возможности сложения двух матриц;

3) максимального элемента матрицы;

4) минимального элемента матрицы.

II. С помощью операторов-элементов класса обеспечить:

1) доступ к элементам матрицы по индексу строки и столбца (чтение/запись), т.е. переопределить оператор [];

2) вывод на экран матрицы в построчной форме, т.е. переопределить оператор вывода на поток <<;

3) математические действия над матрицами A и B без получения новых матриц, т.е. переопределить операторы

а) $A = B$;

б) *A* += *B*;

в) *A* –= *B*;

г) *A* *= *B*;

д) а также *A* *= *k* – умножение матрицы на скаляр.

III. С помощью внешних операторов обеспечить двуместные операции над матрицами *A* и *B* с получением новой матрицы *C*:

1) сложение (*C* = *A* + *B*);

2) вычитание (*C* = *A* – *B*);

3) произведение (*C* = *A* * *B*);

4) умножение матрицы на скаляр (*C* = *A* * *k*).

Выполнению операций сложения, вычитания и умножения матриц должна предшествовать проверка возможности их выполнения над данными объектами.

УКАЗАНИЕ: Для выравнивания позиций при выводе матрицы на экран можно использовать функции-манипуляторы потока (библиотека **iomanip.h**) либо функции-элементы (методы) класса ostream (библиотека **iostream.h**) (табл. 2.1).

Табл. 2.1 – Форматирование вывода

Манипуляторы потока	Методы класса ostream
fixed scientific setprecision(int n)	setf(ios_base::fixed, ios_base::floatfield) setf(ios_base::scientific,

setw(int n)	ios_base::floatfield)
resetiosflags(ios_base::fmtflags flag)	precision(int n)
setiosflags(ios_base::fmtflags flag)	width(int n)
	setf(0, flag)
	setf(flag)

2.3. Задачи для самостоятельного решения

1. Описать класс с двумя полями X и P, инкапсулирующий число $X \cdot 10^P$. Определить операции деления, умножения и возведения в степень таких чисел (/, *, ^).

2. Класс инкапсулирует число N, записанное в системе счисления по основанию P ($2 \le P \le 16$). Определить операции вывода числа на консоль (<<) и присваивания строки (=) такому числу.

3. Класс инкапсулирует вектор из N элементов. Определить операции сравнения векторов (==, !=, >, >=, <, <=). В качестве критерия сравнения использовать норму векторов.

4. Класс инкапсулирует десятичное число, хранящееся в виде строки S, максимальная длина которой равна N. Определить операции сложения (+) и присваивания (=) таких чисел.

5. Класс инкапсулирует точку на декартовой плоскости. Определить операции покоординатного сложения и вычитания точек (+, –), а также унарную операцию обращения знака (–).

6. Класс инкапсулирует точку на декартовой плоскости. Определить операции поворота точки вокруг центра координат на указанный угол (+=, –=), а также поворота на угол ±π (++, —).

7. Класс инкапсулирует двоичное число, хранимое в виде строки S максимальной длины N. Определить операции циклического сдвига двоичного числа вправо или влево, а также инверсии этого числа (<<, >>, ~).

8. Класс инкапсулирует прямоугольник со сторонами A и B. Определить операцию «&», соединяющую два прямоугольника горизонтально, если они имеют одинаковую высоту, и операцию «|», соединяющую два прямоугольника вертикально, если они имеют одинаковую ширину, а также операцию присваивания (=).

9. Описать класс с полем P, инкапсулирующий число e^P. Определить операции деления, умножения и возведения в степень таких чисел (/, *, ^), а также их деления и умножения с числами типа double.

10. Класс инкапсулирует шар радиуса R. Определить операцию сложения (+), в результате которой получается шар, объем которого равен сумме объемов исходных шаров, а также операцию вычитания (–) по схожему принципу. При получении отрицательного объема выдавать ошибку.

11. Класс инкапсулирует дату (в виде номера дня, месяца и года – D, M, Y). Определить операции сравнения дат (<, >), а также увеличения и уменьшения даты на целое количество дней (+=, –=).

12. Класс инкапсулирует рациональную дробь (в виде числителя A и знаменателя B). Определить операции сравнения дробей.

13. Класс инкапсулирует мнимое число. Определить операции деления, умножения и вывода на экран таких чисел.

14. Класс инкапсулирует вектор произвольной размерности. Определить операцию доступа к элементам вектора.

15. Описать класс с двумя полями X и P, инкапсулирующий число X, возведенное в степень P (X^P). Определить операции деления, умножения и возведения в степень таких чисел.

3. Лабораторная работа №3

3.1. Цель работы

Изучение механизма наследования классов и полиморфизма, виртуальных методов, а также абстрактных методов и классов в языке C++.

3.2. Задание к лабораторной работе

Необходимо разработать иерархическую структуру классов, позволяющую создавать два типа объектов – вектора и матрицы. Вектор представляет собой одномерный массив произвольной длины, матрица – двумерный массив произвольной размерности (см. ЛР№2). При этом с обоими типами объектов должны быть предусмотрены:

1. Конструкторы (по умолчанию, с параметрами, копирующий);
2. Проверка возможности сложения и умножения объектов (вектора и вектора, вектора и матрицы, матрицы и вектора, матрицы и матрицы);
3. Оператор доступа к элементам вектора и матрицы [];
4. Оператор вывода в поток <<;
5. Математические операторы (+, –, *, +=, –=, *=, =) с объектами, а также умножение объектов на скаляр;

Указание. Структуру иерархии классов продумать самостоятельно. Возможны, как минимум, три варианта (рис. 3.1). При выборе варианта №3 пользователь не должен иметь возможности создания экземпляров базового класса. Также необходимо самостоятельно решить, какие методы должны быть виртуальными.

Рис. 3.1 – Варианты иерархии классов

3.2. Задачи для самостоятельного решения

Построить иерархию классов для следующей предметной области:

1. Геометрические фигуры;
2. Интервалы (открытые, закрытые, бесконечные и т.п.);
3. Транспорт;
4. Контейнеры для хранения данных;
5. Родственные связи.

Определить конструкторы, деструкторы, методы доступа к инкапсулированным полям, а также все необходимые операции для работы с объектами предметной области.

4. Лабораторная работа №4

4.1. Цель работы

Создание классов математических объектов с перегрузкой операций на языке C++.

4.2. Задание к лабораторной работе

1. Составить описание класса для представления типа «обыкновенная дробь» вида

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

где A и B – целые числа.

а) Обеспечить выполнение 4-х арифметических действий над этими объектами (+=, +, –=, –, *=, *, /=, /);

б) Предусмотреть операцию автоматического «сокращения дроби» – удаления общих множителей числителя и знаменателя;

в) Обеспечить операцию декомпозиции неправильной дроби на целую часть и правильную дробь, используя операцию деления с остатком.

2. Составить описание класса для представления типа «полином» вида

$$P(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n,$$

где a_i – вещественные коэффициенты.

а) Обеспечить выполнение 3-х арифметических действий над этими объектами (+=, +, –=, –, *=, *);

б) Предусмотреть операцию деления полиномов с остатком, используя «деление углом» (алгоритм Эвклида);

в) Предусмотреть функцию обновления, отбрасывающую старшие члены полинома с нулевыми коэффициентами;

г) Перегрузить операцию () для вычисления значения полинома.

3. Для всех типов организовать перегруженный оператор присваивания.

УКАЗАНИЕ: все арифметические операторы реализовать путем перегрузки операций с использованием свойства дружественности.

5. Лабораторная работа №5

5.1. Цель работы

Изучение механизма применения шаблонов функций как средства агрегирования программных модулей в языке C++.

5.2. Задание к лабораторной работе

Данная лабораторная работа базируется на результатах лабораторных работ №1-3.

1. Используя исходный код класса прямоугольных матриц и метод шаблонов, составить описание класса для представления матриц с элементами произвольного типа. Протестировать полученный класс, т.е. проверить правильность выполнения всех функций и операторов класса, используя в качестве шаблонов:

   а) Базовые типы языка C (**int**, **double**, **char** и т.д.);

   б) Класс-вектор, являющийся результатом лабораторной работы №1;

   в) Класс-матрицу, являющуюся результатом лабораторной работы №2;

   г) Класс-матрицу с шаблоном, т.е. шаблоном должна являться матрица, использующая, в

свою очередь, некоторый базовый тип языка С в качестве шаблона.

УКАЗАНИЕ: При выполнении последнего пункта следует учесть, что не все компиляторы поддерживают конструкции вида «класс1<класс2<шаблон>>», в этом случае при помощи оператора **typedef** необходимо сначала создать новый вспомогательный тип, например

typedef класс2<шаблон1> шаблон2.

А уже затем использовать новый тип «шаблон2» как шаблон для класса «класс1».

2. Используя исходный код модуля «обыкновенная дробь», тип «полином» и метод шаблонов составить описание класса для представления объектного типа «полиномиальная дробь»:

$$W(x) = \frac{b_0 + b_1 x + b_2 x^2 + ... + b_m x^m}{a_0 + a_1 x + a_2 x^2 + ... + a_n x^n},$$

где a_i и b_i - вещественные числа, m и n – целые числа.

а) Обеспечить выполнение 4-х арифметических действий над этими объектами (+=, +, –=, –, *=, *, /=, /);

б) Обеспечить операцию декомпозиции неправильной дроби на целую часть и правильную дробь, используя операцию деления с остатком (здесь целая часть – полином, остаток – правильная полиномиальная дробь, у которой степень полинома числителя меньше, чем степень полинома знаменателя).

УКАЗАНИЯ:

1) На основе кода программного модуля «обыкновенная дробь» создать шаблон с параметризацией типа объектов в числителе и знаменателе дроби;
2) Формировать класс «полиномиальная дробь», конкретизируя параметр созданного шаблона классом «полином» с соответствующими перегруженными операторами.

6. Лабораторная работа №6

6.1. Цель работы

Практическое ознакомление с правилами описания классов на языке C#. Освоение описания полей и методов классов, перегрузки операций.

6.2. Задание к лабораторной работе

Составить описание класса для объектов прямоугольных матриц, задаваемых массивом (одномерным или прямоугольным) 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 (static);

2) проверку возможности сложения двух матриц (static);

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).

6.3. Дополнительные задания

1) Обеспечить хранение элементов матрицы в классе в виде одномерного массива. Извне класса пользователь

должен иметь возможность работать с ним как с двумерным объектом.

2) 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.

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

4) 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.

5) 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.

6) 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.

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

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

9) 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.**

10) 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.

11) 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.

12) 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.

13) 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.

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

15) Перегрузить в матрице операторы отношения и реализовать интерфейс IComparable. Сравнение матриц осуществлять на основании количества элементов.

16) Перегрузить в матрице операторы отношения и реализовать интерфейс IComparable. Сравнение матриц осуществлять на основании их норм.

17) 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.

18) Используя оператор **yield**, реализовать в классе итератор для перечисления всех элементов матрицы. Па-

раметр 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.

19) 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.

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

21) 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.

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

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

24) 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.

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

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

27) 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.

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

29) 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.).

30) 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.

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

32) 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.

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

7. Лабораторная работа №7

7.1. Цель работы

Изучение механизма наследования классов и полиморфизма, виртуальных методов, а также абстрактных методов и классов в языке C#.

7.2. Задание к лабораторной работе

Необходимо разработать иерархическую структуру классов, позволяющую создавать два типа объектов – вектора и матрицы. Вектор представляет собой одномерный массив произвольной длины, матрица – двумерный массив произвольной размерности (см. ЛР№2). При этом с обоими типами объектов должны быть предусмотрены те же операции, что и для матрицы в ЛР№2:

6. Конструкторы, деструктор;
7. Проверка возможности сложения и умножения объектов;
8. Поиск минимального и максимального элемента;
9. Индексатор;
10. Перегруженный метод ToString;
11. Перегруженные математические операторы.

Указания:

1) Необходимо обеспечить возможность выполнения математических операторов над объектами разных типов – например, сложение матрицы и вектора (если их размеры соответствуют) и т.п.

2) Структуру иерархии классов продумать самостоятельно. Возможны, как минимум, три варианта (рис. 7.1). При выборе варианта №3 пользователь не должен иметь возможности создания экземпляров базового класса. Также необходимо самостоятельно решить, какие методы должны быть виртуальными.

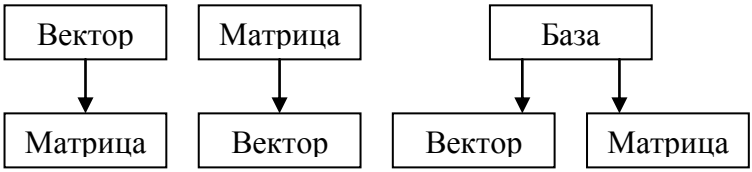

Рис. 7.1 – Варианты иерархии классов

3) Для векторов не должны быть доступны операции, свойственные матрицам, и наоборот. Т.е. либо соответствующих членов не должно быть в базовом классе (при выборе варианта №3), либо их использование должно приводить к исключению (при выборе вариантов №1 и №2).

8. Лабораторная работа №8

8.1. Цель работы

Создание классов математических объектов с перегрузкой операций на языке C#. Изучение механизма универсальных типов в языке C#. Изучение делегатов и событий. Получение навыков документирования кода.

8.2. Задание к лабораторной работе

1. Структура «Дробь». Составить описание структуры для инкапсуляции 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 преобра-

зования 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».

2. Класс «Полином». Составить описание класса для инкапсуляции 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.

3. Универсальные типы. Используя исходный код структуры «Дробь» и класса «Полином», составить описание универсальной структуры «Универсальная дробь» для инкапсуляции объектов-дробей вида

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

где A и B (числитель и знаменатель) – целые числа или полиномы. Значениями по умолчанию должны быть A = 0, B = 1.

В случае с целыми числами (типа **int**) получаем обычную рациональную дробь. В случае с полиномами получаем полиномиальную дробь вида

$$\frac{b_0 + b_1 x + b_2 x^2 + \ldots + b_m x^m}{a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n}.$$

Используя исходный код структуры «Дробь» и класса «Полином», составить описание универсального класса «Универсальный полином» для инкапсуляции объектов-полиномов вида

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n,$$

где a_i – числа с плавающей точкой или дроби.

В случае с числами с плавающей точкой (тип **double**) получаем обычный степенной полином. В случае с дробями получаем полином, коэффициентами которого являются рациональные дроби:

$$\frac{A_0}{B_0} + \frac{A_1}{B_1} x + \frac{A_2}{B_2} x^2 + \ldots + \frac{A_n}{B_n} x^n.$$

Реализовать все операции, доступные для дробей и полиномов в ЛР№4.

Указание. Для универсального типа T, представляющего числитель и знаменатель дроби, а также коэффициенты полинома, следует описать ограничение типа. То есть все эти типы должны реализовывать некоторый общий интерфейс:

```
interface Интерфейс
{
```

```
        …
}
struct Int : Интерфейс, …
{
        …
}
struct Double: Интерфейс, …
{
        …
}
struct Дробь<T> : Интерфейс, … where T: Интерфейс
{
        …
}
class Полином<T> : Интерфейс, … where T: Интерфейс
{
        …
}
```

Либо являться потомками общего абстрактного класса:

```
abstract class Класс
{
        …
}
class Int : Класс, …
{
        …
}
class Double: Класс, …
```

```
{
        …
}
class Дробь<T> : Класс, … where T: Класс
{
        …
}
class Полином<T> : Класс, … where T: Класс
{
        …
}
```

В интерфейсе или абстрактном классе должны быть описаны все операции, которые будут применяться числителю и знаменателю дроби, а также коэффициентам полинома.

Первый вариант предпочтительнее, т.к. типы Int, Double и Дробь не имеет смысла делать ссылочными. Можно интерфейс или абстрактный класс также сделать универсальными:

```
interface Интерфейс<T>
{
        …
}
struct Int : Интерфейс<Int>, …
{
        …
}
…
```

или

```
abstract class Класс<T>
{
        …
}
class Int : Класс<Int>, …
{
        …
}
…
```

При этом типы Int и Double – это обёртки для инкапсуляции целых чисел (типа **int**) и чисел с плавающей точкой (типа **double**). Они должны иметь неявные преобразования в инкапсулируемый тип и обратно.

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

5. 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.

Литература

1. Павловская Т.А. C/C++. Программирование на языке высокого уровня: учебник для вузов. – СПб: Питер, 2013. – 461 с. (36 экз.).

2. Павловская Т.А. C/C++. Программирование на языке высокого уровня: учебник для вузов. – СПб: Питер, 2009 (4 экз.), 2010 (3 экз.), 2011 (1 экз.). – 464 с.

3. Павловская Т.А. C/C++. Программирование на языке высокого уровня для магистров и бакалавров: учебник для вузов. – СПб: Питер, 2012. – 461 с. (3 экз.).

4. Орлов С.А. Технологии разработки программного обеспечения: современный курс по программной инженерии. – СПб: Питер, 2012. – 608 с. (15 экз.).

Приложения

Приложение А. Формат титульного листа отчета

Министерство образования и науки Российской Федерации

Томский государственный университет систем управления и радиоэлектроники (ТУСУР)

Факультет систем управления (ФСУ)

Кафедра автоматизированных систем управления (АСУ)

НАЗВАНИЕ РАБОТЫ

Отчет по лабораторной работе №X по дисциплине «Объектно-ориентированное программирование»

Выполнил: ст. гр. XXX

_____ Иванов И.И.

«_____» _____ 2014 г.

Проверил: доц. каф. АСУ

_____ Романенко В.В.

«_____» _____ 2014 г.

Томск – 2014

Приложение Б. Операции векторно-матричной алгебры

Пусть A, B и C – вектора в трехмерном пространстве с компонентами (A_x, A_y, A_z), (B_x, B_y, B_z) и (C_x, C_y, C_z) соответственно. Тогда для $C = A \pm B$ имеет место:

$$C_x = A_x \pm B_x;$$

$$C_y = A_y \pm B_y;$$

$$C_z = A_z \pm B_z.$$

Модулем вектора A называют число $m = |A|$, определяемое как корень квадратный из суммы квадратов компонентов вектора.

Векторным произведением двух векторов A и B называют вектор $C = [A, B]$, компоненты которого определяются на основе следующих соотношений:

$$C_x = A_y \times B_z - A_z \times B_y;$$

$$C_y = A_z \times B_x - A_x \times B_z;$$

$$C_z = A_x \times B_y - A_y \times B_x;$$

Вектор C перпендикулярен векторам A и B одновременно, его направление совпадает с движением правого винта, вращаемого от A к B, при этом $|C| = |A| \times |B| \times \sin(\alpha)$, где α – угол между векторами.

Скалярным произведением двух векторов A и B называют скалярную величину $s = (A, B)$, определяемую как

$$s = A_x \times B_x + A_y \times B_y + A_z \times B_z.$$

Для скалярного произведения имеет место соотношение: $s = |A| \times |B| \times \cos(\alpha)$, где α – угол между векторами.