МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждениевысшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по выполнению практических занятий
по дисциплине «Информационные технологии и программирование»
Для студентов направления подготовки 15.03.04 Автоматизация технологических процессов и производств,
направленность (профиль) Информационно-управляющие системы

(ЭЛЕКТРОННЫЙ ДОКУМЕНТ)

Невинномысск, 2026

Содержание

Тема 1. Основы C#: синтаксис, типы данных, консольный ввод-вывод
Практическая работа 1 Ввод и вывод данных, оператор присваивания 3
Тема 1. Основы C#: синтаксис, типы данных, консольный ввод-вывод
Практическая работа 2 Целые числа 7
Тема 1. Основы C#: синтаксис, типы данных, консольный ввод-вывод
Практическая работа 3 Логические выражения 10
Тема 1. Основы C#: синтаксис, типы данных, консольный ввод-вывод
Практическая работа 4 Условный оператор 13
Тема 2 Управление потоком: циклы и последовательности
Практическая работа 5 Оператор выбора 16
Тема 2 Управление потоком: циклы и последовательности
Практическая работа 6 Цикл с параметром 19
Тема 2. Управление потоком: циклы и последовательности
Практическая работа 7 Цикл с условием 24
Тема 3. Работа с данными: массивы, строки, рекурсия
Практическая работа 8 Последовательности 28
Тема 3. Работа с данными: массивы, строки, рекурсия
Практическая работа 9 Минимумы и максимумы 33
Тема 3. Работа с данными: массивы, строки, рекурсия
Практическая работа 10 Одномерные массивы 36
Тема 3. Работа с данными: массивы, строки, рекурсия
Практическая работа 11 Двумерные массивы (матрицы) 50
Тема 4. Основы ООП: классы, конструкторы, перегрузка
Практическая работа 12 Символы и строки 60
Тема 4. Основы ООП: классы, конструкторы, перегрузка
Практическая работа 13 Составные типы данных в процедурах и функциях 66
Тема 4. Основы ООП: классы, конструкторы, перегрузка
Практическая работа 14 Рекурсия Простейшие рекурсивные алгоритмы 106

Тема 1. Основы C#: синтаксис, типы данных, консольный ввод-вывод
Практическая работа 1 Ввод и вывод данных, оператор присваивания

Все входные и выходные данные в заданиях этой группы являются вещественными числами.

Begin1°. Дана сторона квадрата a. Найти его периметр $P = 4 \cdot a$.

Begin2°. Дана сторона квадрата a. Найти его площадь $S = a^2$.

Begin3°. Даны стороны прямоугольника a и b. Найти его площадь $S = a \cdot b$ и периметр $P = 2 \cdot (a + b)$.

Begin4°. Дан диаметр окружности d. Найти ее длину $L = \pi \cdot d$. В качестве значения π использовать 3.14.

Begin5°. Дана длина ребра куба a. Найти объем куба $V = a^3$ и площадь его поверхности $S = 6 \cdot a^2$.

Begin6°. Даны длины ребер a, b, c прямоугольного параллелепипеда. Найти его объем $V = a \cdot b \cdot c$ и площадь поверхности $S = 2 \cdot (a \cdot b + b \cdot c + a \cdot c)$.

Begin7°. Найти длину окружности L и площадь круга S заданного радиуса R:

$$L = 2 \cdot \pi \cdot R, \qquad S = \pi \cdot R^2.$$

В качестве значения π использовать 3.14.

Begin8°. Даны два числа a и b. Найти их *среднее арифметическое*: $(a + b)/2$.

Begin9°. Даны два неотрицательных числа a и b. Найти их *среднее геометрическое*, то есть квадратный корень из их произведения: $\sqrt{a \cdot b}$.

Begin10°. Даны два ненулевых числа. Найти сумму, разность, произведение и частное их квадратов.

Begin11°. Даны два ненулевых числа. Найти сумму, разность, произведение и частное их модулей.

Begin12°. Даны катеты прямоугольного треугольника a и b. Найти его гипотенузу c и периметр P

Begin13°. Даны два круга с общим центром и радиусами R_1 и R_2 ($R_1 > R_2$). Найти площади этих кругов S_1 и S_2, а также площадь S_3 кольца, внешний радиус которого равен R_1, а внутренний радиус равен R_2:

$$S_1 = \pi \cdot (R_1)^2, \qquad S_2 = \pi \cdot (R_2)^2, \qquad S_3 = S_1 - S_2.$$

В качестве значения π использовать 3.14.

Begin14°. Дана длина L окружности. Найти ее радиус R и площадь S круга, ограниченного этой окружностью, учитывая, что $L = 2 \cdot \pi \cdot R$, $S = \pi \cdot R^2$. В качестве значения π использовать 3.14.

Begin15°. Дана площадь S круга. Найти его диаметр D и длину L окружности, ограничивающей этот круг, учитывая, что $L = 2 \cdot \pi \cdot R$, $S = \pi \cdot R^2$. В качестве значения π использовать 3.14.

Begin16°. Найти расстояние между двумя точками с заданными координата-

ми x_1 и x_2 на числовой оси: $|x_2 - x_1|$.

Begin17°. Даны три точки *A*, *B*, *C* на числовой оси. Найти длины отрезков *AC* и *BC* и их сумму.

Begin18°. Даны три точки *A*, *B*, *C* на числовой оси. Точка *C* расположена между точками *A* и *B*. Найти произведение длин отрезков *AC* и *BC*.

Begin19°. Даны координаты двух противоположных вершин прямоугольника: (x_1, y_1), (x_2, y_2). Стороны прямоугольника параллельны осям координат. Найти периметр и площадь данного прямоугольника.

Begin20°. Найти расстояние между двумя точками с заданными координатами (x_1, y_1) и (x_2, y_2) на плоскости.

Begin21°. Даны координаты трех вершин треугольника: (x_1, y_1), (x_2, y_2), (x_3, y_3). Найти его периметр и площадь, используя формулу для расстояния между двумя точками на плоскости (см. задание Begin20). Для нахождения площади треугольника со сторонами *a*, *b*, *c* использовать *формулу Герона*:

Begin22°. Поменять местами содержимое переменных *A* и *B* и вывести новые значения *A* и *B*.

Begin23°. Даны переменные *A*, *B*, *C*. Изменить их значения, переместив содержимое *A* в *B*, *B* — в *C*, *C* — в *A*, и вывести новые значения переменных *A*, *B*, *C*.

Begin24°. Даны переменные *A*, *B*, *C*. Изменить их значения, переместив содержимое *A* в *C*, *C* — в *B*, *B* — в *A*, и вывести новые значения переменных *A*, *B*, *C*.

Begin25°. Найти значение функции $y = 3x^6 - 6x^2 - 7$ при данном значении *x*.

Begin26°. Найти значение функции $y = 4(x-3)^6 - 7(x-3)^3 + 2$ при данном значении *x*.

Begin27°. Дано число *A*. Вычислить A^8, используя вспомогательную переменную и три операции умножения. Для этого последовательно находить A^2, A^4, A^8. Вывести все найденные степени числа *A*.

Begin28°. Дано число *A*. Вычислить A^{15}, используя две вспомогательные переменные и пять операций умножения. Для этого последовательно находить A^2, A^3, A^5, A^{10}, A^{15}. Вывести все найденные степени числа *A*.

Begin29°. Дано значение угла α в градусах ($0 < \alpha < 360$). Определить значение этого же угла в радианах, учитывая, что $180^\circ = \pi$ радианов. В качестве значения π использовать 3.14.

Begin30°. Дано значение угла α в радианах ($0 < \alpha < 2 \cdot \pi$). Определить значение этого же угла в градусах, учитывая, что $180^\circ = \pi$ радианов. В качестве значения π использовать 3.14.

Begin31°. Дано значение температуры *T* в градусах Фаренгейта. Определить значение этой же температуры в градусах Цельсия. Температура по Цель-

сию T_C и температура по Фаренгейту T_F связаны следующим соотношением:

$$T_C = (T_F - 32)\cdot 5/9.$$

Begin32°. Дано значение температуры T в градусах Цельсия. Определить значение этой же температуры в градусах Фаренгейта. Температура по Цельсию T_C и температура по Фаренгейту T_F связаны следующим соотношением:

$$T_C = (T_F - 32)\cdot 5/9.$$

Begin33°. Известно, что X кг конфет стоит A рублей. Определить, сколько стоит 1 кг и Y кг этих же конфет.

Begin34°. Известно, что X кг шоколадных конфет стоит A рублей, а Y кг ирисок стоит B рублей. Определить, сколько стоит 1 кг шоколадных конфет, 1 кг ирисок, а также во сколько раз шоколадные конфеты дороже ирисок.

Begin35°. Скорость лодки в стоячей воде V км/ч, скорость течения реки U км/ч ($U < V$). Время движения лодки по озеру T_1 ч, а по реке (против течения) — T_2 ч. Определить путь S, пройденный лодкой (путь = время· скорость). Учесть, что при движении против течения скорость лодки уменьшается на величину скорости течения.

Begin36°. Скорость первого автомобиля V_1 км/ч, второго — V_2 км/ч, расстояние между ними S км. Определить расстояние между ними через T часов, если автомобили удаляются друг от друга. Данное расстояние равно сумме начального расстояния и общего пути, проделанного автомобилями; общий путь = время · суммарная скорость.

Begin37°. Скорость первого автомобиля V_1 км/ч, второго — V_2 км/ч, расстояние между ними S км. Определить расстояние между ними через T часов, если автомобили первоначально движутся навстречу друг другу. Данное расстояние равно модулю разности начального расстояния и общего пути, проделанного автомобилями; общий путь = время · суммарная скорость.

Begin38°. Решить линейное уравнение $A\cdot x + B = 0$, заданное своими коэффициентами A и B (коэффициент A не равен 0).

Begin39°. Найти корни *квадратного уравнения* $A\cdot x^2 + B\cdot x + C = 0$, заданного своими коэффициентами A, B, C (коэффициент A не равен 0), если известно, что дискриминант уравнения положителен. Вывести вначале меньший, а затем больший из найденных корней. Корни квадратного уравнения находятся по формуле

$$x_{1,2} = (-B \pm \sqrt{D})/(2\cdot A),$$

где D — *дискриминант*, равный $B^2 - 4\cdot A\cdot C$.

Begin40°. Найти решение *системы линейных уравнений* вида

$$A_1 \cdot x + B_1 \cdot y = C_1,$$
$$A_2 \cdot x + B_2 \cdot y = C_2,$$

заданной своими коэффициентами A_1, B_1, C_1, A_2, B_2, C_2, если известно, что данная система имеет единственное решение. Воспользоваться формулами

$$x = (C_1 \cdot B_2 - C_2 \cdot B_1)/D, \qquad y = (A_1 \cdot C_2 - A_2 \cdot C_1)/D,$$

где $D = A_1 \cdot B_2 - A_2 \cdot B_1$.

Тема 1. Основы C#: синтаксис, типы данных, консольный ввод-вывод
Практическая работа 2 Целые числа

Все входные и выходные данные в заданиях этой группы являются целыми числами. Все числа, для которых указано количество цифр (двузначное число, трехзначное число и т. д.), считаются положительными.

Integer1°. Дано расстояние L в сантиметрах. Используя операцию деления нацело, найти количество полных метров в нем (1 метр = 100 см).

Integer2°. Дана масса M в килограммах. Используя операцию деления нацело, найти количество полных тонн в ней (1 тонна = 1000 кг).

Integer3°. Дан размер файла в байтах. Используя операцию деления нацело, найти количество полных килобайтов, которые занимает данный файл (1 килобайт = 1024 байта).

Integer4°. Даны целые положительные числа A и B ($A > B$). На отрезке длины A размещено максимально возможное количество отрезков длины B (без наложений). Используя операцию деления нацело, найти количество отрезков B, размещенных на отрезке A.

Integer5°. Даны целые положительные числа A и B ($A > B$). На отрезке длины A размещено максимально возможное количество отрезков длины B (без наложений). Используя операцию взятия остатка от деления нацело, найти длину незанятой части отрезка A.

Integer6°. Дано двузначное число. Вывести вначале его левую цифру (десятки), а затем — его правую цифру (единицы). Для нахождения десятков использовать операцию деления нацело, для нахождения единиц — операцию взятия остатка от деления.

Integer7°. Дано двузначное число. Найти сумму и произведение его цифр.

Integer8°. Дано двузначное число. Вывести число, полученное при перестановке цифр исходного числа.

Integer9°. Дано трехзначное число. Используя одну операцию деления нацело, вывести первую цифру данного числа (сотни).

Integer10°. Дано трехзначное число. Вывести вначале его последнюю цифру (единицы), а затем — его среднюю цифру (десятки).

Integer11°. Дано трехзначное число. Найти сумму и произведение его цифр.

Integer12°. Дано трехзначное число. Вывести число, полученное при прочтении исходного числа справа налево.

Integer13°. Дано трехзначное число. В нем зачеркнули первую слева цифру и приписали ее справа. Вывести полученное число.

Integer14°. Дано трехзначное число. В нем зачеркнули первую справа цифру и приписали ее слева. Вывести полученное число.

Integer15°. Дано трехзначное число. Вывести число, полученное при перестановке цифр сотен и десятков исходного числа (например, 123 перейдет в 213).

Integer16°. Дано трехзначное число. Вывести число, полученное при перестановке цифр десятков и единиц исходного числа (например, 123 перейдет в 132).

Integer17°. Дано целое число, большее 999. Используя одну операцию деления нацело и одну операцию взятия остатка от деления, найти цифру, соответствующую разряду сотен в записи этого числа.

Integer18°. Дано целое число, большее 999. Используя одну операцию деления нацело и одну операцию взятия остатка от деления, найти цифру, соответствующую разряду тысяч в записи этого числа.

Integer19°. С начала суток прошло N секунд (N — целое). Найти количество полных минут, прошедших с начала суток.

Integer20°. С начала суток прошло N секунд (N — целое). Найти количество полных часов, прошедших с начала суток.

Integer21°. С начала суток прошло N секунд (N — целое). Найти количество секунд, прошедших с начала последней минуты.

Integer22°. С начала суток прошло N секунд (N — целое). Найти количество секунд, прошедших с начала последнего часа.

Integer23°. С начала суток прошло N секунд (N — целое). Найти количество полных минут, прошедших с начала последнего часа.

Integer24°. Дни недели пронумерованы следующим образом: 0 — воскресенье, 1 — понедельник, 2 — вторник, . . . , 6 — суббота. Дано целое число K, лежащее в диапазоне 1–365. Определить номер дня недели для K-го дня года, если известно, что в этом году 1 января было понедельником.

Integer25°. Дни недели пронумерованы следующим образом: 0 — воскресенье, 1 — понедельник, 2 — вторник, . . . , 6 — суббота. Дано целое число K, лежащее в диапазоне 1–365. Определить номер дня недели для K-го дня года, если известно, что в этом году 1 января было четвергом.

Integer26°. Дни недели пронумерованы следующим образом: 1 — понедельник, 2 — вторник, . . . , 6 — суббота, 7 — воскресенье. Дано целое число K, лежащее в диапазоне 1–365. Определить номер дня недели для K-го дня года, если известно, что в этом году 1 января было вторником.

Integer27°. Дни недели пронумерованы следующим образом: 1 — понедельник, 2 — вторник, . . . , 6 — суббота, 7 — воскресенье. Дано целое число K, лежащее в диапазоне 1–365. Определить номер дня недели для K-го дня года, если известно, что в этом году 1 января было субботой.

Integer28°. Дни недели пронумерованы следующим образом: 1 — понедель-

ник, 2 — вторник, . . . , 6 — суббота, 7 — воскресенье. Дано целое число K, лежащее в диапазоне 1–365, и целое число N, лежащее в диапазоне 1–7. Определить номер дня недели для K-го дня года, если известно, что в этом году 1 января было днем недели с номером N.

Integer29°. Даны целые положительные числа A, B, C. На прямоугольнике размера $A \times B$ размещено максимально возможное количество квадратов со стороной C (без наложений). Найти количество квадратов, размещенных на прямоугольнике, а также площадь незанятой части прямоугольника.

Integer30°. Дан номер некоторого года (целое положительное число). Определить соответствующий ему номер столетия, учитывая, что, к примеру, началом 20 столетия был 1901 год.

Тема 1. Основы C#: синтаксис, типы данных, консольный ввод-вывод
Практическая работа 3 Логические выражения

Во всех заданиях данной группы требуется вывести логическое значение TRUE, если приведенное высказывание для предложенных исходных данных является истинным, и значение FALSE в противном случае. Все числа, для которых указано количество цифр (двузначное число, трехзначное число и т. д.), считаются целыми положительными.

Boolean1°. Дано целое число A. Проверить истинность высказывания: «Число A является положительным».

Boolean2°. Дано целое число A. Проверить истинность высказывания: «Число A является нечетным».

Boolean3°. Дано целое число A. Проверить истинность высказывания: «Число A является четным».

Boolean4°. Даны два целых числа: A, B. Проверить истинность высказывания: «Справедливы неравенства $A > 2$ и $B \le 3$».

Boolean5°. Даны два целых числа: A, B. Проверить истинность высказывания: «Справедливы неравенства $A \ge 0$ или $B < -2$».

Boolean6°. Даны три целых числа: A, B, C. Проверить истинность высказывания: «Справедливо двойное неравенство $A < B < C$».

Boolean7°. Даны три целых числа: A, B, C. Проверить истинность высказывания: «Число B находится между числами A и C».

Boolean8°. Даны два целых числа: A, B. Проверить истинность высказывания: «Каждое из чисел A и B нечетное».

Boolean9°. Даны два целых числа: A, B. Проверить истинность высказывания: «Хотя бы одно из чисел A и B нечетное».

Boolean10°. Даны два целых числа: A, B. Проверить истинность высказывания: «Ровно одно из чисел A и B нечетное».

Boolean11°. Даны два целых числа: A, B. Проверить истинность высказывания: «Числа A и B имеют одинаковую четность».

Boolean12°. Даны три целых числа: A, B, C. Проверить истинность высказывания: «Каждое из чисел A, B, C положительное».

Boolean13°. Даны три целых числа: A, B, C. Проверить истинность высказывания: «Хотя бы одно из чисел A, B, C положительное».

Boolean14°. Даны три целых числа: A, B, C. Проверить истинность высказывания: «Ровно одно из чисел A, B, C положительное».

Boolean15°. Даны три целых числа: A, B, C. Проверить истинность высказывания: «Ровно два из чисел A, B, C являются положительными».

Boolean16°. Дано целое положительное число. Проверить истинность выска-

зывания: «Данное число является четным двузначным».

Boolean17°. Дано целое положительное число. Проверить истинность высказывания: «Данное число является нечетным трехзначным».

Boolean18°. Проверить истинность высказывания: «Среди трех данных целых чисел есть хотя бы одна пара совпадающих».

Boolean19°. Проверить истинность высказывания: «Среди трех данных целых чисел есть хотя бы одна пара взаимно противоположных».

Boolean20°. Дано трехзначное число. Проверить истинность высказывания: «Все цифры данного числа различны».

Boolean21°. Дано трехзначное число. Проверить истинность высказывания: «Цифры данного числа образуют возрастающую последовательность».

Boolean22°. Дано трехзначное число. Проверить истинность высказывания: «Цифры данного числа образуют возрастающую или убывающую последовательность».

Boolean23°. Дано четырехзначное число. Проверить истинность высказывания: «Данное число читается одинаково слева направо и справа налево».

Boolean24°. Даны числа A, B, C (число A не равно 0). Рассмотрев *дискриминант* $D = B^2 - 4 \cdot A \cdot C$, проверить истинность высказывания: «Квадратное уравнение $A \cdot x^2 + B \cdot x + C = 0$ имеет вещественные корни».

Boolean25°. Даны числа x, y. Проверить истинность высказывания: «Точка с координатами (x, y) лежит во второй координатной четверти».

Boolean26°. Даны числа x, y. Проверить истинность высказывания: «Точка с координатами (x, y) лежит в четвертой координатной четверти».

Boolean27°. Даны числа x, y. Проверить истинность высказывания: «Точка с координатами (x, y) лежит во второй или третьей координатной четверти».

Boolean28°. Даны числа x, y. Проверить истинность высказывания: «Точка с координатами (x, y) лежит в первой или третьей координатной четверти».

Boolean29°. Даны числа x, y, x_1, y_1, x_2, y_2. Проверить истинность высказывания: «Точка с координатами (x, y) лежит внутри прямоугольника, левая верхняя вершина которого имеет координаты (x_1, y_1), правая нижняя — (x_2, y_2), а стороны параллельны координатным осям».

Boolean30°. Даны целые числа a, b, c, являющиеся сторонами некоторого треугольника. Проверить истинность высказывания: «Треугольник со сторонами a, b, c является равносторонним».

Boolean31°. Даны целые числа a, b, c, являющиеся сторонами некоторого треугольника. Проверить истинность высказывания: «Треугольник со сторонами a, b, c является равнобедренным».

Boolean32°. Даны целые числа a, b, c, являющиеся сторонами некоторого треугольника. Проверить истинность высказывания: «Треугольник со сторо-

нами a, b, c является прямоугольным».

Boolean33°. Даны целые числа a, b, c. Проверить истинность высказывания: «Существует треугольник со сторонами a, b, c».

Boolean34°. Даны координаты поля шахматной доски x, y (целые числа, лежащие в диапазоне 1–8). Учитывая, что левое нижнее поле доски (1, 1) является черным, проверить истинность высказывания: «Данное поле является белым».

Boolean35°. Даны координаты двух различных полей шахматной доски x_1, y_1, x_2, y_2 (целые числа, лежащие в диапазоне 1–8). Проверить истинность высказывания: «Данные поля имеют одинаковый цвет».

Boolean36°. Даны координаты двух различных полей шахматной доски x_1, y_1, x_2, y_2 (целые числа, лежащие в диапазоне 1–8). Проверить истинность высказывания: «Ладья за один ход может перейти с одного поля на другое».

Boolean37°. Даны координаты двух различных полей шахматной доски x_1, y_1, x_2, y_2 (целые числа, лежащие в диапазоне 1–8). Проверить истинность высказывания: «Король за один ход может перейти с одного поля на другое».

Boolean38°. Даны координаты двух различных полей шахматной доски x_1, y_1, x_2, y_2 (целые числа, лежащие в диапазоне 1–8). Проверить истинность высказывания: «Слон за один ход может перейти с одного поля на другое».

Boolean39°. Даны координаты двух различных полей шахматной доски x_1, y_1, x_2, y_2 (целые числа, лежащие в диапазоне 1–8). Проверить истинность высказывания: «Ферзь за один ход может перейти с одного поля на другое».

Boolean40°. Даны координаты двух различных полей шахматной доски x_1, y_1, x_2, y_2 (целые числа, лежащие в диапазоне 1–8). Проверить истинность высказывания: «Конь за один ход может перейти с одного поля на другое».

Тема 1. Основы C#: синтаксис, типы данных, консольный ввод-вывод
Практическая работа 4 Условный оператор

If1. Дано целое число. Если оно является положительным, то прибавить к нему 1; в противном случае не изменять его. Вывести полученное число.

If2. Дано целое число. Если оно является положительным, то прибавить к нему 1; в противном случае вычесть из него 2. Вывести полученное число.

If3. Дано целое число. Если оно является положительным, то прибавить к нему 1; если отрицательным, то вычесть из него 2; если нулевым, то заменить его на 10. Вывести полученное число.

If4°. Даны три целых числа. Найти количество положительных чисел в исходном наборе.

If5. Даны три целых числа. Найти количество положительных и количество отрицательных чисел в исходном наборе.

If6°. Даны два числа. Вывести большее из них.

If7. Даны два числа. Вывести порядковый номер меньшего из них.

If8°. Даны два числа. Вывести вначале большее, а затем меньшее из них.

If9. Даны две переменные вещественного типа: *A*, *B*. Перераспределить значения данных переменных так, чтобы в *A* оказалось меньшее из значений, а в *B* — большее. Вывести новые значения переменных *A* и *B*.

If10. Даны две переменные целого типа: *A* и *B*. Если их значения не равны, то присвоить каждой переменной сумму этих значений, а если равны, то присвоить переменным нулевые значения. Вывести новые значения переменных *A* и *B*.

If11. Даны две переменные целого типа: *A* и *B*. Если их значения не равны, то присвоить каждой переменной большее из этих значений, а если равны, то присвоить переменным нулевые значения. Вывести новые значения переменных *A* и *B*.

If12°. Даны три числа. Найти наименьшее из них.

If13. Даны три числа. Найти среднее из них (то есть число, расположенное между наименьшим и наибольшим).

If14. Даны три числа. Вывести вначале наименьшее, а затем наибольшее из данных чисел.

If15. Даны три числа. Найти сумму двух наибольших из них.

If16. Даны три переменные вещественного типа: *A*, *B*, *C*. Если их значения упорядочены по возрастанию, то удвоить их; в противном случае заменить значение каждой переменной на противоположное. Вывести новые значения переменных *A*, *B*, *C*.

If17. Даны три переменные вещественного типа: *A*, *B*, *C*. Если их значения

упорядочены по возрастанию или убыванию, то удвоить их; в противном случае заменить значение каждой переменной на противоположное. Вывести новые значения переменных *A*, *B*, *C*.

If18. Даны три целых числа, одно из которых отлично от двух других, равных между собой. Определить порядковый номер числа, отличного от остальных.

If19. Даны четыре целых числа, одно из которых отлично от трех других, равных между собой. Определить порядковый номер числа, отличного от остальных.

If20. На числовой оси расположены три точки: *A*, *B*, *C*. Определить, какая из двух последних точек (*B* или *C*) расположена ближе к *A*, и вывести эту точку и ее расстояние от точки *A*.

If21. Даны целочисленные координаты точки на плоскости. Если точка совпадает с началом координат, то вывести 0. Если точка не совпадает с началом координат, но лежит на оси *OX* или *OY*, то вывести соответственно 1 или 2. Если точка не лежит на координатных осях, то вывести 3.

If22°. Даны координаты точки, не лежащей на координатных осях *OX* и *OY*. Определить номер координатной четверти, в которой находится данная точка.

If23. Даны целочисленные координаты трех вершин прямоугольника, стороны которого параллельны координатным осям. Найти координаты его четвертой вершины.

If24. Для данного вещественного x найти значение следующей функции f, принимающей вещественные значения:

$$f(x) = \begin{cases} 2\cdot\sin(x), & \text{если } x > 0, \\ 6 - x, & \text{если } x \le 0. \end{cases}$$

If25. Для данного целого x найти значение следующей функции f, принимающей значения целого типа:

$$f(x) = \begin{cases} 2\cdot x, & \text{если } x < -2 \text{ или } x > 2, \\ -3\cdot x, & \text{в противном случае.} \end{cases}$$

If26°. Для данного вещественного x найти значение следующей функции f, принимающей вещественные значения:

$$f(x) = \begin{cases} -x, & \text{если } x \le 0, \\ x^2, & \text{если } 0 < x < 2, \\ 4, & \text{если } x \ge 2. \end{cases}$$

If27. Для данного вещественного x найти значение следующей функции f, принимающей значения целого типа:

$$0, \quad \text{если } x < 0,$$

$$f(x) = \begin{cases} 1, & \text{если } x \text{ принадлежит } [0,1), [2,3), \ldots, \\ -1, & \text{если } x \text{ принадлежит } [1,2), [3,4), \ldots. \end{cases}$$

If28. Дан номер года (положительное целое число). Определить количество дней в этом году, учитывая, что обычный год насчитывает 365 дней, а високосный — 366 дней. Високосным считается год, делящийся на 4, за исключением тех годов, которые делятся на 100 и не делятся на 400 (например, годы 300, 1300 и 1900 не являются високосными, а 1200 и 2000 — являются).

If29. Дано целое число. Вывести его строку-описание вида «отрицательное четное число», «нулевое число», «положительное нечетное число» и т. д.

If30. Дано целое число, лежащее в диапазоне 1–999. Вывести его строку-описание вида «четное двузначное число», «нечетное трехзначное число» и т. д.

Тема 2 Управление потоком: циклы и последовательности
Практическая работа 5 Оператор выбора

Case1. Дано целое число в диапазоне 1–7. Вывести строку — название дня недели, соответствующее данному числу (1 — «понедельник», 2 — «вторник» и т. д.).

Case2. Дано целое число K. Вывести строку-описание оценки, соответствующей числу K (1 — «плохо», 2 — «неудовлетворительно», 3 — «удовлетворительно», 4 — «хорошо», 5 — «отлично»). Если K не лежит в диапазоне 1–5, то вывести строку «ошибка».

Case3. Дан номер месяца — целое число в диапазоне 1–12 (1 — январь, 2 — февраль и т. д.). Вывести название соответствующего времени года («зима», «весна», «лето», «осень»).

Case4°. Дан номер месяца — целое число в диапазоне 1–12 (1 — январь, 2 — февраль и т. д.). Определить количество дней в этом месяце для невисокосного года.

Case5. Арифметические действия над числами пронумерованы следующим образом: 1 — сложение, 2 — вычитание, 3 — умножение, 4 — деление. Дан номер действия N (целое число в диапазоне 1–4) и вещественные числа A и B (B не равно 0). Выполнить над числами указанное действие и вывести результат.

Case6. Единицы длины пронумерованы следующим образом: 1 — дециметр, 2 — километр, 3 — метр, 4 — миллиметр, 5 — сантиметр. Дан номер единицы длины (целое число в диапазоне 1–5) и длина отрезка в этих единицах (вещественное число). Найти длину отрезка в метрах.

Case7. Единицы массы пронумерованы следующим образом: 1 — килограмм, 2 — миллиграмм, 3 — грамм, 4 — тонна, 5 — центнер. Дан номер единицы массы (целое число в диапазоне 1–5) и масса тела в этих единицах (вещественное число). Найти массу тела в килограммах.

Case8. Даны два целых числа: D (день) и M (месяц), определяющие правильную дату невисокосного года. Вывести значения D и M для даты, предшествующей указанной.

Case9°. Даны два целых числа: D (день) и M (месяц), определяющие правильную дату невисокосного года. Вывести значения D и M для даты, следующей за указанной.

Case10. Робот может перемещаться в четырех направлениях («С» — север, «З» — запад, «Ю» — юг, «В» — восток) и принимать три цифровые команды: 0 — продолжать движение, 1 — поворот налево, -1 — поворот направо. Дан символ C — исходное направление робота и целое число N

— посланная ему команда. Вывести направление робота после выполнения полученной команды.

Case11. Локатор ориентирован на одну из сторон света («С» — север, «З» — запад, «Ю» — юг, «В» — восток) и может принимать три цифровые команды поворота: 1 — поворот налево, −1 — поворот направо, 2 — поворот на 180°. Дан символ C — исходная ориентация локатора и целые числа N_1 и N_2 — две посланные команды. Вывести ориентацию локатора после выполнения этих команд.

Case12. Элементы окружности пронумерованы следующим образом: 1 — радиус R, 2 — диаметр $D = 2 \cdot R$, 3 — длина $L = 2 \cdot \pi \cdot R$, 4 — площадь круга $S = \pi \cdot R^2$. Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данной окружности (в том же порядке). В качестве значения π использовать 3.14.

Case13. Элементы равнобедренного прямоугольного треугольника пронумерованы следующим образом: 1 — катет a, 2 — гипотенуза $c = a \cdot \sqrt{2}$, 3 — высота h, опущенная на гипотенузу ($h = c/2$), 4 — площадь $S = c \cdot h/2$. Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данного треугольника (в том же порядке).

Case14. Элементы равностороннего треугольника пронумерованы следующим образом: 1 — сторона a, 2 — радиус R_1 вписанной окружности ($R_1 = a \cdot \sqrt{3}/6$), 3 — радиус R_2 описанной окружности ($R_2 = 2 \cdot R_1$), 4 — площадь $S = a^2 \cdot \sqrt{3}/4$. Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данного треугольника (в том же порядке).

Case15. Мастям игральных карт присвоены порядковые номера: 1 — пики, 2 — трефы, 3 — бубны, 4 — червы. Достоинству карт, старших десятки, присвоены номера: 11 — валет, 12 — дама, 13 — король, 14 — туз. Даны два целых числа: N — достоинство ($6 \le N \le 14$) и M — масть карты ($1 \le M \le 4$). Вывести название соответствующей карты вида «шестерка бубен», «дама червей», «туз треф» и т. п.

Case16. Дано целое число в диапазоне 20–69, определяющее возраст (в годах). Вывести строку-описание указанного возраста, обеспечив правильное согласование числа со словом «год», например: 20 — «двадцать лет», 32 — «тридцать два года», 41 — «сорок один год».

Case17. Дано целое число в диапазоне 10–40, определяющее количество учебных заданий по некоторой теме. Вывести строку-описание указанного количества заданий, обеспечив правильное согласование числа со словами «учебное задание», например: 18 — «восемнадцать учебных заданий»,

23 — «двадцать три учебных задания», 31 — «тридцать одно учебное задание».

Case18. Дано целое число в диапазоне 100–999. Вывести строку-описание данного числа, например: 256 — «двести пятьдесят шесть», 814 — «восемьсот четырнадцать».

Case19. В восточном календаре принят 60-летний цикл, состоящий из 12-летних подциклов, обозначаемых названиями цвета: зеленый, красный, желтый, белый и черный. В каждом подцикле годы носят названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. По номеру года определить его название, если 1984 год — начало цикла: «год зеленой крысы».

Case20. Даны два целых числа: *D* (день) и *M* (месяц), определяющие правильную дату. Вывести знак Зодиака, соответствующий этой дате: «Водолей» (20.1–18.2), «Рыбы» (19.2–20.3), «Овен» (21.3–19.4), «Телец» (20.4–20.5), «Близнецы» (21.5–21.6), «Рак» (22.6–22.7), «Лев» (23.7–22.8), «Дева» (23.8–22.9), «Весы» (23.9–22.10), «Скорпион» (23.10–22.11), «Стрелец» (23.11–21.12), «Козерог» (22.12–19.1).

Тема 2 Управление потоком: циклы и последовательности
Практическая работа 6 Цикл с параметром

For1. Даны целые числа K и N ($N > 0$). Вывести N раз число K.

For2. Даны два целых числа A и B ($A < B$). Вывести в порядке возрастания все целые числа, расположенные между A и B (включая сами числа A и B), а также количество N этих чисел.

For3. Даны два целых числа A и B ($A < B$). Вывести в порядке убывания все целые числа, расположенные между A и B (не включая числа A и B), а также количество N этих чисел.

For4. Дано вещественное число — цена 1 кг конфет. Вывести стоимость 1, 2, . . . , 10 кг конфет.

For5°. Дано вещественное число — цена 1 кг конфет. Вывести стоимость 0.1, 0.2, . . . , 1 кг конфет.

For6. Дано вещественное число — цена 1 кг конфет. Вывести стоимость 1.2, 1.4, . . . , 2 кг конфет.

For7. Даны два целых числа A и B ($A < B$). Найти сумму всех целых чисел от A до B включительно.

For8. Даны два целых числа A и B ($A < B$). Найти произведение всех целых чисел от A до B включительно.

For9. Даны два целых числа A и B ($A < B$). Найти сумму квадратов всех целых чисел от A до B включительно.

For10. Дано целое число N (> 0). Найти сумму

$$1 + 1/2 + 1/3 + \ldots + 1/N$$

(вещественное число).

For11. Дано целое число N (> 0). Найти сумму

$$N^2 + (N + 1)^2 + (N + 2)^2 + \ldots + (2 \cdot N)^2$$

(целое число).

For12°. Дано целое число N (> 0). Найти произведение

$$1.1 \cdot 1.2 \cdot 1.3 \cdot \ldots$$

(N сомножителей).

For13°. Дано целое число N (> 0). Найти значение выражения

$$1.1 - 1.2 + 1.3 - \ldots$$

(N слагаемых, знаки чередуются). Условный оператор не использовать.

For14. Дано целое число N (> 0). Найти квадрат данного числа, используя для его вычисления следующую формулу:

$$N^2 = 1 + 3 + 5 + \ldots + (2 \cdot N - 1).$$

После добавления к сумме каждого слагаемого выводить текущее значе-

ние суммы (в результате будут выведены квадраты всех целых чисел от 1 до N).

For15°. Дано вещественное число A и целое число N (>0). Найти A в степени N:

$$A^N = A \cdot A \cdot \ldots \cdot A$$

(числа A перемножаются N раз).

For16°. Дано вещественное число A и целое число N (>0). Используя один цикл, вывести все целые степени числа A от 1 до N.

For17. Дано вещественное число A и целое число N (>0). Используя один цикл, найти сумму

$$1 + A + A^2 + A^3 + \ldots + A^N.$$

For18. Дано вещественное число A и целое число N (>0). Используя один цикл, найти значение выражения

$$1 - A + A^2 - A^3 + \ldots + (-1)^N \cdot A^N.$$

Условный оператор не использовать.

For19°. Дано целое число N (>0). Найти произведение

$$N! = 1 \cdot 2 \cdot \ldots \cdot N$$

(*N–факториал*). Чтобы избежать целочисленного переполнения, вычислять это произведение с помощью вещественной переменной и вывести его как вещественное число.

For20°. Дано целое число N (>0). Используя один цикл, найти сумму

$$1! + 2! + 3! + \ldots + N!$$

(выражение $N!$ — *N–факториал* — обозначает произведение всех целых чисел от 1 до N: $N! = 1 \cdot 2 \cdot \ldots \cdot N$). Чтобы избежать целочисленного переполнения, проводить вычисления с помощью вещественных переменных и вывести результат как вещественное число.

For21. Дано целое число N (>0). Используя один цикл, найти сумму

$$1 + 1/(1!) + 1/(2!) + 1/(3!) + \ldots + 1/(N!)$$

(выражение $N!$ — *N–факториал* — обозначает произведение всех целых чисел от 1 до N: $N! = 1 \cdot 2 \cdot \ldots \cdot N$). Полученное число является приближенным значением константы $e = \exp(1)$.

For22. Дано вещественное число X и целое число N (>0). Найти значение выражения

$$1 + X + X^2/(2!) + \ldots + X^N/(N!)$$

($N! = 1 \cdot 2 \cdot \ldots \cdot N$). Полученное число является приближенным значением функции exp в точке X.

For23. Дано вещественное число X и целое число N (>0). Найти значение выражения

$$X - X^3/(3!) + X^5/(5!) - \ldots + (-1)^N \cdot X^{2 \cdot N+1}/((2 \cdot N+1)!)$$

($N! = 1 \cdot 2 \cdot \ldots \cdot N$). Полученное число является приближенным значением функции sin в точке X.

For24. Дано вещественное число X и целое число N (> 0). Найти значение выражения

$$1 - X^2/(2!) + X^4/(4!) - \ldots + (-1)^N \cdot X^{2 \cdot N}/((2 \cdot N)!)$$

($N! = 1 \cdot 2 \cdot \ldots \cdot N$). Полученное число является приближенным значением функции cos в точке X.

For25. Дано вещественное число X ($|X| < 1$) и целое число N (> 0). Найти значение выражения

$$X - X^2/2 + X^3/3 - \ldots + (-1)^{N-1} \cdot X^N/N.$$

Полученное число является приближенным значением функции ln в точке $1 + X$.

For26. Дано вещественное число X ($|X| < 1$) и целое число N (> 0). Найти

значение выражения

$$X - X^3/3 + X^5/5 - \ldots + (-1)^N \cdot X^{2 \cdot N+1}/(2 \cdot N+1).$$

Полученное число является приближенным значением функции arctg в точке X.

For27. Дано вещественное число X ($|X| < 1$) и целое число N (> 0). Найти значение выражения

$$X + 1 \cdot X^3/(2 \cdot 3) + 1 \cdot 3 \cdot X^5/(2 \cdot 4 \cdot 5) + \ldots +$$
$$+ 1 \cdot 3 \cdot \ldots \cdot (2 \cdot N - 1) \cdot X^{2 \cdot N+1}/(2 \cdot 4 \cdot \ldots \cdot (2 \cdot N) \cdot (2 \cdot N+1)).$$

Полученное число является приближенным значением функции arcsin в точке X.

For28. Дано вещественное число X ($|X| < 1$) и целое число N (> 0). Найти значение выражения

$$1 + X/2 - 1 \cdot X^2/(2 \cdot 4) + 1 \cdot 3 \cdot X^3/(2 \cdot 4 \cdot 6) - \ldots +$$
$$+ (-1)^{N-1} \cdot 1 \cdot 3 \cdot \ldots \cdot (2 \cdot N - 3) \cdot X^N/(2 \cdot 4 \cdot \ldots \cdot (2 \cdot N)).$$

Полученное число является приближенным значением функции $\sqrt{1+X}$.

For29. Дано целое число N (> 1) и две вещественные точки на числовой оси: A, B ($A < B$). Отрезок $[A, B]$ разбит на N равных отрезков. Вывести H — длину каждого отрезка, а также набор точек

$$A, \quad A + H, \quad A + 2 \cdot H, \quad A + 3 \cdot H, \quad \ldots, \quad B,$$

образующий разбиение отрезка $[A, B]$.

For30. Дано целое число N (> 1) и две вещественные точки на числовой оси: A, B ($A < B$). Отрезок $[A, B]$ разбит на N равных отрезков. Вывести H — длину каждого отрезка, а также значения функции $F(X) = 1 - \sin(X)$ в точках, разбивающих отрезок $[A, B]$:

$$F(A), \quad F(A + H), \quad F(A + 2 \cdot H), \quad \ldots, \quad F(B).$$

For31. Дано целое число N (> 0). Последовательность вещественных чисел A_K определяется следующим образом:

$$A_0 = 2, \qquad A_K = 2 + 1/A_{K-1}, \qquad K = 1, 2, \ldots .$$

Вывести элементы $A_1, A_2, \ldots, A_N$.

For32. Дано целое число N (> 0). Последовательность вещественных чисел A_K определяется следующим образом:

$$A_0 = 1, \qquad A_K = (A_{K-1} + 1)/K, \qquad K = 1, 2, \ldots .$$

Вывести элементы $A_1, A_2, \ldots, A_N$.

For33°. Дано целое число N (> 1). Последовательность *чисел Фибоначчи* F_K (целого типа) определяется следующим образом:

$$F_1 = 1, \qquad F_2 = 1, \qquad F_K = F_{K-2} + F_{K-1}, \qquad K = 3, 4, \ldots .$$

Вывести элементы F_1, F_2, ..., F_N.

For34. Дано целое число N (> 1). Последовательность вещественных чисел A_K определяется следующим образом:

$$A_1 = 1, \qquad A_2 = 2, \qquad A_K = (A_{K-2} + 2 \cdot A_{K-1})/3, \qquad K = 3, 4, \ldots .$$

Вывести элементы A_1, A_2, . . . , A_N.

For35. Дано целое число N (> 2). Последовательность целых чисел A_K определяется следующим образом:

$$A_1 = 1, \qquad A_2 = 2, \qquad A_3 = 3,$$
$$A_K = A_{K-1} + A_{K-2} - 2 \cdot A_{K-3}, \qquad K = 4, 5, \ldots .$$

Вывести элементы A_1, A_2, . . . , A_N.

Вложенные циклы

For36°. Даны целые положительные числа N и K. Найти сумму

$$1^K + 2^K + \quad + N^K.$$

Чтобы избежать целочисленного переполнения, вычислять слагаемые этой суммы с помощью вещественной переменной и выводить результат как вещественное число.

For37. Дано целое число N (> 0). Найти сумму

$$1^1 + 2^2 + \quad + N^N.$$

Чтобы избежать целочисленного переполнения, вычислять слагаемые этой суммы с помощью вещественной переменной и выводить результат как вещественное число.

For38. Дано целое число N (> 0). Найти сумму

$$1^N + 2^{N-1} + \quad + N^1.$$

Чтобы избежать целочисленного переполнения, вычислять слагаемые этой суммы с помощью вещественной переменной и выводить результат как вещественное число.

For39. Даны целые положительные числа A и B ($A < B$). Вывести все целые числа от A до B включительно; при этом каждое число должно выводиться столько раз, каково его значение (например, число 3 выводится 3 раза).

For40. Даны целые числа A и B ($A < B$). Вывести все целые числа от A до B включительно; при этом число A должно выводиться 1 раз, число $A + 1$ должно выводиться 2 раза и т. д.

Тема 2. Управление потоком: циклы и последовательности
Практическая работа 7 Цикл с условием

While1°. Даны положительные числа A и B ($A > B$). На отрезке длины A размещено максимально возможное количество отрезков длины B (без наложений). Не используя операции умножения и деления, найти длину незанятой части отрезка A.

While2°. Даны положительные числа A и B ($A > B$). На отрезке длины A размещено максимально возможное количество отрезков длины B (без наложений). Не используя операции умножения и деления, найти количество отрезков B, размещенных на отрезке A.

While3. Даны целые положительные числа N и K. Используя только операции сложения и вычитания, найти частное от деления нацело N на K, а также остаток от этого деления.

While4°. Дано целое число N (> 0). Если оно является степенью числа 3, то вывести TRUE, если не является — вывести FALSE.

While5. Дано целое число N (> 0), являющееся некоторой степенью числа 2: $N = 2^K$. Найти целое число K — показатель этой степени.

While6. Дано целое число N (> 0). Найти *двойной факториал* N:

$$N!! = N \cdot (N-2) \cdot (N-4) \cdot \ldots$$

(последний сомножитель равен 2, если N — четное, и 1, если N — нечетное). Чтобы избежать целочисленного переполнения, вычислять это произведение с помощью вещественной переменной и вывести его как вещественное число.

While7°. Дано целое число N (> 0). Найти наименьшее целое положительное число K, квадрат которого превосходит N: $K^2 > N$. Функцию извлечения квадратного корня не использовать.

While8. Дано целое число N (> 0). Найти наибольшее целое число K, квадрат которого не превосходит N: $K^2 \leq N$. Функцию извлечения квадратного корня не использовать.

While9. Дано целое число N (> 1). Найти наименьшее целое число K, при котором выполняется неравенство $3^K > N$.

While10. Дано целое число N (> 1). Найти наибольшее целое число K, при котором выполняется неравенство $3^K < N$.

While11°. Дано целое число N (> 1). Вывести наименьшее из целых чисел K, для которых сумма $1 + 2 + \ldots + K$ будет больше или равна N, и саму эту сумму.

While12°. Дано целое число N (> 1). Вывести наибольшее из целых чисел K, для которых сумма $1 + 2 + \ldots + K$ будет меньше или равна N, и саму эту сумму.

While13. Дано число A (> 1). Вывести наименьшее из целых чисел K, для которых сумма $1 + 1/2 + \ldots + 1/K$ будет больше A, и саму эту сумму.

While14. Дано число A (> 1). Вывести наибольшее из целых чисел K, для которых сумма $1 + 1/2 + \ldots + 1/K$ будет меньше A, и саму эту сумму.

While15. Начальный вклад в банке равен 1000 руб. Через каждый месяц размер вклада увеличивается на P процентов от имеющейся суммы (P — вещественное число, $0 < P < 25$). По данному P определить, через сколько месяцев размер вклада превысит 1100 руб., и вывести найденное количество месяцев K (целое число) и итоговый размер вклада S (вещественное число).

While16. Спортсмен-лыжник начал тренировки, пробежав в первый день 10 км. Каждый следующий день он увеличивал длину пробега на P процентов от пробега предыдущего дня (P — вещественное, $0 < P < 50$). По данному P определить, после какого дня суммарный пробег лыжника за все дни превысит 200 км, и вывести найденное количество дней K (целое) и суммарный пробег S (вещественное число).

While17. Дано целое число N (> 0). Используя операции деления нацело и взятия остатка от деления, вывести все его цифры, начиная с самой правой (разряда единиц).

While18. Дано целое число N (> 0). Используя операции деления нацело и взятия остатка от деления, найти количество и сумму его цифр.

While19. Дано целое число N (> 0). Используя операции деления нацело и взятия остатка от деления, найти число, полученное при прочтении числа N справа налево.

While20. Дано целое число N (> 0). С помощью операций деления нацело и взятия остатка от деления определить, имеется ли в записи числа N цифра «2». Если имеется, то вывести TRUE, если нет — вывести FALSE.

While21. Дано целое число N (> 0). С помощью операций деления нацело и взятия остатка от деления определить, имеются ли в записи числа N нечетные цифры. Если имеются, то вывести TRUE, если нет — вывести FALSE.

While22°. Дано целое число N (> 1). Если оно является *простым*, то есть не имеет положительных делителей, кроме 1 и самого себя, то вывести TRUE,

иначе вывести FALSE.

While23°. Даны целые положительные числа A и B. Найти их *наибольший общий делитель* (НОД), используя *алгоритм Евклида*:

$$\text{НОД}(A, B) = \text{НОД}(B, A \bmod B), \quad \text{если } B \neq 0; \qquad \text{НОД}(A, 0) = A.$$

While24. Дано целое число N (> 1). Последовательность *чисел Фибоначчи* F_K определяется следующим образом:

$$F_1 = 1, \qquad F_2 = 1, \qquad F_K = F_{K-2} + F_{K-1}, \qquad K = 3, 4, \ldots .$$

Проверить, является ли число N числом Фибоначчи. Если является, то вывести TRUE, если нет — вывести FALSE.

While25. Дано целое число N (> 1). Найти первое число Фибоначчи, большее N. (определение *чисел Фибоначчи* дано в задании While24).

While26. Дано целое число N (> 1), являющееся числом Фибоначчи: $N = F_K$ (определение *чисел Фибоначчи* дано в задании While24). Найти целые числа F_{K-1} и F_{K+1} — предыдущее и последующее числа Фибоначчи.

While27. Дано целое число N (> 1), являющееся числом Фибоначчи: $N = F_K$ (определение *чисел Фибоначчи* дано в задании While24). Найти целое число K — порядковый номер числа Фибоначчи N.

While28. Дано вещественное число ε (> 0). Последовательность вещественных чисел A_K определяется следующим образом:

$$A_1 = 2, \qquad A_K = 2 + 1/A_{K-1}, \qquad K = 2, 3, \ldots .$$

Найти первый из номеров K, для которых выполняется условие $|A_K - A_{K-1}| < \varepsilon$, и вывести этот номер, а также числа A_{K-1} и A_K.

While29. Дано вещественное число ε (> 0). Последовательность вещественных чисел A_K определяется следующим образом:

$$A_1 = 1, \qquad A_2 = 2, \qquad A_K = (A_{K-2} + 2\,A_{K-1})/3, \qquad K = 3, 4, \ldots .$$

Найти первый из номеров K, для которых выполняется условие $|A_K - A_{K-1}| < \varepsilon$, и вывести этот номер, а также числа A_{K-1} и A_K.

While30. Даны положительные числа A, B, C. На прямоугольнике размера $A \times B$ размещено максимально возможное количество квадратов со стороной C (без наложений). Найти количество квадратов, размещенных на прямоугольнике. Операции умножения и деления не использовать.

Тема 3. Работа с данными: массивы, строки, рекурсия
Практическая работа 8 Последовательности

Во всех заданиях данной группы предполагается, что исходный набор содержит ненулевое число элементов (в частности, число N всегда большенуля). В заданиях на обработку нескольких наборов чисел (Series29–Series40) количество наборов K также всегда является ненулевым.

Series1°. Даны десять вещественных чисел. Найти их сумму.

Series2. Даны десять вещественных чисел. Найти их произведение.

Series3. Даны десять вещественных чисел. Найти их среднее арифметическое.

Series4. Дано целое число N и набор из N вещественных чисел. Вывести сумму и произведение чисел из данного набора.

Series5. Дано целое число N и набор из N положительных вещественных чисел. Вывести в том же порядке целые части всех чисел из данного набора (как вещественные числа с нулевой дробной частью), а также сумму всех целых частей.

Series6. Дано целое число N и набор из N положительных вещественных чисел. Вывести в том же порядке дробные части всех чисел из данного набора (как вещественные числа с нулевой целой частью), а также произведение всех дробных частей.

Series7. Дано целое число N и набор из N вещественных чисел. Вывести в том же порядке округленные значения всех чисел из данного набора (как целые числа), а также сумму всех округленных значений.

Series8. Дано целое число N и набор из N целых чисел. Вывести в том же порядке все четные числа из данного набора и количество K таких чисел.

Series9. Дано целое число N и набор из N целых чисел. Вывести в том же порядке номера всех нечетных чисел из данного набора и количество K таких чисел.

Series10. Дано целое число N и набор из N целых чисел. Если в наборе имеются положительные числа, то вывести TRUE; в противном случае вывести FALSE.

Series11. Даны целые числа K, N и набор из N целых чисел. Если в наборе имеются числа, меньшие K, то вывести TRUE; в противном случае вывести FALSE.

Series12. Дан набор ненулевых целых чисел; признак его завершения — число 0. Вывести количество чисел в наборе.

Series13. Дан набор ненулевых целых чисел; признак его завершения — число 0. Вывести сумму всех положительных четных чисел из данного набора. Если требуемые числа в наборе отсутствуют, то вывести 0.

Series14. Дано целое число K и набор ненулевых целых чисел; признак его завершения — число 0. Вывести количество чисел в наборе, меньших K.

Series15°. Дано целое число K и набор ненулевых целых чисел; признак его завершения — число 0. Вывести номер первого числа в наборе, большего K. Если таких чисел нет, то вывести 0.

Series16°. Дано целое число K и набор ненулевых целых чисел; признак его завершения — число 0. Вывести номер последнего числа в наборе, большего K. Если таких чисел нет, то вывести 0.

Series17°. Дано вещественное число B, целое число N и набор из N вещественных чисел, упорядоченных по возрастанию. Вывести элементы набора вместе с числом B, сохраняя упорядоченность выводимых чисел.

Series18. Дано целое число N и набор из N целых чисел, упорядоченный по возрастанию. Данный набор может содержать одинаковые элементы. Вывести в том же порядке все различные элементы данного набора.

Series19°. Дано целое число N (> 1) и набор из N целых чисел. Вывести те элементы в наборе, которые меньше своего левого соседа, и количество K таких элементов.

Series20. Дано целое число N (> 1) и набор из N целых чисел. Вывести те элементы в наборе, которые меньше своего правого соседа, и количество K таких элементов.

Series21°. Дано целое число N (> 1) и набор из N вещественных чисел. Проверить, образует ли данный набор возрастающую последовательность. Если образует, то вывести TRUE, если нет — вывести FALSE.

Series22. Дано целое число N (> 1) и набор из N вещественных чисел. Если данный набор образует убывающую последовательность, то вывести 0; в противном случае вывести номер первого числа, нарушающего закономерность.

Series23. Дано целое число N (> 2) и набор из N вещественных чисел. Набор называется *пилообразным*, если каждый его внутренний элемент либо больше, либо меньше обоих своих соседей (то есть является «зубцом»). Если данный набор является пилообразным, то вывести 0; в противном случае вывести номер первого элемента, не являющегося зубцом.

Series24. Дано целое число N и набор из N целых чисел, содержащий по крайней мере два нуля. Вывести сумму чисел из данного набора, расположенных между последними двумя нулями (если последние нули идут подряд, то вывести 0).

Series25. Дано целое число N и набор из N целых чисел, содержащий по крайней мере два нуля. Вывести сумму чисел из данного набора, распо-

ложенных между первым и последним нулем (если первый и последний нули идут подряд, то вывести 0).

Вложенные циклы

Series26. Даны целые числа K, N и набор из N вещественных чисел: A_1, A_2, . . ., A_N. Вывести K-е степени чисел из данного набора:

$$(A_1)^K, (A_2)^K, \ldots, (A_N)^K.$$

Series27. Дано целое число N и набор из N вещественных чисел: A_1, A_2, . . ., A_N. Вывести следующие числа:

$$A_1, (A_2)^2, \ldots, (A_{N-1})^{N-1}, (A_N)^N.$$

Series28. Дано целое число N и набор из N вещественных чисел: A_1, A_2, . . ., A_N. Вывести следующие числа:

$$(A_1)^N, (A_2)^{N-1}, \ldots, (A_{N-1})^2, A_N.$$

Series29. Даны целые числа K, N, а также K наборов целых чисел по N элементов в каждом наборе. Вывести общую сумму всех элементов, входящих в данные наборы.

Series30°. Даны целые числа K, N, а также K наборов целых чисел по N элементов в каждом наборе. Для каждого набора вывести сумму его элементов.

Series31. Даны целые числа K, N, а также K наборов целых чисел по N элементов в каждом наборе. Найти количество наборов, содержащих число 2. Если таких наборов нет, то вывести 0.

Series32. Даны целые числа K, N, а также K наборов целых чисел по N элементов в каждом наборе. Для каждого набора вывести номер его первого элемента, равного 2, или число 0, если в данном наборе нет двоек.

Series33. Даны целые числа K, N, а также K наборов целых чисел по N элементов в каждом наборе. Для каждого набора вывести номер его последнего элемента, равного 2, или число 0, если в данном наборе нет двоек.

Series34. Даны целые числа K, N, а также K наборов целых чисел по N элементов в каждом наборе. Для каждого набора выполнить следующее действие: если в наборе содержится число 2, то вывести сумму его элементов; если в наборе нет двоек, то вывести 0.

Series35. Дано целое число K, а также K наборов ненулевых целых чисел. Признаком завершения каждого набора является число 0. Для каждого набора вывести количество его элементов. Вывести также общее количество элементов во всех наборах.

Series36. Дано целое число *K*, а также *K* наборов ненулевых целых чисел. Каждый набор содержит не менее двух элементов, признаком его завершения является число 0. Найти количество наборов, элементы которых возрастают.

Series37. Дано целое число *K*, а также *K* наборов ненулевых целых чисел. Каждый набор содержит не менее двух элементов, признаком его завершения является число 0. Найти количество наборов, элементы которых возрастают или убывают.

Series38. Дано целое число *K*, а также *K* наборов ненулевых целых чисел. Каждый набор содержит не менее двух элементов, признаком его завершения является число 0. Для каждого набора выполнить следующее действие: если элементы набора возрастают, то вывести 1; если элементы набора убывают, то вывести – 1; если элементы набора не возрастают и не убывают, то вывести 0.

Series39. Дано целое число *K*, а также *K* наборов ненулевых целых чисел. Каждый набор содержит не менее трех элементов, признаком его завершения является число 0. Найти количество пилообразных наборов (определение пилообразного набора дано в задании Series23).

Series40. Дано целое число *K*, а также *K* наборов ненулевых целых чисел. Каждый набор содержит не менее трех элементов, признаком его завершения является число 0. Для каждого набора выполнить следующее действие: если набор является пилообразным (см. задание Series23), то вывести количество его элементов; в противном случае вывести номер первого элемента, который не является зубцом.

Тема 3. Работа с данными: массивы, строки, рекурсия
Практическая работа 9 Минимумы и максимумы

Для решения заданий из данной группы следует использовать «однопроходные» алгоритмы, позволяющие получить требуемый результат после *однократного* просмотра набора исходных данных. Однопроходные алгоритмы обладают важным преимуществом: для них не требуется хранить в памяти одновременно весь набор данных, поэтому при программной реализации этих алгоритмов *можно не использовать массивы*.

Во всех заданиях данной группы предполагается, что исходный набор содержит ненулевое количество элементов (в частности, число N всегда больше нуля).

Minmax1°. Дано целое число N и набор из N чисел. Найти минимальный и максимальный из элементов данного набора и вывести их в указанном порядке.

Minmax2. Дано целое число N и набор из N прямоугольников, заданных своими сторонами — парами чисел (a, b). Найти минимальную площадь прямоугольника из данного набора.

Minmax3. Дано целое число N и набор из N прямоугольников, заданных своими сторонами — парами чисел (a, b). Найти максимальный периметр прямоугольника из данного набора.

Minmax4. Дано целое число N и набор из N чисел. Найти номер минимального элемента из данного набора.

Minmax5. Дано целое число N и набор из N пар чисел (m, v) — данные о массе m и объеме v деталей, изготовленных из различных материалов. Вывести номер детали, изготовленной из материала максимальной плотности, а также величину этой максимальной плотности. Плотность P вычисляется по формуле

$$P = m/v.$$

Minmax6°. Дано целое число N и набор из N целых чисел. Найти номера первого минимального и последнего максимального элемента из данного набора и вывести их в указанном порядке.

Minmax7. Дано целое число N и набор из N целых чисел. Найти номера первого максимального и последнего минимального элемента из данного набора и вывести их в указанном порядке.

Minmax8. Дано целое число N и набор из N целых чисел. Найти номера первого и последнего минимального элемента из данного набора и вывести их в указанном порядке.

Minmax9. Дано целое число N и набор из N целых чисел. Найти номера пер-

вого и последнего максимального элемента из данного набора и вывести их в указанном порядке.

Minmax10. Дано целое число N и набор из N целых чисел. Найти номер первого *экстремального* (то есть минимального или максимального) элемента из данного набора.

Minmax11. Дано целое число N и набор из N целых чисел. Найти номер последнего *экстремального* (то есть минимального или максимального) элемента из данного набора.

Minmax12°. Дано целое число N и набор из N чисел. Найти минимальное положительное число из данного набора. Если положительные числа в наборе отсутствуют, то вывести 0.

Minmax13. Дано целое число N и набор из N целых чисел. Найти номер первого максимального нечетного числа из данного набора. Если нечетные числа в наборе отсутствуют, то вывести 0.

Minmax14. Дано число B (> 0) и набор из десяти чисел. Вывести минимальный из тех элементов набора, которые больше B, а также его номер. Если чисел, больших B, в наборе нет, то дважды вывести 0.

Minmax15. Даны числа B, C ($0 < B < C$) и набор из десяти чисел. Вывести максимальный из элементов набора, содержащихся в интервале (B, C), и его номер. Если требуемые числа в наборе отсутствуют, то дважды вывести 0.

Minmax16. Дано целое число N и набор из N целых чисел. Найти количество элементов, расположенных перед первым минимальным элементом.

Minmax17. Дано целое число N и набор из N целых чисел. Найти количество элементов, расположенных после последнего максимального элемента.

Minmax18. Дано целое число N и набор из N целых чисел. Найти количество элементов, содержащихся между первым и последним максимальным элементом. Если в наборе имеется единственный максимальный элемент, то вывести 0.

Minmax19. Дано целое число N и набор из N целых чисел. Найти количество минимальных элементов из данного набора.

Minmax20. Дано целое число N и набор из N целых чисел. Найти общее количество *экстремальных* (то есть минимальных и максимальных) элементов из данного набора.

Minmax21. Дано целое число N (> 2) и набор из N чисел — значений некоторой величины, полученных в N опытах. Найти среднее значение этой величины. При вычислении среднего значения не учитывать минимальное и максимальное из имеющихся в наборе значений.

Minmax22. Дано целое число N (> 2) и набор из N чисел. Найти два наи-

меньших элемента из данного набора и вывести эти элементы в порядке возрастания их значений.

Minmax23. Дано целое число N (> 3) и набор из N чисел. Найти три наибольших элемента из данного набора и вывести эти элементы в порядке убывания их значений.

Minmax24. Дано целое число N (> 1) и набор из N чисел. Найти максимальную сумму двух соседних чисел из данного набора.

Minmax25. Дано целое число N (> 1) и набор из N чисел. Найти номера двух соседних чисел из данного набора, произведение которых является минимальным, и вывести вначале меньший, а затем больший номер.

Minmax26°. Дано целое число N и набор из N целых чисел. Найти максимальное количество четных чисел в наборе, идущих подряд. Если четные числа в наборе отсутствуют, то вывести 0.

Minmax27. Дано целое число N и набор из N целых чисел, содержащий только нули и единицы. Найти номер элемента, с которого начинается самая длинная последовательность одинаковых чисел, и количество элементов в этой последовательности. Если таких последовательностей несколько, то вывести номер первой из них.

Minmax28. Дано целое число N и набор из N целых чисел, содержащий только нули и единицы. Найти номер элемента, с которого начинается самая длинная последовательность единиц, и количество элементов в этой последовательности. Если таких последовательностей несколько, то вывести номер последней из них. Если единицы в исходном наборе отсутствуют, то дважды вывести 0.

Minmax29. Дано целое число N и набор из N целых чисел. Найти максимальное количество подряд идущих минимальных элементов из данного набора.

Minmax30. Дано целое число N и набор из N целых чисел. Найти минимальное количество подряд идущих максимальных элементов из данного набора.

Тема 3. Работа с данными: массивы, строки, рекурсия
Практическая работа 10 Одномерные массивы

Условие вида «дан массив размера N» означает, что вначале дается *фактический размер* массива (целое число N), а затем приводятся все его элементы. Если в задании явно не указывается, какие значения может принимать размер исходного массива, то предполагается, что размер может изменяться в пределах от 2 до 10. Индекс начального элемента массива считается равным 1.

Если в задании, связанном с созданием (преобразованием) массива, не описан результирующий набор данных, то предполагается, что этим набором является созданный (преобразованный) массив, и необходимо вывести все его элементы в порядке возрастания их индексов.

Формирование массива и вывод его элементов

В заданиях на формирование массива предполагается, что размер результирующего массива не превосходит 10.

Array1. Дано целое число N (> 0). Сформировать и вывести целочисленный массив размера N, содержащий N первых положительных нечетных чисел: 1, 3, 5, ...

Array2. Дано целое число N (> 0). Сформировать и вывести целочисленный массив размера N, содержащий степени двойки от первой до N-й: 2, 4, 8, 16, ...

Array3. Дано целое число N (> 1), а также первый член A и разность D *арифметической прогрессии*. Сформировать и вывести массив размера N, содержащий N первых членов данной прогрессии:

$$A, \quad A + D, \quad A + 2 \cdot D, \quad A + 3 \cdot D, \quad \ldots .$$

Array4. Дано целое число N (> 1), а также первый член A и знаменатель D *геометрической прогрессии*. Сформировать и вывести массив размера N, содержащий N первых членов данной прогрессии:

$$A, \quad A \cdot D, \quad A \cdot D^2, \quad A \cdot D^3, \quad \ldots .$$

Array5. Дано целое число N (> 2). Сформировать и вывести целочисленный массив размера N, содержащий N первых элементов последовательности *чисел Фибоначчи* F_K:

$$F_1 = 1, \qquad F_2 = 1, \qquad F_K = F_{K-2} + F_{K-1}, \qquad K = 3, 4, \ldots .$$

Array6. Даны целые числа N (> 2), A и B. Сформировать и вывести целочисленный массив размера N, первый элемент которого равен A, второй равен B, а каждый последующий элемент равен сумме всех предыдущих.

Array7°. Дан массив размера N. Вывести его элементы в обратном порядке.

Array8. Дан целочисленный массив размера N. Вывести все содержащиеся в

данном массиве нечетные числа в порядке возрастания их индексов, а также их количество K.

Array9. Дан целочисленный массив размера N. Вывести все содержащиеся в данном массиве четные числа в порядке убывания их индексов, а также их количество K.

Array10. Дан целочисленный массив размера N. Вывести вначале все содержащиеся в данном массиве четные числа в порядке возрастания их индексов, а затем — все нечетные числа в порядке убывания их индексов.

Array11. Дан массив A размера N и целое число K ($1 \le K \le N$). Вывести элементы массива с порядковыми номерами, кратными K: A_K, $A_{2 \cdot K}$, $A_{3 \cdot K}$, ... Условный оператор не использовать.

Array12. Дан массив A размера N (N — четное число). Вывести его элементы с четными номерами в порядке возрастания номеров: A_2, A_4, A_6, . . ., A_N. Условный оператор не использовать.

Array13. Дан массив A размера N (N — нечетное число). Вывести его элементы с нечетными номерами в порядке убывания номеров: A_N, A_{N-2}, A_{N-4}, , A_1. Условный оператор не использовать.

Array14. Дан массив A размера N. Вывести вначале его элементы с четными номерами (в порядке возрастания номеров), а затем — элементы с нечетными номерами (также в порядке возрастания номеров):

$$A_2, \quad A_4, \quad A_6, \quad \ldots, \quad A_1, \quad A_3, \quad A_5, \quad \ldots .$$

Условный оператор не использовать.

Array15. Дан массив A размера N. Вывести вначале его элементы с нечетными номерами в порядке возрастания номеров, а затем — элементы с четными номерами в порядке убывания номеров:

$$A_1, \quad A_3, \quad A_5, \quad \ldots, \quad A_6, \quad A_4, \quad A_2.$$

Условный оператор не использовать.

Array16. Дан массив A размера N. Вывести его элементы в следующем порядке:

$$A_1, \quad A_N, \quad A_2, \quad A_{N-1}, \quad A_3, \quad A_{N-2}, \quad \ldots .$$

Array17. Дан массив A размера N. Вывести его элементы в следующем порядке:

$$A_1, \quad A_2, \quad A_N, \quad A_{N-1}, \quad A_3, \quad A_4, \quad A_{N-2}, \quad A_{N-3}, \quad \ldots .$$

Анализ элементов массива

Для выполнения некоторых заданий из данного пункта не требуется одновременно хранить в памяти все исходные данные, поэтому использовать при их выполнении массивы, строго говоря, *не нужно*. Однако применение

массивов позволяет сделать алгоритмы решения этих заданий более простыми и наглядными. Задания из данного пункта можно дополнить заданиями из групп Series и Minmax, рассматривая их как задания на обработку массивов. С другой стороны, для тех заданий данного пункта, которые можно выполнить, не используя массивы, полезно реализовать и такие алгоритмы решения.

Array18. Дан массив A ненулевых целых чисел размера 10. Вывести значение первого из тех его элементов A_K, которые удовлетворяют неравенству $A_K < A_{10}$. Если таких элементов нет, то вывести 0.

Array19. Дан целочисленный массив A размера 10. Вывести порядковый номер последнего из тех его элементов A_K, которые удовлетворяют двойному неравенству $A_1 < A_K < A_{10}$. Если таких элементов нет, то вывести 0.

Array20. Дан массив размера N и целые числа K и L ($1 \le K \le L \le N$). Найти сумму элементов массива с номерами от K до L включительно.

Array21. Дан массив размера N и целые числа K и L ($1 \le K \le L \le N$). Найти среднее арифметическое элементов массива с номерами от K до L включительно.

Array22. Дан массив размера N и целые числа K и L ($1 < K \le L \le N$). Найти сумму всех элементов массива, кроме элементов с номерами от K до L включительно.

Array23. Дан массив размера N и целые числа K и L ($1 < K \le L \le N$). Найти среднее арифметическое всех элементов массива, кроме элементов с номерами от K до L включительно.

Array24. Дан целочисленный массив размера N, не содержащий одинаковых чисел. Проверить, образуют ли его элементы *арифметическую прогрессию* (см. задание Array3). Если образуют, то вывести разность прогрессии, если нет — вывести 0.

Array25. Дан массив ненулевых целых чисел размера N. Проверить, образуют ли его элементы *геометрическую прогрессию* (см. задание Array4). Если образуют, то вывести знаменатель прогрессии, если нет — вывести 0.

Array26. Дан целочисленный массив размера N. Проверить, чередуются ли в нем четные и нечетные числа. Если чередуются, то вывести 0, если нет, то вывести порядковый номер первого элемента, нарушающего закономерность.

Array27. Дан массив ненулевых целых чисел размера N. Проверить, чередуются ли в нем положительные и отрицательные числа. Если чередуются, то вывести 0, если нет, то вывести порядковый номер первого элемента, нарушающего закономерность.

Array28. Дан массив A размера N. Найти минимальный элемент из его эле-

ментов с четными номерами: A_2, A_4, A_6, ...

Array29. Дан массив A размера N. Найти максимальный элемент из его элементов с нечетными номерами: A_1, A_3, A_5, ...

Array30. Дан массив размера N. Найти номера тех элементов массива, которые больше своего правого соседа, и количество таких элементов. Найденные номера выводить в порядке их возрастания.

Array31. Дан массив размера N. Найти номера тех элементов массива, которые больше своего левого соседа, и количество таких элементов. Найденные номера выводить в порядке их убывания.

Array32. Дан массив размера N. Найти номер его первого локального минимума (*локальный минимум* — это элемент, который меньше любого из своих соседей).

Array33. Дан массив размера N. Найти номер его последнего локального максимума (*локальный максимум* — это элемент, который больше любого из своих соседей).

Array34. Дан массив размера N. Найти максимальный из его локальных минимумов (определение *локального минимума* дано в задании Array32).

Array35. Дан массив размера N. Найти минимальный из его локальных максимумов (определение *локального максимума* дано в задании Array33).

Array36. Дан массив размера N. Найти максимальный из его элементов, не являющихся ни локальным минимумом, ни локальным максимумом (определения *локального минимума* и *локального максимума* даны в заданиях Array32 и Array33). Если таких элементов в массиве нет, то вывести 0.

Array37. Дан массив размера N. Найти количество участков, на которых его элементы монотонно возрастают.

Array38. Дан массив размера N. Найти количество участков, на которых его элементы монотонно убывают.

Array39. Дан массив размера N. Найти количество его *промежутков монотонности* (то есть участков, на которых его элементы возрастают или убывают).

Array40. Дано число R и массив A размера N. Найти элемент массива, который *наиболее близок* к числу R (то есть такой элемент A_K, для которого величина $|A_K - R|$ является минимальной).

Array41. Дан массив размера N. Найти два соседних элемента, сумма которых максимальна, и вывести эти элементы в порядке возрастания их индексов.

Array42. Дано число R и массив размера N. Найти два соседних элемента массива, сумма которых наиболее близка к числу R, и вывести эти элементы в порядке возрастания их индексов (определение наиболее близких чисел дано в задании Array40).

Array43. Дан целочисленный массив размера N, все элементы которого упорядочены (по возрастанию или по убыванию). Найти количество различных элементов в данном массиве.

Array44. Дан целочисленный массив размера N, содержащий ровно два одинаковых элемента. Найти номера одинаковых элементов и вывести эти номера в порядке возрастания.

Array45. Дан массив размера N. Найти номера двух ближайших элементов из этого массива (то есть элементов с наименьшим модулем разности) и вывести эти номера в порядке возрастания.

Array46. Дано число R и массив размера N. Найти два различных элемента массива, сумма которых наиболее близка к числу R, и вывести эти элементы в порядке возрастания их индексов (определение наиболее близких чисел дано в задании Array40).

Array47°. Дан целочисленный массив размера N. Найти количество различных элементов в данном массиве.

Array48. Дан целочисленный массив размера N. Найти максимальное количество его одинаковых элементов.

Array49. Дан целочисленный массив размера N. Если он является *перестановкой*, то есть содержит все числа от 1 до N, то вывести 0; в противном случае вывести номер первого недопустимого элемента.

Array50. Дан целочисленный массив A размера N, являющийся перестановкой (определение *перестановки* дано в задании Array49). Найти количество *инверсий* в данной перестановке, то есть таких пар элементов A_I и A_J, в которых большее число находится слева от меньшего: $A_I > A_J$ при $I < J$.

Работа с несколькими массивами

Array51. Даны массивы A и B одинакового размера N. Поменять местами их содержимое и вывести вначале элементы преобразованного массива A, а затем — элементы преобразованного массива B.

Array52. Дан массив A размера N. Сформировать новый массив B того же размера, элементы которого определяются следующим образом:

$$B_K = \begin{cases} 2 \cdot A_K, & \text{если } A_K < 5, \\ A_K/2 & \text{в противном случае.} \end{cases}$$

Array53. Даны два массива A и B одинакового размера N. Сформировать новый массив C того же размера, каждый элемент которого равен максимальному из элементов массивов A и B с тем же индексом.

Array54. Дан целочисленный массив A размера N. Переписать в новый целочисленный массив B все четные числа из исходного массива (в том же

порядке) и вывести размер полученного массива B и его содержимое.

Array55. Дан целочисленный массив A размера N (≤ 15). Переписать в новый целочисленный массив B все элементы с нечетными порядковыми номерами (1, 3, . . .) и вывести размер полученного массива B и его содержимое. Условный оператор не использовать.

Array56. Дан целочисленный массив A размера N (≤ 15). Переписать в новый целочисленный массив B все элементы с порядковыми номерами, кратными трем (3, 6, . . .), и вывести размер полученного массива B и его содержимое. Условный оператор не использовать.

Array57. Дан целочисленный массив A размера N. Переписать в новый целочисленный массив B того же размера вначале все элементы исходного массива с четными номерами, а затем — с нечетными:

$$A_2, \quad A_4, \quad A_6, \quad \ldots, \quad A_1, \quad A_3, \quad A_5, \quad \ldots .$$

Условный оператор не использовать.

Array58. Дан массив A размера N. Сформировать новый массив B того же размера по следующему правилу: элемент B_K равен сумме элементов массива A с номерами от 1 до K.

Array59. Дан массив A размера N. Сформировать новый массив B того же размера по следующему правилу: элемент B_K равен среднему арифметическому элементов массива A с номерами от 1 до K.

Array60°. Дан массив A размера N. Сформировать новый массив B того же размера по следующему правилу: элемент B_K равен сумме элементов массива A с номерами от K до N.

Array61. Дан массив A размера N. Сформировать новый массив B того же размера по следующему правилу: элемент B_K равен среднему арифметическому элементов массива A с номерами от K до N.

Array62. Дан массив A размера N. Сформировать два новых массива B и C: в массив B записать все положительные элементы массива A, в массив C — все отрицательные (сохраняя исходный порядок следования элементов). Вывести вначале размер и содержимое массива B, а затем — размер и содержимое массива C.

Array63. Даны два массива A и B размера 5, элементы которых упорядочены по возрастанию. Объединить эти массивы так, чтобы результирующий массив C (размера 10) остался упорядоченным по возрастанию.

Array64. Даны три целочисленных массива A, B и C размера N_A, N_B, N_C соответственно, элементы которых упорядочены по убыванию. Объединить эти массивы так, чтобы результирующий целочисленный массив D (размера $N_A + N_B + N_C$) остался упорядоченным по убыванию.

При выполнении заданий из данного пункта не следует использовать вспомогательные массивы.

Изменение элементов массива

Array65. Дан массив A размера N и целое число K ($1 \le K \le N$). Преобразовать массив, увеличив каждый его элемент на исходное значение элемента A_K.

Array66. Дан целочисленный массив размера N. Увеличить все четные числа, содержащиеся в массиве, на исходное значение первого четного числа. Если четные числа в массиве отсутствуют, то оставить массив без изменений.

Array67. Дан целочисленный массив размера N. Увеличить все нечетные числа, содержащиеся в массиве, на исходное значение последнего нечетного числа. Если нечетные числа в массиве отсутствуют, то оставить массив без изменений.

Array68°. Дан массив размера N. Поменять местами его минимальный и максимальный элементы.

Array69. Дан массив размера N (N — четное число). Поменять местами его первый элемент со вторым, третий — с четвертым и т. д.

Array70. Дан массив размера N (N — четное число). Поменять местами первую и вторую половины массива.

Array71. Дан массив размера N. Поменять порядок его элементов на обратный.

Array72. Дан массив A размера N и целые числа K и L ($1 \le K < L \le N$). Переставить в обратном порядке элементы массива, расположенные между элементами A_K и A_L, включая эти элементы.

Array73. Дан массив A размера N и целые числа K и L ($1 \le K < L \le N$). Переставить в обратном порядке элементы массива, расположенные между элементами A_K и A_L, не включая эти элементы.

Array74. Дан массив размера N. Обнулить элементы массива, расположенные между его минимальным и максимальным элементами (не включая минимальный и максимальный элементы).

Array75. Дан массив размера N. Переставить в обратном порядке элементы массива, расположенные между его минимальным и максимальным элементами, включая минимальный и максимальный элементы.

Array76. Дан массив размера N. Обнулить все его *локальные максимумы* (то есть числа, большие своих соседей).

Array77. Дан массив размера N. Возвести в квадрат все его *локальные минимумы* (то есть числа, меньшие своих соседей).

Array78. Дан массив размера N. Заменить каждый элемент массива на среднее арифметическое этого элемента и его соседей.

Array79. Дан массив размера N. Осуществить *сдвиг* элементов массива вправо на одну позицию (при этом A_1 перейдет в A_2, A_2 — в A_3, . . ., A_{N-1} — в A_N, а исходное значение последнего элемента будет потеряно). Первый элемент полученного массива положить равным 0.

Array80. Дан массив размера N. Осуществить *сдвиг* элементов массива влево на одну позицию (при этом A_N перейдет в A_{N-1}, A_{N-1} — в A_{N-2}, . . ., A_2 — в A_1, а исходное значение первого элемента будет потеряно). Последний элемент полученного массива положить равным 0.

Array81. Дан массив размера N и целое число K ($1 \le K < N$). Осуществить *сдвиг* элементов массива вправо на K позиций (при этом A_1 перейдет в A_{K+1}, A_2 — в A_{K+2}, . . ., A_{N-K} — в A_N, а исходное значение K последних элементов будет потеряно). Первые K элементов полученного массива положить равными 0.

Array82. Дан массив размера N и целое число K ($1 \le K < N$). Осуществить *сдвиг* элементов массива влево на K позиций (при этом A_N перейдет в A_{N-K}, A_{N-1} — в A_{N-K-1}, . . ., A_{K+1} — в A_1, а исходное значение K первых элементов будет потеряно). Последние K элементов полученного массива положить равными 0.

Array83. Дан массив размера N. Осуществить *циклический сдвиг* элементов массива вправо на одну позицию (при этом A_1 перейдет в A_2, A_2 — в A_3, . . ., A_N — в A_1).

Array84. Дан массив размера N. Осуществить *циклический сдвиг* элементов массива влево на одну позицию (при этом A_N перейдет в A_{N-1}, A_{N-1} — в A_{N-2}, . . ., A_1 — в A_N).

Array85. Дан массив A размера N и целое число K ($1 \le K \le 4$, $K < N$). Осуществить *циклический сдвиг* элементов массива вправо на K позиций (при этом A_1 перейдет в A_{K+1}, A_2 — в A_{K+2}, . . ., A_N — в A_K). Допускается использовать вспомогательный массив из 4 элементов.

Array86. Дан массив A размера N и целое число K ($1 \le K \le 4$, $K < N$). Осуществить *циклический сдвиг* элементов массива влево на K позиций (при этом A_N перейдет в A_{N-K}, A_{N-1} — в A_{N-K-1}, . . ., A_1 — в A_{N-K+1}). Допускается использовать вспомогательный массив из 4 элементов.

Array87. Дан массив размера N, все элементы которого, кроме первого, упорядочены по возрастанию. Сделать массив упорядоченным, переместив первый элемент на новую позицию.

Array88. Дан массив размера N, все элементы которого, кроме последнего, упорядочены по возрастанию. Сделать массив упорядоченным, переме-

стив последний элемент на новую позицию.

Array89. Дан массив размера N, все элементы которого, кроме одного, упорядочены по убыванию. Сделать массив упорядоченным, переместив элемент, нарушающий упорядоченность, на новую позицию.

Удаление и вставка элементов

Array90. Дан массив размера N и целое число K ($1 \le K \le N$). Удалить из массива элемент с порядковым номером K.

Array91. Дан массив размера N и целые числа K и L ($1 \le K < L \le N$). Удалить из массива элементы с номерами от K до L включительно и вывести размер полученного массива и его содержимое.

Array92. Дан целочисленный массив размера N. Удалить из массива все нечетные числа и вывести размер полученного массива и его содержимое.

Array93. Дан целочисленный массив размера N (> 2). Удалить из массива все элементы с четными номерами (2, 4, . . .). Условный оператор не использовать.

Array94. Дан целочисленный массив размера N (> 2). Удалить из массива все элементы с нечетными номерами (1, 3, . . .). Условный оператор не использовать.

Array95. Дан целочисленный массив размера N. Удалить из массива все соседние одинаковые элементы, оставив их первые вхождения.

Array96. Дан целочисленный массив размера N. Удалить из массива все одинаковые элементы, оставив их первые вхождения.

Array97. Дан целочисленный массив размера N. Удалить из массива все одинаковые элементы, оставив их последние вхождения.

Array98. Дан целочисленный массив размера N. Удалить из массива все элементы, встречающиеся менее трех раз, и вывести размер полученного массива и его содержимое.

Array99. Дан целочисленный массив размера N. Удалить из массива все элементы, встречающиеся более двух раз, и вывести размер полученного массива и его содержимое.

Array100. Дан целочисленный массив размера N. Удалить из массива все элементы, встречающиеся ровно два раза, и вывести размер полученного массива и его содержимое.

Array101. Дан массив размера N и целое число K ($1 \le K \le N$). Перед элементом массива с порядковым номером K вставить новый элемент с нулевым значением.

Array102. Дан массив размера N и целое число K ($1 \le K \le N$). После элемента

массива с порядковым номером K вставить новый элемент с нулевым значением.

Array103. Дан массив размера N. Вставить элемент с нулевым значением перед минимальным и после максимального элемента массива.

Array104. Дан массив размера N и два целых числа K и M ($1 \le K \le N$, $1 \le M \le 10$). Перед элементом массива с номером K вставить M новых элементов с нулевыми значениями.

Array105. Дан массив размера N и два целых числа K и M ($1 \le K \le N$, $1 \le M \le 10$). После элемента массива с номером K вставить M новых элементов с нулевыми значениями.

Array106. Дан массив размера N. Продублировать в нем элементы с четными номерами (2, 4, . . .). Условный оператор не использовать.

Array107. Дан массив размера N. Утроить в нем вхождения всех элементов с нечетными номерами (1, 3, . . .). Условный оператор не использовать.

Array108. Дан массив размера N. Перед каждым положительным элементом массива вставить элемент с нулевым значением.

Array109. Дан массив размера N. После каждого отрицательного элемента массива вставить элемент с нулевым значением.

Array110. Дан целочисленный массив размера N. Продублировать в нем все четные числа.

Array111. Дан целочисленный массив размера N. Утроить в нем вхождения всех нечетных чисел.

Сортировка массива

Array112°. Дан массив A размера N (≤ 6). Упорядочить его по возрастанию методом сортировки *простым обменом* (*«пузырьковой» сортировкой*): просматривать массив, сравнивая его соседние элементы (A_1 и A_2, A_2 и A_3 и т. д.) и меняя их местами, если левый элемент пары больше правого; повторить описанные действия $N - 1$ раз. Для контроля за выполняемыми действиями выводить содержимое массива после каждого просмотра. Учесть, что при каждом просмотре количество анализируемых пар можно уменьшить на 1.

Array113. Дан массив A_размера N (≤ 6). Упорядочить его по возрастанию методом сортировки *простым выбором*: найти максимальный элемент массива и поменять его местами с последним (N-м) элементом; выполнить описанные действия $N - 1$ раз, каждый раз уменьшая на 1 количество анализируемых элементов и выводя содержимое массива.

Array114. Дан массив A размера N (≤ 6). Упорядочить его по возрастанию методом сортировки *простыми вставками*: сравнить элементы A_1 и A_2 и,

при необходимости меняя их местами, добиться того, чтобы они оказались упорядоченными по возрастанию; затем обратиться к элементу A_3 и переместить его в левую (уже упорядоченную) часть массива, сохранив ее упорядоченность; повторить этот процесс для остальных элементов, выводя содержимое массива после обработки каждого элемента (от 2-го до N -го). При выполнении описанных действий удобно использовать прием «барьера», записывая очередной элемент перед его обработкой в дополнительный элемент массива A_0.

Array115. Дан массив A размера N. Не изменяя данный массив, вывести номера его элементов в том порядке, в котором соответствующие им элементы образуют возрастающую последовательность. Использовать метод *«пузырьковой» сортировки* (см. задание Array112), модифицировав его следующим образом: создать вспомогательный целочисленный *массив номеров* I, заполнив его числами от 1 до N ; просматривать массив A, сравнивая пары элементов массива A с номерами I_1 и I_2, I_2 и I_3, . . . и меняя местами соответствующие элементы массива I, если левый элемент пары больше правого. Повторив описанную процедуру просмотра $N - 1$ раз, получим в массиве I требуемую последовательность номеров.

Серии целых чисел

Array116°. Дан целочисленный массив A размера N. Назовем *серией* группу подряд идущих одинаковых элементов, а *длиной серии* — количество этих элементов (длина серии может быть равна 1). Сформировать два новых целочисленных массива B и C одинакового размера, записав в массив B длины всех серий исходного массива, а в массив C — значения элементов, образующих эти серии.

Array117. Дан целочисленный массив размера N. Вставить перед каждой его серией элемент с нулевым значением (определение серии дано в задании Array116).

Array118. Дан целочисленный массив размера N. Вставить после каждой его серии элемент с нулевым значением (определение серии дано в задании Array116).

Array119. Дан целочисленный массив размера N. Преобразовать массив, увеличив каждую его серию на один элемент (определение серии дано в задании Array116).

Array120. Дан целочисленный массив размера N, содержащий по крайней мере одну серию, длина которой больше 1. Преобразовать массив, уменьшив каждую его серию на один элемент (определение серии дано в задании Array116).

Array121. Дано целое число K (> 0) и целочисленный массив размера N. Преобразовать массив, удвоив длину его серии с номером K (определение серии дано в задании Array116). Если серий в массиве меньше K, то вывести массив без изменений.

Array122. Дано целое число K (> 1) и целочисленный массив размера N. Удалить из массива серию с номером K (определение серии дано в задании Array116). Если серий в массиве меньше K, то вывести массив без изменений.

Array123. Дано целое число K (> 1) и целочисленный массив размера N. Поменять местами первую серию массива и его серию с номером K (определение серии дано в задании Array116). Если серий в массиве меньше K, то вывести массив без изменений.

Array124. Дано целое число K (> 0) и целочисленный массив размера N. Поменять местами последнюю серию массива и его серию с номером K (определение серии дано в задании Array116). Если серий в массиве меньше K, то вывести массив без изменений.

Array125. Дано целое число L (> 1) и целочисленный массив размера N. Заменить каждую серию массива, длина которой меньше L, на один элемент с нулевым значением (определение серии дано в задании Array116).

Array126. Дано целое число L (> 0) и целочисленный массив размера N. Заменить каждую серию массива, длина которой равна L, на один элемент с нулевым значением (определение серии дано в задании Array116).

Array127. Дано целое число L (> 0) и целочисленный массив размера N. Заменить каждую серию массива, длина которой больше L, на один элемент с нулевым значением (определение серии дано в задании Array116).

Array128. Дан целочисленный массив размера N. Преобразовать массив, увеличив его первую серию наибольшей длины на один элемент (определение серии дано в задании Array116).

Array129. Дан целочисленный массив размера N. Преобразовать массив, увеличив его последнюю серию наибольшей длины на один элемент (определение серии дано в задании Array116).

Array130. Дан целочисленный массив размера N. Преобразовать массив, увеличив все его серии наибольшей длины на один элемент (определение серии дано в задании Array116).

Множества точек на плоскости

Для хранения данных о каждом наборе точек следует использовать по два массива: первый массив для хранения абсцисс, второй — для хранения ординат. Можно также использовать массив *записей* с двумя полями (см. задание

Param64).

Array131. Дано множество A из N точек на плоскости и точка B (точки заданы своими координатами x, y). Найти точку из множества A, наиболее близкую к точке B. *Расстояние* R между точками с координатами (x_1, y_1) и (x_2, y_2) вычисляется по формуле:

$$R = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

Array132. Дано множество A из N точек (точки заданы своими координатами x, y). Среди всех точек этого множества, лежащих во второй четверти, найти точку, наиболее удаленную от начала координат. Если таких точек нет, то вывести точку с нулевыми координатами.

Array133. Дано множество A из N точек (точки заданы своими координатами x, y). Среди всех точек этого множества, лежащих в первой или третьей четверти, найти точку, наиболее близкую к началу координат. Если таких точек нет, то вывести точку с нулевыми координатами.

Array134. Дано множество A из N точек (точки заданы своими координатами x, y). Найти пару различных точек этого множества с максимальным расстоянием между ними и само это расстояние (точки выводятся в том же порядке, в котором они перечислены при задании множества A).

Array135. Даны множества A и B, состоящие соответственно из N_1 и N_2 точек (точки заданы своими координатами x, y). Найти минимальное расстояние между точками этих множеств и сами точки, расположенные на этом расстоянии (вначале выводится точка из множества A, затем точка из множества B).

Array136. Дано множество A из N точек ($N > 2$, точки заданы своими координатами x, y). Найти такую точку из данного множества, сумма расстояний от которой до остальных его точек минимальна, и саму эту сумму.

Array137. Дано множество A из N точек ($N > 2$, точки заданы своими координатами x, y). Найти наибольший периметр треугольника, вершины которого принадлежат различным точкам множества A, и сами эти точки (точки выводятся в том же порядке, в котором они перечислены при задании множества A).

Array138. Дано множество A из N точек ($N > 2$, точки заданы своими координатами x, y). Найти наименьший периметр треугольника, вершины которого принадлежат различным точкам множества A, и сами эти точки (точки выводятся в том же порядке, в котором они перечислены при задании множества A).

Array139. Дано множество A из N точек с целочисленными координатами x, y. *Порядок* на координатной плоскости определим следующим образом:

$(x_1, y_1) < (x_2, y_2)$, если либо $x_1 < x_2$, либо $x_1 = x_2$ и $y_1 < y_2$.

Расположить точки данного множества по возрастанию в соответствии с указанным порядком.

Array140. Дано множество A из N точек с целочисленными координатами x, y.

Порядок на координатной плоскости определим следующим образом:

$(x_1, y_1) < (x_2, y_2)$, если либо $x_1 + y_1 < x_2 + y_2$, либо $x_1 + y_1 = x_2 + y_2$ и $x_1 < x_2$.

Расположить точки данного множества по убыванию в соответствии с указанным порядком.

Тема 3. Работа с данными: массивы, строки, рекурсия
Практическая работа 11 Двумерные массивы (матрицы)

Условие вида «дана матрица размера $M \times N$» означает, что вначале дается *фактический размер* двумерного массива-матрицы (количество строк M и количество столбцов N), а затем приводятся элементы этого массива (количество элементов равно $M \cdot N$). Если в задании явно не указывается, какие значения могут принимать размеры исходной матрицы, то предполагается, что и число строк, и число столбцов может изменяться в пределах от 2 до 10. Начальные значения как первого, так и второго индекса двумерного массива-матрицы всегда считаются равными 1. Ввод и вывод элементов матрицы осуществляются *по строкам*.

Квадратной матрицей порядка M называется двумерный массив-матрица размера $M \times M$.

Если в задании, связанном с созданием или преобразованием матрицы, не описан результирующий набор данных, то предполагается, что этим набором является созданная (преобразованная) матрица, и необходимо вывести все ее элементы.

Формирование матрицы и вывод ее элементов

В заданиях на формирование матрицы предполагается, что размер результирующей матрицы не превосходит 10×10.

Matrix1. Даны целые положительные числа M и N. Сформировать целочисленную матрицу размера $M \times N$, у которой все элементы I-й строки имеют значение $10 \cdot I$ ($I = 1, \ldots, M$).

Matrix2. Даны целые положительные числа M и N. Сформировать целочисленную матрицу размера $M \times N$, у которой все элементы J-го столбца имеют значение $5 \cdot J$ ($J = 1, \ldots, N$).

Matrix3. Даны целые положительные числа M, N и набор из M чисел. Сформировать матрицу размера $M \times N$, у которой в каждом столбце содержатся все числа из исходного набора (в том же порядке).

Matrix4. Даны целые положительные числа M, N и набор из N чисел. Сформировать матрицу размера $M \times N$, у которой в каждой строке содержатся все числа из исходного набора (в том же порядке).

Matrix5. Даны целые положительные числа M, N, число D и набор из M чисел. Сформировать матрицу размера $M \times N$, у которой первый столбец совпадает с исходным набором чисел, а элементы каждого следующего столбца равны сумме соответствующего элемента предыдущего столбца

и числа D (в результате каждая строка матрицы будет содержать элементы *арифметической прогрессии*).

Matrix6. Даны целые положительные числа M, N, число Q и набор из N чисел. Сформировать матрицу размера $M \times N$, у которой первая строка совпадает с исходным набором чисел, а элементы каждой следующей строки равны соответствующему элементу предыдущей строки, умноженному на Q (в результате каждый столбец матрицы будет содержать элементы *геометрической прогрессии*).

Matrix7°. Дана матрица размера $M \times N$ и целое число K ($1 \le K \le M$). Вывести элементы K-й строки данной матрицы.

Matrix8. Дана матрица размера $M \times N$ и целое число K ($1 \le K \le N$). Вывести элементы K-го столбца данной матрицы.

Matrix9. Дана матрица размера $M \times N$. Вывести ее элементы, расположенные в строках с четными номерами (2, 4, . . .). Вывод элементов производить по строкам, условный оператор не использовать.

Matrix10. Дана матрица размера $M \times N$. Вывести ее элементы, расположенные в столбцах с нечетными номерами (1, 3, . . .). Вывод элементов производить по столбцам, условный оператор не использовать.

Matrix11. Дана матрица размера $M \times N$. Вывести ее элементы в следующем порядке: первая строка слева направо, вторая строка справа налево, третья строка слева направо, четвертая строка справа налево и т. д.

Matrix12. Дана матрица размера $M \times N$. Вывести ее элементы в следующем порядке: первый столбец сверху вниз, второй столбец снизу вверх, третий столбец сверху вниз, четвертый столбец снизу вверх и т. д.

Matrix13. Дана квадратная матрица A порядка M. Начиная с элемента $A_{1,1}$, вывести ее элементы следующим образом («*уголками*»): все элементы первой строки; элементы последнего столбца, кроме первого (уже выведенного) элемента; оставшиеся элементы второй строки; оставшиеся элементы предпоследнего столбца и т. д.; последним выводится элемент $A_{M,1}$.

Matrix14. Дана квадратная матрица A порядка M. Начиная с элемента $A_{1,1}$, вывести ее элементы следующим образом («*уголками*»): все элементы первого столбца; элементы последней строки, кроме первого (уже выведенного) элемента; оставшиеся элементы второго столбца; оставшиеся элементы предпоследней строки и т. д.; последним выводится элемент $A_{1,M}$.

Matrix15. Дана квадратная матрица A порядка M (M — нечетное число). Начиная с элемента $A_{1,1}$ и перемещаясь по часовой стрелке, вывести все ее элементы *по спирали*: первая строка, последний столбец, последняя строка в обратном порядке, первый столбец в обратном порядке, остав-

шиеся элементы второй строки и т. д.; последним выводится центральный элемент матрицы.

Matrix16. Дана квадратная матрица A порядка M (M — нечетное число). Начиная с элемента $A_{1,1}$ и перемещаясь против часовой стрелки, вывести все ее элементы *по спирали*: первый столбец, последняя строка, последний столбец в обратном порядке, первая строка в обратном порядке, оставшиеся элементы второго столбца и т. д.; последним выводится центральный элемент матрицы.

Анализ элементов матрицы

Matrix17. Дана матрица размера $M \times N$ и целое число K ($1 \le K \le M$). Найти сумму и произведение элементов K-й строки данной матрицы.

Matrix18. Дана матрица размера $M \times N$ и целое число K ($1 \le K \le N$). Найти сумму и произведение элементов K-го столбца данной матрицы.

Matrix19. Дана матрица размера $M \times N$. Для каждой строки матрицы найти сумму ее элементов.

Matrix20. Дана матрица размера $M \times N$. Для каждого столбца матрицы найти произведение его элементов.

Matrix21. Дана матрица размера $M \times N$. Для каждой строки матрицы с нечетным номером (1, 3, . . .) найти среднее арифметическое ее элементов. Условный оператор не использовать.

Matrix22. Дана матрица размера $M \times N$. Для каждого столбца матрицы с четным номером (2, 4, . . .) найти сумму его элементов. Условный оператор не использовать.

Matrix23. Дана матрица размера $M \times N$. В каждой строке матрицы найти минимальный элемент.

Matrix24°. Дана матрица размера $M \times N$. В каждом столбце матрицы найти максимальный элемент.

Matrix25. Дана матрица размера $M \times N$. Найти номер ее строки с наибольшей суммой элементов и вывести данный номер, а также значение наибольшей суммы.

Matrix26. Дана матрица размера $M \times N$. Найти номер ее столбца с наименьшим произведением элементов и вывести данный номер, а также значение наименьшего произведения.

Matrix27. Дана матрица размера $M \times N$. Найти максимальный среди минимальных элементов ее строк.

Matrix28. Дана матрица размера $M \times N$. Найти минимальный среди максимальных элементов ее столбцов.

Matrix29. Дана матрица размера $M \times N$. В каждой ее строке найти количество элементов, меньших среднего арифметического всех элементов этой

строки.

Matrix30. Дана матрица размера $M \times N$. В каждом ее столбце найти количество элементов, больших среднего арифметического всех элементов этого столбца.

Matrix31. Дана матрица размера $M \times N$. Найти номера строки и столбца

для элемента матрицы, наиболее близкого к среднему значению всех ее элементов.

Matrix32. Дана целочисленная матрица размера $M \times N$. Найти номер первой из ее строк, содержащих равное количество положительных и отрицательных элементов (нулевые элементы матрицы не учитываются). Если таких строк нет, то вывести 0.

Matrix33. Дана целочисленная матрица размера $M \times N$. Найти номер последнего из ее столбцов, содержащих равное количество положительных и отрицательных элементов (нулевые элементы матрицы не учитываются). Если таких столбцов нет, то вывести 0.

Matrix34. Дана целочисленная матрица размера $M \times N$. Найти номер последней из ее строк, содержащих только четные числа. Если таких строк нет, то вывести 0.

Matrix35. Дана целочисленная матрица размера $M \times N$. Найти номер первого из ее столбцов, содержащих только нечетные числа. Если таких столбцов нет, то вывести 0.

Matrix36. Дана целочисленная матрица размера $M \times N$, элементы которой могут принимать значения от 0 до 100. Различные строки матрицы назовем *похожими*, если совпадают множества чисел, встречающихся в этих строках. Найти количество строк, похожих на первую строку данной матрицы.

Matrix37. Дана целочисленная матрица размера $M \times N$, элементы которой могут принимать значения от 0 до 100. Различные столбцы матрицы назовем *похожими*, если совпадают множества чисел, встречающихся в этих столбцах. Найти количество столбцов, похожих на последний столбец данной матрицы.

Matrix38. Дана целочисленная матрица размера $M \times N$. Найти количество ее строк, все элементы которых различны.

Matrix39. Дана целочисленная матрица размера $M \times N$. Найти количество ее столбцов, все элементы которых различны.

Matrix40. Дана целочисленная матрица размера $M \times N$. Найти номер последней из ее строк, содержащих максимальное количество одинаковых элементов.

Matrix41. Дана целочисленная матрица размера $M \times N$. Найти номер первого из ее столбцов, содержащих максимальное количество одинаковых элементов.

Matrix42. Дана матрица размера $M \times N$. Найти количество ее строк, элементы которых упорядочены по возрастанию.

Matrix43. Дана матрица размера $M \times N$. Найти количество ее столбцов, элементы которых упорядочены по убыванию.

Matrix44. Дана матрица размера $M \times N$. Найти минимальный среди элементов тех строк, которые упорядочены либо по возрастанию, либо по убыванию. Если упорядоченные строки в матрице отсутствуют, то вывести 0.

Matrix45. Дана матрица размера $M \times N$. Найти максимальный среди элементов тех столбцов, которые упорядочены либо по возрастанию, либо по убыванию. Если упорядоченные столбцы в матрице отсутствуют, то вывести 0.

Matrix46. Дана целочисленная матрица размера $M \times N$. Найти элемент, являющийся максимальным в своей строке и минимальным в своем столбце. Если такой элемент отсутствует, то вывести 0.

Преобразование матрицы

При выполнении заданий из данного пункта (за исключением Matrix74 и MAtrix75) не следует использовать вспомогательные двумерные массивы-матрицы.

Matrix47. Дана матрица размера $M \times N$ и целые числа K_1 и K_2 ($1 \le K_1 < K_2 \le M$). Поменять местами строки матрицы с номерами K_1 и K_2.

Matrix48. Дана матрица размера $M \times N$ и целые числа K_1 и K_2 ($1 \le K_1 < K_2 \le N$). Поменять местами столбцы матрицы с номерами K_1 и K_2.

Matrix49. Дана матрица размера $M \times N$. Преобразовать матрицу, поменяв местами минимальный и максимальный элемент в каждой строке.

Matrix50. Дана матрица размера $M \times N$. Преобразовать матрицу, поменяв местами минимальный и максимальный элемент в каждом столбце.

Matrix51. Дана матрица размера $M \times N$. Поменять местами строки, содержащие минимальный и максимальный элементы матрицы.

Matrix52. Дана матрица размера $M \times N$. Поменять местами столбцы, содержащие минимальный и максимальный элементы матрицы.

Matrix53°. Дана матрица размера $M \times N$. Поменять местами столбец с номером 1 и последний из столбцов, содержащих только положительные

элементы. Если требуемых столбцов нет, то вывести матрицу без изменений.

Matrix54. Дана матрица размера $M \times N$. Поменять местами столбец с номером N и первый из столбцов, содержащих только отрицательные элементы. Если требуемых столбцов нет, то вывести матрицу без изменений.

Matrix55. Дана матрица размера $M \times N$ (M — четное число). Поменять местами верхнюю и нижнюю половины матрицы.

Matrix56. Дана матрица размера $M \times N$ (N — четное число). Поменять местами левую и правую половины матрицы.

Matrix57. Дана матрица размера $M \times N$ (M и N — четные числа). Поменять местами левую верхнюю и правую нижнюю четверти матрицы.

Matrix58. Дана матрица размера $M \times N$ (M и N — четные числа). Поменять местами левую нижнюю и правую верхнюю четверти матрицы.

Matrix59. Дана матрица размера $M \times N$. Зеркально отразить ее элементы относительно горизонтальной оси симметрии матрицы (при этом поменяются местами строки с номерами 1 и M, 2 и $M - 1$ и т. д.).

Matrix60. Дана матрица размера $M \times N$. Зеркально отразить ее элементы относительно вертикальной оси симметрии матрицы (при этом поменяются местами столбцы с номерами 1 и N, 2 и $N - 1$ и т. д.).

Matrix61. Дана матрица размера $M \times N$ и целое число K ($1 \le K \le M$). Удалить строку матрицы с номером K.

Matrix62. Дана матрица размера $M \times N$ и целое число K ($1 \le K \le N$). Удалить столбец матрицы с номером K.

Matrix63. Дана матрица размера $M \times N$. Удалить строку, содержащую минимальный элемент матрицы.

Matrix64. Дана матрица размера $M \times N$. Удалить столбец, содержащий максимальный элемент матрицы.

Matrix65. Дана матрица размера $M \times N$. Удалить ее первый столбец, содержащий только положительные элементы. Если требуемых столбцов нет, то вывести матрицу без изменений.

Matrix66. Дана матрица размера $M \times N$. Удалить ее последний столбец, содержащий только отрицательные элементы. Если требуемых столбцов нет, то вывести матрицу без изменений.

Matrix67. Дана матрица размера $M \times N$, содержащая как положительные, так и отрицательные элементы. Удалить все ее столбцы, содержащие только положительные элементы. Если требуемых столбцов нет, то вывести матрицу без изменений.

Matrix68. Дана матрица размера $M \times N$ и целое число K ($1 \le K \le M$). Перед строкой матрицы с номером K вставить строку из нулей.

Matrix69. Дана матрица размера $M \times N$ и целое число K ($1 \le K \le N$). После столбца матрицы с номером K вставить столбец из единиц.

Matrix70. Дана матрица размера $M \times N$. Продублировать строку матрицы, содержащую ее максимальный элемент.

Matrix71. Дана матрица размера $M \times N$. Продублировать столбец матрицы, содержащий ее минимальный элемент.

Matrix72. Дана матрица размера $M \times N$. Перед первым столбцом, содержащим только положительные элементы, вставить столбец из единиц. Если требуемых столбцов нет, то вывести матрицу без изменений.

Matrix73. Дана матрица размера $M \times N$. После последнего столбца, содержащего только отрицательные элементы, вставить столбец из нулей. Если требуемых столбцов нет, то вывести матрицу без изменений.

Matrix74. Дана матрица размера $M \times N$. Элемент матрицы называется ее *локальным минимумом*, если он меньше всех окружающих его элементов. Заменить все локальные минимумы данной матрицы на нули. При решении допускается использовать вспомогательную матрицу.

Matrix75. Дана матрица размера $M \times N$. Элемент матрицы называется ее *локальным максимумом*, если он больше всех окружающих его элементов. Поменять знак всех локальных максимумов данной матрицы на противоположный. При решении допускается использовать вспомогательную матрицу.

Matrix76. Дана матрица размера $M \times N$. Упорядочить ее строки так, чтобы их первые элементы образовывали возрастающую последовательность.

Matrix77. Дана матрица размера $M \times N$. Упорядочить ее столбцы так, чтобы их последние элементы образовывали убывающую последовательность.

Matrix78. Дана матрица размера $M \times N$. Упорядочить ее строки так, чтобы их минимальные элементы образовывали убывающую последовательность.

Matrix79. Дана матрица размера $M \times N$. Упорядочить ее столбцы так, чтобы их максимальные элементы образовывали возрастающую последовательность.

Диагонали квадратной матрицы

Matrix80. Дана квадратная матрица A порядка M. Найти сумму элементов ее *главной диагонали*, то есть диагонали, содержащей следующие элементы:

$$A_{1,1}, \quad A_{2,2}, \quad A_{3,3}, \quad \ldots, \quad A_{M,M}.$$

Matrix81. Дана квадратная матрица A порядка M. Найти среднее арифметическое элементов ее *побочной диагонали*, то есть диагонали, содержащей

следующие элементы:

$$A_{1,M}, \quad A_{2,M-1}, \quad A_{3,M-2}, \quad \ldots, \quad A_{M,1}.$$

Matrix82°. Дана квадратная матрица A порядка M. Найти сумму элементов каждой ее диагонали, параллельной главной (начиная с одноэлементной диагонали $A_{1,M}$).

Matrix83. Дана квадратная матрица A порядка M. Найти сумму элементов каждой ее диагонали, параллельной побочной (начиная с одноэлементной диагонали $A_{1,1}$).

Matrix84. Дана квадратная матрица A порядка M. Найти среднее арифметическое элементов каждой ее диагонали, параллельной главной (начиная с одноэлементной диагонали $A_{1,M}$).

Matrix85. Дана квадратная матрица A порядка M. Найти среднее арифметическое элементов каждой ее диагонали, параллельной побочной (начиная с одноэлементной диагонали $A_{1,1}$).

Matrix86. Дана квадратная матрица A порядка M. Найти минимальный элемент для каждой ее диагонали, параллельной главной (начиная с одноэлементной диагонали $A_{1,M}$).

Matrix87. Дана квадратная матрица A порядка M. Найти максимальный элемент для каждой ее диагонали, параллельной побочной (начиная с одноэлементной диагонали $A_{1,1}$).

Matrix88°. Дана квадратная матрица порядка M. Обнулить элементы матрицы, лежащие ниже главной диагонали. Условный оператор не использовать.

Matrix89. Дана квадратная матрица порядка M. Обнулить элементы матрицы, лежащие выше побочной диагонали. Условный оператор не использовать.

Matrix90. Дана квадратная матрица порядка M. Обнулить элементы матрицы, лежащие на побочной диагонали и ниже нее. Условный оператор не использовать.

Matrix91. Дана квадратная матрица порядка M. Обнулить элементы матрицы, лежащие на главной диагонали и выше нее. Условный оператор не использовать.

Matrix92. Дана квадратная матрица порядка M. Обнулить элементы матрицы, лежащие одновременно выше главной диагонали и выше побочной диагонали. Условный оператор не использовать.

Matrix93. Дана квадратная матрица порядка M. Обнулить элементы матрицы, лежащие одновременно выше главной диагонали и ниже побочной диагонали. Условный оператор не использовать.

Matrix94. Дана квадратная матрица порядка M. Обнулить элементы матрицы, лежащие одновременно ниже главной диагонали (включая эту диагональ) и выше побочной диагонали (также включая эту диагональ). Условный

оператор не использовать.

Matrix95. Дана квадратная матрица порядка M. Обнулить элементы матрицы, лежащие одновременно ниже главной диагонали (включая эту диагональ) и ниже побочной диагонали (также включая эту диагональ). Условный оператор не использовать.

Matrix96. Дана квадратная матрица A порядка M. Зеркально отразить ее элементы относительно главной диагонали (при этом элементы главной диагонали останутся на прежнем месте, элемент $A_{1,2}$ поменяется местами с $A_{2,1}$, элемент $A_{1,3}$ — с $A_{3,1}$ и т. д.). Вспомогательную матрицу не использовать.

Matrix97. Дана квадратная матрица A порядка M. Зеркально отразить ее элементы относительно побочной диагонали. (при этом элементы побочной диагонали останутся на прежнем месте, элемент $A_{1,1}$ поменяется местами с $A_{M,M}$, элемент $A_{1,2}$ — с $A_{M-1,M}$ и т. д.). Вспомогательную матрицу не использовать.

Matrix98. Дана квадратная матрица A порядка M. Повернуть ее на угол 180° (при этом элемент $A_{1,1}$ поменяется местами с $A_{M,M}$, элемент $A_{1,2}$ — с $A_{M,M-1}$ и т. д.). Вспомогательную матрицу не использовать.

Matrix99. Дана квадратная матрица A порядка M. Повернуть ее на угол 90° в положительном направлении, то есть против часовой стрелки (при этом элемент $A_{1,1}$ перейдет в $A_{M,1}$, элемент $A_{M,1}$ — в $A_{M,M}$ и т. д.). Вспомогательную матрицу не использовать.

Matrix100. Дана квадратная матрица A порядка M. Повернуть ее на угол 90° в отрицательном направлении, то есть по часовой стрелке (при этом элемент $A_{1,1}$ перейдет в $A_{1,M}$, элемент $A_{1,M}$ — в $A_{M,M}$ и т. д.). Вспомогательную матрицу не использовать.

Тема 4. Основы ООП: классы, конструкторы, перегрузка
Практическая работа 12 Символы и строки

При выполнении заданий на обработку русских букв можно считать, что буква «ё» в исходных строковых данных отсутствует.

Символы и их коды. Формирование строк

String1. Дан символ *C*. Вывести его *код* (то есть номер в кодовой таблице).

String2. Дано целое число N ($32 \le N \le 126$). Вывести символ с кодом, равным N.

String3. Дан символ *C*. Вывести два символа, первый из которых предшествует символу *C* в кодовой таблице, а второй следует за символом *C*.

String4. Дано целое число N ($1 \le N \le 26$). Вывести N первых *прописных* (то есть заглавных) букв латинского алфавита.

String5. Дано целое число N ($1 \le N \le 26$). Вывести N последних *строчных* (то есть маленьких) букв латинского алфавита в обратном порядке (начиная с буквы «z»).

String6. Дан символ *C*, изображающий цифру или букву (латинскую или русскую). Если *C* изображает цифру, то вывести строку «digit», если латинскую букву — вывести строку «lat», если русскую — вывести строку «rus».

String7. Дана непустая строка. Вывести коды ее первого и последнего символа.

String8. Дано целое число N (> 0) и символ *C*. Вывести строку длины N, которая состоит из символов *C*.

String9°. Дано четное число N (> 0) и символы C_1 и C_2. Вывести строку длины N, которая состоит из чередующихся символов C_1 и C_2, начиная с C_1.

String10°. Дана строка. Вывести строку, содержащую те же символы, но расположенные в обратном порядке.

String11. Дана непустая строка *S*. Вывести строку, содержащую символы строки *S*, между которыми вставлено по одному пробелу.

String12. Дана непустая строка *S* и целое число N (> 0). Вывести строку, содержащую символы строки *S*, между которыми вставлено по N символов «*» (звездочка).

Посимвольный анализ и преобразование строк. Строки и числа

String13. Дана строка. Подсчитать количество содержащихся в ней цифр.

String14. Дана строка. Подсчитать количество содержащихся в ней прописных латинских букв.

String15. Дана строка. Подсчитать общее количество содержащихся в ней

строчных латинских и русских букв.

String16. Дана строка. Преобразовать в ней все прописные латинские буквы в строчные.

String17. Дана строка. Преобразовать в ней все строчные буквы (как латинские, так и русские) в прописные.

String18. Дана строка. Преобразовать в ней все строчные буквы (как латинские, так и русские) в прописные, а прописные — в строчные.

String19. Дана строка. Если она представляет собой запись целого числа, то вывести 1, если вещественного (с дробной частью) — вывести 2; если строку нельзя преобразовать в число, то вывести 0. Считать, что дробная часть вещественного числа отделяется от его целой части десятичной *точкой* «.».

String20. Дано целое положительное число. Вывести символы, изображающие цифры этого числа (в порядке слева направо).

String21. Дано целое положительное число. Вывести символы, изображающие цифры этого числа (в порядке справа налево).

String22. Дана строка, изображающая целое положительное число. Вывести сумму цифр этого числа.

String23. Дана строка, изображающая арифметическое выражение вида «<цифра>±<цифра> ± . .±<цифра>», где на месте знака операции «±» находится символ «+» или «—» (например, «4+7– 2 8»). Вывести значение данного выражения (целое число).

String24. Дана строка, изображающая двоичную запись целого положительного числа. Вывести строку, изображающую десятичную запись этого же числа.

String25. Дана строка, изображающая десятичную запись целого положительного числа. Вывести строку, изображающую двоичную запись этого же числа.

Обработка строк с помощью стандартных функций. Поиск и замена

В заданиях, связанных с поиском и заменой подстрок, можно считать, что исходная строка не содержит *перекрывающихся* вхождений требуемых подстрок. В заданиях String32, String35 и String38, кроме этого, можно также считать, что удаление (в String32 и String35) или замена (в String38) любого вхождения подстроки не приведет к появлению в строке *новых* вхождений данной подстроки.

String26. Дано целое число N (> 0) и строка S. Преобразовать строку S в

строку длины N следующим образом: если длина строки S больше N, то отбросить первые символы, если длина строки S меньше N, то в ее начало добавить символы «.» (точка).

String27. Даны целые положительные числа N_1 и N_2 и строки S_1 и S_2. Получить из этих строк новую строку, содержащую первые N_1 символов строки S_1 и последние N_2 символов строки S_2 (в указанном порядке).

String28. Дан символ C и строка S. Удвоить каждое вхождение символа C в строку S.

String29°. Дан символ C и строки S, S_0. Перед каждым вхождением символа C в строку S вставить строку S_0.

String30. Дан символ C и строки S, S_0. После каждого вхождения символа C в строку S вставить строку S_0.

String31. Даны строки S и S_0. Проверить, содержится ли строка S_0 в строке S. Если содержится, то вывести TRUE, если не содержится, то вывести FALSE.

String32. Даны строки S и S_0. Найти количество вхождений строки S_0 в строку S.

String33. Даны строки S и S_0. Удалить из строки S первую подстроку, совпадающую с S_0. Если совпадающих подстрок нет, то вывести строку S без изменений.

String34. Даны строки S и S_0. Удалить из строки S последнюю подстроку, совпадающую с S_0. Если совпадающих подстрок нет, то вывести строку S без изменений.

String35. Даны строки S и S_0. Удалить из строки S все подстроки, совпадающие с S_0. Если совпадающих подстрок нет, то вывести строку S без изменений.

String36. Даны строки S, S_1 и S_2. Заменить в строке S первое вхождение строки S_1 на строку S_2.

String37. Даны строки S, S_1 и S_2. Заменить в строке S последнее вхождение строки S_1 на строку S_2.

String38. Даны строки S, S_1 и S_2. Заменить в строке S все вхождения строки S_1 на строку S_2.

String39. Дана строка, содержащая по крайней мере один символ пробела. Вывести подстроку, расположенную между первым и вторым пробелом исходной строки. Если строка содержит только один пробел, то вывести пустую строку.

String40. Дана строка, содержащая по крайней мере один символ пробела. Вывести подстроку, расположенную между первым и последним пробелом исходной строки. Если строка содержит только один пробел, то вывести пустую строку.

Во всех заданиях данного пункта предполагается, что исходные строки являются непустыми и не содержат начальных и конечных пробелов.

String41°. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Найти количество слов в строке.

String42. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Найти количество слов, которые начинаются и заканчиваются одной и той же буквой.

String43. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Найти количество слов, которые содержат хотя бы одну букву «А».

String44. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Найти количество слов, которые содержат ровно три буквы «А».

String45. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Найти длину самого короткого слова.

String46. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Найти длину самого длинного слова.

String47. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова, разделенные одним символом «.» (точка). В конце строки точку не ставить.

String48. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Преобразовать каждое слово в строке, заменив в нем все последующие вхождения его первой буквы на символ «.» (точка). Например, слово «МИНИМУМ» надо преобразовать в «МИНИ.У.». Количество пробелов между словами не изменять.

String49. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Преобразовать каждое слово в строке, заменив в нем все предыдущие вхождения его последней буквы на символ «.» (точка). Например, слово «МИНИМУМ» надо преобразовать в «.ИНИ.УМ». Количество пробелов между словами не изменять.

String50. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова, разделенные одним пробелом и расположенные в обратном порядке.

String51. Дана строка, состоящая из русских слов, набранных заглавными

буквами и разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова, разделенные одним пробелом и расположенные в алфавитном порядке.

String52. Дана строка-предложение на русском языке. Преобразовать строку так, чтобы каждое слово начиналось с заглавной буквы. *Словом* считать набор символов, не содержащий пробелов и ограниченный пробелами или началом/концом строки. Слова, не начинающиеся с буквы, не изменять.

String53. Дана строка-предложение на русском языке. Подсчитать количество содержащихся в строке знаков препинания.

String54. Дана строка-предложение на русском языке. Подсчитать количество содержащихся в строке гласных букв.

String55. Дана строка-предложение на русском языке. Вывести самое длинное слово в предложении. Если таких слов несколько, то вывести первое из них. *Словом* считать набор символов, не содержащий пробелов, знаков препинания и ограниченный пробелами, знаками препинания или началом/концом строки.

String56. Дана строка-предложение на русском языке. Вывести самое короткое слово в предложении. Если таких слов несколько, то вывести последнее из них. *Словом* считать набор символов, не содержащий пробелов, знаков препинания и ограниченный пробелами, знаками препинания или началом/концом строки.

String57. Дана строка-предложение с избыточными пробелами между словами. Преобразовать ее так, чтобы между словами был ровно один пробел.

Дополнительные задания на обработку строк

String58. Дана строка, содержащая *полное имя файла*, то есть имя диска, список каталогов (путь), собственно имя и расширение. Выделить из этой строки имя файла (без расширения).

String59. Дана строка, содержащая *полное имя файла*, то есть имя диска, список каталогов (путь), собственно имя и расширение. Выделить из этой строки расширение файла (без предшествующей точки).

String60. Дана строка, содержащая полное имя файла. Выделить из этой строки название первого каталога (без символов «\»). Если файл содержится в корневом каталоге, то вывести символ «\».

String61. Дана строка, содержащая полное имя файла. Выделить из этой строки название последнего каталога (без символов «\»). Если файл содержится в корневом каталоге, то вывести символ «\».

String62. Дана строка-предложение на русском языке. Зашифровать ее, выполнив циклическую замену каждой буквы на следующую за ней в алфавите

и сохранив при этом регистр букв («А» перейдет в «Б», «а» — в «б», «Б» — в «В», «я» — в «а» и т. д.). Букву «ё» в алфавите не учитывать («е» должна переходить в «ж»). Знаки препинания и пробелы не изменять.

String63. Дана строка-предложение на русском языке и число K ($0 < K < 10$). Зашифровать строку, выполнив циклическую замену каждой буквы на букву того же регистра, расположенную в алфавите на K-й позиции после шифруемой буквы (например, для $K = 2$ «А» перейдет в «В», «а» — в «в», «Б» — в «Г», «я» — в «б» и т. д.). Букву «ё» в алфавите не учитывать, знаки препинания и пробелы не изменять.

String64. Дано зашифрованное предложение на русском языке (способ шифрования описан в задании String63) и кодовое смещение K ($0 < K < 10$). Расшифровать предложение.

String65. Дано зашифрованное предложение на русском языке (способ шифрования описан в задании String63) и его расшифрованный первый символ C. Найти кодовое смещение K и расшифровать предложение.

String66. Дана строка-предложение. Зашифровать ее, поместив вначале все символы, расположенные на четных позициях строки, а затем, в обратном порядке, все символы, расположенные на нечетных позициях (например, строка «Программа» превратится в «ргамамроП»).

String67. Дано предложение, зашифрованное по правилу, описанному в задании String66. Расшифровать это предложение.

String68. Дана строка, содержащая цифры и строчные латинские буквы. Если буквы в строке упорядочены по алфавиту, то вывести 0; в противном случае вывести номер первого символа строки, нарушающего алфавитный порядок.

String69. Дана строка, содержащая латинские буквы и круглые скобки. Если скобки расставлены правильно (то есть каждой открывающей соответствует одна закрывающая), то вывести число 0. В противном случае вывести или номер позиции, в которой расположена первая ошибочная закрывающая скобка, или, если закрывающих скобок не хватает, число −1.

String70°. Дана строка, содержащая латинские буквы и скобки трех видов: «()», «[]», «{}». Если скобки расставлены правильно (то есть каждой открывающей соответствует закрывающая скобка того же вида), то вывести число 0. В противном случае вывести или номер позиции, в которой расположена первая ошибочная скобка, или, если закрывающих скобок не хватает, число −1.

Тема 4. Основы ООП: классы, конструкторы, перегрузка
Практическая работа 13 Составные типы данных в процедурах и функциях

В каждом задании данного раздела требуется описать процедуру или функцию и затем использовать ее для обработки исходных данных. Все параметры любой *функции* считаются входными. Для *процедур* всегда указывается, какие параметры являются выходными (или одновременно входными и выходными); если о виде параметра процедуры ничего не сказано, то он считается входным.

Одномерные и двумерные массивы

При вводе исходного массива вначале следует ввести его размер (одно число для одномерных массивов, два числа — количество строк и столбцов — для двумерных массивов-матриц), а затем — все его элементы.

Если в задании явно не указывается размер одномерного массива, являющегося параметром процедуры или функции, то предполагается, что этот размер может изменяться в пределах от 1 до 10. Для двумерных массивов-матриц предполагается, что число их строк и столбцов может меняться от 1 до 10. Индексы начальных элементов как одномерных, так и двумерных массивов всегда считаются равными 1.

При описании процедур, выполняющих преобразование массива, не следует использовать вспомогательный массив того же размера.

Param1°. Описать функцию MinElem(*A, N*) целого типа, находящую минимальный элемент целочисленного массива *A* размера *N*. С помощью этой функции найти минимальные элементы массивов *A, B, C* размера N_A, N_B, N_C соответственно.

Param2. Описать функцию MaxNum(*A, N*) целого типа, находящую номер максимального элемента вещественного массива *A* размера *N*. С помощью этой функции найти номера максимальных элементов массивов *A, B, C* размера N_A, N_B, N_C соответственно.

Param3. Описать процедуру MinmaxNum(*A, N, NMin, NMax*), находящую номера минимального и максимального элемента вещественного массива *A* размера *N*. Выходные параметры целого типа: *NMin* (номер минимального элемента) и *NMax* (номер максимального элемента). С помощью этой процедуры найти номера минимальных и максимальных элементов массивов *A, B, C* размера N_A, N_B, N_C соответственно.

Param4. Описать процедуру Invert(A, N), меняющую порядок следования элементов вещественного массива A размера N на противоположный (*инвертирование* массива). Массив A является входным и выходным параметром. С помощью этой процедуры инвертировать массивы A, B, C размера N_A, N_B, N_C соответственно.

Param5. Описать процедуру Smooth1(A, N), выполняющую *сглаживание* вещественного массива A размера N следующим образом: элемент A_K заменяется на среднее арифметическое первых K исходных элементов массива A. Массив A является входным и выходным параметром. С помощью этой процедуры выполнить пятикратное сглаживание данного массива A размера N, выводя результаты каждого сглаживания.

Param6. Описать процедуру Smooth2(A, N), выполняющую *сглаживание* вещественного массива A размера N следующим образом: элемент A_1 не изменяется, элемент A_K ($K = 2, \ldots, N$) заменяется на полусумму исходных элементов A_{K-1} и A_K. Массив A является входным и выходным параметром. С помощью этой процедуры выполнить пятикратное сглаживание данного массива A размера N, выводя результаты каждого сглаживания.

Param7. Описать процедуру Smooth3(A, N), выполняющую *сглаживание* вещественного массива A размера N следующим образом: каждый элемент массива заменяется на его среднее арифметическое с соседними элементами (при вычислении среднего арифметического используются *исходные* значения соседних элементов). Массив A является входным и выходным параметром. С помощью этой процедуры выполнить пятикратное сглаживание данного массива A размера N, выводя результаты каждого сглаживания.

Param8. Описать процедуру RemoveX(A, N, X), удаляющую из целочисленного массива A размера N элементы, равные целому числу X. Массив A и число N являются входными и выходными параметрами. С помощью этой процедуры удалить числа X_A, X_B, X_C из массивов A, B, C размера N_A, N_B, N_C соответственно и вывести размер и содержимое полученных массивов.

Param9. Описать процедуру RemoveForInc(A, N), удаляющую из вещественного массива A размера N «лишние» элементы так, чтобы оставшиеся элементы оказались упорядоченными по возрастанию: первый элемент не удаляется, второй элемент удаляется, если он меньше первого, третий — если он меньше предыдущего элемента, оставленного в массиве, и т. д.

Например, массив 5.5, 2.5, 4.6, 7.2, 5.8, 9.4 должен быть преобразован к виду 5.5, 7.2, 9.4. Массив A и число N являются входными и выходными параметрами. С помощью этой процедуры преобразовать массивы A, B, C размера N_A, N_B, N_C соответственно и вывести размер и содержимое полученных массивов.

Param10. Описать процедуру DoubleX(A, N, X), дублирующую в целочисленном массиве A размера N элементы, равные целому числу X. Массив A и число N являются входными и выходными параметрами. С помощью этой процедуры продублировать числа X_A, X_B, X_C в массивах A, B, C размера N_A, N_B, N_C соответственно и вывести размер и содержимое полученных массивов.

Param11. Описать процедуру SortArray(A, N), выполняющую сортировку по возрастанию вещественного массива A размера N. Массив A является входным и выходным параметром. С помощью этой процедуры отсортировать массивы A, B, C размера N_A, N_B, N_C соответственно.

Param12. Описать процедуру SortIndex(A, N, I), формирующую для вещественного массива A размера N *индексный массив* I — массив целых чисел того же размера, содержащий номера элементов массива A в том порядке, который соответствует возрастанию элементов массива A (сам массив A при этом не изменяется). Индексный массив I является выходным параметром. С помощью этой процедуры создать индексные массивы для массивов A, B, C размера N_A, N_B, N_C соответственно.

Param13. Описать процедуру Bell(A, N), меняющую порядок элементов вещественного массива A размера N на следующий: наименьший элемент массива располагается на первом месте, наименьший из оставшихся элементов — на последнем, следующий по величине располагается на втором месте, следующий — на предпоследнем и т. д. (в результате график значений элементов будет напоминать *колокол*). Массив A является входным и выходным параметром. С помощью этой процедуры преобразовать массивы A, B, C размера N_A, N_B, N_C соответственно.

Param14. Описать процедуру Split1(A, N_A, B, N_B, C, N_C), формирующую по вещественному массиву A размера N_A два вещественных массива B и C размера N_B и N_C соответственно; при этом массив B содержит все элементы массива A с нечетными порядковыми номерами (1, 3, . . .), а массив C — все элементы массива A с четными номерами (2, 4, . . .). Массивы B и C и числа N_B и N_C являются выходными параметрами.

Применить эту процедуру к данному массиву A размера N_A и вывести размер и содержимое полученных массивов B и C.

Param15. Описать процедуру Split2(A, N_A, B, N_B, C, N_C), формирующую по целочисленному массиву A размера N_A два целочисленных массива B и C размера N_B и N_C соответственно; при этом массив B содержит все четные числа из массива A, а массив C — все нечетные числа (в том же порядке). Массивы B и C и числа N_B и N_C являются выходными параметрами. Применить эту процедуру к данному массиву A размера N_A и вывести размер и содержимое полученных массивов B и C.

Param16. Описать процедуру ArrayToMatrRow(A, K, M, N, B), формирующую по вещественному массиву A размера K матрицу B размера $M \times N$ (матрица заполняется элементами массива A по строкам). «Лишние» элементы массива игнорируются; если элементов массива недостаточно, то оставшиеся элементы матрицы полагаются равными 0. Двумерный массив B является выходным параметром. С помощью этой процедуры на основе данного массива A размера K и целых чисел M и N сформировать матрицу B размера $M \times N$.

Param17°. Описать процедуру ArrayToMatrCol(A, K, M, N, B), формирующую по вещественному массиву A размера K матрицу B размера $M \times N$ (матрица заполняется элементами массива A по столбцам). «Лишние» элементы массива игнорируются; если элементов массива недостаточно, то оставшиеся элементы матрицы полагаются равными 0. Двумерный массив B является выходным параметром. С помощью этой процедуры на основе данного массива A размера K и целых чисел M и N сформировать матрицу B размера $M \times N$.

Param18. Описать процедуру Chessboard(M, N, A), формирующую по целым положительным числам M и N матрицу A размера $M \times N$, которая содержит числа 0 и 1, расположенные в «шахматном» порядке, причем $A_{1,1} = 0$. Двумерный целочисленный массив A является выходным параметром. С помощью этой процедуры по данным целым числам M и N сформировать матрицу A размера $M \times N$.

Param19. Описать функцию Norm1(A, M, N) вещественного типа, вычисляющую *норму* вещественной матрицы A размера $M \times N$:

$$\text{Norm1}(A, M, N) = \max \{|A_{1,J}| + |A_{2,J}| + \ldots + |A_{M,J}|\},$$

где максимум берется по всем J от 1 до N. Для данной матрицы A размера $M \times N$ найти Norm1(A, K, N), $K = 1, \ldots, M$.

Param20. Описать функцию Norm2(A, M, N) вещественного типа, вычисляющую *норму* вещественной матрицы A размера $M \times N$:

$$\text{Norm2}(A, M, N) = \max \{|A_{I,1}| + |A_{I,2}| + \ldots + |A_{I,N}|\},$$

где максимум берется по всем I от 1 до M. Для данной матрицы A размера $M \times N$ найти Norm2(A, K, N), $K = 1, \ldots, M$.

Param21. Описать функцию SumRow(A, M, N, K) вещественного типа, вычисляющую сумму элементов вещественной матрицы A размера $M \times N$, расположенных в K-й строке (если $K > M$, то функция возвращает 0). Для данной матрицы A размера $M \times N$ и трех данных K найти SumRow(A, M, N, K).

Param22. Описать функцию SumCol(A, M, N, K) вещественного типа, вычисляющую сумму элементов вещественной матрицы A размера $M \times N$, расположенных в K-м столбце (если $K > N$, то функция возвращает 0). Для данной матрицы A размера $M \times N$ и трех данных K найти SumCol(A, M, N, K).

Param23. Описать процедуру SwapRow(A, M, N, K_1, K_2), осуществляющую перемену местами строк вещественной матрицы A размера $M \times N$ с номерами K_1 и K_2. Матрица A является входным и выходным параметром; если K_1 или K_2 больше M, то матрица не изменяется. Используя эту процедуру, поменять для данной матрицы A размера $M \times N$ строки с данными номерами K_1 и K_2.

Param24. Описать процедуру SwapCol(A, M, N, K_1, K_2), осуществляющую перемену местами столбцов вещественной матрицы A размера $M \times N$ с номерами K_1 и K_2. Матрица A является входным и выходным параметром; если K_1 или K_2 больше N, то матрица не изменяется. Используя эту процедуру, поменять для данной матрицы A размера $M \times N$ столбцы с данными номерами K_1 и K_2.

Param25. Описать процедуру Transp(A, M), выполняющую *транспонирование* (то есть зеркальное отражение относительно главной диагонали) квадратной вещественной матрицы A порядка M. Матрица A является входным и выходным параметром. Используя эту процедуру, транспонировать данную матрицу A порядка M.

Param26. Описать процедуру RemoveRows(A, M, N, K_1, K_2), удаляющую из вещественной матрицы A размера $M \times N$ строки с номерами от K_1 до K_2 включительно (предполагается, что $1 < K_1 \le K_2$). Если $K_1 > M$, то матрица не изменяется; если $K_2 > M$, то удаляются строки матрицы с номерами

от K_1 до M. Двумерный массив A и числа M, N являются входными и выходными параметрами. Используя процедуру RemoveRows, удалить из данной матрицы A размера $M \times N$ строки с номерами от K_1 до K_2 и вывести размер полученной матрицы и ее элементы.

Param27. Описать процедуру RemoveCols(A, M, N, K_1, K_2), удаляющую из вещественной матрицы A размера $M \times N$ столбцы с номерами от K_1 до K_2 включительно (предполагается, что $1 < K_1 \le K_2$). Если $K_1 > N$, то матрица не изменяется; если $K_2 > N$, то удаляются столбцы матрицы с номерами от K_1 до N. Двумерный массив A и числа M, N являются входными и выходными параметрами. Используя процедуру RemoveCols, удалить из данной матрицы A размера $M \times N$ столбцы с номерами от K_1 до K_2 и вывести размер полученной матрицы и ее элементы.

Param28. Описать процедуру RemoveRowCol(A, M, N, K, L), удаляющую из вещественной матрицы A размера $M \times N$ строку и столбец, которые содержат элемент $A_{K,L}$ (предполагается, что $M > 1$ и $N > 1$; если $K > M$ или $L > N$, то матрица не изменяется). Двумерный массив A и числа M, N являются входными и выходными параметрами. Дана матрица A размера $M \times N$ и числа K, L. Применить к матрице A процедуру RemoveRowCol и вывести размер полученной матрицы и ее элементы.

Param29. Описать процедуру SortCols(A, M, N), выполняющую сортировку по возрастанию столбцов целочисленной матрицы A размера $M \times N$ (столбцы сравниваются *лексикографически*: если первые элементы столбцов различны, то меньшим считается столбец, содержащий меньший первый элемент; если первые элементы столбцов равны, то анализируются их вторые элементы и т. д.). Двумерный массив A является входным и выходным параметром. Используя процедуру SortCols, отсортировать столбцы данной матрицы A размера $M \times N$.

Строки

Param30. Описать функцию IsIdent(S) целого типа, проверяющую, является ли строка S допустимым *идентификатором*, то есть непустой строкой, которая содержит только латинские буквы, цифры и символ подчеркивания «» и не начинается с цифры. Если S является допустимым идентификатором, то функция возвращает 0. Если S является пустой строкой, то возвращается —1, если S начинается с цифры, то возвращается – 2. Если S содержит недопустимые символы, то возвращается номер первого недо-

пустимого символа. Проверить с помощью функции IsIdent пять данных строк.

Param31. Описать функцию FillStr(*S*, *N*) строкового типа, возвращающую строку длины *N*, заполненную повторяющимися копиями строки-*шаблона S* (последняя копия строки-шаблона может входить в результирующую строку частично). Используя эту функцию, сформировать по данному числу *N* и пяти данным строкам-шаблонам пять результирующих строк длины *N*.

Param32. Описать процедуру UpCaseRus(*S*), преобразующую все строчные русские буквы строки *S* в прописные (остальные символы строки *S* не изменяются). Строка *S* является входным и выходным параметром. Используя процедуру UpCaseRus, преобразовать пять данных строк.

Param33. Описать процедуру LowCaseRus(*S*), преобразующую все прописные русские буквы строки *S* в строчные (остальные символы строки *S* не изменяются). Строка *S* является входным и выходным параметром. Используя процедуру LowCaseRus, преобразовать пять данных строк.

Param34. Описать процедуру TrimLeftC(*S, C*), удаляющую в строке *S* начальные символы, совпадающие с символом *C*. Строка *S* является входным и выходным параметром. Дан символ *C* и пять строк. Используя процедуру TrimLeftC, преобразовать данные строки.

Param35. Описать процедуру TrimRightC(*S, C*), удаляющую в строке *S* конечные символы, совпадающие с символом *C*. Строка *S* является входным и выходным параметром. Дан символ *C* и пять строк. Используя процедуру TrimRightC, преобразовать данные строки.

Param36. Описать функцию InvertStr(*S, K, N*) строкового типа, возвращающую *инвертированную подстроку* строки *S*, содержащую в обратном порядке *N* символов строки *S*, начиная с ее *K*-го символа. Если *K* превосходит длину строки *S*, то возвращается пустая строка; если длина строки меньше *K* + *N*, то инвертируются все символы строки, начиная с ее *K*-го символа. Вывести значения функции InvertStr для данной строки *S* и каждой из трех пар положительных целых чисел: (K_1, N_1), (K_2, N_2), (K_3, N_3).

Param37. Описать функцию PosSub(S_0, *S*, *K*, *N*) целого типа, возвращающую номер позиции, начиная с которой в строке *S* содержится первое вхождение строки S_0, причем анализируются только *N* символов строки *S*, начиная с ее *K*-го символа (таким образом, PosSub обеспечивает *поиск в подстроке*). Если *K* превосходит длину строки *S*, то возвращается 0,

если длина строки меньше $K + N$, то анализируются все символы строки, начиная с ее K-го символа. Если в требуемой подстроке строки S вхождения S_0 отсутствуют, то функция возвращает 0. Вывести значения функции PosSub для данных строк S_0, S и каждой из трех пар положительных целых чисел: (K_1, N_1), (K_2, N_2), (K_3, N_3).

Param38. Описать функцию PosLast(S_0, S) целого типа, возвращающую номер позиции, начиная с которой в строке S содержится последнее вхождение подстроки S_0. Считать, что перекрывающихся вхождений подстрок S_0 строка S не содержит. Если в строке S отсутствуют подстроки S_0, то функция возвращает 0. Вывести значения этой функции для пяти данных пар строк S_0 и S.

Param39. Описать функцию PosK(S_0, S, K) целого типа, возвращающую номер позиции, начиная с которой в строке S содержится K-е вхождение подстроки S_0 ($K > 0$). Если количество вхождений S_0 в строке S меньше K, то функция возвращает 0. Считать, что перекрывающихся вхождений подстрок S_0 строка S не содержит. Вывести значения этой функции для пяти данных троек: S_0, S и K.

Param40. Описать функцию WordK(S, K) строкового типа, возвращающую K-е слово строки S (*словом* считается набор символов, не содержащий пробелов и ограниченный пробелами или началом/концом строки). Если количество слов в строке меньше K, то функция возвращает пустую строку. Используя эту функцию, выделить из данной строки S слова с данными номерами K_1, K_2, K_3.

Param41°. Описать процедуру SplitStr(S, W, N), которая формирует по данной строке S массив W слов, входящих в S (массив W и его размер N являются выходными параметрами). *Словом* считается набор символов, не содержащий пробелов и ограниченный пробелами или началом/концом строки; предполагается, что строка S содержит не более 10 слов. Используя функцию SplitStr, найти количество слов N, содержащихся в данной строке S, и сами эти слова.

Param42. Описать функцию CompressStr(S) строкового типа, выполняющую *сжатие* строки S по следующему правилу: каждая подстрока строки S, состоящая из более чем четырех одинаковых символов C, заменяется текстом вида «C{K}», где K — количество символов C (предполагается, что строка S не содержит фигурных скобок «{» и «}»). Например, для строки S = «bbbccccce» функция вернет строку «bbbc{5}e». С помощью функции

CompressStr сжать пять данных строк.

Param43. Описать функцию DecompressStr(*S*) строкового типа, восстанавливающую строку, сжатую процедурой CompressStr (см. задание Param42). Параметр *S* содержит сжатую строку; восстановленная строка является возвращаемым значением функции. С помощью функции DecompressStr восстановить пять данных сжатых строк.

Param44. Описать функцию DecToBin(*N*) строкового типа, возвращающую строковое представление целого неотрицательного числа *N* в двоичной системе счисления. Результирующая строка состоит из символов «0»–«1» и не содержит ведущих нулей (за исключением представления числа 0). Используя эту функцию, получить двоичные представления пяти данных чисел.

Param45. Описать функцию DecToHex(*N*) строкового типа, возвращающую строковое представление целого неотрицательного числа *N* в 16-ричной системе счисления. Результирующая строка состоит из символов «0»–«9», «A»–«F» и не содержит ведущих нулей (за исключением представления числа 0). Используя эту функцию, получить 16-ричные представления пяти данных чисел.

Param46. Описать функцию BinToDec(*S*) целого типа, определяющую целое неотрицательное число по его строковому представлению *S* в двоичной системе счисления. Параметр *S* имеет строковый тип, состоит из символов «0»–«1» и не содержит ведущих нулей (за исключением значения «0»). Используя эту функцию, вывести пять чисел, для которых даны их двоичные представления.

Param47. Описать функцию HexToDec(*S*) целого типа, определяющую целое неотрицательное число по его строковому представлению *S* в 16-ричной системе счисления. Параметр *S* имеет строковый тип, состоит из символов «0»–«9», «A»–«F» и не содержит ведущих нулей (за исключением значения «0»). Используя эту функцию, вывести пять чисел, для которых даны их 16-ричные представления.

Файлы

Param48. Описать функцию IntFileSize(*S*) целого типа, возвращающую количество элементов в файле целых чисел с именем *S*. Если файл не существует, то функция возвращает –1. С помощью этой функции найти количество элементов в трех файлах с данными именами.

Param49. Описать функцию LineCount(*S*) целого типа, возвращающую количество строк в текстовом файле с именем *S*. Если файл не существует, то функция возвращает –1. С помощью этой функции найти количество строк в трех файлах с данными именами.

Param50. Описать процедуру InvertIntFile(*S*), меняющую порядок следования элементов файла целого типа с именем *S* на противоположный. Если файл не существует или содержит менее двух элементов, то процедура не выполняет никаких действий. Обработать с помощью этой процедуры три файла с данными именами.

Param51°. Описать процедуру AddLineNumbers(*S*, *N*, *K*, *L*), добавляющую в начало каждой строки существующего текстового файла с именем *S* ее порядковый номер: первая строка получает номер *N*, вторая — $N + 1$ и т. д. Номер отображается в *K* позициях, выравнивается по правому краю и отделяется от последующего текста *L* пробелами ($K > 0$, $L > 0$). Если строка файла является пустой, то она также нумеруется, но пробелы после номера не добавляются. Применить эту процедуру к данному файлу, используя указанные значения *N*, *K* и *L*.

Param52. Описать процедуру RemoveLineNumbers(*S*), удаляющую из начала каждой строки существующего текстового файла с именем *S* ее порядковый номер, добавленный процедурой AddLineNumbers (см. задание Param51), а также пробелы, отделяющие номер от последующего текста. Если строки не содержат номеров, то процедура не выполняет никаких действий. Применить эту процедуру к файлу с данным именем.

Param53. Описать процедуру SplitIntFile(S_0, *K*, S_1, S_2), копирующую первые *K* (≥ 0) элементов существующего файла целых чисел с именем S_0 в новый файл целых чисел с именем S_1, а остальные элементы — в новый файл целых чисел с именем S_2. Один из созданных файлов может остаться пустым. Применить эту процедуру к файлу с данным именем S_0, используя указанные значения *K*, S_1 и S_2.

Param54. Описать процедуру SplitText(S_0, *K*, S_1, S_2), копирующую первые *K* (≥ 0) строк существующего текстового файла с именем S_0 в новый текстовый файл с именем S_1, а остальные строки — в новый текстовый файл с именем S_2. Один из созданных файлов может остаться пустым. Применить эту процедуру к файлу с данным именем S_0, используя указанные значения *K*, S_1 и S_2.

Param55. Описать процедуру StringFileToText(*S*), преобразующую двоичный

строковый файл с именем *S* в текстовый файл с тем же именем. Используя эту процедуру, преобразовать два данных строковых файла с именами S_1 и S_2 в текстовые.

Param56. Описать процедуру TextToStringFile(*S*), преобразующую текстовый файл с именем *S* в двоичный строковый файл с тем же именем. Используя эту процедуру, преобразовать два данных текстовых файла с именами S_1 и S_2 в строковые.

Param57. Описать процедуру EncodeText(*S, K*), которая шифрует текстовый файл с именем *S*, выполняя циклическую замену каждой русской буквы на букву того же регистра, расположенную в алфавите на *K*-й позиции после шифруемой буквы ($0 < K < 10$). Например, при $K = 3$ «А» перейдет в «Г», «я» — в «в». Букву «ё» в алфавите не учитывать, считая, что за буквой «е» сразу идет «ж». Символы, не являющиеся русскими буквами, при шифровании не изменять. Используя эту процедуру и зная кодовое смещение *K*, зашифровать файл с указанным именем.

Param58. Описать процедуру DecodeText(*S, K*), которая дешифрует текстовый файл с именем *S*, зашифрованный с использованием кодового смещения *K* (способ шифрования описан в задании Param57). Используя эту процедуру и зная кодовое смещение *K*, расшифровать файл с указанным именем.

Записи

При вводе и выводе каждой даты в заданиях Param59–Param63 вначале указывается день, затем номер месяца, затем год. При вводе каждой точки в заданиях Param64–Param70 вначале указывается ее абсцисса (*x*-координата), затем ее ордината (*y*-координата).

Param59. Описать тип TDate — запись с полями целого типа Day (день), Month (месяц) и Year (год) — и функцию LeapYear(*D*) логического типа с параметром типа TDate, которая возвращает TRUE, если год в дате *D* является високосным, и FALSE в противном случае. Вывести значение функции LeapYear для пяти данных дат (предполагается, что все даты являются правильными). *Високосным* считается год, делящийся на 4, за исключением тех годов, которые делятся на 100 и не делятся на 400.

Param60. Используя тип TDate и функцию LeapYear (см. задание Param59), описать функцию DaysInMonth(*D*) целого типа с параметром типа TDate, которая возвращает количество дней для месяца, указанного в дате *D*.

Вывести значение функции DaysInMonth для пяти данных дат (предполагается, что все даты являются правильными).

Param61. Используя тип TDate и функцию DaysInMonth (см. задания Param59 и Param60), описать функцию CheckDate(*D*) целого типа с параметром типа TDate, которая проверяет правильность даты, указанной в параметре *D*. Если дата *D* является правильной, то функция возвращает 0; если в дате указан неверный номер месяца, то функция возвращает 1; если в дате указан неверный день для данного месяца, то возвращается 2. Вывести значение функции CheckDate для пяти данных дат.

Param62. Используя тип TDate и функции DaysInMonth и CheckDate (см. задания Param59–Param61), описать процедуру PrevDate(*D*) с параметром типа TDate, которая преобразует дату *D* к предыдущей дате (если дата *D* является неправильной, то она не изменяется). Запись *D* является входным и выходным параметром. Применить процедуру PrevDate к пяти данным датам.

Param63. Используя тип TDate и функции DaysInMonth и CheckDate (см. задания Param59–Param61), описать процедуру NextDate(*D*) с параметром типа TDate, которая преобразует дату *D* к следующей дате (если дата *D* является неправильной, то она не изменяется). Запись *D* является входным и выходным параметром. Применить процедуру NextDate к пяти данным датам.

Param64. Описать тип TPoint — запись с полями вещественного типа *X* и *Y* (координаты точки на плоскости) — и функцию Leng(*A*, *B*) вещественного типа, находящую длину отрезка *AB* на плоскости по координатам его концов:

$$|AB| = \sqrt{(A.X - B.X)^2 + (A.Y - B.Y)^2}$$

(*A* и *B* — параметры типа TPoint). С помощью этой функции найти длины отрезков *AB*, *AC*, *AD*, если даны координаты точек *A*, *B*, *C*, *D*.

Param65. Используя тип TPoint и функцию Leng (см. задание Param64), описать тип TTriangle — запись с полями *A*, *B*, *C* типа TPoint (вершины треугольника) — и функцию Perim(*T*) вещественного типа, находящую периметр треугольника *T* (*T* — параметр типа TTriangle). С помощью этой функции найти периметры треугольников *ABC*, *ABD*, *ACD*, если даны координаты точек *A*, *B*, *C*, *D*.

Param66. Используя типы TPoint, TTriangle и функции Leng и Perim (см. задания Param64 и Param65), описать функцию Area(*T*) вещественного

типа, находящую площадь треугольника T (T — параметр типа TTriangle) по *формуле Герона*:

$$S_{ABC} = \sqrt{p\cdot(p - |AB|)\cdot(p - |AC|)\cdot(p - |BC|)},$$

где p — *полупериметр*. С помощью этой функции найти площади треугольников *ABC*, *ABD*, *ACD*, если даны координаты точек *A*, *B*, *C*, *D*.

Param67. Используя типы TPoint, TTriangle и функции Leng и Area (см. задания Param64–Param66), описать функцию Dist(P, A, B) вещественного типа (P, A, B — параметры типа TPoint), находящую расстояние $D(P, AB)$ от точки P до прямой AB по формуле

$$D(P, AB) = 2\cdot S_{PAB}/|AB|,$$

где S_{PAB} — площадь треугольника *PAB*. С помощью этой функции найти расстояния от точки P до прямых *AB*, *AC*, *BC*, если даны координаты точек *P*, *A*, *B*, *C*.

Param68. Используя типы TPoint, TTriangle и функцию Dist (см. задания Param64, Param65, Param67), описать процедуру Heights(T, h_1, h_2, h_3), находящую высоты h_1, h_2, h_3 треугольника T (T — входной параметр типа TTriangle, h_1, h_2, h_3 — выходные вещественные параметры), проведенные соответственно из вершин *T.A*, *T.B*, *T.C*. С помощью этой процедуры найти высоты треугольников *ABC*, *ABD*, *ACD*, если даны координаты точек *A*, *B*, *C*, *D*.

Param69. Используя тип TPoint и функцию Leng (см. задание Param64), описать функцию PerimN(P, N) вещественного типа, находящую периметр N-угольника, вершины которого (в порядке их обхода) передаются в массиве P размера N (> 2) с элементами типа TPoint. С помощью этой функции найти периметры трех многоугольников, если дано число их сторон и координаты их вершин.

Param70. Используя типы TPoint, TTriangle и функцию Area (см. задания Param64–Param66), описать функцию AreaN(P, N) вещественного типа, находящую площадь выпуклого N-угольника, вершины которого (в порядке их обхода) передаются в массиве P размера N (> 2) с элементами типа TPoint. С помощью этой функции найти площади трех многоугольников, если дано число их сторон и координаты их вершин.

Тема 4. Основы ООП: классы, конструкторы, перегрузка
Практическая работа 14 Рекурсия Простейшие рекурсивные алгоритмы

Задания этого раздела можно легко решить и *без использования рекурсии*. Данное обстоятельство связано с тем, что в заданиях рассматриваются *простейшие* примеры рекурсии, легко сводимые к итерационным алгоритмам. Более того, в некоторых случаях непосредственные вычисления по рекурсивным формулам оказываются весьма неэффективными (см., например, задания Recur4 и Recur6). Однако именно на подобных примерах проще всего получить первоначальные навыки разработки рекурсивных алгоритмов.

Recur1°. Описать рекурсивную функцию Fact(N) вещественного типа, вычисляющую значение *факториала*

$$N! = 1 \cdot 2 \cdot \ldots \cdot N$$

($N > 0$ — параметр целого типа). С помощью этой функции вычислить факториалы пяти данных чисел.

Recur2. Описать рекурсивную функцию Fact2(N) вещественного типа, вычисляющую значение *двойного факториала*

$$N!! = N \cdot (N-2) \cdot (N-4) \cdot \ldots$$

($N > 0$ — параметр целого типа; последний сомножитель в произведении равен 2, если N — четное число, и 1, если N — нечетное). С помощью этой функции вычислить двойные факториалы пяти данных чисел.

Recur3. Описать рекурсивную функцию PowerN(X, N) вещественного типа, находящую значение N-й степени числа X по формулам:

$$X^0 = 1,$$

$$X^N = (X^{N/2})^2 \text{ при } \textit{четных } N > 0, \qquad X^N = X \cdot X^{N-1} \text{ при } \textit{нечетных } N > 0,$$

$$X^N = 1/X^{-N} \text{ при } N < 0$$

($X \neq 0$ — вещественное число, N — целое; в формуле для четных N должна использоваться операция *целочисленного деления*). С помощью этой функции найти значения X^N для данного X при пяти данных значениях N.

Recur4. Описать рекурсивную функцию Fib1(N) целого типа, вычисляющую N-й элемент последовательности *чисел Фибоначчи* (N — целое число):

$$F_1 = F_2 = 1, \qquad F_K = F_{K-2} + F_{K-1}, \qquad K = 3, 4, \ldots .$$

С помощью этой функции найти пять чисел Фибоначчи с данными номерами, и вывести эти числа вместе с количеством рекурсивных вызовов

функции Fib1, потребовавшихся для их нахождения.

Recur5. Описать рекурсивную функцию Fib2(N) целого типа, вычисляющую N-й элемент последовательности *чисел Фибоначчи* (N — целое число):

$$F_1 = F_2 = 1, \qquad F_K = F_{K-2} + F_{K-1}, \qquad K = 3, 4, \ldots .$$

Считать, что номер N не превосходит 20. Для уменьшения количества рекурсивных вызовов по сравнению с функцией Fib1 (см. задание Recur4) создать вспомогательный массив для хранения *уже вычисленных* чисел Фибоначчи и обращаться к нему при выполнении функции Fib2. С помощью функции Fib2 найти пять чисел Фибоначчи с данными номерами.

Recur6. Описать рекурсивную функцию Combin1(N, K) целого типа, находящую $C(N, K)$ — *число сочетаний* из N элементов по K — с помощью рекуррентного соотношения:

$$C(N, 0) = C(N, N) = 1,$$
$$C(N, K) = C(N - 1, K) + C(N - 1, K - 1) \qquad \text{при } 0 < K < N.$$

Параметры функции — целые числа; $N > 0$, $0 \le K \le N$. Дано число N и пять различных значений K. Вывести числа $C(N, K)$ вместе с количеством рекурсивных вызовов функции Combin1, потребовавшихся для их нахождения.

Recur7. Описать рекурсивную функцию Combin2(N, K) целого типа, находящую $C(N, K)$ — *число сочетаний* из N элементов по K — с помощью рекуррентного соотношения:

$$C(N, 0) = C(N, N) = 1,$$
$$C(N, K) = C(N - 1, K) + C(N - 1, K - 1) \qquad \text{при } 0 < K < N.$$

Параметры функции — целые числа; $N > 0$, $0 \le K \le N$. Считать, что параметр N не превосходит 20. Для уменьшения количества рекурсивных вызовов по сравнению с функцией Combin1 (см. задание Recur6) описать вспомогательный двумерный массив для хранения *уже вычисленных* чисел $C(N, K)$ и обращаться к нему при выполнении функции Combin2. С помощью функции Combin2 найти числа $C(N, K)$ для данного значения N и пяти различных значений K.

Recur8. Описать рекурсивную функцию RootK(X, K, N) вещественного типа, находящую приближенное значение корня K-й степени из числа X по формуле:

$$Y_0 = 1, \qquad Y_{N+1} = Y_N - (Y_N - X/(Y_N)^{K-1})/K,$$

где Y_N обозначает RootK(X, K, N) при фиксированных X и K. Параметры функции: X (> 0) — вещественное число, K (> 1) и N (> 0) — целые.

С помощью функции RootK найти для данного числа X приближенные значения его корня K-й степени при шести данных значениях N.

Recur9. Описать рекурсивную функцию NOD(A, B) целого типа, находящую *наибольший общий делитель* (НОД) двух целых положительных чисел A и B, используя *алгоритм Евклида*:

$$НОД(A, B) = НОД(B, A \bmod B), \quad \text{если } B = 0; \qquad НОД(A, 0) = A.$$

С помощью этой функции найти НОД(A, B), НОД(A, C), НОД(A, D), если даны числа A, B, C, D.

Recur10°. Описать рекурсивную функцию DigitSum(K) целого типа, которая находит сумму цифр целого числа K, не используя оператор цикла. С помощью этой функции найти суммы цифр для пяти данных целых чисел.

Recur11. Описать рекурсивную функцию MaxElem(A, N) целого типа, которая находит максимальный элемент целочисленного массива A размера N ($1 \le N \le 10$), не используя оператор цикла. С помощью этой функции найти максимальные элементы массивов A, B, C размера N_A, N_B, N_C соответственно.

Recur12. Описать рекурсивную функцию DigitCount(S) целого типа, которая находит количество цифр в строке S, не используя оператор цикла. С помощью этой функции найти количество цифр в каждой из пяти данных строк.

Recur13. Описать рекурсивную функцию Palindrom(S) логического типа, возвращающую TRUE, если строка S является *палиндромом* (то есть читается одинаково слева направо и справа налево), и FALSE в противном случае. Оператор цикла в теле функции не использовать. Вывести значения функции Palindrom для пяти данных строк.

Разбор выражений

Во всех заданиях данного пункта предполагается, что исходные строки, определяющие выражения, не содержат пробелов. При выполнении заданий не следует использовать оператор цикла.

Recur14°. Вывести значение целочисленного выражения, заданного в виде строки S. Выражение определяется следующим образом:

<выражение> ::= <цифра> | <выражение> + <цифра> |
<выражение> — <цифра>

Recur15°. Вывести значение целочисленного выражения, заданного в виде строки *S*. Выражение определяется следующим образом:

<выражение> ::= <терм> | <выражение> + <терм> |
<выражение> — <терм>
<терм> ::= <цифра> | <терм> * <цифра>

Recur16°. Вывести значение целочисленного выражения, заданного в виде строки *S*. Выражение определяется следующим образом:

<выражение> ::= <терм> | <выражение> + <терм> |
<выражение> — <терм>
<терм> ::= <элемент> | <терм> * <элемент>
<элемент> ::= <цифра> | (<выражение>)

Recur17°. Вывести значение целочисленного выражения, заданного в виде строки *S*. Выражение определяется следующим образом:

<выражение> ::= <цифра> |
(<выражение><знак><выражение>)
<знак> ::= + | — | *

Recur18°. Проверить правильность выражения, заданного в виде непустой строки *S* (выражение определяется по тем же правилам, что и в задании Recur17). Если выражение составлено правильно, то вывести TRUE, иначе вывести FALSE.

Recur19. Проверить правильность выражения, заданного в виде непустой строки *S* (выражение определяется по тем же правилам, что и в задании Recur17). Если выражение составлено правильно, то вывести 0, в противном случае вывести номер первого ошибочного, лишнего или недостающего символа в строке *S*.

Recur20. Вывести значение целочисленного выражения, заданного в виде строки *S*. Выражение определяется следующим образом (функция M возвращает максимальный из своих параметров, а функция m — минимальный):

<выражение> ::= <цифра> | M(<выражение> , <выражение>) |
m(<выражение> , <выражение>)

Recur21. Вывести значение логического выражения, заданного в виде строки *S*. Выражение определяется следующим образом («T» — TRUE, «F» — FALSE):

<выражение> ::= T | F | And(<выражение> , <выражение>) |
Or(<выражение> , <выражение>)

Recur22. Вывести значение целочисленного выражения, заданного в виде строки *S*. Выражение определяется следующим образом (функция M возвращает максимальный из своих параметров, а функция m — минимальный):

<выражение> ::= <цифра> | M(<параметры>) | m(<параметры>)
<параметры> ::= <выражение> | <выражение> , <параметры>

Recur23. Вывести значение логического выражения, заданного в виде строки *S*. Выражение определяется следующим образом («T» — TRUE, «F» — FALSE):

<выражение> ::= T | F | And(<параметры>) | Or(<параметры>)
<параметры> ::= <выражение> | <выражение> , <параметры>

Recur24. Вывести значение логического выражения, заданного в виде строки *S*. Выражение определяется следующим образом («T» — TRUE, «F» — FALSE):

<выражение> ::= T | F | And(<параметры>) |
Or(<параметры>) | Not(<выражение>)
<параметры> ::= <выражение> | <выражение> , <параметры>

Перебор с возвратом

Recur25°. Дано дерево глубины *N*, каждая внутренняя вершина которого имеет *K* (< 10) непосредственных потомков (нумеруются от 1 до *K*). Корень дерева имеет номер 0. Записать в текстовый файл с данным именем все возможные пути, ведущие от корня к листьям. Перебирать пути, начиная с «самого левого» и заканчивая «самым правым» (при этом первыми заменять конечные элементы пути).

Recur26. Дано дерево глубины *N*, каждая внутренняя вершина которого имеет *K* (< 10) непосредственных потомков (нумеруются от 1 до *K*). Корень дерева имеет номер 0. Записать в текстовый файл с данным именем все пути, ведущие от корня к листьям и удовлетворяющие следующему условию: никакие соседние элементы пути не нумеруются одной и той же цифрой. Порядок перебора путей такой же, как в задании Recur25.

Recur27. Дано дерево глубины *N* (*N* — четное), каждая внутренняя вершина

которого имеет 2 непосредственных потомка: A с весом 1 и B с весом 1. Корень дерева C имеет вес 0. Записать в текстовый файл с данным именем все пути от корня к листьям, удовлетворяющие следующему условию: суммарный вес элементов пути равен 0. Порядок перебора путей такой же, как в задании Recur25.

Recur28. Дано дерево глубины N того же типа, что и в задании Recur27. Записать в текстовый файл с данным именем все пути от корня к листьям, удовлетворяющие следующему условию: суммарный вес элементов для любого начального отрезка пути неотрицателен. Порядок перебора путей такой же, как в задании Recur25.

Recur29. Дано дерево глубины N, каждая внутренняя вершина которого имеет 3 непосредственных потомка: A с весом 1, B с весом 0 и C с весом—1. Корень дерева D имеет вес 0. Записать в текстовый файл с данным именем все пути от корня к листьям, удовлетворяющие следующим условиям: суммарный вес элементов для любого начального отрезка пути неположителен, а суммарный вес всех элементов пути равен 0. Порядок перебора путей такой же, как в задании Recur25.

Recur30. Дано дерево глубины N того же типа, что и в задании Recur29. Записать в текстовый файл с данным именем все пути от корня к листьям, удовлетворяющие следующим условиям: никакие соседние элементы пути не обозначаются одной и той же буквой, а суммарный вес всех элементов пути равен 0. Порядок перебора путей такой же, как в задании Recur25.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждениевысшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по выполнению лабораторных работ
по дисциплине «Информационные технологии и программирование»
Для студентов направления подготовки 15.03.04 Автоматизация технологических процессов и производств,
направленность (профиль) Информационно-управляющие системы

Невинномысск, 2026

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 6
ТЕМА 1 ОСНОВЫ C#: СИНТАКСИС, ТИПЫ ДАННЫХ, КОНСОЛЬНЫЙ ВВОД-ВЫВОД
ЛАБОРАТОРНАЯ РАБОТА 1. СТРУКТУРА КОНСОЛЬНОГО ПРИЛОЖЕНИЯ В C# 8
ТЕМА 1 ОСНОВЫ C#: СИНТАКСИС, ТИПЫ ДАННЫХ, КОНСОЛЬНЫЙ ВВОД-ВЫВОД
ЛАБОРАТОРНАЯ РАБОТА 2. ПРЕДОПРЕДЕЛЕННЫЕ ТИПЫДАННЫХ, ПЕРЕМЕННЫЕ, КОНСТАНТЫ 17
ТЕМА 1 ОСНОВЫ C#: СИНТАКСИС, ТИПЫ ДАННЫХ, КОНСОЛЬНЫЙ ВВОД-ВЫВОД
ЛАБОРАТОРНАЯ РАБОТА 3. ИСПОЛЬЗОВАНИЕ ВОЗМОЖНОСТЕЙКОНСОЛЬНОГО ВВОДА-ВЫВОДА. 27
ТЕМА 1 ОСНОВЫ C#: СИНТАКСИС, ТИПЫ ДАННЫХ, КОНСОЛЬНЫЙ ВВОД-ВЫВОД
ЛАБОРАТОРНАЯ РАБОТА 4. УПРАВЛЕНИЕ ПОТОКОМ ВЫПОЛНЕНИЯ С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРОВ IF, SWITCH 34
ТЕМА 2. УПРАВЛЕНИЕ ПОТОКОМ: ЦИКЛЫ И ПОСЛЕДОВАТЕЛЬНОСТИ
ЛАБОРАТОРНАЯ РАБОТА 5. УПРАВЛЕНИЕ ПОТОКОМ ВЫПОЛНЕНИЯ С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРА ЦИКЛА FOR 42
ТЕМА 2. УПРАВЛЕНИЕ ПОТОКОМ: ЦИКЛЫ И ПОСЛЕДОВАТЕЛЬНОСТИ
ЛАБОРАТОРНАЯ РАБОТА 6. УПРАВЛЕНИЕ ПОТОКОМВЫПОЛНЕНИЯ С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРОВ WHILE 47
ТЕМА 2. УПРАВЛЕНИЕ ПОТОКОМ: ЦИКЛЫ И ПОСЛЕДОВАТЕЛЬНОСТИ
ЛАБОРАТОРНАЯ РАБОТА 7. УПРАВЛЕНИЕ ПОТОКОМ ВЫПОЛНЕНИЯ С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРА DO …WHILE 53
ТЕМА 3. РАБОТА С ДАННЫМИ: МАССИВЫ, СТРОКИ, РЕКУРСИЯ
ЛАБОРАТОРНАЯ РАБОТА 8. КЛАССЫ. СТРУКТУРЫ. 57
ТЕМА 4. ОСНОВЫ ООП: КЛАССЫ, КОНСТРУКТОРЫ, ПЕРЕГРУЗКА
ЛАБОРАТОРНАЯ РАБОТА 9. КОНСТРУКТОР КЛАССА. ПЕРЕГРУЗКА КОНСТРУКТОРОВ КЛАССА. 71
ТЕМА 4. ОСНОВЫ ООП: КЛАССЫ, КОНСТРУКТОРЫ, ПЕРЕГРУЗКА
ЛАБОРАТОРНАЯ РАБОТА 10. МНОГОМОДУЛЬНЫЕ ПРИЛОЖЕНИЯ.....74
ТЕМА 4. ОСНОВЫ ООП: КЛАССЫ, КОНСТРУКТОРЫ, ПЕРЕГРУЗКА
ЛАБОРАТОРНАЯ РАБОТА 11. ОПЕРАЦИИ КЛАССОВ. ПЕРЕГРУЗКА ОПЕРАЦИЙ. 87
ТЕМА 4. ОСНОВЫ ООП: КЛАССЫ, КОНСТРУКТОРЫ, ПЕРЕГРУЗКА
ЛАБОРАТОРНАЯ РАБОТА 12. ПОСТРОЕНИЕ ИЕРАРХИИ КЛАССОВ.....93
ТЕМА 5. ПРОДВИНУТОЕ ООП: МОДУЛЬНОСТЬ И ИНТЕРФЕЙСЫ
ЛАБОРАТОРНАЯ РАБОТА 13. РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ.....107
ТЕМА 5. ПРОДВИНУТОЕ ООП: МОДУЛЬНОСТЬ И ИНТЕРФЕЙСЫ
ЛАБОРАТОРНАЯ РАБОТА 14. ФАЙЛОВЫЙ ВВОД-ВЫВОД. РАБОТА С КАТАЛОГАМИ. РАБОТА С ФАЙЛАМИ. 120
ТЕМА 5. ПРОДВИНУТОЕ ООП: МОДУЛЬНОСТЬ И ИНТЕРФЕЙСЫ
ЛАБОРАТОРНАЯ РАБОТА 16. СОЗДАНИЕ ПРИЛОЖЕНИЯ ПО ТЕХНОЛОГИИ WINDOWS FORMS.....151
ТЕМА 5. ПРОДВИНУТОЕ ООП: МОДУЛЬНОСТЬ И ИНТЕРФЕЙСЫ
ЛАБОРАТОРНАЯ РАБОТА 17. ПРИМЕНЕНИЕ ЭЛЕМЕНТОВУПРАВЛЕНИЯ В ПРИЛОЖЕНИЯХ WINDOWS.....161
ЗАКЛЮЧЕНИЕ.....175
СПИСОК ЛИТЕРАТУРЫ 176

ВВЕДЕНИЕ

В настоящее время особое значение для всех отраслей экономики принимает автоматизация производственных процессов. Для разработки приложений масштаба предприятия необходимо владеть современными методами создания сложных программных продуктов, а также инструментарием разработки программных средств.

Особое значение при подготовке специалиста в сфере информационных систем и технологий имеет практическая подготовка специалиста: производственные задачи автоматизации связаны с многопоточным программированием, использованием баз данных, применением сложных структур данных.

Сложность освоения технологий программирования, в совокупности с постоянным ростом значимости информационных технологий, указывает на необходимость подготовки специалистов в области программирования.

Учебное пособие (лабораторный практикум) направлено на развитие следующих профессиональных компетенций:

– способность к проектированию базовых и прикладных информационных технологий (ПК-11);

– способность разрабатывать средства реализации информационных технологий (методические, информационные, математические, алгоритмические, технические и программные) (ПК-12).

Основная цель изучения дисциплины «Основы алгоритмизации и программирования»: изучение и получение практических навыков в области современных средств и методов разработки, алгоритмов, основных синтаксических конструкция языка высокого уровня.

Задачами изучения дисциплины «Основы алгоритмизации и программирования» являются:

– усвоение теоретических знаний о структуре и принципах построения современных программных комплексов, а также об архитектуре современных

программных платформ для автоматизации проектирования информационных систем;

– получение практических навыков разработки приложений с использованием языка высокого уровня C#;

– изучение основных алгоритмических приемов решения задач с использованием C#;

– освоение принципов разработки программного обеспечения на основе технологий Microsoft .NET Framework;

– освоение принципов использования перспективных инструментальных средств разработки и проектирования приложений.

Знания, полученные студентами в результате выполнения заданий лабораторного практикума, позволит студентам применять полученные знания при разработке программных средств, как для образовательных целей, так и в профессиональной деятельности.

ТЕМА 1 ОСНОВЫ C#: СИНТАКСИС, ТИПЫ ДАННЫХ, КОНСОЛЬНЫЙ ВВОД-ВЫВОД

ЛАБОРАТОРНАЯ РАБОТА 1. СТРУКТУРА КОНСОЛЬНОГО ПРИЛОЖЕНИЯ В C#

1. Цель и содержание

Цель лабораторной работы: научиться работать с переменными и константами простых типов в C#.

Задачи лабораторной работы:

– научиться объявлять переменные простых типов в языке C#;

– научиться объявлять константы простых типов в языке C#;

– научиться выполнять простейшие действия с переменными и константами.

2. Формируемые компетенции

Лабораторная работа направлена на формирование следующих компетенций:

– способность к проектированию базовых и прикладных информационных технологий (ПК-11);

– способность разрабатывать средства реализации информационных технологий (методические, информационные, математические, алгоритмические, технические и программные) (ПК-12).

3. Теоретическая часть

Рассмотрим структуру консольного приложения на языке C#, созданного с использованием средств MS Visual Studio (рис. 1.1).

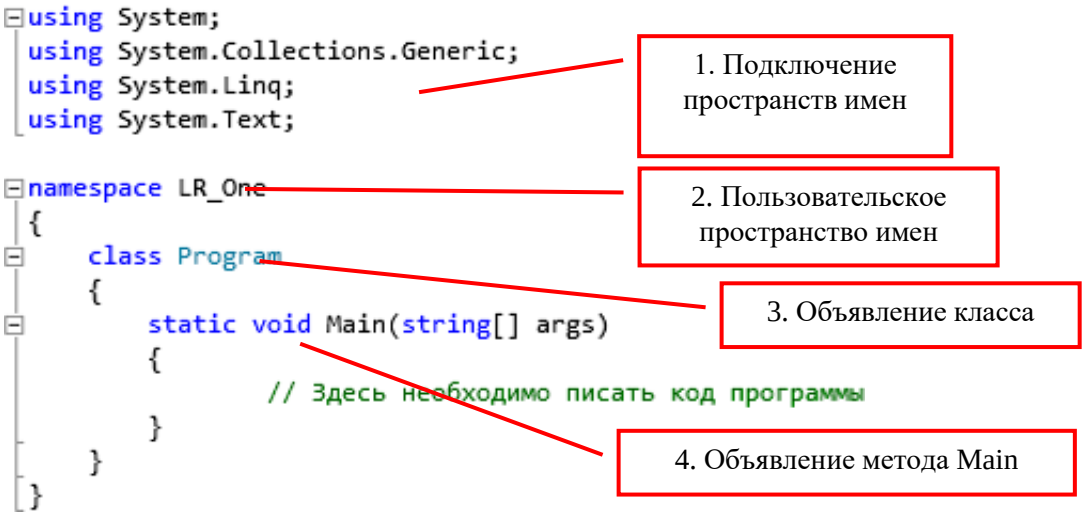


Рисунок 1.1 – Исходный код консольного приложения.

В интерактивном редакторе кода среды разработки код может быть свернут/ развернут с использованием кнопок +/-, внедренных в код. На рис. 1.1 в области 1 показано подключение пространств имен (библиотек, содержащих стандартные инструменты) с использованием зарезервированного слова using.

Программист сам создает свое пространство имен с именем LR_One (2), в котором объявляется класс с именем Program.

В классе Program объявлен один метод – функция Main (параметры функции не рассматриваем).

Функция Main имеет особенное значение в программировании на языках C, C++ и C#.

Функцию Main называют «точкой входа», то есть началом выполнения программы. Далее мы рассмотрим приложения, содержащие множество функций. Операционная система знает с какой именно функции начать выполнение программы – с функции Main. Очевидно, что имя этой функции менять нельзя. Это должен быть статический метод класса (или структуры), возвращающий либо значение типа int, либо void. Хотя нередко модификатор public указывается явно, поскольку по определению этот метод должен быть вызван извне программы, на самом деле неважно, какой уровень доступа вы назначите методу точки входа. Он запустится, даже если вы пометите его как private.

Когда компилируется консольное или Windows-приложение C#, по умолчанию компилятор ищет в точности один метод Main () с описанной выше сигнатурой в любом классе и делает его точкой входа программы. Если существует более одного метода Main ( ), компилятор возвращает сообщение об ошибке.

Обратите внимание на вложенность конструкций на рис. 1.1: в пространство имен LR_One вложен класс Program, в класс Program вложена функция Main. В свою очередь, в функции Main содержатся инструкции на языке C#-код, который начнет выполняться при старте программы.

В C#, как и в других C-подобных языках, большинство операторов завершаются точкой с запятой (;) и могут продолжаться в нескольких строках без необходимости указания знака переноса. Операторы могут быть объединены в блоки с помощью фигурных скобок ({ }). Однострочные комментарии начинаются с двойного слеша (//), а многострочные – сослеша со звездочкой (/*) и заканчиваются противоположной комбинацией (*/). В этих аспектах язык C# идентичен C++ и Java.

Следует также помнить о том, что язык C# чувствителен к регистру символов. Это значит, что переменные с именами myVar и MyVar являются разными.

Причина присутствия оператора using в файле Program.cs связана с использованием библиотечных классов.

Весь код C# должен содержаться внутри класса. Объявление класса состоит из ключевого слова class, за которым следует имя класса и пара фигурных скобок. Весь код, ассоциированный с этим классом, размещается между этими скобками.

4. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный

(x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

5. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

6. Методика и порядок выполнения работы

1. Создайте консольное приложение, для этого выполните следующие действия:

1.1 Выберите команду главного меню $File \rightarrow New \rightarrow Project...$

1.2 В открывшемся диалоговом окне (рис. 1.2) выберите необходимые настройки для создаваемого проекта: язык Visual C#; фреймворк: .NET Framework 4; шаблон: Console Application.

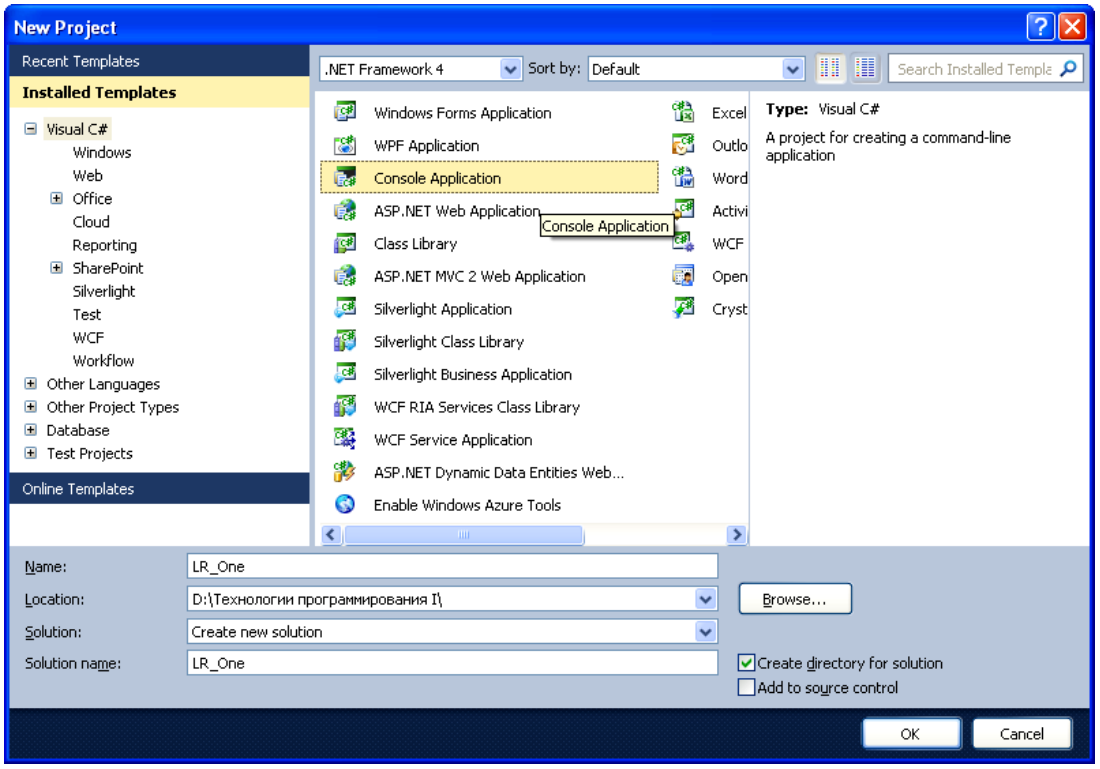


Рисунок 1.2 – Создание нового проекта консольного приложения.

1.3 В текстовом поле Name введите имя проекта (например LR_One).

1.4 В текстовом поле Location выберите место сохранения нового проекта.

1.5 Установите флажок-переключатель «Create directory for solution».

1.6 Нажмите кнопку «OK».

2. После выполнения пункта 1 в среде разработки откроется новый созданный проект (рис. 1.3).

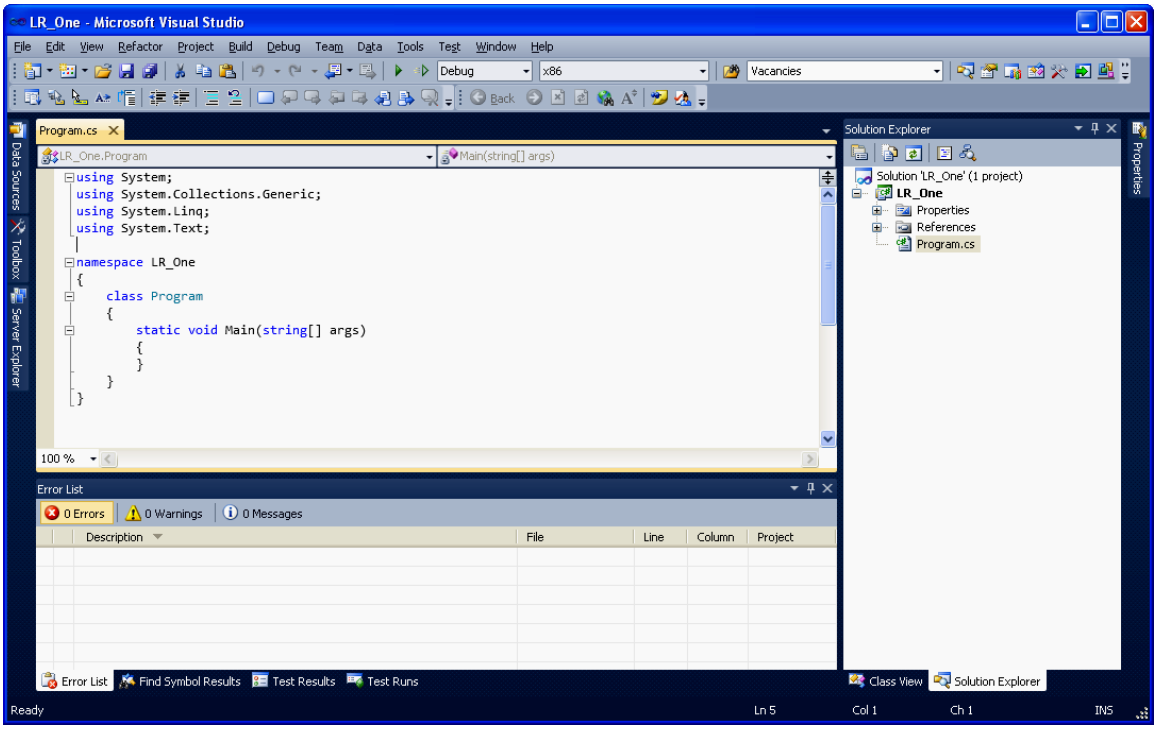


Рисунок 1.3 – Новый проект, загруженный в среду разработки: вкладка «Solution Explorer» отображает состав проекта (нас интересует только файл Program.cs); в левой части окна (сверху) открыт файл Program.cs в редакторе кода.

3. На данном этапе необходимо ознакомиться со структурой исходного файла консольного приложения (рис. 1.4).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR_One
{
    class Program
    {
        static void Main(string[] args)
        {
                // Здесь необходимо писать код программы
        }
    }
}
```

Рисунок 1.4 – Исходный файл консольного приложения.

4. Весь код программы необходимо писать внутри функции Main.

5. Для построения сборки (исполняемого exe-файла) выполните команду главного меню *Build → Build Solution* (или использовать горячую клавишу *F6*). После этого сборка создана, но приложение не будет запущено автоматически.

6. Для создания сборки и последующего запуска программы можно воспользоваться командой *Debug → Start Debugging* главного меню среды разработки или нажать кнопку панели инструментов *Start Debugging* (*F5*). Можно также использовать горячую клавишу *F5*.

7. Запустите приложение на выполнение одним из методов, указанных в пункте 6. Окно консольного приложения появится и исчезнет. Это означает, что приложение выполнило все команды, написанные программистом, и завершило свою работу.

8. Для удержания окна на экране измените исходный файл в соответствии с рисунком 1.5. В функции Main добавлен вызов только одной команды Console.DeadKey ( ) – эта функция останавливает выполнение программы и ждет, когда пользователь нажмет любую клавишу.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR_One
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.ReadKey();
        }
    }
}
```

Рисунок 1.5 – Исходный файл консольного приложения для предотвращения закрытия окна консольного приложения.

9. Запустите измененное приложение, убедитесь, что окно удерживается на экране.

10. Добавьте несколько строк кода в исходный файл (рис. 1.6).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR_One
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Лабораторная работа №1");
            Console.WriteLine("");
            Console.WriteLine("Выполнил: Иванов Иван иванович");
            Console.WriteLine("Группа: ИСТБ-121");
            Console.WriteLine("Наименование ЛР: Структура консольного приложения");
            Console.WriteLine("");
            Console.WriteLine("для завершения работы программы нажмите любую клавишу...");

            Console.ReadKey();
        }
    }
}
```

Рисунок 1.6 – Исходный файл консольного приложения для вывода информации на экран.

11. Внимательно изучите исходный код примера на рис. 1.5. Запустите приложение и убедитесь, что отсутствуют ошибки и информация выводится.

12. Выполните индивидуальное задание.

Индивидуальное задание.

Измените приложение, созданное в ходе выполнения данной лабораторной работы таким образом, чтобы программа выводила на экран следующую информацию (каждый студент должен использовать персональную информацию о себе):

– Название и номер лабораторной работы;

– ФИО студента;

– Группа студента и шифр специальности;

– Дата рождения студента;

– Населенный пункт постоянного места жительства студента;

– Любимый предмет в школе;

– Краткое описание увлечений, хобби, интересов.

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Какая функция имеет особенное значение при выполнении программы на языках C, C++, C#?

2. Что такое «точка входа» в программе?

3. Как вы понимаете термины «пространства имен», «класс», «метод», «функция»?

4. Чем отличается структура консольного приложения от приложения, построенного с использованием технологий Windows Forms, WPF?

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [1-3].

ТЕМА 1 ОСНОВЫ C#: СИНТАКСИС, ТИПЫ ДАННЫХ, КОНСОЛЬНЫЙ ВВОД-ВЫВОД
ЛАБОРАТОРНАЯ РАБОТА 2. ПРЕДОПРЕДЕЛЕННЫЕ ТИПЫДАННЫХ, ПЕРЕМЕННЫЕ, КОНСТАНТЫ

1. Цель и содержание

Цель лабораторной работы: научиться работать с переменными и константами простых типов в C#.

Задачи лабораторной работы:

– научиться объявлять переменные простых типов в языке C#;

– научиться объявлять константы простых типов в языке C#;

– научиться выполнять простейшие действия с переменными и константами.

2. Теоретическая часть

Перед выполнением лабораторной работы необходимо изучить материалы лекций. Следует понимать принцип деления типов .NET на типы значений и ссылочные типы.

В данной лабораторной работе необходимо освоить приемы работы с предопределенными типами значений.

2.1 Типы значений C#

Язык C# поддерживает 8 предопределенных целочисленных типов (таблица 2.1).

Таблица 2.1 – Целочисленные типы C#.

Имя типа	Тип CTS	Описание	Диапазон (минимум : максимум)
sbyte	System.SByte	8-битное целое со знаком	-128 : 127
short	System.Int16	16-битное целое со знаком	-32 768 : 32 767
int	System.Int32	32-битное целое со знаком	-2 147 483 648 : 2 147 483 647

long	System.Int64	64-битное целое со знаком	$-2^{63} : 2^{63}-1$
byte	System.Byte	8-битное целое без знака	0 : 255
ushort	System.UInt16	16-битное целое без знака	0 : 65 535
uint	System.UInt32	32-битное целое без знака	$0 : 2^{32}-1$
ulong	System.UInt64	64-битное целое без знака	$0 : 2^{64}-1$

Язык C# также поддерживает и типы с плавающей точкой (таблица 2.2).

Таблица 2.2 – Типы с плавающей точкой C#.

Имя типа	Тип CTS	Описание	Кол-во знаков	Диапазон (минимум : максимум)
float	System.Single	32-битное с плавающей точкой одинарной точности	7	от $\pm 1.5\times10^{-45}$ до $\pm 3.4\times10^{38}$
double	System.Double	64-битное с плавающей точкой двойной точности	15/16	от $\pm 5.0\times10^{-324}$ до $\pm 1.7\times10^{308}$

В таблице 2.3 представлен десятичный тип C#. Данный тип реализован для финансовых операций.

Таблица 2.3 – Десятичный тип C#.

Имя типа	Тип CTS	Описание	Кол-во знаков	Диапазон (минимум : максимум)
decimal	System.Decimal	128-битное с плавающей точкой в десятичной нотации с высокой точностью	28	от $\pm 1.0\times10^{-28}$ до $\pm 7.9\times10^{28}$

Как и во многих языках программирования существует булевский тип (таблица 2.4).

Таблица 2.4 – Булевский тип.

Имя типа	Тип CTS	Значения
bool	System.Boolean	true или false

Для хранения одиночных символов в языке C# используется тип char (таблица 2.5)

Таблица 2.5 – Булевский тип.

Имя типа	Тип CTS	Значения
char	System.Char	Представляет отдельный 16-битный (Unicode) символ

Литералы типа char записываются как одиночные, заключенные в одинарные кавычки символы: ‘F’, ‘w’, ‘ц’, ‘Я’ и т.д.

В переменных типа char можно хранить и специальные символы в виде управляющих последовательностей (таблица 2.6).

Таблица 2.6 – Представление символов в виде управляющих последовательностей.

Управляющая последовательность	Символ
\’	Одиночная кавычка
\”	Двойная кавычка

\\	Обратный слэш
\0	Пусто
\a	Предупреждение (звуковой сигнал)
\b	Забой
\f	Подача формы
\n	Новая строка
\r	Возврат каретки
\t	Символ табуляции
\v	Вертикальная табуляция

Если отдельные символы объединены в строку, то необходимо использовать тип string, который отображается на тип CTS – System.String.

2.2 Объявление и инициализация переменных в C#

Синтаксис объявления переменных в C# выглядит следующим образом:

ТипДанных ИдентификаторПеременной;

Например:

```
int a;
```

Этот код объявляет переменную типа int с именем a. Компилятор не позволит использовать эту переменную до тех пор, пока она не будет инициализирована (т.е. пока ей не будет присвоено значение).

Для инициализации переменной a необходимо написать следующий код:

```
a = 123;
```

Переменную можно инициализировать во время объявления:

```
int b = 7;
```

или

```
string str = "Hello, World!!!";
```

Синтаксис C# позволяет объявить несколько переменных (и инициализировать их) одного типа в одной синтаксической конструкции.

Например:

```
float b, i, myPerem, U_t = 0.3F, z_11 = 23.56F;
```

В данном примере объявляется 5 переменных типа float, некоторые из них инициализируются в процессе объявления.

2.3 Объявление и инициализация констант в C#

Константа – это переменная, значение которой не меняется за время выполнения программы. Для объявления константы необходимо воспользоваться ключевым словом const. Например:

```
const char simv = 'A';
const double pi = 3.14;
```

Очевидно, что при таком объявлении, поменять значения simv и pi в дальнейшем будет нельзя.

3. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место

пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмом описанном в лабораторной работе №1.

2. Изучите материал в разделе «Теоретическое обоснование» данной лабораторной работы.

3. Модифицируйте исходный файл как показано на рис. 2.1.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5
6  namespace LR_One
7  {
8      class Program
9      {
10         static void Main(string[] args)
11         {
12             int a;
13             int b = 7;
14
15             string str = "Hello, World!!!";
16
17             a = 123;
18
19             Console.ReadKey();
20         }
21     }
22 }
```

Рисунок 2.1 – Объявление переменных в C#.

4. Рассмотрим приведенный пример подробнее:

4.1. В строке 12 объявляется переменная типа int с именем a.

4.2. В строке 13 объявляется переменная типа int с именем b, причем при объявлении для нее устанавливается начальное значение, равное 7.

4.3. В строке 15 объявляется переменная типа string с именем str и инициализируется строковым значением «Hello, World!!!».

4.4. В строке 17 переменной a присваивается целочисленное значение 123.

5. Запустите приложение на выполнение. У вас должно появиться пустое окно консольного приложения. Очевидно, что исходный код, представленный на рис. 2.1 не предполагает вывода какой-либо информации на экран.

6. Для вывода информации на экран воспользуемся функцией Console.WriteLine, изученной в лабораторной работе 1. Добавим в исходный файл следующие строки (строки 19 – 21 на рис. 2.2).

```
6  namespace LR_One
7  {
8      class Program
9      {
10         static void Main(string[] args)
11         {
12             int a;
13             int b = 7;
14
15             string str = "Hello, World!!!";
16
17             a = 123;
18
19             Console.WriteLine(a);
20             Console.WriteLine(b);
21             Console.WriteLine(str);
22
23             Console.ReadKey();
24         }
25     }
26 }
27
```

Рисунок 2.2 – Добавление строк кода для вывода значений объявленных переменных на экран.

7. В примере на рис. 2.2 используется простой вывод, то есть имя переменной просто передается в качестве параметра функции Console.WriteLine.

8. Для выполнения форматного вывода (изменения формата представления выводимой информации) необходимо реализовать вывод в следующем виде (рис. 2.3 изменены строки 19-21):

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR_One
{
    class Program
    {
        static void Main(string[] args)
        {
            int a;
            int b = 7;

            string str = "Hello, World!!!";

            a = 123;

            Console.WriteLine("Значение переменной a равно {0}", a);
            Console.WriteLine("a значение переменной b равно {0}", b);
            Console.WriteLine("значение a+b = {0}+{1} = {2}", a, b, a+b);
            Console.WriteLine(str);

            Console.ReadKey();
        }
    }
}
```

Рисунок 2.3 – Вывод информации с использованием форматных строк.

9. Внимательно изучите код полученной программы, Затем выполните индивидуальное задание.

Индивидуальное задание.

Объявите требуемые переменные, присвойте им начальные значения (определите самостоятельно, значения какого типа могут принимать переменные), выведите на экран с использованием форматной строки значения переменных и результат вычисления выражения в соответствие с вариантом:

Вариант	Выражение для вычисления
1	$F = a_1 + b - a \cdot (x + y5)$
2	$Se = w111 + bt - x + y \cdot w$
3	$R_x = a \cdot b + b / t - x + f \cdot i_2$
4	$c = gh + b \cdot q3 - x + y / w$
5	$H = \frac{g \cdot h}{d17} + b / h1 - \frac{x + y}{4}$
6	$Z = \frac{35}{f} + y \cdot f - \frac{f + y}{4}$

7	$a = \frac{35}{a} \cdot z + z \cdot a - \frac{a + Et}{4}$
8	$A0 = \frac{35}{G_1} \cdot Zvcw + G_1 \cdot a - \frac{a + Zvcw}{a}$
9	$Se = w111 + bt - x + y \cdot w$
10	$R_x = a \cdot b + b / t - x + f \cdot i_2$
11	$g = w \cdot h + b \cdot q3 - q3 + y / w$
12	$ee = \frac{g \cdot h}{d17} + d17 / h - \frac{g + d17 + h}{4}$
13	$Zze = \frac{35}{f} + y \cdot f - \frac{f + y}{4}$
14	$t = \frac{35}{a} \cdot z + z \cdot a - \frac{a + Et}{4}$
15	$A0 = \frac{35}{G_1} \cdot Zvcw + G_1 \cdot a - \frac{a + Zvcw}{a}$
16	$Zxy = a_5 + b - a_5 \cdot (b + y)$
17	$ch = \frac{6}{f1} \cdot z + z \cdot f1 - \frac{f1 + Et}{6}$
18	$ch = rx \cdot z + z \cdot f1 - \frac{f1 + Et}{rx}$
19	$_H = k1 + k2 - k3 + \frac{k1 \cdot k2}{k3}$
20	$Fy = a_1 + b - a_1 \cdot (x + y)$
21	$_G_1 = y1 + y2 + y3 - \frac{y1 + y2 + y3}{3}$
22	$R = 2 \cdot x + 2 \cdot y - 4 \cdot x \cdot y + z$
23	$g11 = t \cdot z + z \cdot f1 - \frac{f1 + U}{t}$
24	$Y = Z \cdot (z + x) + z / Z - \frac{z + Z}{x + X}$
25	$U = rx \cdot z + z \cdot Y - \frac{Y + z}{rx}$

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Что такое переменная? Как объявляется переменная?

2. Как объявляется константа? Чем константа отличается от переменной?

3. Какие типы значений применяются C#?

4. Чем тип char отличается от типа string?

5. Как производится инициализация переменных? Как производится инициализация констант?

6. Что такое управляющие последовательности?

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [1], [7].

ТЕМА 1 ОСНОВЫ C#: СИНТАКСИС, ТИПЫ ДАННЫХ, КОНСОЛЬНЫЙ ВВОД-ВЫВОД

ЛАБОРАТОРНАЯ РАБОТА 3. ИСПОЛЬЗОВАНИЕ ВОЗМОЖНОСТЕЙ КОНСОЛЬНОГО ВВОДА-ВЫВОДА.

1. Цель и содержание

Цель лабораторной работы: изучить команды ввода и вывода данных в консольном приложении.

Задачи лабораторной работы:

– научиться применять функции Console.Write и Console.WriteLine;

– научиться применять функции Console.Read и Console.ReadLine;

– научиться использовать форматы вывода для различных типов данных.

2. Теоретическая часть

В предыдущих лабораторных работах уже использовалась команда Console.WriteLine. Рассмотрим ее синтаксис подробнее:

```
Console.WriteLine("Строка текста");
```

Эта запись означает вызов статического метода класса Console. В качестве параметра функции передается строка, которая выводится в консоль.

Существует также функция:

```
Console.Write("Строка текста");
```

Этот метод также осуществляет вывод, только не переводит каретку на следующую строку.

Рассмотренные методы также позволяют осуществлять форматный вывод, аналогичный функции printf языка Си, например:

```
int a = 100, b = 13, c = 23, d = 0, e = 0;
Console.WriteLine("Переменная a = {0}, переменная b = {1}," +
"переменная c = {2}, " +
"переменная d = {3}, " +
"переменная e = {4}", a, b, c, d, e);
```

В данном примере в качестве первого параметра передается строка, содержащая маркеры в фигурных скобках. При выводе на места маркеров будут подставлены параметры, которые следуют за строкой.

При форматном выводе можно задавать ширину поля вывода, для этого необходимо воспользоваться следующим приемом:

```
int a = 100, b = 13, c = 23, d = 0, e = 0;
Console.WriteLine("Переменная a = {0, 5}, переменная b = {1, 5}," +
"переменная c = {2, 5}, " +
"переменная d = {3, 5}, " +
"переменная e = {4, 5}", a, b, c, d, e);
```

В данном примере видно, чтобы задать ширину вывода необходимо использовать в формате {n, w}, где n – порядковый номер параметра, а w – ширина области вывода (знак w позволяет осуществлять выравнивание выводимого значения по левому или правому краю).

При выводе также можно добавлять строку формата вместе с необязательным значением точности (таблица 2.1).

Таблица 3.1 – Основные строки формата.

Стока	Описание
C	Локальный формат валюты
D	Десятичный формат. Преобразует целое к основанию 10 и снабжает ведущими нулями, если есть спецификатор точности.
E	Научный (экспоненциальный) формат. Спецификатор точности устанавливает количество десятичных

	разрядов (по умолчанию 6).
F	Формат с фиксированной запятой. Спецификатор точности задает количество десятичных разрядов.
G	Общий формат. Форматирование E или F.
N	Числовой формат. Форматирует число с разделителями тысяч – запятыми.
P	Процентный формат
X	Шестнадцатиричный формат. Спецификатор точности используется для указания ведущих нулей.

Пример использования формата:

```
float a, b, c, res;

Console.Write("Введите a > ");
a = Convert.ToSingle(Console.ReadLine());

Console.Write("Введите b > ");
b = Convert.ToSingle(Console.ReadLine());

Console.Write("Введите c > ");
c = Convert.ToSingle(Console.ReadLine());

res = (a + b) * c;

Console.WriteLine("\n\n");

Console.WriteLine("Денежный формат {0:C} \n", res);
// Console.WriteLine("Десятичный формат {0:D10} \n", res);
Console.WriteLine("Экспоненциальный формат {0:E} \n", res);
Console.WriteLine("Формат с фиксированной запятой {0:F3, 7} \n", res);
Console.WriteLine("Общий формат {0:G} \n", res);
Console.WriteLine("Числовой формат {0:N} \n", res);
Console.WriteLine("Процентный формат {0:P} \n", res);
// Console.WriteLine("Шестнадцатиричный формат {0:X} \n", res);
```

Изучите представленный пример. Выполните данный код в среде выполнения.

В классе Console также существуют методы, позволяющие считывать информацию, вводимую пользователем:

```
string str;

str = Console.ReadLine();

Console.WriteLine("вы ввели строку: " + str);
```

В данном примере строка, введенная пользователем, считывается в переменную str, затее осуществляется ее вывод. Наберите данный пример в среде VS и изучите код.

3. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе №1.
2. Выполните индивидуальное задание, выполнив вывод результата с использованием всех возможных форматов, представленных в таблице 2.1.

Индивидуальное задание.

Объявите требуемые переменные, значения переменным пользователь должен присваивать в процессе выполнения программы (считываются с консоли), выведите на экран с использованием форматной строки значения переменных и результат вычисления выражения во всех возможных форматах:

Вариант	Выражение для вычисления
1.	$F = a_1 + \sin(b) - a \cdot (\lvert x \rvert + y5)$
2.	$Se = \frac{tg(w111) + bt}{x} - \frac{x}{bt} + y \cdot w$
3.	$R_x = \frac{a \cdot b}{\sin(f)} + b/t - x + f \cdot i_2$
4.	$c = \cos(gh) + b \cdot \sqrt{q3 - \lvert x - y \rvert / w}$
5.	$H = \frac{\sqrt{g \cdot h}}{d17} + \cos(b)/h1 - \frac{x + y}{4}$
6.	$Z = \frac{35}{\lvert f \rvert} + \cos(y \cdot f) - \frac{f + y}{4}$
7.	$a_q = \frac{35}{a} \cdot z + z \cdot a - \frac{a + Et}{4}$
8.	$A0 = \frac{35}{G_1} \cdot \cos(Zvcw) + G_1 \cdot a - \frac{a + \sin(Zvcw)}{a}$
9.	$Se = \sqrt{w111 + \sin(bt)} - x + y \cdot w$
10.	$R_x = a \cdot b + \sqrt{\frac{b}{t} - x + f \cdot \lvert i_2 \rvert}$
11.	$g = \sqrt{w \cdot h + \lvert b \rvert} \cdot q3 - \cos(q3) + y/w$
12.	$ee = \frac{g \cdot h}{d17} + d17/h - \sqrt{\frac{g + d17 + h}{4}}$
13.	$Zze = \sqrt{\frac{35}{f}} + \cos(y \cdot f) - \frac{f + y}{4}$

14.	$t = \frac{35}{a} \cdot z + \sin(z \cdot a) - \frac{a + \sqrt{Et}}{4}$
15.	$A0 = \frac{35}{G_1} \cdot \sqrt{Zvcw} + G_1 \cdot a - \left\lvert \frac{a - Zvcw}{a} \right\rvert$
16.	$Zxy = \sqrt{\frac{a_5}{b}} + b - a_5 \cdot (\sin(b) + y)$
17.	$ch = \frac{6}{f1} \cdot \sqrt{z} + z \cdot \sin(f1) - \frac{f1 + Et}{6}$
18.	$ch = \cos(rx) \cdot z + \left\lvert z \cdot f1 - \frac{f1 + Et}{rx} \right\rvert$
19.	$_H = k1 + \sin(k2) - k3 + \sqrt{\frac{k1 \cdot k2}{k3}}$
20.	$Fy = a_1 + \lvert b - \sin(a_1) \cdot (x + y) \rvert$
21.	$_G_1 = y1 + \sqrt{y2} + \left\lvert y3 - \frac{y1 + y2 + y3}{3} \right\rvert$
22.	$R = \sqrt{2 \cdot \sin(x)} + \lvert 2 \cdot y - 4 \cdot x \cdot y \rvert + z$
23.	$g11 = t \cdot z + \sqrt{z} \cdot \left\lvert f1 - \frac{f1 + U}{t} \right\rvert$
24.	$Y = Z \cdot \sin(z + x) + \left\lvert z / Z - \frac{\sin(z) + Z}{x + X} \right\rvert$
25.	$U = \sqrt{\lvert rx - z \rvert} + z \cdot \left\lvert Y - \frac{Y + z}{rx} \right\rvert$

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Как оформляется комментарий?
2. Какие функции используются для вывода информации в консоль?
3. Какие функции используются для считывания информации с консоли?
4. Какие форматы вывода вы знаете? Опишите их.
5. Какой класс содержит статические методы для конвертирования значений и приведения их к требуемому типу?
6. Какой класс содержит статические методы – математические функции? Что такое управляющие последовательности?

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [4], [7], [8].

ТЕМА 1 ОСНОВЫ C#: СИНТАКСИС, ТИПЫ ДАННЫХ, КОНСОЛЬНЫЙ ВВОД-ВЫВОД

ЛАБОРАТОРНАЯ РАБОТА 4. УПРАВЛЕНИЕ ПОТОКОМ ВЫПОЛНЕНИЯ С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРОВ IF, SWITCH

1. Цель и содержание

Цель лабораторной работы: изучить операторы, позволяющие организовывать непоследовательное выполнение программного кода.

Задачи лабораторной работы:

– научиться применять условныq оператор if;

– научиться применять оператор цикла for;

– научиться применять операторы выбора switch.

2. Теоретическая часть

2.1 Условный оператор.

В предыдущих лабораторных работах были рассмотрены вопросы программирования на языке C# с использованием только последовательного выполнения операторов программы.

В языке C# (как и в большинстве языков) можно вводить условия, циклы, ветвления.

Условный оператор в C# позволяет организовать условие типа «если …, то …, иначе …».

Синтаксис условного оператора:

if (*условие*)
    оператор
[else if (*условие*)
    оператор
...
else
    оператор]

В данном синтаксисе *оператор* может быть составным оператором (группа операторов языка, заключенные в фигурные скобки). Условный оператор содержит только одно обязательное зарезервированное слово – if. Все остальные конструкции не являются обязательными (в синтаксическом описании на это указывают скобки [ ]).

Пример использования условного оператора:

```
int a = 100, b = 13, c = 23, d = 0, e = 0;


if (a == 100) // Если а равно 100
{
    // Увеличиваем b на 1000
    b += 1000;
}
else if (a < 100) // иначе если а меньше 100
{
    // изменяем значение d
    d = b + c;
    // и изменяем значение e
    e = a + b;
}
else // иначе (остальные варианты не предусмотренные первыми двумя условиями)
    // edtkbxbdftv a на 1
    a++;

// продолжается последовательное выполнение программы
Console.WriteLine("{0}", a);
```

В представленном примере сначала проверяется является ли значение переменной a равным 100.

1. Если «Да», то выполняется единственный оператор (увеличение b на 1000). На этом выполнение всего условного оператора завершено и программа переходит к выполнению операторов, следующих за ним, то есть к выводу значения a.

2. Если первое условие не выполнилось, проверяется второе – является ли a меньше 100 (конструкция else if). Если это утверждение истинно, то выполняется составной оператор, состоящий из двух – изменение значений переменных d и e. Затем осуществляется переход к выводу переменной a.

3. Если не выполнилось ни одно условие с оператором if, то выполняется оператор, указанный после зарезервированного слова else. В данном случае это несоставной оператор (a увеличивается на 1), поэтому фигурные скобки можно не писать.

Во всей этой конструкции только оператор if является обязательным. Конструкций else if может быть любое количество, а if и else – только по одному.

Еще один пример условного оператора:

```
int a = 100;

if (a >= 100)
{
    Console.WriteLine("Значение переменной a больше или равно 100");
}
```

В данном случае вывод в консоль выполнится только если a больше либо равно 100.Иначе условный оператор ничего не выведет. Фигурные скобки в данном примере можно опустить.

Обратите внимание, что для проверки на равенство используется оператор ==, а не =. Знак «равно» используется в C# для присваивания значений. Следует понимать, что условное выражение, стоящее в конструкции if должно возвращать булево значение.

Например, следующий условный оператор всегда будет выполняться:

```
if (true)
    Console.WriteLine("Всегда выводится");
else
    Console.WriteLine("Никогда не выводится");
```

2.2 Оператор выбора.

Синтаксис оператора:

```
switch  (перем)
{
    case  константа1:
        оператор _1;
        break;
    case  константа2 :
        оператор _ 2;
        break;
...
    default :
        оператор _ n;
        break;
}
```

В данном операторе осуществляется выбор оператора (который может быть составным) в зависимости от значения переменной *перем*. Если *перем* равна значению *константа*1, то выполнится *оператор* _1, если *константа*2 – *оператор* _ 2, и т.д. Если ни одно значение, указанное в каком-либо операторе case, не совпало со значением *перем*, то выполнится оператор, указанный в секции default . Внутри каждой секции case следует указать оператор break, который предотвращает проверку других условий после выполнения данного оператора. Следует обратить внимание, что в секциях case следует использовать только константы (переменные не допускаются).

Пример использование оператора switch … case:

```
// Организация ответа на действие пользователя:
string text = "1 - Вывод группы \n" +
    "2 - Вывод фамилии преподавателя \n" +
    "3 - Вывод названия предмета \n" +
    "4 - Вывод приветствия \n";

Console.Write(text + "Введите команду > ");
int result = Convert.ToInt32(Console.ReadLine());

switch (result)
{
    case 1: { Console.WriteLine("ИСТБ - 101"); break; }
    case 2: Console.WriteLine("Николаев Евгений Иванович"); break;
    case 3: { Console.WriteLine("Технологии программирования"); break; }
    case 4: Console.WriteLine("Привет !"); break;
    default: Console.WriteLine("Недопустимый вариант !!!"); break;
}
```

Изучите представленный пример самостоятельно.

4. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

5. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

6. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе №1.

2. Выполните индивидуальное задание. Во всех заданиях переменные X, Y являются вещественными и вводятся пользователем. Количество слагаемых также вводится пользователем. Программа должна вывести сумму заданного числа членов последовательности.

Индивидуальное задание.

Перед выполнением задания требуется самостоятельно определить закономерность изменения членов последовательности, чтобы применить цикл, условный оператор или, если потребуется, оператор выбора.

Вариант	Выражение для вычисления
1	$j = -\frac{\sin^3(Y)}{1\cdot 3} + \frac{\sin^5(X^2)}{3\cdot 5} - \frac{\sin^7(Y^3)}{5\cdot 7} + \frac{\sin^9(X^4)}{7\cdot 9} - \ldots$
2	$Z = \frac{1}{1\cdot 3\cdot 5} - \frac{X}{2\cdot 4\cdot 6} + \frac{Y^2}{3\cdot 5\cdot 7} - \frac{X^3}{4\cdot 6\cdot 8} + \frac{Y^4}{5\cdot 7\cdot 9} - \ldots$
3	$Z = \frac{X^2}{1\cdot 3} - \frac{Y^4}{3\cdot 5} + \frac{X^6}{5\cdot 7} - \frac{Y^8}{7\cdot 9} + \ldots$
4	$Z = \frac{Y^2\cdot X}{2} - \frac{Y^2\cdot X^4}{4} + \frac{X^4\cdot Y^6}{6} - \frac{X^8\cdot Y^6}{8} + \ldots$
5	$A = -\frac{\sin(X)\cdot\lg(Y)}{1!} + \frac{\ln(X^3)\cdot\cos(Y^2)}{3!} - \frac{\sin(X^5)\cdot\lg(Y^3)}{5!} + \frac{\ln(X^7)\cdot\cos(Y^4)}{7!} - \ldots$
6	$Z = 1 - \frac{X}{2} + \frac{Y^2}{6} - \frac{X^3}{24} + \frac{Y^4}{120} - \ldots$ (в знаменателе факториал)
7	$s = -\frac{Y\cdot X^2}{1\cdot 3} + \frac{X\cdot Y^3}{2\cdot 4} - \frac{Y\cdot X^4}{3\cdot 5} + \frac{X\cdot Y^5}{4\cdot 6} - \ldots$
8	$Z = 1 - \frac{\sin(X^2)}{2} + \frac{\cos(Y^3)}{3} - \frac{\sin(X^4)}{4} + \frac{\cos(Y^5)}{5} - \ldots$
9	$Z = \frac{1}{1\cdot 3} - \frac{X}{2\cdot 4} + \frac{X^2}{3\cdot 5} - \frac{X^3}{4\cdot 6} + \ldots$
10	$Z = \frac{X^2}{1\cdot 3} - \frac{Y^4}{3\cdot 5} + \frac{X^6}{5\cdot 7} - \frac{Y^8}{7\cdot 9} + \ldots$

11	$p = -\frac{X \cdot \sin(Y)}{1 \cdot 2} + \frac{X^3 \cdot \cos(Y^2)}{3 \cdot 4} - \frac{X^5 \cdot \sin(Y^3)}{5 \cdot 6} + \frac{X^7 \cdot \cos(Y^4)}{7 \cdot 8} - \ldots$
12	$p = -\frac{X}{1 \cdot 2 \cdot 3} + \frac{Y^3}{3 \cdot 4 \cdot 5} - \frac{X^5}{5 \cdot 6 \cdot 7} + \frac{Y^7}{7 \cdot 8 \cdot 9} - \ldots$
13	$Z = 1 - \frac{\sin(X^2) + Y}{2} + \frac{\cos(Y^3) + X}{3} - \frac{\sin(X^4) + Y}{4} + \frac{\cos(Y^5) + X}{5} - \ldots$
14	$Z = \frac{Y^2 + X}{1 \cdot 2} - \frac{Y^2 - X^4}{2 \cdot 4} + \frac{X^4 + Y^6}{4 \cdot 6} - \frac{X^8 - Y^6}{6 \cdot 8} + \ldots$
15	$p = 1 - \frac{\cos(X) \cdot \sin^2(Y)}{1 \cdot 2} + \frac{\cos^4(X) \cdot \sin^3(Y)}{3 \cdot 4} - \frac{\cos^5(X) \cdot \sin^6(Y)}{5 \cdot 6} + \ldots$
16	$s = -\frac{\sin(Y) \cdot X^2}{1 \cdot 3} + \frac{X \cdot \cos(Y^3)}{2 \cdot 4} - \frac{\sin(Y) \cdot X^4}{3 \cdot 5} + \frac{X \cdot \cos(Y^5)}{4 \cdot 6} - \ldots$
17	$Z = -\frac{tg(Y) \cdot ctg(X^2)}{1 \cdot 3} + \frac{tg(X) \cdot ctg(Y^3)}{2 \cdot 4} - \frac{tg(Y) \cdot ctg(X^4)}{3 \cdot 5} + \frac{tg(X) \cdot ctg(Y^5)}{4 \cdot 6} - \ldots$
18	$Z = \frac{1}{1!} - \frac{X}{2!} + \frac{Y^2}{3!} - \frac{X^3}{4!} + \frac{Y^4}{5!} - \ldots$
19	$Z = \frac{\cos(X) + \lvert Y \rvert}{1!} - \frac{\cos^2(Y) + \lvert X \rvert}{2!} + \frac{\cos^3(X) + \lvert Y \rvert}{3!} - \frac{\cos^4(X) + \lvert Y \rvert}{4!} + \ldots$
20	$Z = \frac{Y^2 \cdot X}{2+1} - \frac{Y^2 \cdot X^4}{4-2} + \frac{X^4 \cdot Y^6}{6+4} - \frac{X^8 \cdot Y^6}{8-6} + \ldots$
21	$Z = 1 + \frac{1 - \cos(Y)}{1 \cdot 3 \cdot 5} - \frac{Y + \cos^2(X)}{2 \cdot 4 \cdot 6} + \frac{X^2 - \cos^3(Y)}{3 \cdot 5 \cdot 7} - \frac{Y^3 + \cos^4(X)}{4 \cdot 6 \cdot 8} + \ldots$
22	$Z = \frac{X - Y}{1 \cdot 3} - \frac{Y^2 - X^2}{2 \cdot 4} + \frac{X^3 - Y^3}{3 \cdot 5} - \frac{Y^4 - X^4}{4 \cdot 6} + \ldots$
23	$Z = 1 + \frac{1 - \cos(Y)}{1 \cdot 3 \cdot 5} - \frac{Y + \cos^2(X)}{2 \cdot 4 \cdot 6} + \frac{X^2 - \cos^3(Y)}{3 \cdot 5 \cdot 7} - \frac{Y^3 + \cos^4(X)}{4 \cdot 6 \cdot 8} + \ldots$
24	$Z = \frac{(X - Y)^2}{1 \cdot 3} - \frac{(Y - X)^4}{3 \cdot 5} + \frac{(X - Y)^6}{5 \cdot 7} - \frac{(Y - X)^8}{7 \cdot 9} + \ldots$
25	$Z = \frac{Y^2 \cdot \sin(X)}{2} - \frac{\cos(Y^2) \cdot X^4}{4} + \frac{\sin(X^4) \cdot Y^6}{6} - \frac{\cos(Y^6) \cdot X^8}{8} + \ldots$

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Как оформляется комментарий?
2. Опишите синтаксис условного оператора. Какие части данного оператора являются обязательными?
3. Опишите синтаксис оператора выбора. Поясните почему следует использовать оператор break внутри каждого выбора.
4. Можно ли с помощью for реализовать бесконечный цикл? Поясните ответ на примерах.

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [2-4].

ТЕМА 2. УПРАВЛЕНИЕ ПОТОКОМ: ЦИКЛЫ И ПОСЛЕДОВАТЕЛЬНОСТИ

ЛАБОРАТОРНАЯ РАБОТА 5. УПРАВЛЕНИЕ ПОТОКОМ ВЫПОЛНЕНИЯ С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРА ЦИКЛА FOR

1. Цель и содержание

Цель лабораторной работы: изучить операторы, позволяющие организовывать циклическое выполнение программного кода.

Задачи лабораторной работы:

– научиться применять оператор цикла for во вложенных циклах;

2. Теоретическая часть

Циклы позволяют выполнять определенную последовательность операторов до тех пор, пока выполняется некоторое условие. Каждый «круг» выполнения этого блока называется итерацией.

Синтаксис оператора:

for (*инициализатор*; *условие*; *итератор*)
    оператор

инициализатор – выражение, выполняемое перед первой итерацией цикла.

условие – выражение булевого типа, которое проверяется перед каждой итерацией. Если оно истинно то итерация выполняется, иначе – цикл завершается.

итератор – выражение, вычисляемое после каждой итерации.

Пример применения оператора цикла for (пример 1):

```
/*
* Вычислить сумму чисел от 1 до 1000
*/

// Объявляем переменную, которая хранит значение суммы чисел
System.Int32 Sum = 0;
// Объявляем счетчик цикла
int i;

for (i = 1; i <= 1000; i++)
    Sum += i;

Console.WriteLine("Итог: {0}", Sum);
```

Изучите код самостоятельно (создайте такую программу в VS). Еще один пример для самостоятельного изучения (пример 2):

```
/*
* вычислить сумму всех чисел, делящихся на 3 без остатка
* и находящихся в диапазоне от 1 до 1000000
*/

long Sum = 0;
int i;

for (i = 1; i <= 1000000; i++)
    if (i % 3 == 0)
        Sum += i;
```

Пример, отображающий возможность применения итератора (пример 3):

```
/*
* вычислить сумму всех чисел, делящихся на 3 без остатка
* и находящихся в диапазоне от 1 до 1000000
*/

long Sum = 0;
int i;

for (i = 0; i <= 1000000; i += 3)
        Sum += i;
```

Обратите внимание, что примеры 2 и 3 решают одну и ту же задачу.

3. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения работы

Необходимо модифицировать приложение, разработанное в рамках лабораторной работы №4 следцющим образом:

1. Программа должна запросить исходные данные для вычисления выражения.

2. Программа вычисляет выражение и выводит результат.

3. Затем программа снова запрашивает ввод исходных данных и цикл повторяется.

С использованием цикла for организуйте указанное поведение программы и самостоятельно определите команду, которая приведет к завершению работы программы (например, нажата клавиша «Escape», пользователь должен ответить на вопрос «Продолжить?» и т.п.).

Индивидуальное задание.

В качестве индивидуального задания необходимо изменить код из лабораторной работы №4.

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Как оформляется комментарий?

2. В разделе «Теоретическое обоснование» приведены примеры 1, 2 и 3. Почему в примерах 2 и 3 для переменной Sum выбран тип long, хотя в примере 1 для Sum выбран тип int? Обоснуйте ответ.

3. Опешите синтаксис оператора цикла for.

4. Можно ли с помощью for реализовать бесконечный цикл? Поясните ответ на примерах.

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [3], [7].

ТЕМА 2. УПРАВЛЕНИЕ ПОТОКОМ: ЦИКЛЫ И ПОСЛЕДОВАТЕЛЬНОСТИ

ЛАБОРАТОРНАЯ РАБОТА 6. УПРАВЛЕНИЕ ПОТОКОМВЫПОЛНЕНИЯ С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРОВ WHILE

1. Цель и содержание

Цель лабораторной работы: изучить операторы, позволяющие организовывать непоследовательное выполнение программного кода.

Задачи лабораторной работы:

– научиться применять оператор цикла с предусловием while;

– научиться применять оператор цикла с постусловием do … while.

2. Теоретическая часть

Цикл while похож на for тем, что является конструкцией с предварительной проверкой условия продолжения цикла. Но синтаксис цикла while более лаконичен:

while (*условие*)
оператор

В данном синтаксисе *оператор* может быть составным оператором (группа операторов языка, заключенные в фигурные скобки).

Следует понимать, что while используется для заранее неизвестного количества повторных выполнений операторов.

Пример выполнения цикла (пример 1):

```
/*
 * Цикл будет выполняться пока пользователь не введет 0
 */

bool Stop = false;
int chislo;

while (!Stop)
{
    Console.Write("Введите число > ");
    chislo = Convert.ToInt32(Console.ReadLine());

    if (chislo == 0)
        Stop = true;
}
```

Оператор break уже встречался в конструкции case оператора switch для выхода из case. Но break часто применяется внутри циклов for, while и do … while. Данный оператор прерывает выполнение цикла и передает управление оператору, следующему за циклом.

Пример, выполняющий то же самое, что и пример 1 (пример 2).

```
/*
 * Цикл будет выполняться пока пользователь не введет 0
 */

bool Stop = true;
int chislo;

while(Stop)
{
    Console.Write("Введите число > ");
    chislo = Convert.ToInt32(Console.ReadLine());

    if (chislo == 0)
        break;
}
```

Следует обратить внимание, что цикл является бесконечным (логическая переменная Stop всегда является true), но выполнение цикла все равно прервется, если пользователь введет 0.

Оператор continue также применяется внутри цикла – он останавливает выполнение текущей итерации и немедленно переходит к выполнению следующей итерации.

3. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе №1.
2. Выполните индивидуальное задание. Во всех заданиях переменныеX, Y являются вещественными и вводятся пользователем. Количество

слагаемых пользователем не вводится. Программа должна работать следующим образом:

– пользователю выводится приглашение на ввод X и Y;

– пользователь вводит X и Y;

– программа начинает расчет суммы, при этом выводится результат расчета, полученный на каждой итерации;

– сначала выводится номер итерации (1) и сумма 1-го слагаемого, затем программа останавливается и ждет ввода команды пользователя (1 – продолжить, 0 – прекратить расчет);

– если пользователь продолжает, выводится номер итерации (2) и сумма 2-х слагаемых и снова программа ждет команды пользователя, и т.д.

3. Программа не должна использовать цикл for.

Индивидуальное задание.

Перед выполнением задания требуется самостоятельно определить закономерность изменения членов последовательности, чтобы применить цикл, условный оператор или, если потребуется, оператор выбора.

Вариант	Выражение для вычисления
1	$Z = 1 - \frac{X}{2} + \frac{Y^2}{6} - \frac{X^3}{24} + \frac{Y^4}{120} - \ldots$ (в знаменателе факториал)
2	$s = -\frac{Y \cdot X^2}{1 \cdot 3} + \frac{X \cdot Y^3}{2 \cdot 4} - \frac{Y \cdot X^4}{3 \cdot 5} + \frac{X \cdot Y^5}{4 \cdot 6} - \ldots$
3	$Z = 1 - \frac{\sin(X^2)}{2} + \frac{\cos(Y^3)}{3} - \frac{\sin(X^4)}{4} + \frac{\cos(Y^5)}{5} - \ldots$
4	$Z = \frac{1}{1 \cdot 3} - \frac{X}{2 \cdot 4} + \frac{X^2}{3 \cdot 5} - \frac{X^3}{4 \cdot 6} + \ldots$
5	$Z = \frac{X^2}{1 \cdot 3} - \frac{Y^4}{3 \cdot 5} + \frac{X^6}{5 \cdot 7} - \frac{Y^8}{7 \cdot 9} + \ldots$
6	$p = -\frac{X \cdot \sin(Y)}{1 \cdot 2} + \frac{X^3 \cdot \cos(Y^2)}{3 \cdot 4} - \frac{X^5 \cdot \sin(Y^3)}{5 \cdot 6} + \frac{X^7 \cdot \cos(Y^4)}{7 \cdot 8} - \ldots$
7	$p = -\frac{X}{1 \cdot 2 \cdot 3} + \frac{Y^3}{3 \cdot 4 \cdot 5} - \frac{X^5}{5 \cdot 6 \cdot 7} + \frac{Y^7}{7 \cdot 8 \cdot 9} - \ldots$

8	$Z = 1 - \frac{\sin(X^2)+Y}{2} + \frac{\cos(Y^3)+X}{3} - \frac{\sin(X^4)+Y}{4} + \frac{\cos(Y^5)+X}{5} - \ldots$
9	$Z = \frac{Y^2+X}{1\cdot 2} - \frac{Y^2-X^4}{2\cdot 4} + \frac{X^4+Y^6}{4\cdot 6} - \frac{X^8-Y^6}{6\cdot 8} + \ldots$
10	$j = -\frac{\sin^3(Y)}{1\cdot 3} + \frac{\sin^5(X^2)}{3\cdot 5} - \frac{\sin^7(Y^3)}{5\cdot 7} + \frac{\sin^9(X^4)}{7\cdot 9} - \ldots$
11	$Z = \frac{1}{1\cdot 3\cdot 5} - \frac{X}{2\cdot 4\cdot 6} + \frac{Y^2}{3\cdot 5\cdot 7} - \frac{X^3}{4\cdot 6\cdot 8} + \frac{Y^4}{5\cdot 7\cdot 9} - \ldots$
12	$Z = \frac{X^2}{1\cdot 3} - \frac{Y^4}{3\cdot 5} + \frac{X^6}{5\cdot 7} - \frac{Y^8}{7\cdot 9} + \ldots$
13	$Z = \frac{Y^2\cdot X}{2} - \frac{Y^2\cdot X^4}{4} + \frac{X^4\cdot Y^6}{6} - \frac{X^8\cdot Y^6}{8} + \ldots$
14	$A = -\frac{\sin(X)\cdot\lg(Y)}{1!} + \frac{\ln(X^3)\cdot\cos(Y^2)}{3!} - \frac{\sin(X^5)\cdot\lg(Y^3)}{5!} + \frac{\ln(X^7)\cdot\cos(Y^4)}{7!} - \ldots$
15	$Z = \frac{\cos(X)+\lvert Y\rvert}{1!} - \frac{\cos^2(Y)+\lvert X\rvert}{2!} + \frac{\cos^3(X)+\lvert Y\rvert}{3!} - \frac{\cos^4(X)+\lvert Y\rvert}{4!} + \ldots$
16	$Z = 1 + \frac{1-\cos(Y)}{1\cdot 3\cdot 5} - \frac{Y+\cos^2(X)}{2\cdot 4\cdot 6} + \frac{X^2-\cos^3(Y)}{3\cdot 5\cdot 7} - \frac{Y^3+\cos^4(X)}{4\cdot 6\cdot 8} + \ldots$
17	$Z = \frac{X-Y}{1\cdot 3} - \frac{Y^2-X^2}{2\cdot 4} + \frac{X^3-Y^3}{3\cdot 5} - \frac{Y^4-X^4}{4\cdot 6} + \ldots$
18	$Z = 1 + \frac{1-\cos(Y)}{1\cdot 3\cdot 5} - \frac{Y+\cos^2(X)}{2\cdot 4\cdot 6} + \frac{X^2-\cos^3(Y)}{3\cdot 5\cdot 7} - \frac{Y^3+\cos^4(X)}{4\cdot 6\cdot 8} + \ldots$
19	$Z = \frac{(X-Y)^2}{1\cdot 3} - \frac{(Y-X)^4}{3\cdot 5} + \frac{(X-Y)^6}{5\cdot 7} - \frac{(Y-X)^8}{7\cdot 9} + \ldots$
20	$Z = \frac{Y^2\cdot\sin(X)}{2} - \frac{\cos(Y^2)\cdot X^4}{4} + \frac{\sin(X^4)\cdot Y^6}{6} - \frac{\cos(Y^6)\cdot X^8}{8} + \ldots$
21	$p = 1 - \frac{\cos(X)\cdot\sin^2(Y)}{1\cdot 2} + \frac{\cos^4(X)\cdot\sin^3(Y)}{3\cdot 4} - \frac{\cos^5(X)\cdot\sin^6(Y)}{5\cdot 6} + \ldots$
22	$s = -\frac{\sin(Y)\cdot X^2}{1\cdot 3} + \frac{X\cdot\cos(Y^3)}{2\cdot 4} - \frac{\sin(Y)\cdot X^4}{3\cdot 5} + \frac{X\cdot\cos(Y^5)}{4\cdot 6} - \ldots$
23	$Z = -\frac{tg(Y)\cdot ctg(X^2)}{1\cdot 3} + \frac{tg(X)\cdot ctg(Y^3)}{2\cdot 4} - \frac{tg(Y)\cdot ctg(X^4)}{3\cdot 5} + \frac{tg(X)\cdot ctg(Y^5)}{4\cdot 6} - \ldots$
24	$Z = \frac{1}{1!} - \frac{X}{2!} + \frac{Y^2}{3!} - \frac{X^3}{4!} + \frac{Y^4}{5!} - \ldots$
25	$Z = \frac{Y^2\cdot X}{2+1} - \frac{Y^2\cdot X^4}{4-2} + \frac{X^4\cdot Y^6}{6+4} - \frac{X^8\cdot Y^6}{8-6} + \ldots$

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Какой цикл лучше while или for?

2. Опишите синтаксис оператора while.

3. Опишите синтаксис условного оператора. Какие части данного оператора являются обязательными?

4. Опишите синтаксис оператора выбора. Поясните почему следует использовать оператор break внутри каждого выбора.

5. Поясните назначение операторов break и continue.

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [3], [7].

ТЕМА 2. УПРАВЛЕНИЕ ПОТОКОМ: ЦИКЛЫ И ПОСЛЕДОВАТЕЛЬНОСТИ

ЛАБОРАТОРНАЯ РАБОТА 7. УПРАВЛЕНИЕ ПОТОКОМ ВЫПОЛНЕНИЯ С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРА DO …WHILE

1. Цель и содержание

Цель лабораторной работы: изучить операторы, позволяющие организовывать непоследовательное выполнение программного кода.

Задачи лабораторной работы:

– научиться применять оператор цикла с постусловием do … while.

– изучить особенности оператора do…while;

– научиться использовать оператор цикла do…while для решения практических задач.

2. Теоретическая часть

Цикл do … while полностью повторяет функциональные возможности while, но предполагает проверку условия окончания цикла после выполнения тела цикла. То есть блок операторов тела цикла всегда выполнится хотя бы один раз.

Синтаксис оператора:

```
do
{
        оператор (операторы)
}
while  (условие);
```

Пример использования цикла (пример 1):

```
/*
 * Цикл будет выполняться пока пользователь не введет 0
 */

bool Stop = false;
int chislo;

do
{
    Console.Write("Введите число > ");
    chislo = Convert.ToInt32(Console.ReadLine());

    if (chislo == 0)
        Stop = true;
} while (!Stop);
```

3. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального

компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе №1.
2. Выполните индивидуальное задание. Во всех заданиях переменные X, Y являются вещественными и вводятся пользователем. Количество слагаемых пользователем не вводится. Программа должна работать следующим образом:
   – пользователю выводится приглашение на ввод X и Y;
   – пользователь вводит X и Y;
   – программа начинает расчет суммы, при этом выводится результат расчета, полученный на каждой итерации;
   – сначала выводится номер итерации (1) и сумма 1-го слагаемого, затем программа останавливается и ждет ввода команды пользователя (1 – продолжить, 0 – прекратить расчет);
   – если пользователь продолжает, выводится номер итерации (2) и сумма 2-х слагаемых и снова программа ждет команды пользователя, и т.д.
3. Программа не должна использовать цикл for или while.

Индивидуальное задание.

Необходимо использовать индивидуальное задание из лабораторной работы №6.

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Какой цикл лучше do…while или while?

2. Опишите синтаксис оператора do…while.

3. Опишите синтаксис условного оператора. Какие части данного оператора являются обязательными?

4. Опишите синтаксис оператора выбора. Поясните почему следует использовать оператор break внутри каждого выбора.

5. Опешите синтаксис оператора do … while.

6. Поясните назначение операторов break и continue.

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [6-8].

ТЕМА 3. РАБОТА С ДАННЫМИ: МАССИВЫ, СТРОКИ, РЕКУРСИЯ
ЛАБОРАТОРНАЯ РАБОТА 8. КЛАССЫ. СТРУКТУРЫ.

1. Цель и содержание

Цель лабораторной работы: изучить структуру и принципы объявления классов, освоить технологию создания экземпляров классов (объектов).

Задачи лабораторной работы:

– научиться объявлять классы;

– научиться создавать объекты классов;

– научиться работать с полями данных и методами классов.

2. Теоретическая часть

2.1 Классы и структуры.

Класс – это тип данных, объединяющий данные и методы их обработки. Класс – это пользовательский шаблон, в соответствии с которым можно создавать объекты. То есть класс – это правило, по которому будет строиться объект. Сам класс не содержит данных.

Объект класса (экземпляр класса) – переменная типа класс. Объект содержит данные и методы, манипулирующие этими данными. Класс

определяет, какие данные содержит объект и каким образом он ими манипулирует.

Все что справедливо для классов можно распространить и на структуры. Отличие состоит в методе хранения объектов данных типов в оперативной памяти: структуры – это типы по значению, они размещаются в стеке; классы – это ссылочные типы, объекты классов размещаются в куче. Структуры не поддерживают наследование.

Структуры применяются для представления небольших объемов данных. Объявление структур происходит с использованием ключевого слова struct, объявление классов – с помощью ключевого слова class.

Пример объявления класса:

```
// Объявление класса
public class MyFirstClass
{
    // Данные-члены класса

    // Доступные на уровне экземпляра
    public int a;
    public float b__;
    public string fio;

    // Доступные только на уровне класса
    private bool IsOK;
    private double precision;
}
```

При создании как классов, так и структур, используется ключевое слово new, например:

```
MyFirstClass obj = new MyFirstClass();
```

2.2 Структура класса.

Данные и функции, объявленные внутри класса, называются членами класса (class members). Доступность членов класса может быть описана как public, private, protected, internal или internal protected.

2.2.1 Данные-члены.

Данные-члены – это те структуры внутри класса, которые содержат данные класса – поля, константы события.

Поля – это любые переменные, ассоциированные с классом. После создания экземпляра класса к полям можно обращаться с использованием синтаксиса , например: ИмяПоля ИмяОбъекта.

```
MyFirstClass obj = new MyFirstClass();

obj.a = 5;
int cc = obj.a;

obj.fio = "Novak E.I.";
```

Аналогичным образом с классом ассоциируются константы.

События будут рассмотрены в следующих лабораторных работах.

2.2.2 Функции-члены.

Функции-члены – это члены, которые обеспечивают некоторую функциональность для манипулирования данными классов. Они делятся на следующие виды: методы, свойства, конструкторы, финализаторы, операции и индексаторы.

Методы (method) – это функции, ассоциированные с определенным классом. Как и данные-члены, по умолчанию они являются членами экземпляра. Они могут быть объявлены статическими с помощью модификатора static.

Свойства (property) – это наборы функций, которые могут быть доступны клиенту таким же способом, как общедоступные поля класса. В C# предусмотрен специальный синтаксис для реализации чтения и записи свойств для классов, поэтому писать собственные методы с именами, начинающимися на Set и Get, не понадобится. Поскольку не существует какого-то отдельного синтаксиса для свойств, который отличал бы их от нормальных функций, создается иллюзия объектов как реальных сущностей, предоставляемых клиентскому коду.

Конструкторы (constructor) – это специальные функции, вызываемые

автоматически при инициализации объекта. Их имена совпадают с именами классов, которым они принадлежат, и они не имеют типа возврата. Конструкторы полезны для инициализации полей класса.

Финализаторы (finalizer) похожи на конструкторы, но вызываются, когда среда CLR определяет, что объект больше не нужен. Они имеют то же имя, что и класс, но с предшествующим символом тильды (~). Предсказать точно, когда будет вызван финализатор, невозможно.

Операции (operator) – это простейшие действия вроде + или -. Когда вы складываете два целых числа, то, строго говоря, применяете операцию + к целым. Однако C# позволяет указать, как существующие операции будут работать с пользовательскими классами (так называемая перегрузка операций).

Индексаторы (indexer) позволяют индексировать объекты таким же способом, как массив или коллекцию.

В данной лабораторной работе рассматриваются только методы класса – это функции, ассоциированные с определенным классом.

В C# объявление метода класса состоит из спецификатора доступности, возвращаемого значения, имени метода, списка формальных параметров и тела метода:

```
[модификатор] тип_возврата имя_метода ([список_параметров])
{
    // тело метода
}
```

Например, добавим методы для объявленного ранее класса MyFirstClass:

```
// Объявление класса
public class MyFirstClass
{
    // Данные-члены класса

    // Доступные на уровне экземпляра
    public int a;
    public float b__;
    public string fio;

    // Доступные только на уровне класса
    private bool IsOK;
    private double precision;

    // Метод для инициализации некоторых полей класса
    public void InitClassMembers(int pA, float pB__, string pFio)
    {
        a = pA;
        b__ = pB__;
        fio = pFio;
    }

    // Метод, возвращающий значение вычисленное на основе полей класса
    public int GetAbsA()
    {
        return Math.Abs(a);
    }
}
```

Синтаксис вызова методов аналогичен синтаксису обращения к данным-членам:

```
MyFirstClass obj = new MyFirstClass();

obj.InitClassMembers(10, 0.8F, "Новиков П.Е.");

int abs_a = obj.GetAbsA();
```

В данном примере метод InitClassMembers не возвращает никаких данных, но требует передачи ему фактических параметров. В свою очередь, метод GetAbsA возвращает значение типа int и не предполагает никаких параметров.

В общем случае параметры могут передаваться методу либо по значению, либо по ссылке. Когда переменная передается по ссылке, вызываемый метод получает саму переменную, поэтому любые изменения, которым она подвергнется внутри метода, останутся в силе после его завершения. Но если переменная передается по значению, вызываемый метод получает копию этой переменной, а это значит, что все изменения в ней по завершении метода будут

утеряны. Для сложных типов данных передача по ссылке более эффективна из-за большого объема данных, который приходится копировать при передаче по значению.

Если не указано обратное, то в C# все параметры передаются по значению. Тем не менее, можно принудительно передавать значения по ссылке, для чего используется ключевое слово ref. Если параметр передается в метод, и входной аргумент этого метода снабжен префиксом ref, то любые изменения этой переменной, которые сделает метод, отразятся на исходном объекте.

В C-подобных языках функции часто возвращают более одного значения. Это обеспечивается применением выходных параметров, за счет присваивания значений переменным, переданным в метод по ссылке. Часто первоначальное значение таких переменных не важно. Эти значения перезаписываются в функции, которая может даже не обращать внимания на то, что в них хранилось первоначально.

Было бы удобно использовать то же соглашение в C#. Однако в C# требуется, чтобы переменные были инициализированы каким-то начальным значением перед тем, как к ним будет выполнено обращение. Хотя можно инициализировать входные переменные какими-то бессмысленными значениями до передачи их в функцию, которая наполнит их осмысленными значениями, этот прием выглядит в лучшем случае излишним, а в худшем – сбивающим с толку. Тем не менее, существует способ обойти требование компилятора C# относительно начальной инициализации переменных.

Это достигается ключевым словом out. Когда входной аргумент снабжен префиксом out, этому методу можно передать неинициализированную переменную. Переменная передается по ссылке, поэтому любые изменения, выполненные методом в переменной, сохраняются после того, как он вернет управление. Ключевое слово out также должно указываться при вызове метода – так же, как при его определении.

Частичные классы.

Ключевое слово partial (частичный) позволяет определить класс, структуру или интерфейс, распределенный по нескольким файлам. Но ситуации, когда множеству разработчиков требуется доступ к одному и тому же классу, или же в ситуации, когда некоторый генератор кода генерирует часть класса, такое разделение класса на несколько файлов может оказаться полезным. Ключевое слово partial просто помещается перед классом, структурой или интерфейсом.

Статические классы.

Статический класс функционально представляет собой то же самое, что и класс с приватным статическим конструктором. Создать экземпляр такого класса невозможно. Если указать ключевое слово static в объявлении класса, компилятор будет гарантировать, что к этому классу никогда не будут добавлены нестатические члены.

Класс Object.

Rлассы .NET изначально унаследованы от System.Object. Фактически, если при определении нового класса базовый класс не указан, компилятор автоматически предполагает, что он наследуется от Object.

Практическое значение этого в том, что помимо методов и свойств, которые программист определяет самостоятельно, также появляется доступ к множеству общедоступных и защищенных методов-членов, которые определены в классе Object. Эти методы присутствуют во всех определяемых классах.

3. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое

устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе №1.

2. Изучите пример выполнения задания, представленный в данном разделе.

3. Выполните индивидуальное задание. Задания ориентированы на работу с классами.

Пример выполнения задания.

Разработать класс для представления объекта «Прямоугольный параллелепипед». Реализуйте все необходимые поля данных (закрытые) и методы позволяющие:

– устанавливать и считывать значения полей данных;

– вычислять объем прямоугольного параллелепипеда;

– вычислять площадь поверхности прямоугольного параллелепипеда;

– выводить полную информацию об объекте в консоль.

Решение данной задачи состоит из двух этапов: объявление класса Parallelepiped и демонстрация использования объекта данного класса.

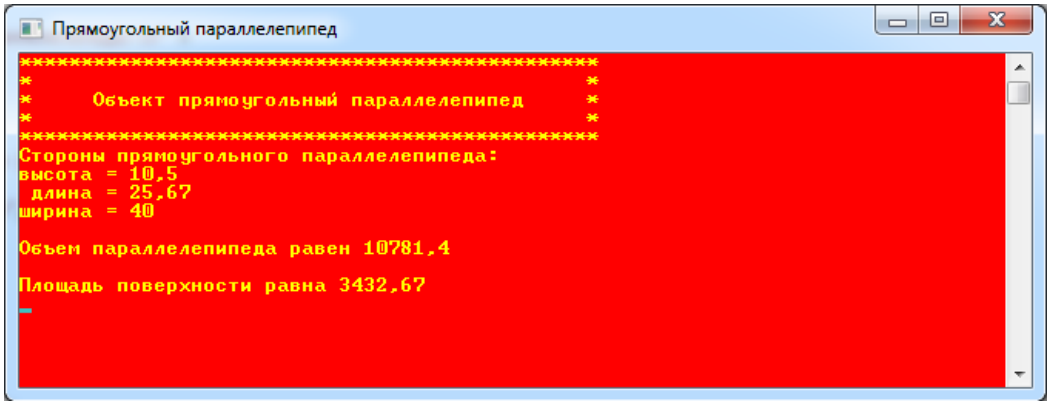
Полный листинг примера:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR_Three
{
    class Parallelepiped
    {
        // Поля данных представляют стороны параллелепипеда
        private double a, b, c;

        // Методы для установки и считывания значений полей
        public void Set_a(double pa) { a = pa; }
        public double Get_a() { return a; }

        public void Set_b(double pb) { b = pb; }
        public double Get_b() { return b; }

        public void Set_c(double pc) { c = pc; }
        public double Get_c() { return c; }


        // Метод для вычисления объема прямоугольного параллелепипеда
        public double GetV()
        {
            return a * b * c;
        }

        // Метод для вычисления площади поверхности прямоугольного параллелепипеда
        public double GetS()
        {
            return 2 * (a * b + b * c + a * c);
        }
```

```
        // Метод для вывода полной информации об объектк в консоль
        public void PrintFullInformation()
        {
            string str =    "*********************************************\n" +
                            "*                                           *\n" +
                            "*     Объект прямоугольный параллелепипед   *\n" +
                            "*                                           *\n" +
                            "*********************************************";
            Console.WriteLine(str);

            Console.WriteLine("Стороны прямоугольного параллелепипеда:\n" +
                "высота = {0}\n "+
                "длина = {1}"+
                "ширина = {2}", a, b, c);

            Console.WriteLine("Объем параллелепипеда равен {0}", GetV());

            Console.Write("Площадь поверхности равна {0}", GetS());
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Console.Title = "Прямоугольный параллелепипед";
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.BackgroundColor = ConsoleColor.Red;
            Console.Clear();

            Parallelepiped p;

            p = new Parallelepiped();
            p.Set_a(10.5);
            p.Set_b(25.67);
            p.Set_c(40);

            p.PrintFullInformation();

            Console.ReadKey();
        }
    }


}
```

В данном примере необходимо обратит внимание на тот факт, что все вычисления выполняются внутри класса. Метод Main содержит только вызовы методов класса, то есть вся реализация скрыта.

В результате выполнения программы отобразится следующее консольное окно с выводом информации:



Индивидуальное задание.

Спроектируйте класс, наполните его требуемой функциональностью, продемонстрируйте работоспособность класса.

Вариант	Выражение для вычисления
1.	Класс «Шар». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте.
2.	Класс «Куб». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, длины диагонали, а также вывод информации об объекте.
3.	Класс «Сфера». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте.
4.	Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте.
5.	Класс «График y=x». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте.
6.	Класс «Шар». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте.
7.	Класс «Матрица MxN». Реализовать инициализацию элементов матрицы случайными числами, вывод матрицы, нахождение максимального и минимального элементов, а также вывод информации об объекте.
8.	Класс «Прямоугольный треугольник». Реализовать ввод и вывод полей данных, вычисление гипотенузы, площади и периметра, а также вывод информации об объекте.

9.	Класс «Отрезок». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте.
10.	Класс «Цилиндр». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
11.	Класс «Ромб». Реализовать ввод и вывод полей данных (диагонали ромба), вычисление площади, периметра, а также вывод информации об объекте.
12.	Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте.
13.	Класс «График y=3x+5». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте.
14.	Класс «Матрица MxN». Реализовать инициализацию элементов матрицы случайными числами, вывод транспонированной матрицы, нахождение среднего арифметического всех элементов, а также вывод информации об объекте.
15.	Класс «Отрезок в пространстве». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте.
16.	Класс «График y=x-10». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте.
17.	Класс «Матрица MxN». Реализовать инициализацию элементов матрицы случайными числами, вывод транспонированной матрицы, нахождение и вывод среднего арифметического элементов в каждом столбце.
18.	Класс «Куб». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, длины диагонали, а также вывод информации об объекте.
19.	Класс «Сфера». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте.
20.	Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте.

21.	Класс «График y=x». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте.
22.	Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте.
23.	Класс «График y=-x». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте.
24.	Класс «Цилиндр». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
25.	Класс «Отрезок в пространстве». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте.

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Что такое класс?

2. Что такое структура? Чем структура отличается от класса?

3. Что такое члены класса? Какие группы членов класса вы знаете?

4. Какие типы членов-данных вы знаете?

5. Какие типы функций-членов класса вы знаете?

6. Как поменять цвет текста в консольном приложении?

7. Как поменять цвет фона в консольном приложении?

8. Какие модификаторы доступности членов класса вы знаете?

9. Какое ключевое слово используется для создания объекта класса?

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [8].

ТЕМА 4. ОСНОВЫ ООП: КЛАССЫ, КОНСТРУКТОРЫ, ПЕРЕГРУЗКА
ЛАБОРАТОРНАЯ РАБОТА 9. КОНСТРУКТОР КЛАССА. ПЕРЕГРУЗКА КОНСТРУКТОРОВ КЛАССА.

1. Цель и содержание

Цель лабораторной работы: понять принципы работы конструктора.

Задачи лабораторной работы:

– научиться объявлять конструктор класса;

– научиться создавать перегруженные конструкторы.

2. Теоретическая часть

Конструктор – это метод класса, который не возвращает значения и имеет то же самое имя, что и класс. Если конструктор класса не определен программистом явно, то компилятор создаст конструктор по умолчанию.

Конструкторы подчиняются тем же правилам перегрузки, что и все методы.

В C# поддерживается перегрузка методов – то есть может существовать несколько версий одного метода, но с разными сигнатурами (методы

отличаются количеством и / или типом параметров). Чтобы перегрузить метод, просто объявляются методы с одинаковыми именами, но разными сигнатурами.

3. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения работы

1. Для выполнения лабораторной работы необходимо модифицировать приложение, полученное в результате выполнения индивидуального задания лабораторной работы №13.

2. Модификация сводится к следующему: необходимо объявить и продемострировать использование 3-4 перегруженных конструкторов класса.

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Что такое класс?

2. Что такое конструктор класса?

3. Что такое перегрузка методов?

4. Может ли один конструктор класса вызывать другой конструктор? Прежде чем отвечать попробуйте реализовать такой вызов в своем разработанном классе.

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [7-8].

ТЕМА 4. ОСНОВЫ ООП: КЛАССЫ, КОНСТРУКТОРЫ, ПЕРЕГРУЗКА
ЛАБОРАТОРНАЯ РАБОТА 10. МНОГОМОДУЛЬНЫЕ ПРИЛОЖЕНИЯ

1. Цель и содержание

Цель лабораторной работы: научиться проектировать консольные приложения на основе нескольких сборок.

Задачи лабораторной работы:

– научиться управлять несколькими разрабатываемыми проектами в одном решении;

– научиться организовывать взаимодействие нескольких модулей приложения.

2. Теоретическая часть

В терминологии технологической платформы .NET не используется понятие «исполняемый файл». Подобные файлы (*.dll или *.exe) характерны для компиляторов небезопасного кода.

В технологии .NET используется термин «сборка». Сборка – это модуль, в котором хранится компилированный управляемый код. Она похожа на классический исполняемый файл (*.exe) или dll-библиотеку, но имеет важное

свойство – она полностью себя описывает. Сборки содержат метаданные, которые включают в себя сведения о сборке и обо всех определенных внутри нее типах, методах и т.д. Сборка может быть частной (доступной только одному приложению) или разделяемой (доступной любому приложению Windows).

В предыдущих лабораторных работах студентам предлагалось спроектировать консольные приложения, которые представляют собой одномодульные приложения – exe-файлы.

В данной лабораторной работе требуется спроектировать приложение, которое состоит из нескольких сборок (один модуль – файл *.exe, остальные – dll).

3. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место

пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения работы

Для выполнения лабораторной работы спроектируем следующее приложение:

Требуется разработать приложение-опрос. Пользователю последовательно задаются вопросы, за каждый вариант ответа начисляются определенные баллы. После завершения опросы выводится сумма набранных баллов и какое-либо резюме. Такая программа может использоваться для оценки остаточных знаний в форме тестирования, для составления опросов, для самообследования и т.п.

Перед началом создания приложения, то есть перед непосредственным программированием, необходимо определиться: какие объекты и явления предметной области и связи между ними должен выявить программист. Очевидно, что это:

– вопрос;

– ответ (несколько для каждого вопроса);

– балл соответствующий конкретному ответу;

– итоговые резюме или оценки, которые выводятся в зависимости от количества набранных баллов.

Второе, что необходимо сделать осуществить физическую декомпозицию данного приложения, то есть сколько отдельных модулей будет содержать приложение. В данном случае:

– 1-ый модуль в виде файла *.exe для запуска приложения;

– 2-ой модуль в виде файла *.dll, содержащий вопросы и ответы;

– 3-ий модуль в виде файла *.dll, содержащий сообщения о результатах тестирования (опроса).

Декомпозиция, логическая и физическая, завершена. Для реализации прложения необходимо выполнить следующие шаги:

1. Создайте консольное приложение в среде MS VS (проект назовите Quiz). После выполнения всех необходимых действий окно Solution Explorer примет следующий вид:

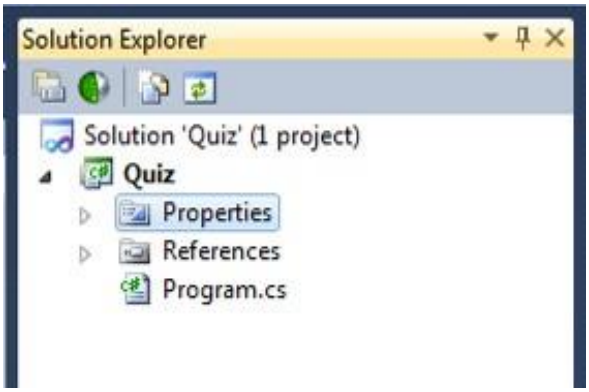


Рисунок 15.1 – Структура одномодульного приложения.

В окне отображено одно решение (Solution) с именем «Quiz» и один проект в рамках решения (с именем «Quiz»). Проект содержит файл кода Program.cs, в котором определена функция Main. Этот проект будет откомпилирован в файл Quiz.exe (то есть файл для запуска).

2. Создадим еще один файл в рамках проекта Quiz, который будет содержать класс Testing, позволяющий реализовывать логику тестирования, то есть вывод вопросов в определенной последовательности, ввод выбора пользователя и т.д. Для этого вызовем контекстное меню проекта (рис. 13.2) и создадим новый файл с именем Testing.cs

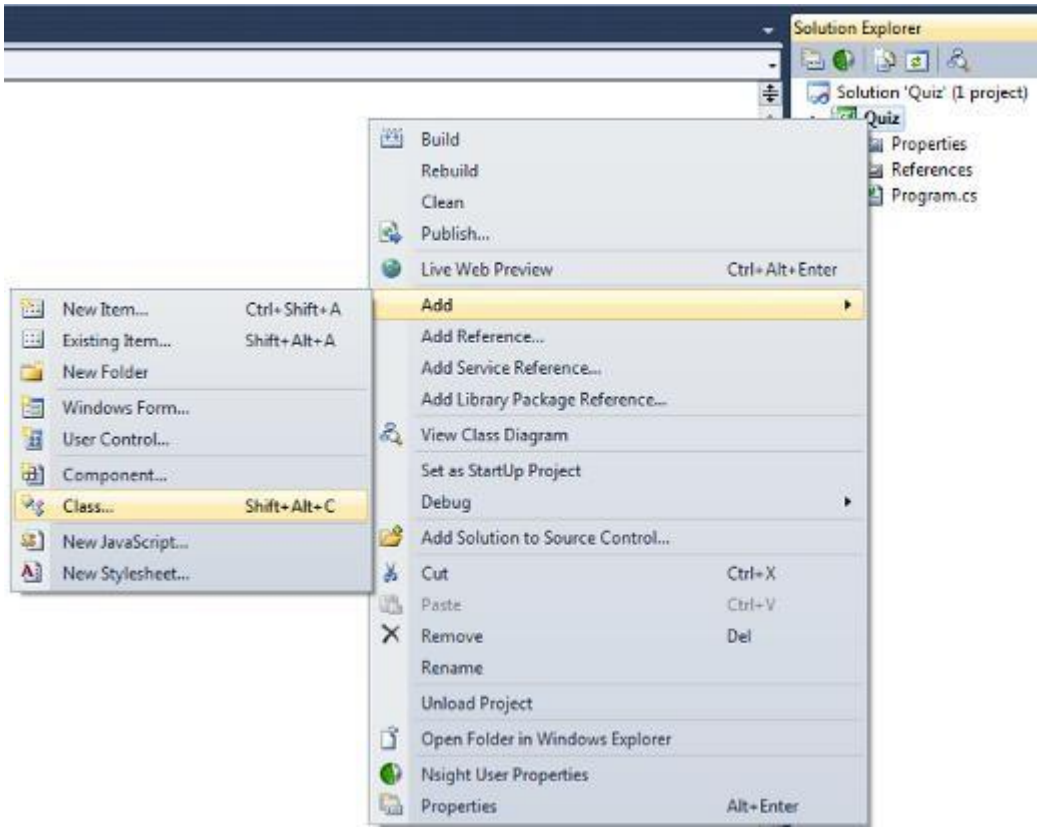


Рисунок 15.2 – Добавления нового класса к проекту.

После выполнения данного действия в окне Solution Explorer состав проекта изменится (рис. 15.3).

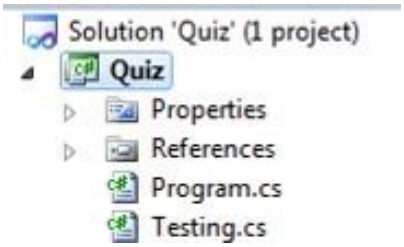


Рисунок 15.3 – Проект с несколькими файлами исходного кода.

На данном этапе класс Testing не содержит никакой логики.

3. Создадим второй модуль в виде dll-библиотеки, которая будет содержать вопросы и ответы. Для этого вызовем команду контекстного меню решения Quiz (рис. 15.4).

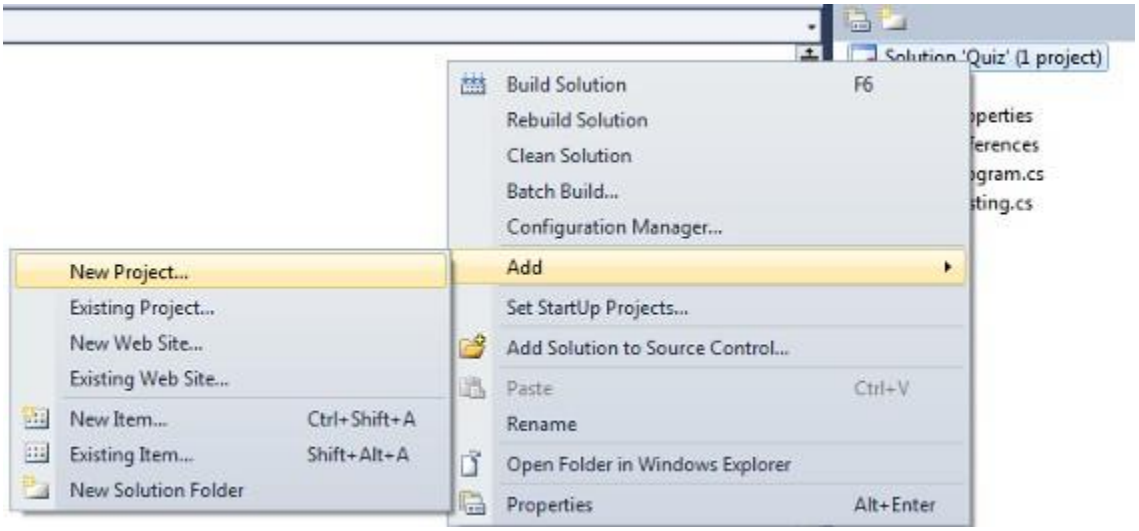


Рисунок 15.4 – Добавление проекта к решению VS.

4. В появившемся диалоговом окне выберем тип проекта «Class Library», в качестве имени проекта укажем «Task» (рис. 15.5)

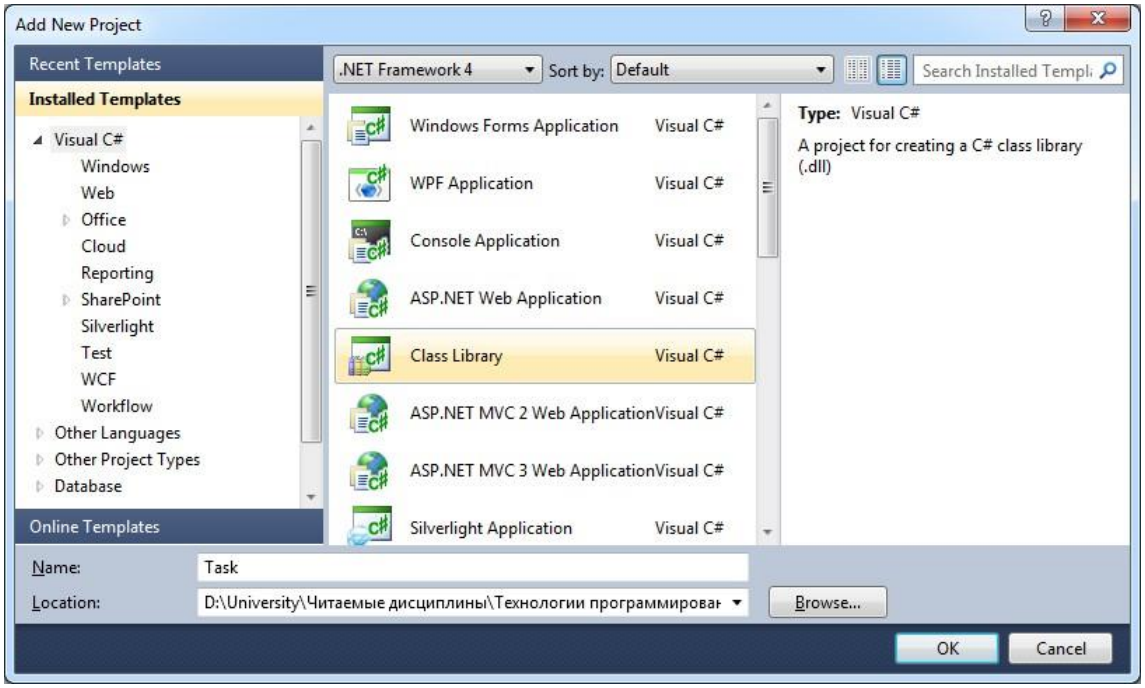


Рисунок 15.5 –Добавление библиотеки классов к решению Quiz.

После добавления нового проекта окно Solution Explorer примет вид, показанный на рис. 15.6

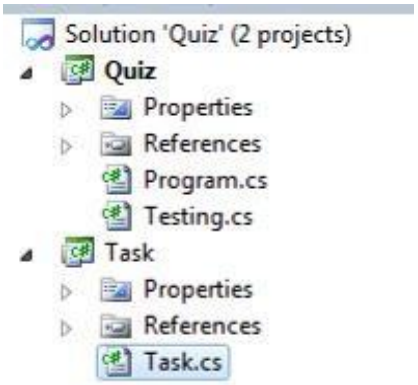


Рисунок 15.6 –Решение с несколькими проектами.

Два проекта в одном решении Quiz, при этом проект Quiz помечен полужирным шрифтом. Это означает, что при вызове команды Run будет запущен именно этот проект. Следует обратить внимание, что файл Task.cs проекта Task содержит только пустой класс без кода. Проект Task в целом не содержит функции Main, то есть не может быть запушен на выполнение.

5. Аналогичным образом добавим к решению еще один проект с именем «Result» и типом «Class Library». Окно Solution Explorer примет вид:

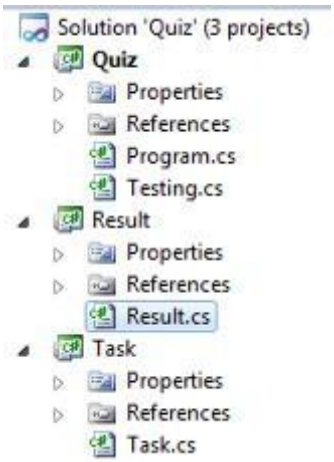


Рисунок 15.7 – Все требуемые проекты, в соответствие с физической декомпозицией, добавлены к решению.

Далее переходим к наполнению классов требуемым функционвалом.

6. Модифицируем файл Task.cs проекта Task следующим образом:

```
namespace Task
{
    // Класс для представления одного тестового задания
    public class SingleTest
    {
        public string Question {get; set;} // вопрос
        public List<string> Answers; // Ответы
        public List<int> Balls; // Баллы за ответы
    }

    // Статический класс для представления списка тестовых заданий
    public static class Task
    {
        // Список отдельных вопросов с ответами
        public static List<SingleTest> AllQuestions;

        // Статический конструктор для инициализации коллекции
        static Task()
        {
            AllQuestions = new List<SingleTest>()
            {
                new SingleTest()
                {
                    Question = "Main - точка входа программы ...",
                    Answers = new List<string>() { "1", "7", "Не знаю" },
                    Balls = new List<int>() { 20, 5, 0 }
                },
                new SingleTest()
                {
                    Question = "Какой оператор не является оператором цикла?",
                    Answers = new List<string>() { "while", "for", "if" },
                    Balls = new List<int>() { -5, 0, 10 }
                },
                new SingleTest()
                {
                    Question = "Как обозначить составной оператор?",
                    Answers = new List<string>() { "{ ... }", "( ... )", "< ... >" },
                    Balls = new List<int>() { 50, 0, 0 }
                },
                new SingleTest()
                {
                    Question = "Что обозначает символ ; ",
                    Answers = new List<string>() { "Пустой оператор", "Конец программы", "Не знаю" },
                    Balls = new List<int>() { 50, 20, 0 }
                }
            };
        }
    }
}
```

Рисунок 15.8 – Классы проекта Task.

Следует обратить внимание на то, что класс Task статический, то есть не требует создания объектов.

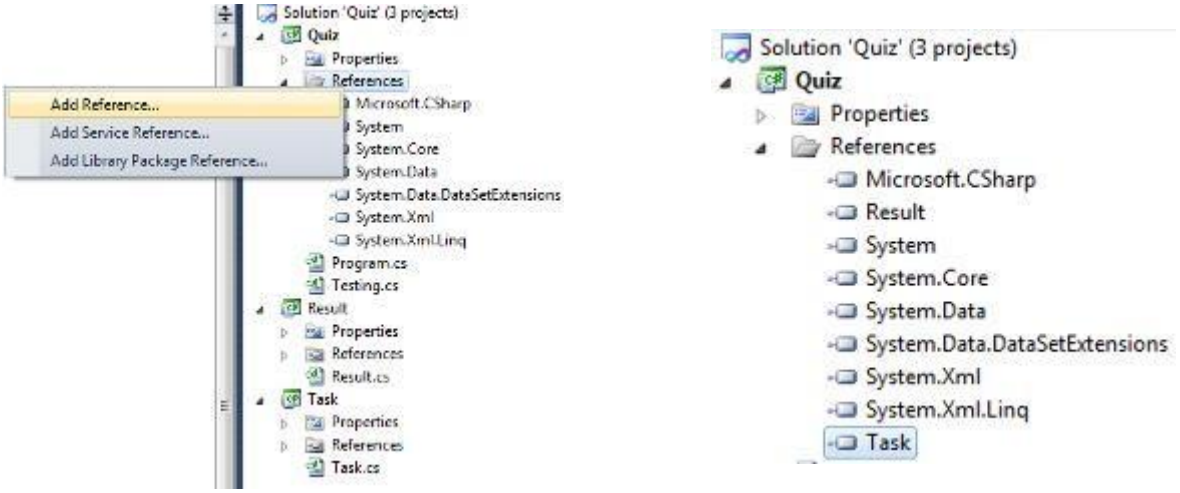
7. Модифицируйте файл Result.cs проекта Result следующим образом:

```
namespace Result
{
    public static class Result
    {
        public static List<string> Messages;

        // Статический конструктор
        static Result()
        {
            Messages = new List<string>()
            {
                "Вы набрали менее 10 баллов! Плохо!",
                "Вы набрали не менее 10 и менее 40 баллов! Нормально!",
                "Вы набрали не менее 40 баллов! Отлично"
            };
        }

        // Метод для возвращения сообщения
        public static string GetMessage(int Ball)
        {
            string mes = "";

            if (Ball < 10)
                mes = Messages[0];
            else if (Ball < 40)
                mes = Messages[1];
            else
                mes = Messages[2];

            return mes;
        }
    }
}
```

Рисунок 15.9 – Класс проекта Result.

Создан один статический файл, для представления результирующих сообщений.

8. Теперь модифицируем класс Tasting проекта Quiz. В своей работе класс будет использовать ранее созданные классы Task и Result, которые находятся в других проектах. Поэтому необходимо установит ссылки на данные проекты. Для этого вызовите команду (рис. 15.10а) контекстного меню папки References проекта Quiz. После этого ссылки на сборки (откомпилированные проекты Task и Result) появятся в списке ссылок Reference (рис. 15.10б).



а *б*

Рисунок 15.10 – Добавление ссылки на проекты: а – команда контекстного меню; б – список ссылок на проекты.

9. Модифицируем файл Testing.cs для реализации логики тестирования (рис. 15.11).

```
class Testing
{
    public string DoTesting()
    {
        int SumBal = 0;
        int ans = 0;

        foreach (Task.SingleTest p in Task.Task.AllQuestions)
        {
            Console.WriteLine("//////////////////////////");
            Console.WriteLine(p.Question);
            Console.WriteLine("//////////////////////////");
            Console.WriteLine("1. "+p.Answers[0]);
            Console.WriteLine("2. " + p.Answers[1]);
            Console.WriteLine("3. " + p.Answers[2]);
            Console.Write("Выберите номер ответа > ");
            ans = Convert.ToInt32(Console.ReadLine());
            SumBal += p.Balls[ans-1];
        }

        return Result.Result.GetMessage(SumBal);
    }
}
```

Рисунок 15.11 – Класс Testing.

10. Модифицируем функцию Main:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Quiz
{
    class Program
    {
        static void Main(string[] args)
        {
            Testing newTest = new Testing();

            Console.WriteLine("\n\n"+newTest.DoTesting());

            Console.ReadKey();
        }
    }
}
```

Рисунок 15.12 – Функция Main.

Запустите полученное приложение. Внимательно проанализируйте код приложения. Попробуйте изменить вопросы и их количество.

Затем, выполните индивидуальное задание.

Индивидуальное задание.

Спроектируйте многомодульное приложение (3 модуля), реализующее поставленную задачу. Перед выполнением приложения необходимо выполнить декомпозицию задачи и выявить основные объекты предметной области.

Вариант	Структура данных
1, 6, 11, 16, 21	Приложение осуществляет сложение обыкновенных дробей, то есть чисел $\frac{1}{3}$, $\frac{7}{12}$ и т.д. Требуется реализовать только сложение. Исходные слагаемые пользователь вводит с клавиатуры.
2, 7, 12, 17, 22	Приложение осуществляет умножение обыкновенных дробей, то есть чисел $\frac{1}{3}$, $\frac{7}{12}$ и т.д. Требуется реализовать только умножение. Исходные множители пользователь вводит с клавиатуры.

Вариант	Структура данных
3, 8, 13, 18, 23	Приложение осуществляет деление обыкновенных дробей, то есть чисел $\frac{1}{3}$, $\frac{7}{12}$ и т.д. Требуется реализовать только деление. Исходные дроби пользователь вводит с клавиатуры.
4, 9, 14, 19, 24	Реализуйте приложение, осуществляющее сокращение обыкновенной дроби. Дробь пользователь вводит с клавиатуры.
5, 10, 15, 20, 25	Реализуйте программу для выполнения операций над комплексными числами: сложение, умножение, вычисление модуля комплексного числа. Исходные комплексные числа пользователь вводит с клавиатуры.

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Что такое статический класс?

2. Что такое сборка?

3. Как определить проект по умолчанию в многомодульном решении?

4. Какими способами можно разместить в файлах определение класса?

5. Какой проект начнет выполняться первым если несколько из них в одном решении содержат функцию Main?

6. Что необходимо сделать для того, чтобы появилась возможность использовать классы, определенные в другом проекте?

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [1-3].

ТЕМА 4. ОСНОВЫ ООП: КЛАССЫ, КОНСТРУКТОРЫ, ПЕРЕГРУЗКА
ЛАБОРАТОРНАЯ РАБОТА 11. ОПЕРАЦИИ КЛАССОВ. ПЕРЕГРУЗКА ОПЕРАЦИЙ.

1. Цель и содержание

Цель лабораторной работы: научиться осуществлять перегрузку операторов относительно пользовательских типов.

Задачи лабораторной работы:

– изучить операции, подлежащие перегрузке;

– получить практические навыки перегрузки операторов.

2. Теоретическая часть

2.1 Операции. Большинство операций пришло в C# из языков C, C++:

Категория	Операции
Арифметические	+, -, *, /, %
Логические	&, |, ^, ~, &&, ||, !
Конкатенация строк	+
Инкремент и декремент	++, --
Сравнение	<, >, >=, >=, ==, !=
Присвоение	+=, -=, /=, *=, %=, =, &=, |=, ^=, <<=, >>=

Доступ к члену класса	.
Индексация	[ ]
Приведение	( )
Условная (тернарная) операция	? :
Создание объектов	new
Информация о типе	sizeof, is, as, typeof,
Контроль исключения, связанного с переполнением	checked, unchecked

2.2 Перегрузка операций относительно класса. Рассмотрим простой класс, например, для представления объекта «Вектор» (для простоты возьмем вектор на плоскости). Пример 1:

```
class Vector
{
    // Поля данных представляют стороны параллелепипеда
    public double X, Y;

    // Конструкторы
    public Vector()
    {

    }

    public Vector(double XCoord, double YCoord)
    {
        X = XCoord;
        Y = YCoord;
    }
}
```

Класс создан, но в математике можно выполнять сложение (вычитание) векторов и умножение вектора на число, а также находить скалярное произведение векторов. Необходимо добавить эту возможность для нашего класса «Вектор». Это значит, что операции, указанные в примере 2 должны иметь смысл.

Пример 2:

```
Vector a = new Vector(4, 10);
Vector b = new Vector(10, 15);

// вычисление суммы векторов
Vector SumVector = a + b;

// Вычисление произведения вектора на число
Vector MultNum = 2 * a;

// вычисление скалярного произведения
double Skalar = a * b;
```

В данной реализации (пример 1) такие операции в коде невозможны. Добавим в класс перегруженные операции.

Пример 3:

```
class Vector
{
    // Поля данных представляют стороны параллелепипеда
    public double X, Y;

    // Конструкторы
    public Vector()
    {

    }

    public Vector(double XCoord, double YCoord)
    {
        X = XCoord;
        Y = YCoord;
    }

    // Перегрузка операции сложения
    public static Vector operator +(Vector left, Vector right)
    {
        return new Vector(left.X + right.X, left.Y + right.Y);
    }

    // Перегрузка операции умножения вектора на число
    public static Vector operator *(double k, Vector vek)
    {
        return new Vector(k*vek.X, k*vek.Y);
    }

    // Перегрузка операции умножения двух векторов (скалярное умножение)
    public static double operator *(Vector left, Vector right)
    {
        return left.X * right.X + left.Y * right.Y;
    }

}
```

Перегрузка операции похожа на объявление метода класса, но существуют отличия: используется ключевое слово static; используется ключевое слово operator и символ операции (+, *, - и т.д.) вместо имени метода.

После того как операции перегружены, возможно выполять следующие действия:

```
class Program
{
    static void Main(string[] args)
    {
        Vector a = new Vector(4, 10);
        Vector b = new Vector(10, 15);

        // вычисление суммы векторов
        Vector SumVector = a + b;

        // Вычисление произведения вектора на число
        Vector MultNum = 2 * a;

        // вычисление скалярного произведения
        double Skalar = a * b;

        // Проведение сложных расчетов
        Vector RES = 2 * a + 3 * b + (a * b) * a;


        Console.ReadKey();
    }
}
```

3. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку

и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения работы

1. Создайте консольное приложение.

2. Совместно с преподавателем спроектируйте и разработайте класс, относительно которого перегрузите операции +, - /, *, %, ==, > (каждый студент разрабатывает свой класс).

Индивидуальные задания (возможные варианты классов):

«Вектор в пространстве», «Матрица», «Сотрудник», «Компьютер», «Товар», «Объект недвижимости», «Комплексное число», «Куб», «Автомобиль», «Студент», «Книга», «Дисциплина (предмет)».

Возможно предложение своего класса или доработка класса из лабораторной работы №13.

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Что такое конструктор?

2. Что такое перегрузка операторов? Когда применяется данный механизм?

3. Допустим, что для класса создана перегруженная операция сложения. Может ли быть создана еще одна операция сложения для данного класса?

4. Какие операции нельзя перегрузить?

5. Какие операции необходимо перегружать попарно?

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [3], [5].

ТЕМА 4. ОСНОВЫ ООП: КЛАССЫ, КОНСТРУКТОРЫ, ПЕРЕГРУЗКА
ЛАБОРАТОРНАЯ РАБОТА 12. ПОСТРОЕНИЕ ИЕРАРХИИ КЛАССОВ

1. Цель и содержание

Цель лабораторной работы: изучить механизм организации наследования классов.

Задачи лабораторной работы:

– научиться объявлять производные классы;

– научиться создавать иерархии классов;

– научиться использовать механизм полиморфизма.

2. Теоретическая часть

2.1 Наследование реализации. Наследование реализации (implementation inheritance) означает, что тип происходит от базового типа, получая от него все поля-члены и функции-члены.

Синтаксис наследования реализации:

```
class ПроизводныйКласс : БазовыйКласс
{
    // Данные– членыи функции – члены
}
```

Если при определении класса не указан базовый класс, то C# предполагает, что базовым классом является System.Object.

2.2 Создание иерархии классов. При наследовании реализации производный класс наследует реализацию каждой функции базового типа, если только в его определении не указано, что реализация функции должна быть переопределена.

Определим следующую иерархию классов (рис. 17.1) и продемонстрируем, как **наследуется реализация** и как **переопределяются** свойств и методов.

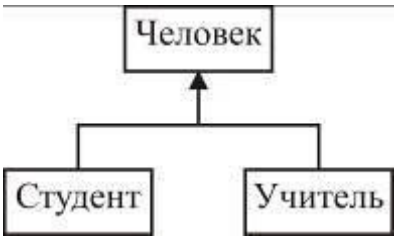


Рисунок 17.1 – Иерархия классов.

Создадим код, описывающий данную иерархию. Определим базовый класс:

```
class Человек
{
    public Человек(string Ф, string И, string О, int Возр)
    {
        Фамилия = Ф; Имя = И; Отчество = О; Возраст = Возр;
    }

    public Человек()
    {
        Фамилия = "нет данных"; Имя = ""; Отчество = "";
        Возраст = 18;
    }

    public string Фамилия, Имя, Отчество;
    private DateTime ДатаРождения;

    public virtual string ФИО
    {
        get { return Фамилия + " " + Имя + " " + Отчество; }
    }

    public int Возраст
    {
        get { return DateTime.Now.Year - ДатаРождения.Year; }
        set
        {
            int ГодРождения = DateTime.Now.Year - value;

            ДатаРождения = Convert.ToDateTime(ГодРождения.ToString()+".01.01");
        }
    }
}
```

Обратите внимание на наличие двух конструкторов, механизм хранения возраста человека и ключевое слово virtual у свойства «ФИО». Ключевое слово virtual указывает, что данное свойство будет переопределено в производном классе.

Определим производный класс «Учитель»:

```
class Учитель: Человек
{
    public Учитель()
        :base()
    {
        УченоеЗвание = УченыеЗвания.Без_Звания;
        УченаяСтепень = УченыеСтепени.Без_Степени;
    }

    public Учитель(string Ф, string И, string О, int Возр, УченыеЗвания УЗ, УченыеСтепени УС)
        :base(Ф, И, О, Возр)
    {
        УченоеЗвание = УЗ;
        УченаяСтепень = УС;
    }

    public УченыеЗвания УченоеЗвание;
    public УченыеСтепени УченаяСтепень;

    public override string ФИО
    {
        get
        {
            return УченаяСтепень.ToString() + ", " + УченоеЗвание.ToString() + ", " + base.ФИО;
        }
    }
}
```

Следует обратить внимание на использование ключевого слова override у свойства «ФИО», которое указывает на то, что данное свойство имеет новую реализацию, отличающуюся от реализации базового класса.

Определим второй производный класс «Студент»:

```
class Студент: Человек
{
    public Студент(string Ф, string И, string О, int Возр, Специальности Спец)
        :base(Ф, И, О, Возр)
    {
        Специальность = Спец;
    }

    public Специальности Специальность;

    public override string ФИО
    {
        get
        {
            return base.ФИО + ", " + Специальность.ToString();
        }
    }
}
```

В обоих производных классах следует обратить внимание на реализацию конструкторов производных классов, использование ключевого слова base в коде свойства «ФИО» и при объявлении конструкторов.

Также интерес представляют типы данных, используемые для членов-данных: «Специальности», «УченыеЗвания», «УченыеСтепени». Вот определения данных типов-перечислений:

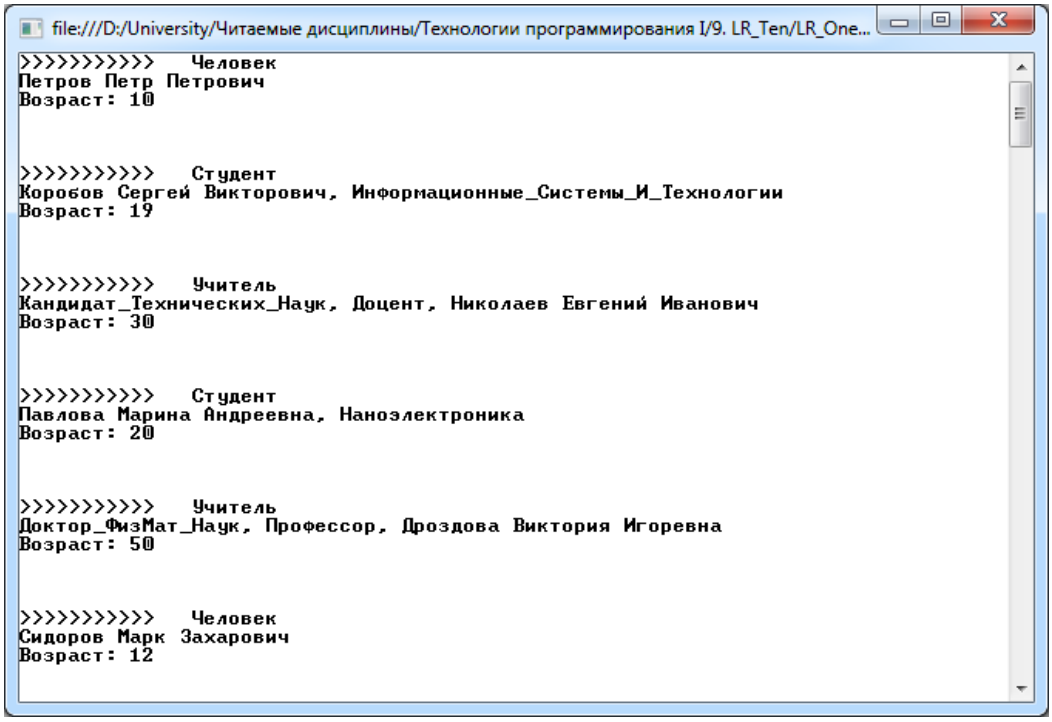
```
public enum УченыеЗвания
{
    Доцент,
    Профессор,
    Академик,
    Без_Звания
}

public enum УченыеСтепени
{
    Кандидат_Технических_Наук,
    Кандидат_ФизМат_Наук,
    Кандидат_Педагогических_Наук,
    Доктор_ФизМат_Наук,
    Без_Степени
}

public enum Специальности
{
    Информационные_Системы_И_Технологии,
    Безопасность_Информационных_Систем,
    Технология_Защиты_Информации,
    Психология,
    Наноэлектроника
}
```

Наконец, продемонстрируем использование объявленной иерархии классов.

В программе объявим массив объектов типа «Человек». Следует понимать, что при объявлении такого массива, его элементам можно присваивать объекты любого производного класса, причем каждый объект будет вести себя по-своему (за счет переопределения свойств и методов).

```
static void Main(string[] args)
{
    // Объявим массив людей:
    Человек[] mas = new Человек[6];

    // Заполним массив значениями разных (!) типов
    mas[0] = new Человек("Петров", "Петр", "Петрович", 10);

    mas[1] = new Студент("Коробов", "Сергей", "Викторович",
        19,
        Специальности.Информационные_Системы_И_Технологии);

    mas[2] = new Учитель("Николаев", "Евгений", "Иванович",
        30,
        УченыеЗвания.Доцент,
        УченыеСтепени.Кандидат_Технических_Наук);

    mas[3] = new Студент("Павлова", "Марина", "Андреевна",
        20,
        Специальности.Наноэлектроника);

    mas[4] = new Учитель("Дроздова", "Виктория", "Игоревна",
        50,
        УченыеЗвания.Профессор,
        УченыеСтепени.Доктор_ФизМат_Наук);

    mas[5] = new Человек("Сидоров", "Марк", "Захарович", 12);

    // Вывод инфорации
    for (int i = 0; i < mas.Length; i++)
    {
        Console.WriteLine(">>>>>>>>>>>    " + mas[i].GetType().Name);
        Console.WriteLine(mas[i].ФИО);
        Console.WriteLine("Возраст: " + mas[i].Возраст.ToString()+"\n");
        Console.WriteLine("\n");
    }

        Console.ReadKey();
}
```

В результате работы программы будет осуществлен следующий вывод:



Полная диаграмма типов в полученном приложении показана на рис. 17.2.

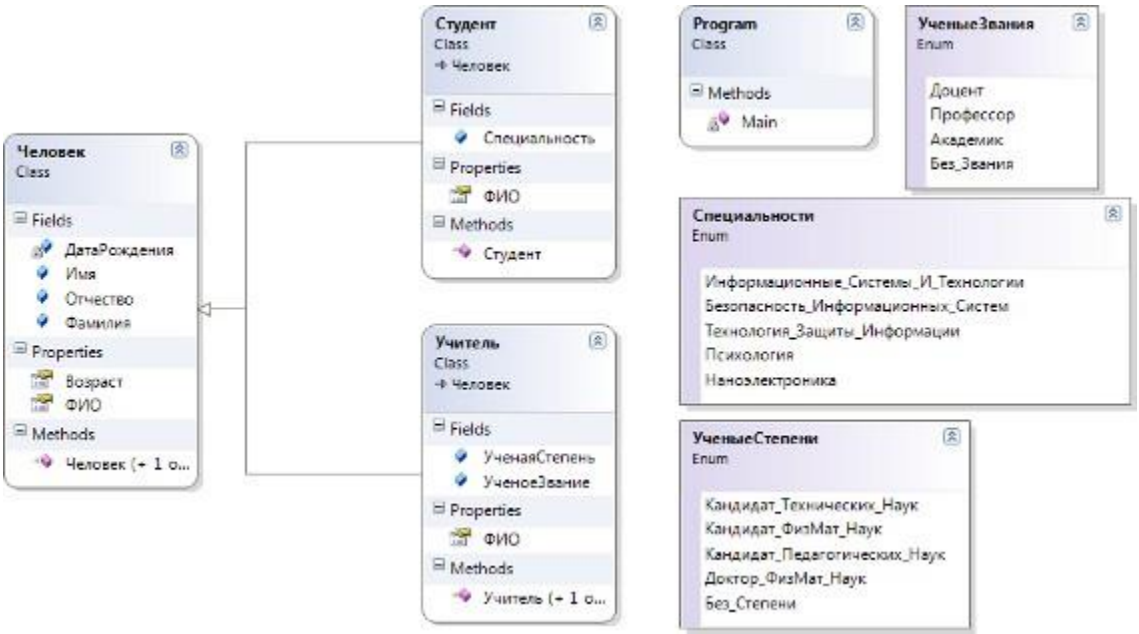


Рисунок 17.2 – Диаграмма типов приложения (создана средствами VS).

Структуры всегда наследуются от System.ValueType. Они могут также наследовать любое количество интерфейсов. Классы всегда наследуются от

одного класса по вашему выбору. Они также могут наследовать любое количество интерфейсов.

3. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе №1.

2. Изучите пример создания иерархии классов, представленный в разделе «Теоретическое обоснование» данной лабораторной работы.

3. Постройте свою иерархию классов в соответствии с индивидуальным заданием. В результате выполнения лабораторной работы должны быть реализованы следующие механизмы:

– использование типа-перечисления (хотя бы одного);

– использование переопределенного свойства (хотя бы одного);

– использование переопределенного метода (хотя бы одного);

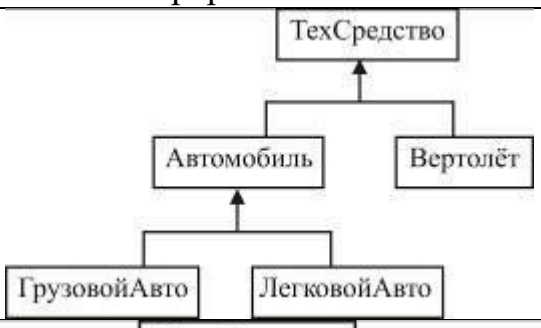
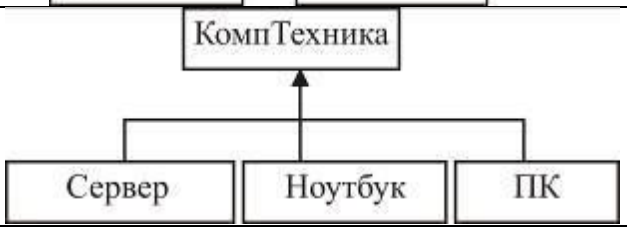
– использование вызова базового конструктора;

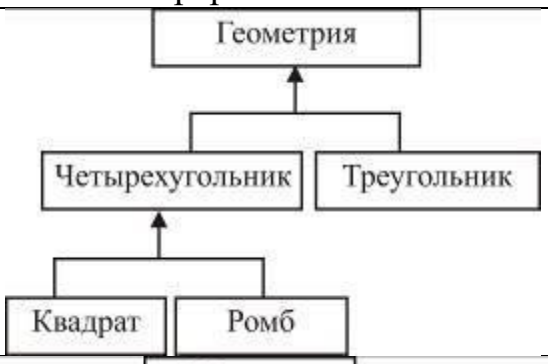
– использование вызова любого базового метода (отличного от конструктора).

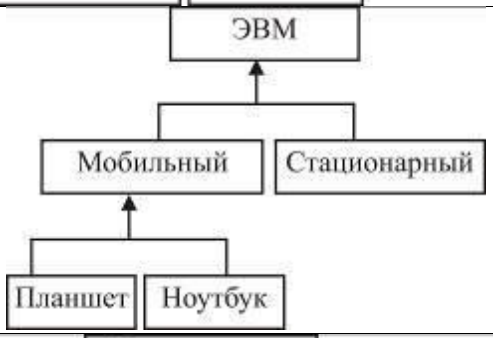
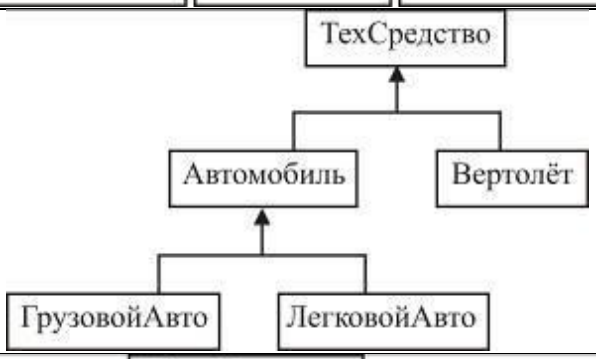
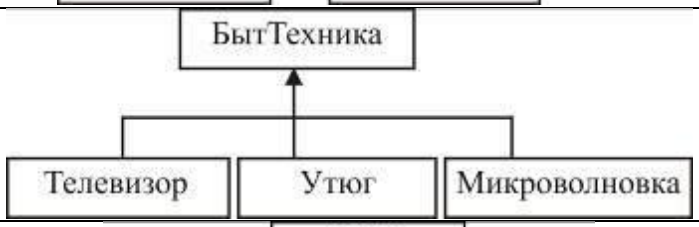
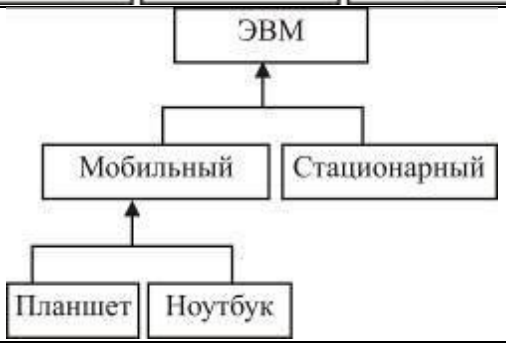
4. Продемонстрируйте использование классов, созданной иерархии (легче всего это сделать с использованием массивов). При защите работы укажите признаки присутствия полиморфного поведения в программе (реализация полиморфизма).

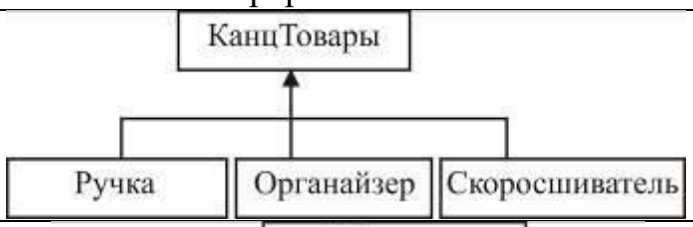
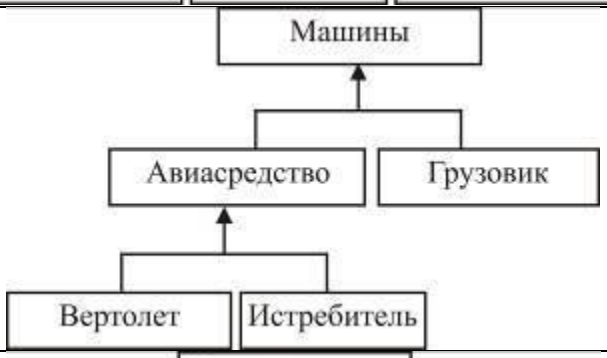
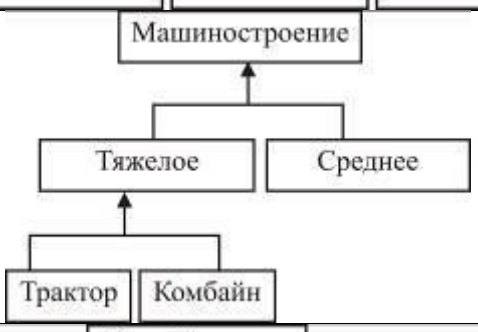
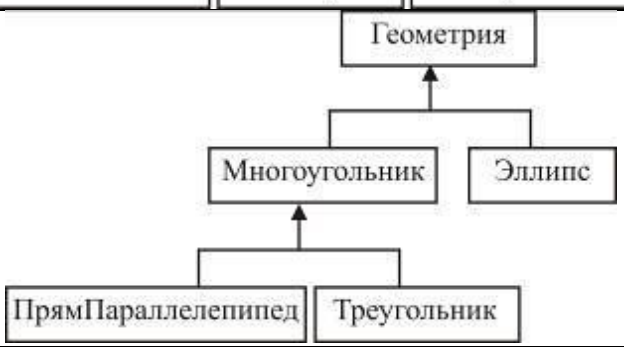
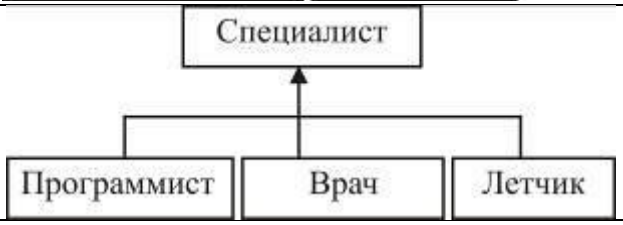
Индивидуальное задание.

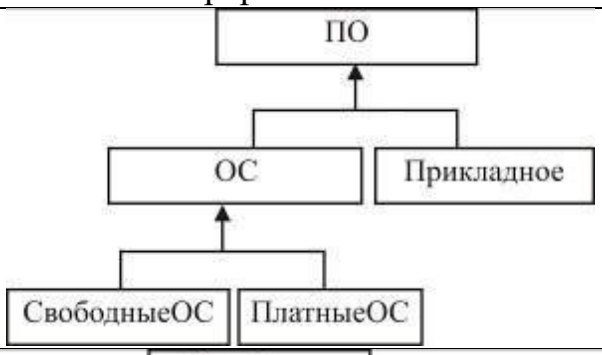
Спроектируйте класс, наполните его требуемой функциональностью, продемонстрируйте работоспособность класса.

Вариант	Иерархия классов
1	
2	

Вариант	Иерархия классов
3	
4	
5	
6	
7	
8	

Вариант	Иерархия классов
9	
10	
11	
12	
13	
14	

Вариант	Иерархия классов
15	
16	
17	
18	
19	
20	
21	

Вариант	Иерархия классов
22	
23	
24	
25	

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Что такое наследование реализации? Как описать синтаксически наследование реализации?

2. Для чего используется ключевое слово base?

3. Можно ли переопределить метод класса? Свойства класса? Данные класса?

4. Как переопределить метод в производном классе?

5. Для чего используется ключевое слово virtual?

6. Для чего используется ключевое слово override?

7. Как поменять цвет фона в консольном приложении?

8. Как построить диаграмму типов данных, используемых в приложении средствами Visual Studio 2008/ 2010?

9. Сколько базовых классов может быть у любого класса в C#?

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [5-7].

ТЕМА 5. ПРОДВИНУТОЕ ООП: МОДУЛЬНОСТЬ И ИНТЕРФЕЙСЫ ЛАБОРАТОРНАЯ РАБОТА 13. РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ

1. Цель и содержание

Цель лабораторной работы: изучить принципы работы с типами интерфейсов в C#.

Задачи лабораторной работы:

– научиться объявлять интерфейсы в C#;

– научиться создавать классы, реализующие интерфейсы;

2. Теоретическая часть

2.1 Наследование интерфейсов. Кроме наследования реализации (implementation inheritance) в языке C# реализован механизм наследования интерфейсов (interface inheritance). Необходимо понимать, что не все объектно-ориентированные языки поддерживают интерфейсы.

Если класс наследует интерфейс, класс как бы берет на себя обязательства реализовать некоторый функционал.

Синтаксис наследования интерфейса не отличается от синтаксиса наследования реализации:

class *КлассРеализующийИнтерфейс* : *Интерфейс*
{
    // *Данные– членыи функции – члены*
}

Отличие от механизма наследования реализации состоит в том, что:

- класс обязательно должен реализовать методы и свойства, продекларированные в интерфейсе (никакой реализации в интерфейсе не существует);

- класс может реализовывать (наследовать) несколько интерфейсов, тогда как базовый класс может быть только один.

2.2 Объявление интерфейсов. Для объявления типа интерфейса используется ключевое слово interface:

```
public interface ICalculate
{
    void Plus(int pPlus);
    void Minus(int pMinus);
}

public interface IVisual
{
    string Name { get; set; }
    void DrawObject();
}
```

В приведенном примере объявлены два интерфейса: ICalculate и IVisual.

Интерфейс ICalculate включает два метода с одним параметром. Из названий видно, что один метод что-то увеличивает, второй – что-то уменьшает на величину параметра.

Второй интерфейс IVisual содержит метод DrawObject без параметров, который призван нарисовать объект, а также свойство Name, которое доступно как для чтения, так и для записи.

Следует понимать, что интерфейсы не предполагают конкретную реализацию методов, а свойства интерфейсов не хранят данных. Интерфейс

должен быть реализован классом, который и определит конкретное наполнение методов и свойств интерфейса.

Для использования объявленных интерфейсов создадим два класса Human (человек) и Car (автомобиль).

```
public class Human
{
    public Human(string pFIO, int pAge)
    {
        FIO = pFIO;
        Age = pAge;
    }

    private string FIO;
    private int Age;
}

public class Car
{
    public Car(string pManufacturer, string pModel, int pVelocity)
    {
        Manufacturer = pManufacturer;
        Model = pModel;
        Velocity = pVelocity;
    }

    private string Manufacturer;
    private string Model;
    private int Velocity;
}
```

В каждом классе определен конструктор, который инициализирует закрытые данные-члены класса.

Требуется для каждого класса реализовать вывод в консоль, а также возможности увеличения скорости автомобиля и возможности изменения возраста у человека.

Этот общий функционал можно реализовать с использованием следующих способов:

1. Добавить все необходимые функции в каждый класс независимо. Этот метод приводит к «разбуханию кода», сложной управляемости кода.

2. Реализовать один класс, например Base (базовый), в котором определить общие методы и свойства (Plus, Minus, DrawObject). Затем использовать класс Base в качестве базового для классов Human и Car, причем в каждом классе переопределить методы базового класса с использованием

ключевого слова override. Этот метод – наследование реализации. Получается иерархия классов, которые по смыслу и наполнению сильно отличаются. Хотя в данном подходе классы и код более упорядочены, все равно возникает избыточность за счет многократного переопределения методов.

3. Определение интерфейсов и реализация возможностей интерфейсов данными классами.

Именно метод 3 будет использован в данной лабораторной работе.

4. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

5. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

6. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе №1.

2. Определите в приложении классы Human и Car, а также интерфейсы ICalculate и IVisual, представленные в разделе «Теоретическое обоснование» данной лабораторной работы.

3. Реализуем механизм наследования интерфейса ICalculate классом Human. Для этого выполним следующие действия:

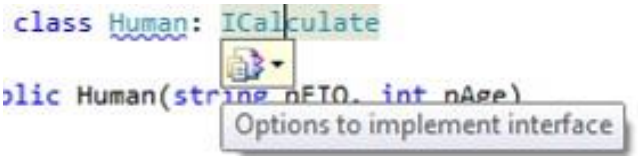
3.1. В определении класса укажем, что класс Human наследует интерфейс ICalculate.

```
public class Human: ICalculate
```

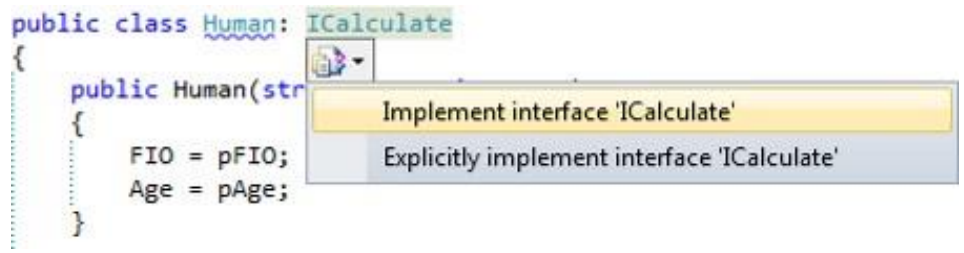
3.2. Обратите внимание, что после этого попытка перекомпилировать проект приведет к ошибкам:

То есть компилятор сообщает, что интерфейс наследуется классом, но методы, заявленные в интерфейсе, классом не реализованы. Реализуем их.

3.3. Наведите курсор мыши на интерактивное подчеркивание и появится выпадающий список:



Раскройте его и выберите команду «Implement interface» (реализовать интерфейс).



3.4. В классе Human появятся два новых метода, как и было объявлено в интерфейсе ICalculate. Определение класса примет вид:

```
public class Human: ICalculate
{
    public Human(string pFIO, int pAge)
    {
        FIO = pFIO;
        Age = pAge;
    }

    private string FIO;
    private int Age;

    public void Plus(int pPlus)
    {
        throw new NotImplementedException();
    }

    public void Minus(int pMinus)
    {
        throw new NotImplementedException();
    }
}
```

Обратите внимание, что в каждый сгенерированный метод среда разработки добавила код вызова исключения, то есть проект скомпилируется без ошибок но в процессе выполнения, при попытке использовать методы Plus или Minus программа завершится с ошибками. Это сделано для того, чтобы программист не забыл реализовать данные методы интерфейсов.

В процессе выполнения пп. 3.1 – 3.3 были использованы возможности Visual Studio по автоматизации реализации интерфейса. Очевидно, что интерфейс можно было реализовать, самостоятельно написав данный код.

– использование вызова базового конструктора;

– использование вызова любого базового метода (отличного от конструктора).

4. Определим окончательную реализацию для методов Plus и Minus:

```
public void Plus(int pPlus)
{
    Age += pPlus;
}

public void Minus(int pMinus)
{
    Age -= pMinus;
}
```

5. Аналогичным образом реализуем наследование интерфейса IVisual для класса Human. Окончательно для класса Human получим:

```
public class Human: ICalculate, IVisual
{
    public Human(string pFIO, int pAge)
    {
        FIO = pFIO;
        Age = pAge;
    }

    private string FIO;
    private int Age;

    public void Plus(int pPlus)
    {
        Age += pPlus;
    }

    public void Minus(int pMinus)
    {
        Age -= pMinus;
    }

    public string Name
    {
        get
        {
            return FIO + " : " +Age.ToString();
        }
        set
        {
            FIO = value;
        }
    }

    public void DrawObject()
    {
        Console.WriteLine
            (
                "    o    \n" +
                "  -----  \n" +
                "    |    \n" +
                "   / \\   \n" +
                "  /   \\  \n"
            );
        Console.WriteLine(Name);
    }
}
```

6. Реализуем интерфейсы ICalculate и IVisual для класса Car:

```
public class Car: IVisual, ICalculate
{
    public Car(string pManufacturer, string pModel, int pVelocity)
    {
        Manufacturer = pManufacturer;
        Model = pModel;
        Velocity = pVelocity;
    }

    private string Manufacturer;
    private string Model;
    private int Velocity;

    public void Plus(int pPlus)
    {
        Velocity += pPlus;
    }

    public void Minus(int pMinus)
    {
        Velocity += pMinus;
    }

    public string Name
    {
        get
        {
            return Manufacturer + " - " + Model +
                " : " + Velocity.ToString() + "km/h";
        }
        set
        {
            Model = value;
        }
    }

    public void DrawObject()
    {
        Console.WriteLine
            (
                "     -----------\n" +
                "____/           \\_____\n" +
                "|                     |\n" +
                "---(@)-----------(@)--- \n"
            );
        Console.WriteLine(Name);
    }
}
```

7. Когда все классы и интерфейсы определены и реализованы можно их использовать. Продемонстрируем использование типов данных созданием соответствующих объектов в функции main.

```
static void Main(string[] args)
{
    Console.BackgroundColor = ConsoleColor.White;
    Console.ForegroundColor = ConsoleColor.Blue;
    Console.Clear();

    Console.Title = "Лабораторная работа №11";

    Human h = new Human("Иванов Иван Иванович", 50);
    h.Plus(5);
    h.Minus(1);
    h.DrawObject();

    Console.WriteLine("\n\n\n");

    Car car = new Car("Hyundai", "ix35", 120);
    car.Plus(25);
    car.Minus(11);
    car.DrawObject();

    Console.ReadKey();
}
```

8. Вид окна разработанного приложения представлен на рис. 18.1:

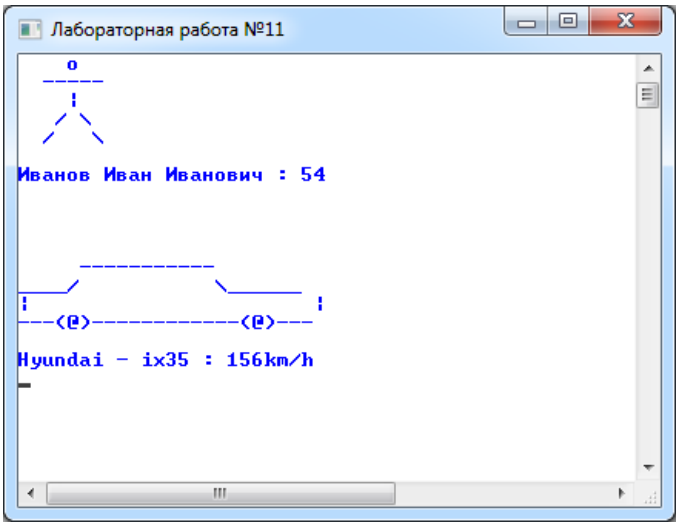


Рисунок 18.1 – Консольное приложение на основе интерфейсных типов.

Диаграмма типов данных, созданная редактором Visual Studio, для разработанного приложения, показана на рис. 18.2 (сравните ее с диаграммой на рис. 17.2 в лабораторной работе №17):

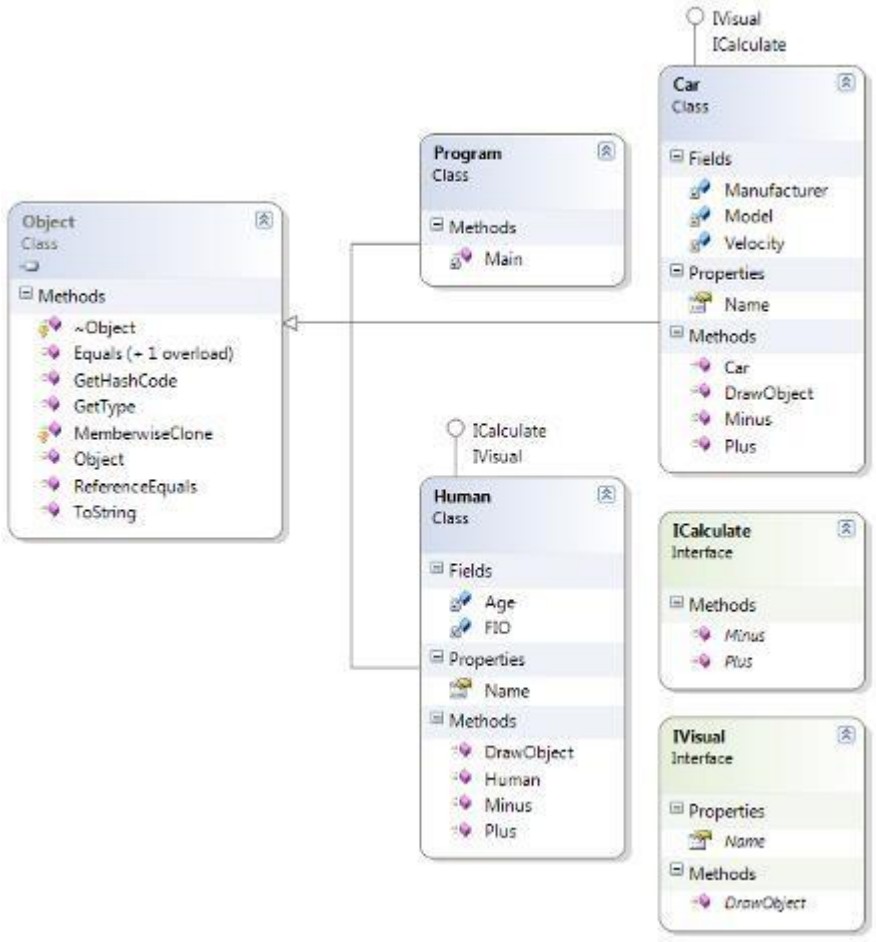


Рисунок 18.2 – Диаграмма классов приложения.

Индивидуальное задание.

Спроектируйте классы, наполните их требуемой функциональностью, определите интерфейс, продемонстрируйте использование объектов класса и методов интерфейса. Постройте диаграмму классов своего приложения средствами Visual Studio.

В качестве классов можно использовать иерархию классов, разработанную в лабораторной работе №17. Интерфейс спроектируйте и реализуйте самостоятельно.

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Что такое наследование реализации? Как описать синтаксически наследование реализации?

2. Для чего используется ключевое слово base?

3. Можно ли переопределить метод класса? Свойства класса? Данные класса?

4. Что такое наследование интерфейса? Укажите основные отличия от наследования реализации.

5. Для чего используется ключевое слово virtual? Для чего используется ключевое слово override?

6. Может ли один класс наследовать несколько классов? Несколько интерфейсов?

7. Внимательно изучите код примеров данной лабораторной работы и подумайте, каким образом специфицируются методы интерфейса public, private или protected? В чем причина применения именно такого модификатора доступа?

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [6].

ТЕМА 5. ПРОДВИНУТОЕ ООП: МОДУЛЬНОСТЬ И ИНТЕРФЕЙСЫ ЛАБОРАТОРНАЯ РАБОТА 14. ФАЙЛОВЫЙ ВВОД-ВЫВОД. РАБОТА С КАТАЛОГАМИ. РАБОТА С ФАЙЛАМИ.

1. Цель и содержание

Цель лабораторной работы: научиться использовать механизмы файлового ввода-вывода.

Задачи лабораторной работы:

– научиться применять классы для работы с файлами;

– научиться применять классы для работы с каталогами;

– научиться использовать потоки ввода-вывода.

2. Теоретическая часть

2.1 Классы .NET Framework для реализации операций ввода-вывода.

Весь ввод и вывод в .NET Framework подразумевает использование потоков. Поток – это абстрактное представление последовательного устройства. Последовательное устройство – это нечто такое, что хранит данные в линейной структуре и точно таким же образом обеспечивает доступ к ним: считывает или записывает по одному байту за одну единицу времени.

Сохранение устройства абстрактным означает, что лежащие в основе источник/приемник данных могут быть скрыты. Такой уровень абстракции обеспечивает повторное использование кода и позволяет писать более обобщенные процедуры, потому что нет необходимости заботиться о действительной специфике передачи данных.

Для обработки файлов в C# необходима ссылка на пространство имен System.IO. При открытии файла создается объект, с которым ассоциируется поток. Потоки обеспечивают механизмы связи между файлами и программами.

Среда .NET Framework предоставляет все необходимые инструменты для эффективного использования файлов в приложениях.

Основные классы, необходимые программисту:

1. Object – исходный базовый класс для всех классов платформы .NET Framework и корень иерархии типов.

2. File – статический служебный класс, предоставляющий множество статических методов, которые дают возможность перемещать, создавать, копировать, удалять файлы, опрашивать и обновлять атрибуты, а также создавать объекты потоков FileStream.

3. Directory – статический служебный класс, предоставляющий множество статических методов для перемещения, копирования и удаления каталогов.

4. Path – служебный класс, используемый для манипулирования путевыми именами.

5. MarshalByRefObject – разрешает доступ к объектам через границы доменов приложения в приложениях, поддерживающих удаленное взаимодействие, это базовый класс для всех классов .NET, позволяющих удаленное взаимодействие.

6. FileInfo – представляет физический файл на диске, имеет методы для манипулирования этим файлом. Для любого объекта, который читает или пишет в этот файл, должен быть создан объект Stream. Все методы FileInfo доступны из объектной переменной, поэтому, если необходимо выполнить

только одно действие, более эффективным может оказаться использование метода File, а не соответствующего экземпляра метода FileInfo. Для всех методов FileInfo требуется путь к файлу, с которым проводится операция. Все статические методы класса FileInfo выполняют проверку безопасности для всех методов. Если необходимо использовать объект неоднократно, рекомендуется использовать соответствующий метод экземпляра FileInfo, поскольку в этом случае проверка безопасности будет требоваться не всегда.

7. DirectoryInfo – представляет физический каталог на диске и предоставляет методы уровня экземпляра для манипулирования каталогом. Класс DirectoryInfo работает точно так же, как класс FileInfo. Это объект, представляющий отдельный каталог на машине. Подобно классу FileInfo, многие из вызовов методов дублируются между Directory и DirectoryInfo.

8. FileSystemInfo – служит базовым классом для FileInfo и DirectoryInfo, обеспечивая возможность работы с файлами и каталогами одновременно, используя полиморфизм.

9. Stream – предоставляет универсальное представление последовательности байтов. Класс Stream является абстрактным базовым классом всех потоков.

10. FileStream – представляет файл, который может быть записан, прочитан или то и другое.

11. TextReader – представляет средство чтения, позволяющее считывать последовательные наборы знаков. Этот класс является абстрактным базовым классом для StreamReader, который считывает символы из потоков.

12. TextWriter – представляет средство записи, позволяющее записывать последовательные наборы символов. Этот класс является абстрактным базовым классом для StreamWriter, который записывают символы в потоки.

13. StreamReader – читает символьные данные из потока и может быть создан с использованием класса FileStream в качестве базового.

14. StreamWriter – пишет символьные данные в поток и может быть создан с использованием класса FileStream в качестве базового.

15. Component – предоставляет базовую реализацию интерфейса IComponent и делает возможным совместное использование объектов разными приложениями.

16. FileSystemWatcher – используется для мониторинга файлов и каталогов и представляет события, которые приложение может перехватить, когда в этих объектах происходят какие-то изменения.

Таким образом, эта система классов включает в себя классы для работы с файлами (File, FileInfo), каталогами (Directory, DirectoryInfo) и потоками (FileStream, StreamReader, StreamWriter).

В большинстве случаев для разработки бизнес-приложений достаточно лишь четырех классов для манипулирования файловой системой. Эти классы расположены пространстве имен System.IO и предназначены для работы с файловой системой компьютера, то есть для создания, удаления переноса файлов и каталогов.

Первые два типа – Directory и File реализуют свои возможности с помощью статических методов, поэтому данные классы можно использовать без создания соответствующих объектов (экземпляров классов).

Следующие типы – DirectoryInfo и FileInfo обладают схожими функциональными возможностями с Directory и File, но порождены от класса FileSystemInfo, поэтому реализуются путем создания соответствующих экземпляров классов.

Класс FileSystemInfo предоставляет базовый функционал. Значительная часть членов FileSystemInfo предназначена для работы с общими характеристиками файла или каталога (метками времени, атрибутами и т. п.). Рассмотрим некоторые свойства FileSystemInfo (таблица 19.1).

Таблица 19.1 – Свойства класса FileSystemInfo.

Свойство	Описание
Attributes	Позволяет получить или установить атрибуты для данного объекта файловой системы. Для этого свойства используются значения и перечисления FileAttributes

Свойство	Описание
CreationTime	Позволяет получить или установить время создания объекта файловой системы
Exists	Может быть использовано для того, чтобы определить, существует ли данный объект файловой системы
Extension	Позволяет получить расширение для файла
FullName	Возвращает имя файла или каталога с указанием пути к нему в файловой системе
LastAccessTime	Позволяет получить или установить время последнего обращения к объекту файловой системы
LastWriteTime	Позволяет получить или установить время последнего внесения изменений в объект файловой системы
Name	Возвращает имя указанного файла. Это свойство доступно только для чтения. Для каталогов возвращает имя последнего каталога в иерархии, если это возможно. Если нет, возвращает полностью определенное имя

В FileSystemInfo предусмотрен набор методов. Например, метод Delete() – позволяет удалить объект файловой системы с жесткого диска, а Refresh() – обновить информацию об объекте файловой системы.

2.2 Классы для работы с каталогами файловой системы.

Класс DirectoryInfo наследует члены класса FileSystemInfo и содержит дополнительный набор членов, которые предназначены для создания, перемещения, удаления, получения информации о каталогах и подкаталогах в файловой системе. Наиболее важные члены класса содержатся в таблице 2.2.

Таблица 19.2 – Доступные члены класса DirectoryInfo.

Член	Описание
Create() CreateSubDirectory()	Создают каталог (или подкаталог) по указанному пути в файловой системе
Delete()	Удаляет пустой каталог
GetDirectories()	Позволяет получить доступ к подкаталогам текущего каталога (в виде массива объектов DirectoryInfo)
GetFiles()	Позволяет получить доступ к файлам текущего каталога

Член	Описание
	(в виде массива объектов FileInfo)
MoveTo()	Перемещает каталог и все его содержимое на новый адрес в файловой системе
Parent	Возвращает родительский каталог в иерархии файловой системы

Работа с типом DirectoryInfo начинается с того, что создается экземпляр класса (объект), при вызове конструктора в качестве параметра указывается путь к нужному каталогу. Если необходимо обратиться к текущему каталогу (то есть каталогу, в котором в настоящее время производится выполнение приложения), вместо параметра используется обозначение ".". Например:

```
// Создаем объект DirectoryInfo,
// которому будет обращаться к текущему каталогу
DirectoryInfo dir1 = new DirectoryInfo(".");

// Создаем объект DirectoryInfo,
// которому будет обращаться к каталогу d:\prim
DirectoryInfo dir2 = new DirectoryInfo(@"d:\prim");
```

Если создается объект DirectoryInfo, который связывается с несуществующим каталогом, то будет сгенерировано исключение System.IO.DirectoryNotFoundException.

Свойство Attributes класса DirectoryInfo позволяет получить информацию об атрибутах объекта файловой системы. Возможные значения данного свойства приведены в следующей таблице 19.3.

Таблица 19.3 – Значения свойства Attributes.

Значение	Описание
Archive	Этот атрибут используется приложениями при проведении резервного копирования, а в некоторых случаях - удаления старых файлов
Compressed	Определяет, что файл является сжатым
Directory	Определяет, что объект файловой системы является каталогом
Encrypted	Определяет, что файл является зашифрованным
Hidden	Определяет, что файл является скрытым (такой файл не будет выводиться при обычном просмотре каталога)
Normal	Определяет, что файл находится в обычном состоянии и для него установлены любые другие атрибуты. Этот атрибут не может

Значение	Описание
	использоваться с другими атрибутами
Offline	Файл (расположенный на сервере) кэширован в хранилище off-line на клиентском компьютере. Возможно, что данные этого файла уже устарели
Readonly	Файл доступен только для чтения
System	Файл является системным (то есть файл является частью операционной системы или используется исключительно операционной системой)

Через класс DirectoryInfo программист может собрать информацию о дочерних подкаталогах. Например:

```
static void printDirect(DirectoryInfo dir)
{
    Console.WriteLine("***** " + dir.Name + " *****");
    Console.WriteLine("FullName: {0}", dir.FullName);
    Console.WriteLine("Name: {0}", dir.Name);
    Console.WriteLine("Parent: {0}", dir.Parent);
    Console.WriteLine("Creation: {0}", dir.CreationTime);
    Console.WriteLine("Attributes: {0}", dir.Attributes.ToString());
    Console.WriteLine("Root: {0}", dir.Root);
}

static void Main(string[] args)
{
    DirectoryInfo dir = new DirectoryInfo(@"d:\prim");
    printDirect(dir);
    DirectoryInfo[] subDirects = dir.GetDirectories();
    Console.WriteLine("Найдено {0} подкаталогов", subDirects.Length)
    foreach (DirectoryInfo d in subDirects)
    {
        printDirect(d);
    }
}
```

Метод CreateSubdirectory() позволяет создать в выбранном каталоге как единственный подкаталог, так и множество подкаталогов (в том числе, и вложенных друг в друга). Например:

```
DirectoryInfo dir = new DirectoryInfo(@"d:\prim");

//создали подкаталог
dir.CreateSubdirectory("doc");

//создали вложенный подкаталог
dir.CreateSubdirectory(@"book\2008");
```

Метод MoveTo() позволяет переместить текущий каталог по заданному в качестве параметра адресу. При этом возможно произвести переименование каталога. Например:

```
DirectoryInfo dir = new DirectoryInfo(@"d:\prim\bmp");
dir.MoveTo(@"d:\prim\letter\Николаев");
```

В данном случае каталог «bmp» перемещается по адресу «d:\prim\letter\Николаев». Так как имя перемещаемого каталога не совпадает с крайним правым именем в адресе нового местоположения каталога, то производится переименование.

Работать с каталогами файловой системы компьютера можно и при помощи класса Directory, функциональные возможности которого во многом совпадают с возможностями DirectoryInfo. Следует учитывать, что члены данного класса реализованы статически, поэтому для их использования нет необходимости создавать объект. Например:

```
// Создание подкаталога Новый
Directory.CreateDirectory(@"d:\prim\Новый");

// Перемещение каталога Новый в каталог 2012
Directory.Move(@"d:\prim\Новый",
               @"d:\prim\2012\Новый");

//переименовали каталог Новый в Старый
Directory.Move(@"d:\prim\2012\Новый",
               @"d:\prim\2012\Старый");
```

Следует учитывать, что удаление каталога возможно только когда он пуст. На практике комбинируют использование классов Directory и DirectoryInfo.

2.3 Классы для работы с файлами.

Класс FileInfo предназначен для организации доступа к физическому файлу, который содержится на жестком диске компьютера. Он позволяет получать информацию об этом файле (например, о времени его создания, размере, атрибутах), а также производить различные операции, например, по созданию файла или его удалению. Класс FileInfo наследует члены класса FileSystemInfo и содержит дополнительный набор членов, который приведен в следующей таблице 19.4

Таблица 19.4 – Члены класса FileInfo.

Член	Описание
AppendText()	Создает объект StreamWriter для добавления текста к файлу.
CopyTo()	Копирует уже существующий файл в новый файл.
Create()	Создает новый файл и возвращает объект FileStream для взаимодействия с этим файлом.
CreateText()	Создает объект StreamWriter для записи текстовых данных в новый файл.
Delete()	Удаляет файл, которому соответствует объект FileInfo.
Directory	Возвращает каталог, в котором расположен данный файл.
DirectoryName	Возвращает полный путь к данному файлу в файловой системе.
Length	Возвращает размер файла.
MoveTo()	Перемещает файл в указанное пользователем место (этот метод позволяет одновременно переименовать данный файл).
Name	Позволяет получить имя файла.
Open()	Открывает файл с указанными пользователем правами доступа на чтение, запись или совместное использование с другими пользователями.
OpenRead()	Создает объект FileStream, доступный только для чтения.
OpenText()	Создает объект StreamReader (о нем также будет рассказано ниже), который позволяет считывать информацию из существующего текстового файла.
OpenWrite()	Создает объект FileStream, доступный для чтения и записи.

Большинство методов FileInfo возвращает объекты классов FileStream, StreamWriter, StreamReader и т. п., которые позволяют различным образом взаимодействовать с файлом, например, производить чтение или запись в него. Например:

```
// Создание объекта FileInfo
FileInfo f = new FileInfo("text.txt");

// Создание файла и потока
StreamWriter fOut =
    new StreamWriter(f.Create());

// Запись текста в файл
fOut.WriteLine("ОДИН ДВА ТРИ...");

// Закрытие файла
fOut.Close();

// Получение информации о файле
Console.WriteLine("Name: {0}", f.Name);
Console.WriteLine("File size: {0}",
    f.Length);
Console.WriteLine("Creation: {0}",
    f.CreationTime);
Console.WriteLine("Attributes: {0}",
    f.Attributes.ToString());
```

Доступ к физическим файлам можно получать и через статические методы класса File. Большинство методов объекта FileInfo представляют в этом смысле зеркальное отражение методов объекта File.

2.4 Потоки в системе ввода-вывода.

Программы на языке C# выполняют операции ввода-вывода посредством потоков, которые построены на иерархии классов. Поток (stream) – это абстракция, которая генерирует и принимает данные. С помощью потока можно читать данные из различных источников (клавиатура, файл, память) и записывать в различные источники (принтер, экран, файл, память).

Центральную часть потоковой системы C# занимает класс Stream пространства имен System.IO. Класс Stream представляет байтовый поток и является базовым для всех остальных потоковых классов. Производными от класса Stream являются классы потоков:

1. FileStream – байтовый поток, разработанный для файлового ввода-вывода.

2. BufferedStream – заключает в оболочку байтовый поток и добавляет буферизацию, которая во многих случаях увеличивает производительность программы.

3. MemoryStream – байтовый поток, который использует память для хранения данных.

Программист может реализовать собственные потоковые классы. Однако для подавляющего большинства приложений достаточно встроенных потоков.

2.4.1 Байтовый поток.

Чтобы создать байтовый поток, связанный с файлом, создается объект класса FileStream. При этом в классе определено несколько конструкторов. Чаще всего используется конструктор, который открывает поток для чтения и (или) записи:

```
public FileStream(string path, FileMode mode);
```

Параметр path определяет имя файла, с которым будет связан поток ввода-вывода данных. Параметр mode определяет режим открытия файла, который может принимать одно из возможных значений, определенных перечислением FileMode:

– FileMode.Append – предназначен для добавления данных в конец файла;

– FileMode.Create – предназначен для создания нового файла, при этом если существует файл с таким же именем, то он будет предварительно удален;

– FileMode.CreateNew – предназначен для создания нового файла, при этом файл с таким же именем не должен существовать;

– FileMode.Open – предназначен для открытия существующего файла;

– FileMode.OpenOrCreate – если файл существует, то открывает его, в противном случае создает новый;

– FileMode.Truncate – открывает существующий файл, но усекает его длину до нуля.

Если попытка открыть файл оказалась неудачной, то генерируется одно из исключений:

– FileNotFoundException – файл невозможно открыть по причине его отсутствия;

– IOException – файл невозможно открыть из-за ошибки ввода-вывода;

– ArgumentNullException – имя файла представляет собой null-значение;

– ArgumentException – некорректен параметр mode;

– SecurityException – пользователь не обладает правами доступа;

– DirectoryNotFoundException – некорректно задан каталог.

Другая версия конструктора позволяет ограничить доступ только чтением или только записью:

```
public FileStream(string path, FileMode mode, FileAccess access);
```

Параметры path и mode имеют то же назначение, что и в предыдущей версии конструктора. Параметр access, определяет способ доступа к файлу и может принимать одно из значений, определенных перечислением FileAccess:

– FileAccess.Read – только чтение;

– FileAccess.Write – только запись;

– FileAccess.ReadWrite – и чтение, и запись.

После установления связи байтового потока с физическим файлом внутренний указатель потока устанавливается на начальный байт файла.

Для чтения очередного байта из потока, связанного с физическим файлом, используется метод ReadByte( ). После прочтения очередного байта внутренний указатель перемещается на следующий байт файла. Если достигнут конец файла, то метод ReadByte( ) возвращает значение -1.

Для побайтовой записи данных в поток используется метод WriteByte( ).

По завершении работы с файлом его необходимо закрыть. Для этого достаточно вызвать метод Close( ). При закрытии файла освобождаются системные ресурсы, ранее выделенные для этого файла, что дает возможность использовать их для работы с другими файлами.

```
static void Main(string[] args)
{
    try
    {
        // Создание потока для чтения из файла
        FileStream fileIn =
            new FileStream("ФайлСТекстом.txt",
                FileMode.Open,
                FileAccess.Read);
        // Создание потока для записи в файл
        FileStream fileOut =
            new FileStream("ФайлКопия.txt",
                FileMode.Create,
                FileAccess.Write);
        int i;

        // В цикле производится чтение одного файла
        // и одновременная запись содержимого
        // во второй файл
        while ((i = fileIn.ReadByte()) != -1)
        {
            // запись очередного байта в поток
            fileOut.WriteByte((byte)i);
        }

        // Закрыте файлов
        fileIn.Close();
        fileOut.Close();
    }
    catch (Exception EX)
    {
        Console.WriteLine(EX.Message);
    }
}
```

В данном примере создаются два потока fileIn (для чтения байтов из файла в поток) и fileOut (для записи байтов из потока в файл). Каждый из потоков ассоциируется со своим файлом. Затем в цикле производится побайтовое чтение файла «ФайлСТекстом.txt» до момента, когда функция ReadByte( ) вернет значение -1 (то есть достигнут конец файла). При этом на каждой итерации цикла считывается один байт и на этой же итерации этот байт записывается в файл «ФайлКопия.txt». После всех операция оба потока закрываются. Весь код, осуществляющий работу с файлами заключен в конструкцию try … catch. Это позволяет повысить устойчивость программы к ошибкам.

2.4.2 Символьный поток.

Чтобы создать символьный поток нужно поместить объект класса Stream (например FileStream) внутрь объекта класса StreamWriter или объекта класса StreamReader. В этом случае байтовый поток будет автоматически преобразовываться в символьный.

Класс StreamWriter предназначен для организации выходного символьного потока. В нем определено несколько конструкторов. Один из них записывается следующим образом:

```
public StreamWriter(Stream stream);
```

Параметр stream определяет имя уже открытого байтового потока. Этот конструктор генерирует исключение типа ArgumentException, если поток stream не открыт для вывода, и исключение типа ArgumentNullException, если он (поток) имеет null-значение.

Другой вид конструктора позволяет открыть поток сразу через обращения к файлу:

```
public StreamWriter(string path);
```

Параметр path определяет имя открываемого файла.

Например, создать экземпляр класса StreamWriter можно следующим образом:

```
StreamWriter fileOut =
    new StreamWriter(
        new FileStream(
            "ФайлНиколаева.txt",
            FileMode.Create,
            FileAccess.Write));
```

И еще один вариант конструктора StreamWriter:

```
public StreamWriter(string path, bool append);
```

Параметр path определяет имя открываемого файла, а параметр append может принимать значение true – если нужно добавлять данные в конец файла, или false – если файл необходимо перезаписать.

Например:

```
// Добавляем символы в конец файла
StreamWriter fileOut_1 =
    new StreamWriter("МойФ.txt", true);
```

Объявив таким образом переменную fileOut_1 для записи данных в поток можно обратиться к методу WriteLine:

```
// Использование WriteLine
fileOut_1.WriteLine("Строка для записи");
```

В данном случае для записи используется метод, аналогичный статическому методу класса Console. Это действительно схожие механизмы ввода-вывода.

Класс StreamReader предназначен для организации входного символьного потока. Один из его конструкторов выглядит следующим образом:

```
public StreamReader(Stream stream);
```

Параметр stream определяет имя уже открытого байтового потока.

Этот конструктор генерирует исключение типа ArgumentException, если поток stream не открыт для ввода. Создать экземпляр класса StreamReader можно следующим образом:

```
StreamReader fileIn =
    new StreamReader(
        new FileStream(
            "text.txt",
            FileMode.Open,
            FileAccess.Read));
```

Как и в случае с классом StreamWriter у класса StreamReader есть и другой вид конструктора, который позволяет открыть файл напрямую:

```
public StreamReader(string path);
```

Параметр path определяет имя открываемого файла. Обратиться к данному конструктору можно следующим образом:

```
StreamReader fileIn =
    new StreamReader
        (@"c:\temp\Николаев.txt");
```

В C# символы реализуются кодировкой Unicode. Для того, чтобы можно было обрабатывать текстовые файлы, содержащие русские символы, созданные, например, в Блокноте, рекомендуется вызывать следующий вид конструктора StreamReader:

```
StreamReader fileIn =
    new StreamReader(
        @"c:\temp\t.txt",
        Encoding.GetEncoding(1251));
```

Параметр Encoding.GetEncoding(1251) говорит о том, что будет выполняться преобразование из кода Windows-1251 (одна из модификаций кода ASCII, содержащая русские символы) в Unicode. Тип Encoding реализован в пространстве имен System.Text.

Для чтения данных из потока fileIn можно воспользоваться методом ReadLine. При этом если будет достигнут конец файла, то метод ReadLine вернет значение null.

Рассмотрим пример, в котором данные из одного файла копируются в другой, но уже с использованием классов StreamWriter и StreamReader.

```
static void Main(string[] args)
{
    // Создание символьных потоков
    StreamReader fileIn =
        new StreamReader(
            "ФайлСТекстом.txt",
            Encoding.GetEncoding(1251));
    StreamWriter fileOut =
        new StreamWriter(
            "НовыйФайл.txt",
            false);
    string line;

    // Построчное считывание
    while ((line = fileIn.ReadLine()) != null)
    {
        // ... и запись строки
        fileOut.WriteLine(line);
    }

    fileIn.Close();
    fileOut.Close();
}
```

В данном примере осуществляется копирование содержимого одного символьного файла в другой.

Таким образом, данный способ копирования одного файла в другой, дает тот же результат, что и при использовании байтовых потоков. Однако, его работа будет менее эффективной, т.к. будет тратиться дополнительное время на преобразование байтов в символы. У символьных потоков есть и свои преимущества. Например, можно использовать регулярные выражения для поиска заданных фрагментов текста в файле.

2.4.3 Перенаправление стандартных потоков.

Тремя стандартными потоками, доступ к которым осуществляется через свойства Console.Out, Console.In и Console.Error, могут пользоваться все программы, работающие в пространстве имен System. Свойство Console.Out относится к стандартному выходному потоку. По умолчанию это консоль. Например, при вызове метода Console.WriteLine() информация автоматически передается в поток Console.Out. Свойство Console.In относится к стандартному входному потоку, источником которого по умолчанию является клавиатура. Например, при вводе данных с клавиатуры информация автоматически передается потоку Console.In, к которому можно обратиться с помощью метода Console.ReadLine(). Свойство Console.Error относится к ошибкам в стандартном потоке, источником которого также по умолчанию является консоль. Однако эти потоки могут быть перенаправлены на любое совместимое устройство ввода-вывода, например, на работу с физическими файлами.

Перенаправить стандартный поток можно с помощью методов SetIn(), SetOut() и SetError(), которые являются членами класса Console:

```
static void SetIn(TextReader input);
static void SetOut(TextWriter output);
static void SetError(TextWriter output);
```

Пример перенаправления потоков представлен в следующей программе, демонстрирующей, что стандартный поток вывода перенаправляется в один файл, а поток ввода – в другой:

```
static void Main(string[] args)
{
    int[,] MyArray;
    StreamReader file = new StreamReader("input.txt");
    // Перенаправление на file
    Console.SetIn(file);
    string line = Console.ReadLine();
    string[] mas = line.Split(' ');
    int n = int.Parse(mas[0]);
    int m = int.Parse(mas[1]);
    MyArray = new int[n, m];
    for (int i = 0; i < n; i++)
    {
        line = Console.ReadLine();
        mas = line.Split(' ');
        for (int j = 0; j < m; j++)
        {
            MyArray[i, j] = int.Parse(mas[j]);
        }
    }
    PrintArray("Исходный массив:", MyArray, n, m);
    file.Close();
}

static void PrintArray(string a, int[,] mas, int n, int m)
{
    // Перенаправление на file
    StreamWriter file=new StreamWriter("output.txt");
    Console.SetOut(file);
    Console.WriteLine(a);
    for (int i = 0; i < n; i++)
    {
        for (int j=0; j<m; j++) Console.Write("{0} ", mas[i,j]);
        Console.WriteLine();
    }
    file.Close();
}
```

4. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

5. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

6. Методика и порядок выполнения работы

Для выполнения лабораторной работы необходимо спроектировать многомодульное приложение, использующее файлы для ввода и вывода информации.

Например, требуется разработать приложение, которое позволяет:

– генерировать матрицу случайных чисел с сохранением ее в файл;

– считывать матрицу из файла;

– выводить матрицу на экран.

Для выполнения данного задания-примера достаточно в отдельном проекте реализовать класс. Этот проект следует скомпилировать в виде dll-файла. Затем, в основной программе просто необходимо использовать данную библиотеку.

Класс библиотеки назовем Matrix. В нем определим следующие функции:

```
class Matrix
{
    private float[,] matrix;
    int m, n;

    // Конструктор
    public Matrix()...

    // Генерация матрицы заданного размера
    public void GenerateMatrix(int M, int N)...

    // Сохранение сгенерированной матрицы в файл
    public void SaveMatrix(string pFileName)...

    // Загрузка сохраненной матрицы из файла
    public Boolean LoadMatrix(string pFileName)...

    // Вывод матрицы в консоль
    public void PrintMatrix()...
}
```

Рисунок 19.1 – Основные методы класса Matrix.

Листинг метода GenegateMatrix:

```
// Генерация матрицы заданного размера
public void GenerateMatrix(int M, int N)
{
    m = M; n = N;

    Random r = new Random(DateTime.Now.Millisecond);

    matrix = new float[M, N];

    for (int i = 0; i < M; i++)
        for (int j = 0; j < N; j++)
            matrix[i, j] = (float)r.Next(1000) / 973f;
}
```

Рисунок 19.2 – Метод для создания матрицы заданного размера и заполнения ее случайными числами.

Листинг метода SaveMatrix:

```
// Сохранение сгенерированной матрицы в файл
public void SaveMatrix(string pFileName)
{
    if (matrix.Length > 0)
    {
        if (File.Exists(pFileName))
            File.Delete(pFileName);

        FileInfo f = new FileInfo(pFileName);
        TextWriter tw = f.CreateText();

        tw.WriteLine(m.ToString());
        tw.WriteLine(n.ToString());

        for (int i = 0; i < m; i++)
            for (int j = 0; j < n; j++)
                tw.WriteLine(i.ToString() + " " + j.ToString() + " " + matrix[i, j].ToString("E10"));

        tw.Close();
    }
}
```

Рисунок 19.3 – Метод для сохранения матрицы в файл.

```
// Загрузка сохраненной матрицы из файла
public Boolean LoadMatrix(string pFileName)
{
    if (File.Exists(pFileName))
    {
        try
        {
            TextReader tr = File.OpenText(pFileName);
            m = Convert.ToInt32(tr.ReadLine());
            n = Convert.ToInt32(tr.ReadLine());

            matrix = new float[m, n];
            string line;
            string[] substring;

            for (int i = 0; i < m; i++)
                for (int j = 0; j < n; j++)
                {
                    line = tr.ReadLine();
                    substring = line.Split(new char[] {' '}, 3);
                    matrix[i, j] = Convert.ToSingle(substring[2]);
                }

            tr.Close();

            return true;

        }
        catch
        {
            return false;
        }
    }

    return false;
}
```

Рисунок 19.4 – Метод загрузки матрицы из файла.

На рис. 19.4 представлен листинг метода LoadMatrix. Листинг метода PrintMatrix:

```
// Вывод матрицы в консоль
public void PrintMatrix()
{
    if (matrix.Length > 0)
    {
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < n; j++)
                Console.Write(matrix[i, j].ToString("E3") + "  ");

            Console.WriteLine();
        }

    }
}
```

Рисунок 19.5 – Метод для вывода матрицы.

Основная программа для использования класса-матрицы представлена на рис. 19.6

```
class Program
{
    static void Main(string[] args)
    {
        Console.BackgroundColor = ConsoleColor.White;
        Console.ForegroundColor = ConsoleColor.Black;
        Console.Clear();

        Matrix m = new Matrix();

        m.GenerateMatrix(10, 5);
        m.SaveMatrix("FileForMatrix.txt");

        if (m.LoadMatrix("FileForMatrix.txt"))
            m.PrintMatrix();

        Console.ReadKey();
    }
}
```

Рисунок 19.6 – Работа с библиотечным классом.

В результате работы данного кода будет создан файл FileForMatrix.txt следующего содержания:

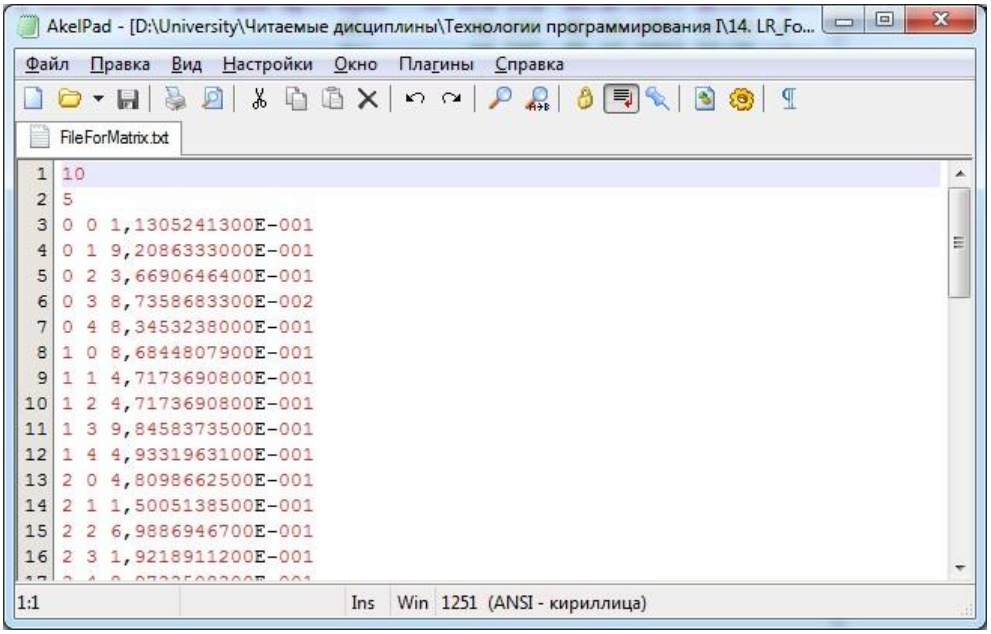


Рисунок 19.7 – Результирующий файл.

На экран будет выведена следующая информация:

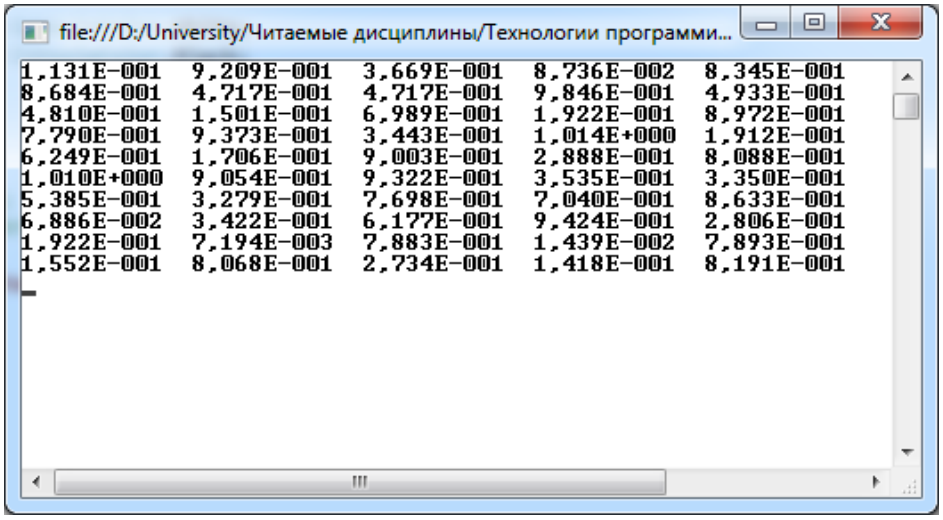


Рисунок 19.8 – Вывод программы в консоль.

Выполните индивидуальное задание, согласно предложенному варианту. В каждом варианте необходимо спроектировать многомодульное приложение (оптимально 2 модуля). Исходные данные в каждом варианте программа получает из входного файла.

Индивидуальное задание.

Вариант	Структура данных
1	Программа рассчитывает произведение двух матриц, которые хранятся в разных файлах.
2	Программа рассчитывает сумму двух матриц, которые хранятся в разных файлах.
3	Программа находит максимальный элемент в двух матрицах, которые хранятся в разных файлах.
4	Программа рассчитывает сумму диагональных элементов двух матриц, которые хранятся в разных файлах.
5	Программа рассчитывает сумму элементов с четной суммой индексов в двух матрицах, которые хранятся в разных файлах.
6	Программа рассчитывает разность сумм элементов матриц, которые хранятся в разных файлах.
7	Программа рассчитывает сумму элементов главной диагонали первой матрицы и сумму элементов второстепенной диагонали второй матрицы. Матрицы хранятся в разных файлах.
8	Программа рассчитывает сумму элементов четных столбцов в двух матрицах, которые хранятся в разных файлах.
9	Программа рассчитывает сумму диагональных элементов четных столбцов в двух матрицах, которые хранятся в разных файлах.
10	Программа рассчитывает сумму элементов четных строк в двух матрицах, которые хранятся в разных файлах.
11	Программа рассчитывает сумму элементов , находящихся в четном столбце и нечетной строке, в двух матрицах, которые хранятся в разных файлах.
12	Программа рассчитывает сумму элементов четных строк и расположенных на второстепенной диагонали в обеих матрицах, которые хранятся в разных файлах.
13	Программа рассчитывает произведение элементов в двух матрицах, которые хранятся в разных файлах.
14	Программа рассчитывает сумму первой матрицы и матрицы транспонированной относительно второй. Обе матрицы хранятся в отдельных файлах.
15	Программа рассчитывает произведение элементов диагонали первой матрицы и сумму всех элементов второй матрицы. Обе матрицы хранятся в отдельных файлах.
16	Программа рассчитывает сумму элементов четных строк в двух матрицах, которые хранятся в разных файлах.
17	Программа находит максимальный элемент в двух матрицах, которые хранятся в разных файлах.

Вариант	Структура данных
18	Программа рассчитывает сумму элементов четных строк и расположенных на второстепенной диагонали в обеих матрицах, которые хранятся в разных файлах.
19	Программа рассчитывает сумму первой матрицы и матрицы транспонированной относительно второй. Обе матрицы хранятся в отдельных файлах.
20	Программа рассчитывает произведение двух матриц, которые хранятся в разных файлах.
21	Программа рассчитывает сумму элементов четных столбцов в двух матрицах, которые хранятся в разных файлах.
22	Программа рассчитывает сумму диагональных элементов четных столбцов в двух матрицах, которые хранятся в разных файлах.
23	Программа рассчитывает произведение элементов диагонали первой матрицы и сумму всех элементов второй матрицы. Обе матрицы хранятся в отдельных файлах.
24	Программа рассчитывает сумму элементов с четной суммой индексов в двух матрицах, которые хранятся в разных файлах.
25	Программа рассчитывает разность сумм элементов матриц, которые хранятся в разных файлах.

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Какие классы для работы с файловой системой вы знаете?

2. Что такое сборка?

3. Как определить проект по умолчанию в многомодульном решении?

4. Какие классы отвечают за представление файлов в программе?

5. Что такое поток? Какие типы классов потоков используются при работе с файлами?

6. Опишите последовательность действий при необходимости записать одну строку в файл. Приведите примеры использования различных классов.

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [5], [6-8].

ЛАБОРАТОРНАЯ РАБОТА 15. РЕШЕНИЕ ВЫЧИСЛИТЕЛЬНОЙ ЗАДАЧИ С ПРИМЕНЕНИЕМ ФАЙЛОВОГО ВВОДА-ВЫВОДА.

1. Цель и содержание

Цель лабораторной работы: научиться использовать возможности файлового ввода-вывода для решения практических задач.

Задачи лабораторной работы:

– научиться проектировать приложение для реализации хранения данных программы;

– научиться производить выбор оптимальных инструментов для обеспечения сериализации.

2. Теоретическая часть

Перед выполнением лабораторной работы необходимо повторить теоретический материал лабораторной работы №19.

3. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения работы

1. Создайте консольное приложение.

2. Выполните индивидуальное задание. В каждом задании необходимо реализовать вычисление значений функции $f(\mathrm{x},\mathrm{y})$ с заданными шагами Δx и Δy на заданных диапазонах изменения независимых переменных $[x_0;x_1]$, $[y_0;\mathrm{y}_1]$.

3. Необходимо реализовать сохранение рассчитанных значений в файле.

Индивидуальное задание.

Вариант	Выражение для вычисления
1	$f(x,y)=a\cdot\sin(x)+b\cdot\cos(y)$; $\Delta x=0{,}1$; $\Delta y=0{,}1$; $x\in[-5;5]$; $y\in[-5;5]$.
2	$f(x,y)=a\cdot x\cdot\lvert\sin(x+y)\rvert+b\cdot x\cdot\lvert\cos(y+x)\rvert$; $\Delta x=0{,}1$; $\Delta y=0{,}1$; $x\in[-1;10]$; $y\in[-5;5]$.
3	$f(x,y)=a^2\cdot\sin^3(x+y)+\lvert b^3\cdot\cos^2(x+y)\rvert$; $\Delta x=0{,}01$; $\Delta y=0{,}01$; $x\in[-2;2]$; $y\in[-2;2]$.
4	$f(x,y)=a^2\cdot\sqrt{\lvert\sin(x)\rvert}+b\cdot\sqrt{\lvert\cos(y)\rvert}$; $\Delta x=0{,}01$; $\Delta y=0{,}01$; $x\in[-2;2]$; $y\in[-2;2]$.
5	$f(x,y)=a\cdot\sqrt{\lvert\sin(x)+a\cdot x\rvert}+b\cdot\sqrt{\lvert\cos(y)+b\cdot y\rvert}$; $\Delta x=0{,}01$; $\Delta y=0{,}01$; $x\in[-2;2]$; $y\in[-2;2]$.
6	$f(x,y)=a\cdot\sqrt{\lvert\sin(x)+a\cdot x\rvert}+b\cdot\sqrt{\lvert\cos(y)+b\cdot y\rvert}$; $\Delta x=0{,}01$; $\Delta y=0{,}01$; $x\in[-2;2]$; $y\in[-2;2]$.
7	$f(x,y)=\sqrt{x^2+y^2-2xy}$; $\Delta x=0{,}01$; $\Delta y=0{,}01$; $x\in[-2;2]$; $y\in[-2;2]$.
8	$f(x,y)=a\cdot\sin(x)+b\cdot\cos(y)$; $\Delta x=0{,}1$; $\Delta y=0{,}1$; $x\in[-5;5]$; $y\in[-5;5]$.
9	$f(x,y)=a\cdot y\cdot\sin(x+y)+b\cdot x\cdot\cos(y+x)$; $\Delta x=0{,}1$; $\Delta y=0{,}1$; $x\in[-5;5]$; $y\in[-5;5]$.
10	$f(x,y)=a\sqrt{\lvert x+y\rvert}+b\sqrt{\lvert x-y\rvert}$; $\Delta x=0{,}01$; $\Delta y=0{,}01$; $x\in[-2;2]$; $y\in[-2;2]$.
11	$f(x,y)=a\cdot\sqrt{\lvert\sin(x)+a\cdot x\rvert}+b\cdot\sqrt{\lvert\cos(y)+b\cdot y\rvert}$; $\Delta x=0{,}01$; $\Delta y=0{,}01$; $x\in[-2;2]$; $y\in[-2;2]$.
12	$f(x,y)=a\cdot\sqrt{\lvert\sin(x)+a\cdot x\rvert}+b\cdot\sqrt{\lvert\cos(y)+b\cdot y\rvert}$; $\Delta x=0{,}01$; $\Delta y=0{,}01$; $x\in[-2;2]$; $y\in[-2;2]$.
13	$f(x,y)=\sqrt{x^2+y^2-2xy}$; $\Delta x=0{,}01$; $\Delta y=0{,}01$; $x\in[-2;2]$; $y\in[-2;2]$.
14	$f(x,y)=a\cdot\sin(x)+b\cdot\cos(y)$; $\Delta x=0{,}1$; $\Delta y=0{,}1$; $x\in[-5;5]$; $y\in[-5;5]$.
15	$f(x,y)=a\cdot x\cdot\lvert\sin(x+y)\rvert+b\cdot x\cdot\lvert\cos(y+x)\rvert$; $\Delta x=0{,}1$; $\Delta y=0{,}1$; $x\in[-1;10]$; $y\in[-5;5]$.

16	$f(x,y)=a\cdot(x+y)\cdot\sin^2(x)+b\cdot(x-y)\cdot\cos^2(y)$; $\Delta x=0{,}1$; $\Delta y=0{,}1$; $x\in[-5;5]$; $y\in[-5;5]$.
17	$f(x,y)=a^2\cdot x+b^3\cdot\cos^2(y+x)$; $\Delta x=0{,}05$; $\Delta y=0{,}05$; $x\in[-2;2]$; $y\in[-2;2]$.
18	$f(x,y)=a^2\cdot\sqrt{\lvert b\cdot\sin(x)\rvert}+b^2\cdot\sqrt{\lvert a\cdot\cos(y)\rvert}$; $\Delta x=0{,}01$; $\Delta y=0{,}01$; $x\in[-2;2]$; $y\in[-2;2]$.
19	$f(x,y)=a\cdot x\cdot\lvert\sin(x+y)\rvert+b\cdot x\cdot\lvert\cos(y+x)\rvert$; $\Delta x=0{,}1$; $\Delta y=0{,}1$; $x\in[-1;10]$; $y\in[-5;5]$.
20	$f(x,y)=a\cdot\sin(x)+b\cdot\cos(y)$; $\Delta x=0{,}1$; $\Delta y=0{,}1$; $x\in[-5;5]$; $y\in[-5;5]$.
21	$f(x,y)=a^2\cdot\sin^3(x+y)+b^3\cdot\cos^2(x+y)$; $\Delta x=0{,}05$; $\Delta y=0{,}05$; $x\in[-2;2]$; $y\in[-2;2]$.
22	$f(x,y)=a\cdot\sqrt{\lvert\sin(x)+a\cdot x\rvert}+b\cdot\sqrt{\lvert\cos(y)+b\cdot y\rvert}$; $\Delta x=0{,}01$; $\Delta y=0{,}01$; $x\in[-2;2]$; $y\in[-2;2]$.
23	$f(x,y)=a\sqrt{\lvert x+y^2\rvert}+b\sqrt{\lvert x^2-y\rvert}$; $\Delta x=0{,}01$; $\Delta y=0{,}01$; $x\in[-2;2]$; $y\in[-2;2]$.
24	$f(x,y)=a^2\cdot x+b^3\cdot y-\lvert\sin(x)+\cos(y)\rvert$; $\Delta x=0{,}01$; $\Delta y=0{,}01$; $x\in[-1;1]$; $y\in[-1;1]$.
25	$f(x,y)=\sqrt{x^2+y^2-2xy}$; $\Delta x=0{,}01$; $\Delta y=0{,}01$; $x\in[-2;2]$; $y\in[-2;2]$.

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Что такое поток? Какие классы потоков вы знаете?
2. Какие классы пространства System.IO предназначены для работы с каталогами?
3. Какие классы пространства System.IO предназначены для работы с объектами файловой системы – файлами?
4. Как перенаправить стандартный консольный поток?
5. Поясните разницу между классами File и FileInfo?

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [5].

ТЕМА 5. ПРОДВИНУТОЕ ООП: МОДУЛЬНОСТЬ И ИНТЕРФЕЙСЫ
ЛАБОРАТОРНАЯ РАБОТА 16. СОЗДАНИЕ ПРИЛОЖЕНИЯ ПО ТЕХНОЛОГИИ WINDOWS FORMS

1. Цель и содержание

Цель лабораторной работы: научиться создавать приложение Windows в среде MS Visual Studio и программировать алгоритмы с использованием простых элементов управления.

Задачами лабораторной работы являются:

– разработка приложения с использованием Windows Forms,

– изучение основных членов класса Form;

– обработка событий от простых элементов управления.

2. Теоретическая часть

Файловая структура проекта MS Visual Studio в случае использования типа поекта Windows Form Application выглядит следующим образом (рис. 21.1):

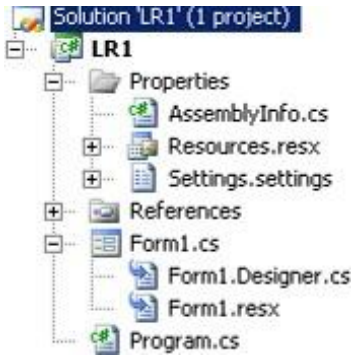


Рисунок 21.1 – Вид окна Solution Explorer для Windows Form Application.

Файл Program.cs содержит класс Program и статический метод Main(), с которого начинается выполнение приложения (рис. 21.2).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace LR1
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Рисунок 21.2 – Листинг Program.cs.

В самом начале файла Program.cs выполняется объявление используемых пространств имен с использованием using.

Класс Program содержит метод Main(), который используя статический метод Run класса Application создает и выводит на экран главную форму приложения: Application.Run (new Form1()).

Таким образом, реализуется один из принципов объектно-ориентированного программирования: разграничение обязанностей (т.е. каждый класс выполняет минимально возможное количество операций).

Класс главной формы Form1 (по умолчанию) представлен двумя связанными C#-файлами. Для отображения содержимого Form1.cs необходимо щелкнутьправой кнопкой мыши в окне проектирования главной формы на самой форме или на пиктограмме Form1.cs в окне Solution Explorer (рис. 21.1) и в появившемся контекстном меню выбрать пункт «View Code».

Листинг Form1.cs представлен на рис 21.3.

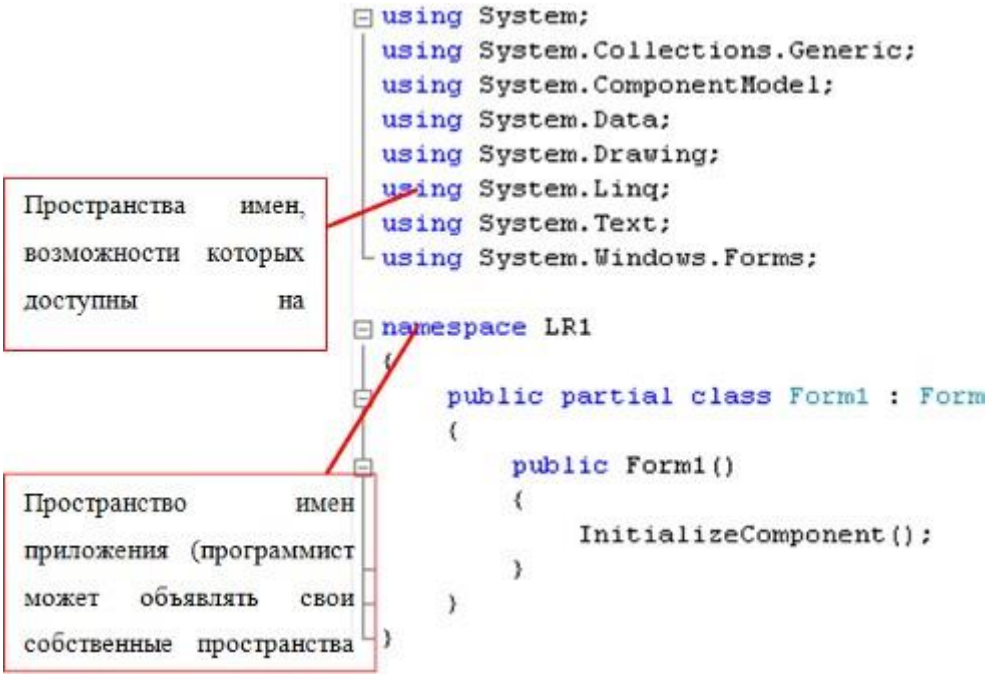


Рисунок 21.3 –Листинг Form1.cs.

Конструктор, созданный по умолчанию, вызывает метод InitializeComponent, который определен в соответствующем файле Form1.Designer.cs (рис. 21.1).

Этот метод создается автоматически, в нем отображаются все изменения, производимые программистом в окне визуального проектирования формы.

Пространства имен, доступные для использования, объявляются в начале файла.

System является базовым пространством имен – в него входят все остальные типы и пространство имен. Конструкция using System; в начале файла программы указывает на то, что весь программный код будет выполняться в данном пространстве имен, поэтому при использовании типов (например Int32), определенных в пространстве System нет необходимости указывать само имя пространства (то есть нет необходимости писать System.Int32).

Программист может самостоятельно вводить пространства имен при написании приложений.

Обработка событий в режиме проектирования. Окно проектирования главной формы приложения (рис. 21.4) можно открыть дважды щелкнув на пиктограмме Form1.cs в окне Solution Explorer (рис. 21.1). При этом откроется вкладка Form1.cs [Design].

Окно свойств примет вид, показанный на рисунке 21.4.

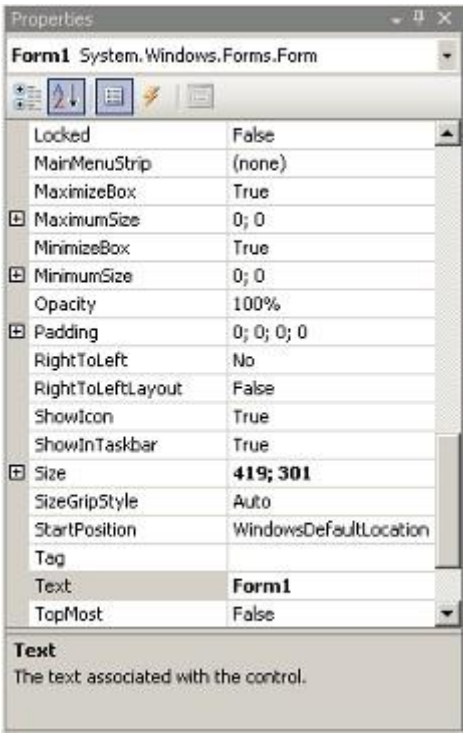


Рисунок 21.4 – Вид окна Свойств (Properties) в режиме проектирования формы.

Кнопка позволяет отобразить список событий, характерных для элемента управления, активного в данный момент в окне проектирования. Например, если выделить главную форму приложения, то отобразятся события, характерные для класса Form.

Для отработки механизма регистрации событий, рассмотрим обработку наиболее простых событий – событий от мыши.

Для добавления обработчика события мыши необходимо выбрать соответствующее событие (Click, DoubleClick, MouseEnter, MouseLeave, MouseDown, MouseUp, MouseMove,. MouseHover, MeaseWheel) и дважды щелкнуть на поле, расположенном в соседнем от него столбце в окне свойств (таким образом обработчику присвоится имя по умолчанию). В данном поле можно ввести свое название для нового обработчика, или выбрать из выпадающего списка уже существующие обработчики.

После регистрации события в окне «Properties», автоматически добавляются строки кода в файлы Form1.Designer.cs (регистрируется обработчик события) и Form1.cs (добавляетсяметод, ассоциированный с данным событием).

Форма приложения должна быть функциональной и максимально эргономичной. Такие простые вещи, как цветовые комбинации, размеры шрифтов и окон могут сделать приложение намного более привлекательным для пользователя.

В случае если свойство Icon формы установлено, а для свойства ControlBox не указано значение false, то значок появится в левом верхнем углу формы. Обычно принято устанавливать здесь значок приложения.

Свойство FormBorderStyle задает тип рамки, которая появляется вокруг формы. Для этого используется перечисление FormBorderStyle. Допустимые значения этого перечисления: Fixed3D, FixedDialog, FixedSingle, FixedToolWindow, None, Sizable, SizableToolWindow.

Для выполнения лабораторной работы потребуется основная информация о событиях мыши:

Все обработчики событий мыши принимают параметр типа MouseEventArgs. Поступающий на вход обработчика событий мыши объект типа MouseEventArgs позволяет получить дополнительную информацию о действии мыши, путем введения ряда специальных членов класса (табл. 21.1).

Таблица 21.1 – Свойства типа MouseEventArgs

Свойство	Описание
Button	Содержит информацию о том, какая клавиша была нажата, в соответствии с определением перечня MouseButtons
Clicks	Содержит информацию о том, сколько раз была нажата и отпущена клавиша мыши
Delta	Содержит информацию со знаком, соответствующее числу щелчков, произошедших при вращении колесика мыши
X	Содержит информацию о координате X
Y	Содержит информацию о координате Y

3. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального

компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения работы

1. Создайте в среде разработки MS VS проект. В качестве типа проекта выбрать «Windows Application» (или «Windows Forms Application» в зависимости от версии .NET Framework)

2. После создания и сохранения проекта измените программу таким образом, чтобы координаты курсора мыши выводились в заголовке главного окна приложения.

Для этого в обработчик события движения мыши над оконной формой (MouseMove) добавьте строку кода:

Text = string.Format("Координаты:{0}, {1}", e.X, e.Y);

3. Добавьте текстовое поле (TextBox) в режиме разработки (для этого необходимо использовать панель элементов управления «ToolBox»). Дополните обработчик движения мыши таким образом, чтобы в текстовом поле отображалась сумма координат указателя мыши.

4. Работающую программу необходимо представить преподавателю.

5. После этого выполните индивидуальное задание в соответствие с вариантом. В каждом задании необходимо вывести значение выражения, предварительно введя значения переменных в соответствующие текстовые поля формы главного окна приложения. Результат выводится в заголовок окна в ответ на нажатие кнопки (кнопку также необходимо поместить на форму).

6. Для выполнения индивидуального задания необходимо использовать математические функции, которые доступны в виде статических методов класса Math.

Индивидуальное задание.

Перед выполнением задания требуется самостоятельно определить закономерность изменения членов последовательности, чтобы применить цикл, условный оператор или, если потребуется, оператор выбора.

Вариант	Выражение для вычисления
1	$Z = \sqrt{\left\|\frac{a \cdot x}{w^a}\right\|} + \sqrt{\|b\|} - \|x + \cos(y)\|$
2	$F = \sqrt{\|d2\|} \cdot x + \sqrt{\|b^3\|} - \left\|x^2 + \cos(y)\right\|$
3	$A = -d \cdot \frac{z}{\sqrt{\|e\|}} + \|\sin(e) + \cos(y)\|$
4	$U = \sqrt{\left\|\frac{f - e}{w}\right\|} + \left\|\sin^2(\frac{e}{w}) + \cos(y)\right\|$
5	$t = \lg(f) - e + \left\|\sin(\frac{w}{t}) + \sqrt{\|e\|}\right\|$
6	$V = \frac{h - e}{q + a} + \left\|\sin(f) + \sqrt{\|\sin(b)\|}\right\|$
7	$Q = \sin(h + \frac{d}{e^{o1}}) - o1 + \left\|\sin(f) + \sqrt{\|\sin(b)\|}\right\|$
8	$E = \sin(h11 + \frac{d12}{\ln(h11)}) - v6 + \left\|\sin(h3) + \sqrt{\|\sin(f)\|}\right\|$
9	$Z = z \cdot ax1 + \sqrt{\left\|e7 + \frac{x}{y}\right\|} - \|x + \cos(as_9 + y)\|$
10	$RES = \frac{\arg01 - \arg02 + \left\|\sin(\arg03) + \sqrt{\left\|\arg04\right\|}\right\|}{\arg01^{\arg 5} - \lg(\arg 02)}$
11	$U = \frac{v - e_2 + \left\|\cos(\arg3) + \sqrt{\left\|t\right\|}\right\|}{a_del * tg(t)}$
12	$t = W + \frac{\cos(g \cdot q)}{W} - e + \left\|\sin(e) + \sqrt{\|o\|}\right\|$
13	$Z = z \cdot \cos(y) + \sqrt{e7} - \|x + \cos(as_9 + y)\|$
14	$R = q + \left\|\sin^2(e) + \cos(y)\right\| \cdot \cos(s + g)$
15	$F = \sin(\frac{v \cdot x}{y}) \cdot x + \sqrt{\|b^3\|} - \left\|x^2 + \cos(y)\right\|$

16	$res = \frac{F}{\cos(w)} - e + \frac{\left\lvert \sin(\frac{w}{t}) + \sqrt{\lvert e \rvert} \right\rvert}{e}$
17	$t = W + \frac{tg(g \cdot q)}{\ln(20 \cdot x)} - \frac{e}{W} + \left\lvert \sin(e) + \sqrt{\lvert W - e \rvert} \right\rvert$
18	$t = W + \frac{tg(g + q)}{\ln(20 \cdot x)} - \frac{e}{W} + \left\lvert \sin(e) + \sqrt{\lvert W - e \rvert} \right\rvert$
19	$F = \sqrt{\lvert d2 - x \rvert} \cdot x + \sqrt{\lvert b^3 \rvert} - \left\lvert x^2 + \cos(y) \right\rvert$
20	$t = W + \frac{\cos(g \cdot q)}{W} - e + \left\lvert \sin(e) + \sqrt{\left\lvert \frac{o}{e^4} \right\rvert} \right\rvert$
21	$U = \frac{v^3 - e_2 + \left\lvert \cos(\arg 3) + \sqrt{\lvert t \rvert} \right\rvert}{a_del * tg(t)}$
22	$F = \sin(\frac{v \cdot x}{y}) \cdot x + \sqrt{\lvert b^3 \rvert} - \left\lvert x^2 + \cos(y) \right\rvert$
23	$v = \sqrt[4]{e_2} \cdot \sqrt{\frac{\sin(12 * \arg 3)}{e_2 - 1}} + \left\lvert \cos(\arg 3) + \sqrt{\lvert t \rvert} \right\rvert$
24	$res = \frac{F^{\lvert e \rvert}}{\cos(w + e)} - e + \frac{\left\lvert \sin(\frac{w}{t}) \right\rvert}{25 + \sqrt{\lvert e \rvert}}$
25	$t = \frac{az}{\sqrt{g^3}} + \frac{\cos(g \cdot q)}{az} - g + \left\lvert \sin(q) + \sqrt{\lvert 12 - q \rvert} \right\rvert$

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Какие файлы описывают класс формы?
2. Какие действия необходимо выполнить для создания обработчика события?
3. Где описывается код обработчика события? В каком файле регистрируется обработчик события (метод привязывается к событию)?
4. Как получить доступ к координатам курсора мыши?
5. Какой класс содержит методы, реализующие математические функции?

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [6].

ТЕМА 5. ПРОДВИНУТОЕ ООП: МОДУЛЬНОСТЬ И ИНТЕРФЕЙСЫ ЛАБОРАТОРНАЯ РАБОТА 17. ПРИМЕНЕНИЕ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ В ПРИЛОЖЕНИЯХ WINDOWS

1. Цель и содержание

Цель лабораторной работы: изучить принципы использования стандартных элементов управления в приложениях Windows.

Задачи лабораторной работы:

– научиться использовать кнопки, фражки, переключатели;

– научиться использовать списки, выпадающие списки;

– научиться .

2. Теоретическая часть

Понимание иерархии классов элементов управления Windows имеет важное значение при проектировании интерфейсов пользователя, особенно при разработке собственных элементов управления.

В пространстве имен System.Windoiws .Forms есть один особенный класс, служащий базовым для почти каждого создаваемого элемента управления и формы. Это System.Windows.Forms.Control. Класс Control реализует базовую

функциональность создания и отображения всего, что видит пользователь. Класс Control унаследован от класса System.ComponentModel.Component. Класс Component обеспечивает Control всей необходимой инфраструктурой, которая потребуется для того, чтобы его можно было перетаскивать на поверхность проектирования в визуальном конструкторе и помещать в другой объект. Класс Control предоставляет огромный список функциональности классам, унаследованным от него.

Размер и местоположение элемента управления определяется свойствами Height, Width, Top, Bottom, Left и Right, наряду с дополняющими их свойствами Size и Location. Отличие между ними в том, что свойства Height, Width, Top, Bottom, Left и Right принимают целочисленные значения, Size – структуру Size, а Location – структуру Point. Структуры Srze и Point содержат версии координат X,Y. Структура Point обычно связана с местоположением, а Size задает высоту и широту объекта. Size и Point определены в пространстве имен System. Drawing. Обе структуры очень похожи в том отношении, что представляют пару координат X,Y, но также имеют перегруженные операции для облегчения сравнения и преобразования. Так, например, две структуры Size можно складывать вместе. В структуре Point операция Addition переопределена так, что можно прибавлять структуру Size к Point и получать новое значение Point. Это обеспечивает эффект прибавления расстояния к местоположению с получением нового местоположения, что очень удобно, если нужно динамически создавать формы и элементы управления.

Свойство Bounds возвращает объект Rectangle, представляющий область элемента управления. Эта область включает линейки прокрутки и панель заголовка. Rectangle – также часть пространства имен System. Drawing. Свойство ClientSize – это структура Size, представляющая клиентскую область элемента управления, минус линейки прокрутки и панель заголовка.

PointToClient и PointToScreen – методы преобразования, которые принимают Point и возвращают Point. Метод PointToClient принимает Point, представляющую экранные координаты, и транслирует их в координаты,

основанные на текущем клиентском объекте. Это удобно для операций перетаскивания. Метод PointToScreen выполняет прямо противоположную операцию – принимает координаты клиентского объекта и транслирует их в экранные координаты. Методы RectangleToScreen и ScreenToRectangle выполняют те же функции со структурами Rectangle вместо Point.

Свойство Anchor прикрепляет край элемента управления к краю родительского элемента. Это отличается от стыковки тем, что не устанавливает грань по родительскому элементу, а устанавливает некоторую постоянную дистанцию от края. Например, если после прикрепления правого края элемента управления к правому краю родительского элемента размер родительского элемента будет изменен, правый край дочернего элемента останется на том же расстоянии от правого края родительского элемента. Свойство Anchor принимает значение типа перечисления AnchorStyle. Перечисление включает следующие значения: Top, Bottom, Left, Right и None.

С внешним видом элемента управления связаны свойства BackColor и ForeColor, которые принимают значения типа System. Drawing.Color. Свойство BackGroundlmage принимает в качестве значения объект на базе Image. Класс System. Drawing. Image – это абстрактный класс, используемый в качестве базового для классов Bitmap и Metafile.

Свойство BackgroundlmageLayout использует перечисление ImageLayout для установки режима вывода графического изображения в элементе управления. Допустимыми значениями являются Center,Tile, Stretch, Zoom и None.

Свойства Font и Text имеют дело с отображением текста. Чтобы изменить Font, потребуется создать объект Font. При создании объекта Font указывается имя шрифта, размер и стиль.

Для выполнения лабораторной работы необходимо изучить теоретические основы проектирования с использованием стандартных элементов управления библиотеки .NET Framework.

Класс. Button. Класс Button представляет простую командную кнопку и наследуется от класса ButtonBase. Наиболее часто приходится писать код для обработки события Click кнопки. В следующем фрагменте кода реализован обработчик события Click (обработка данного события не представляет сложностей). С помощью метода PerformClick можно эмулировать событие Click на кнопке без реального щелчка на ней пользователем, что может пригодиться для тестирования построенного пользовательского интерфейса. В окне также имеется понятие кнопки по умолчанию, на которой автоматически совершается щелчок, если пользователь нажимает в этом окне клавишу «Enter». Чтобы идентифицировать кнопку как кнопку по умолчанию, в форме, содержащей эту кнопку, необходимо установить свойство AcceptButton в объект кнопки.

Класс CheckBox (рис. 22.1). Элемент управления CheckBox также унаследован от ButtonBase и используется для приема двух или трех состояний от пользователя.

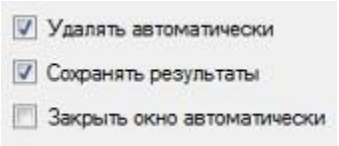


Рисунок 22.1 – Внешний вид CheckBox.

Если установить свойство ThreeState в true, то свойство CheckState элемента CheckBox сможет принимать одно их трех значений перечисления CheckState, описанных в табл. 22.1.

Таблица 22.1 – Состоятия CheckBox (возможные значения перечисления CheckState).

Значение	Описание
Checked	Флажок имеет отметку.
Unchecked	Флажок не имеет отметки.
Indeterminate	В этом состоянии флажок становится серым (неопределенное состояние).

Состояние Indeterminate удобно, когда пользователь должен быть уведомлен, что опция не установлена. Если нужно булевское значение, можно проверить свойство Checked.

События CheckedChanged и CheckStateChanged возникают, когда изменяются значения свойств CheckState и Checked. Для флажка с тремя состояниями понадобится присоединиться к событию CheckStateChanged. Перехват этих событий может быть полезен для установки других значений, основанных на новом состоянии CheckBox.

Класс RadioButton (рис. 22.2). Переключатель (кнопки опций) – элемент управления, унаследованный от ButtonBase.

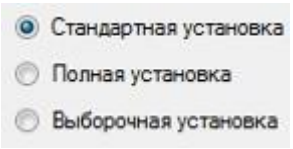


Рисунок 22.2 – Внешний вид RadioButton.

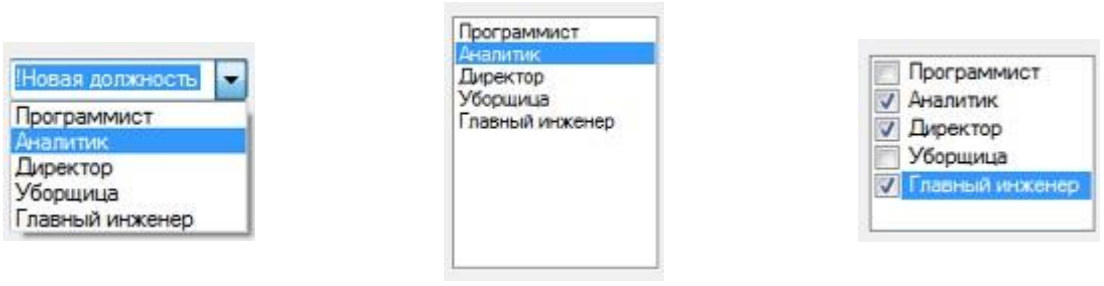
Переключатели обычно используются в группе. Переключатели позволяют пользователю выбирать одну из нескольких опций. При наличии нескольких элементов управления RadioButton в одном контейнере, только один из них может быть выбран в один и тот же момент времени.

Свойство Appearance принимает значение перечисления Appearance, которым может быть Button либо Normal. В случае установки значения Normal переключатель выглядит как маленький кружочек с меткой рядом с ним. Выбор переключателя заполняет этот кружок, а выбор другого переключателя снимает отметку с текущего выбранного переключателя и кружок делается пустым. В случае установки значения Button элемент управления выглядит подобно стандартной кнопке, но работает как переключатель – его выбор означает включение (вдавливание кнопки), а отказ от выбора – выключение (стандартное положение кнопки).

Свойство CheckedAlign определяет местоположение кружка по отношению к тексту метки. Он может быть над меткой, с любой стороны от нее либо под меткой.

Событие CheckedChanged инициируется при любом изменении свойства Checked. Это позволяет предпринимать другие действия на основе нового значения элемента управления.

Классы ComboBox, ListBox и CheckedListBox (рис. 22.3). Элементы управления ComboBox, ListBox и CheckedListBox унаследованы от класса ListControl. Этот класс предоставляет некоторую базовую функциональность управления списками. Наиболее важные аспекты использования списочных элементов управления состоят в добавлении данных и выборе данных из списка.



а) б) в)

Рисунок 22.3 – Внешний вид списков: а – ComboBox; б – ListBox; в – CheckedListBox.

На выбор списка влияет то, как он должен использоваться, и тип данных, которые будут помещены в список. Если есть необходимость во множественном выборе, или если пользователю нужно видеть несколько элементов в списке в одно и то же время, то для этого лучше всего подойдут ListBox и CheckedListBox. Если в любой момент времени в списке может быть выбран только один элемент, то предпочесть следует ComboBox.

Для добавления элементов в списки необходимо обратиться к коллекции ListBox.ObjectCollection, которая доступна через свойство Items списка.

Поскольку коллекция хранит объекты, в список может быть добавлен любой допустимый тип .NET. Чтобы идентифицировать элементы, необходимо установить два важных свойства. Первым свойством является DisplayMember. Его настройка сообщает ListControl, какое свойство объекта должно быть отображено в списке. Другое свойство – ValueMember. Оно представляет собой свойство объекта, которое должно возвращаться в качестве значения. Если в список были добавлены строки, для обоих свойств по умолчанию используются строковые значения.

После загрузки данных в список для их получения могут быть использованы свойства Selectedltem и Selectedlndex. Свойство Selectedltem возвращает объект, выбранный в данный момент. Если список предполагает множественный выбор, то должна применяться коллекция Selectedltems для получения выбранных элементов.

Если для наполнения списка применяется DataBinding, то свойство SelectedValue вернет значение свойства выбранного объекта, которое было установлено в свойстве ValueMember.

Элемент ComboBox – это комбинация поля редактирования и окна списка. Стиль ComboBox задается установкой в свойстве DropDownStyle значения перечисления DropDownStyle (таблица 22.2).

Таблица 22.2 – Стили ComboBox (возможные значения перечисления DropDownStyle).

Значение	Описание
DropDown	Текстовая часть комбинированного списка допускает редактирование, и пользователи могут вводить значение. Также они могут щелкать на кнопке со стрелочкой, чтобы отобразить раскрывающийся список элементов.
DropDownList	Текстовая часть не допускает редактирование. Пользователи должны выбирать значения из списка.
Simple	Подобно DropDown, но список является постоянно видимым.

Класс DateTimePicker (рис. 22.4). Элемент управления DateTimePicker позволяет выбирать значение даты или времени (или то и другое) во множестве разных форматов. Значение DateTime можно отобразить в любом стандартном формате времени и даты. Свойство Format принимает значение перечисления DateTimePickerFormat, которое устанавливает формат Long, Short, Time или Custom. Если свойство Format установлено в DateTimePickerFormat .Custom, с помощью свойства CustomFormat можно задать строку, представляющую формат.

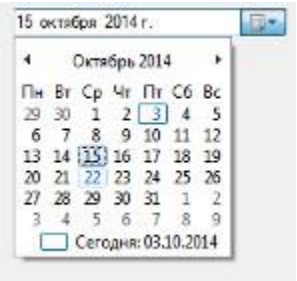


Рисунок 22.4 – Внешний вид DateTimePicker.

Свойство Text возвращает строковое представление значения DateTime, а свойство Value – объект DateTime. С помощью свойств MinDate и MaxDate можно устанавливать максимальное и минимальное значения дат. Когда пользователь щелкает на стрелке вниз, отображается календарь, позволяющий выбрать дату. Доступны свойства, которые позволяют изменить внешний вид календаря, указать фоновые цвета заголовка и месяца, а также цвета переднего плана.

3. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое

устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

4. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

5. Методика и порядок выполнения работы

Для выполнения лабораторной работы необходимо выполнить следующую последовательность действий:

1. Создайте проект приложения Windows Forms в среде MS Visual Studio.
2. Реализуйте методы для вычисления выражений в соответствии с вариантом индивидуального задания.
3. При проектировании формы необходимо учитывать следующие особенности задачи:

– выражение может быть вычислено двумя различными способами, то есть пользователь должен иметь возможность выбрать метод расчета (подумайте, какой элемент управления подходит для этого больше всего?);

– в правой части выражения присутствуют неизвестные переменные, которые пользователь может вводить с клавиатуры в текстовые поля;

– в некоторых выражениях присутствуют коэффициенты, выбор значений которых необходимо реализовать с помощью различных списков.

Индивидуальное задание.

Вариант	Выражение для вычисления
1	$Z = \begin{cases} 1 - \frac{X}{2} + \frac{Y^2}{6} - \frac{X^3}{24} + \frac{Y^4}{120} - ..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R} \frac{3 + a \cdot j}{b \cdot i}. \end{cases}$
2	$Z = \begin{cases} -\frac{1}{2} + \frac{1}{6} - \frac{X}{24} + \frac{Y}{120} - ..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R} \frac{i^2 + j}{i}. \end{cases}$
3	$Z = \begin{cases} -\frac{1}{1} + \frac{Y}{2} - \frac{X}{3} + \frac{Y}{4} - ..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R} \frac{j^2 + h \cdot i}{i + c \cdot j}. \end{cases}$
4	$Z = \begin{cases} -\frac{2}{2} + \frac{}{6} - \frac{}{24} + \frac{}{120} - ..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R} \frac{a \cdot i + b \cdot j}{c \cdot i}. \end{cases}$
5	$Z = \begin{cases} -\frac{p}{2} + \frac{p}{6} - \frac{p}{24} + \frac{p}{120} - ..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R} \frac{i^2 + b \cdot j}{c \cdot i} \end{cases}$
6	$Z = \begin{cases} -\frac{X}{2} + \frac{p \cdot X^2}{6} - \frac{p^2 \cdot X^3}{24} + \frac{p^3 \cdot X^4}{120} - ..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R} \frac{i^2 + a \cdot j^2}{i + j} \end{cases}$

7	$Z=\begin{cases} -\frac{X}{2}+\frac{X^2}{6}-\frac{X^3}{24}+\frac{X^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot i^2}{\sqrt[3]{i^3}+b\cdot j}. \end{cases}$
8	$Z=\begin{cases} -\frac{1}{2}+\frac{X}{6}-\frac{X^2}{24}+\frac{X^3}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot i}{\sqrt[3]{i^3}+j}. \end{cases}$
9	$Z=\begin{cases} -\frac{f}{2}+\frac{X\cdot f^2}{6}-\frac{X^2\cdot f^3}{24}+\frac{X^3\cdot f^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot(i+j)}{i\cdot(i+2)}. \end{cases}$
10	$Z=\begin{cases} 1-\frac{X}{2}+\frac{Y^2}{6}-\frac{X^3}{24}+\frac{Y^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{3+j}{c\cdot i}. \end{cases}$
11	$Z=\begin{cases} -\frac{1}{2}+\frac{Y}{6}-\frac{X}{24}+\frac{Y}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+j}{\sqrt{j}}. \end{cases}$
12	$Z=\begin{cases} -\frac{1}{2}+\frac{Y}{6}-\frac{X}{\sqrt{3}}+\frac{Y}{\sqrt{4}}- \\ \qquad\qquad ..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{j^2+c\cdot i}{\sqrt{j}+d\cdot j}. \end{cases}$
13	$Z=\begin{cases} -\frac{1}{2}+\frac{1}{6}-\frac{1}{24}+\frac{1}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot i+b\cdot j}{c\cdot\sqrt{i}}. \end{cases}$
14	$Z=\begin{cases} -\frac{P}{2}+\frac{P}{6}-\frac{P}{24}+\frac{P}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+b\cdot j}{c\cdot\sqrt[3]{i}}. \end{cases}$

15	$Z = \begin{cases} -\frac{X}{2} + \frac{p \cdot X^2}{6} - \frac{p^2 \cdot X^3}{24} + \frac{p^3 \cdot X^4}{120} - ..;. \\ \sum_{i=1}^{N} \sum_{j=1}^{R} \frac{i^2 + j^2}{\sqrt[3]{i^3 + j}}. \end{cases}$
16	$Z = \begin{cases} -\frac{X}{2} + \frac{2 \cdot X^2}{6} - \frac{3 \cdot X^3}{24} + \frac{4 \cdot X^4}{120} - ..;. \\ \sum_{i=1}^{N} \sum_{j=1}^{R} \frac{(i+1)^2}{\sqrt[3]{a \cdot i^3 + b \cdot j}}. \end{cases}$
17	$Z = \begin{cases} -\frac{1}{2} + \frac{X}{6} - \frac{X^2}{24} + \frac{X^3}{120} - ..;. \\ \sum_{i=1}^{N} \sum_{j=1}^{R} \frac{a \cdot i}{\sqrt[3]{i^3 + b \cdot j}}. \end{cases}$
18	$Z = \begin{cases} -\frac{f}{2} + \frac{X \cdot f^2}{6} - \frac{X^2 \cdot f^3}{24} + \frac{X^3 \cdot f^4}{120} - ..;. \\ \sum_{i=1}^{N} \sum_{j=1}^{R} \frac{a \cdot (i + j)}{b \cdot i \cdot (i + 2)}. \end{cases}$
19	$Z = \begin{cases} -\frac{1}{1} + \frac{Y}{3} - \frac{X^2}{\sqrt{5}} + \frac{Y^3}{\sqrt{7}} - ..;. \\ \sum_{i=1}^{N} \sum_{j=1}^{R} \frac{a \cdot j^2 + 2}{\sqrt[j]{b \cdot i^3 + 3}}. \end{cases}$
20	$Z = \begin{cases} -\frac{1}{2} + \frac{X}{6} - \frac{X^2}{24} + \frac{X^3}{120} - ..;. \\ \sum_{i=1}^{N} \sum_{j=1}^{R} \frac{a \cdot i + b \cdot j}{\sqrt[j]{c \cdot i}}. \end{cases}$
21	$Z = \begin{cases} -\frac{p}{1 \cdot 2} + \frac{p}{2 \cdot 6} - \frac{p}{3 \cdot 24} + \frac{p}{4 \cdot 120} - ..;. \\ \sum_{i=1}^{N} \sum_{j=1}^{R} \frac{i^2 + d \cdot j}{\sqrt[3]{c^i \cdot i}}. \end{cases}$
22	$Z = \begin{cases} -\frac{X}{2} + \frac{p \cdot X^2}{6} - \frac{p^2 \cdot X^3}{24} + \frac{p^3 \cdot X^4}{120} - ..;. \\ \sum_{i=1}^{N} \sum_{j=1}^{R} \frac{i^2 + j^2}{\sqrt[3]{i}}. \end{cases}$

23	$Z=\begin{cases} -\frac{3\cdot Y}{2}+\frac{5\cdot Y^2}{6}-\frac{7\cdot X^3}{24}+\frac{9\cdot Y^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i+k\cdot j}{c\cdot i}. \end{cases}$
24	$Z=\begin{cases} -\frac{3\cdot p^2}{2}+\frac{5\cdot p^3}{6}-\frac{7\cdot p^4}{24}+\frac{9\cdot p^5}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+b}{c^{\,i}\cdot(i+1)}. \end{cases}$
25	$Z=\begin{cases} -\frac{X}{2}+\frac{p\cdot X^2}{6}-\frac{p^2\cdot X^3}{24}+\frac{p^3\cdot X^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{(a\cdot i+j)^2}{\sqrt[3]{i}+b\cdot j}. \end{cases}$

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Какое событие элемента управления Button обрабатывается в программах чаще всего?

2. Для чего предназначен компонент CheckBox? Назовите основные

свойства класса CheckBox.

3. Опишите назначение элемента RadioButton. Какой внешний вид может принимать данный компонент? Назовите основные свойства класса RadioButton.

4. Назовите классы компонентов для представления списочной информации.

5. Опишите основные свойства класса ListBox. Чем комопнет ListBox отличается от CheckedListBox?

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [6-8].

ЗАКЛЮЧЕНИЕ

В учебном пособии представлены методические указанияк лабораторным работам, которые помогают студентам освоить программирование как род деятельности. Учебное пособие (лабораторный практикум) по дисциплине «Основы алгоритмизации и программирования» предоставляют студентам необходимый теоретический материал, а также позволяют получить практические навыки для дальнейшего самообучения, развития своих навыков.

Материал лабораторных работ представлен на базе современной среды разработки программных средств – интегрированной среды разработки Microsoft Visual Studio. Технологии, доступные разработчику с использованием данной IDE, непрерывно совершенствуются и требуют от программиста непрерывного повышения профессионального уровня. Полный спектр возможностй IDE MS VS выходит далеко за рамки данного учебного пособия, но выполнение заданий лабораторного практикума обеспечивает студентов знаниями, востребованными на современном рынке труда.

В качестве основы построения лабораторного практикума предложена технология MS .NET Framework. Данная технология предлагает широкий спектр алгоритмов и технологий разработки программного обеспечения: широкий набор библиотечных типов; специализированные синтаксические конструкции; современные примитивы программирования; компоненты разработки графических приложений; классы для работы с объектами файловой системы. Все представленные технологии в рамках одного языка позволяют студентам получить практический опыт использования современных инструментов разработки программ.

СПИСОК ЛИТЕРАТУРЫ

1. Рихтер. Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. / Дж. Рихтер. – СПб.: Питер, 2014. – 896 с.

2. Петцольд Ч. Программирование для Microsoft Windows 8. 6-е изд. / Ч. Петцольд. – СПб.: Питер. 2013 – 1008 с.

3. Хейлсберг А. Язык программирования C#. Классика Computers Science. 4-е изд. / А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд. – СПб.: Питер. 2011. – 784 с.

4. Стиллмен Э. Изучаем C#. 3-е изд. / Э. Стиллмен, Дж. Грин. – СПб.: Питер. 2014. – 816 с.

5. Язык программирования C# 5.0 и платформа .NET 4.5. 6-е изд. Пер. с англ. / Э. Троелсен. – М.:Изд. «Издательский дом Вильямс». 2013. – 1312 с.

6. Албахари Дж. C# 5.0. Справочник. Полное описание языка / Дж. Албахари, Б. Албахари. – М.:Изд. «Издательский дом Вильямс». 2013. – 1008 с.

7. Флентов, М. Библия C#. 2-е изд. / М. Флентов. – СПб.: БХВ-Петербург, 2011. – 560 с.

8. Ватсон, Б. C# 4.0 на примерах: пер. с англ. / Б. Ватсон. – СПб.: БХВ-Петербург, 2011. – 608 с.