**3 ПРОЦЕДУРЫ И ФУНКЦИИ – МЕТОДЫ КЛАССА**

Задание №1. Разработать методы f(x1, y1, x2, y2), который вычисляет длину отрезка по координатам вершин (x1, y1) и (x2, y2), и метод d(a, b, c), который вычисляет периметр треугольника по длинам сторон a, b, c. С помощью данных методов найти периметр треугольника, заданного координатами своих вершин.

Листинг программы:

try

{

int x1, y1, x2, y2, x3, y3;

Console.Write("x1=");

x1 = Convert.ToInt32(Console.ReadLine());

Console.Write("y1=");

y1 = Convert.ToInt32(Console.ReadLine());

Console.Write("x2=");

x2 = Convert.ToInt32(Console.ReadLine());

Console.Write("y2=");

y2 = Convert.ToInt32(Console.ReadLine());

Console.Write("x3=");

x3 = Convert.ToInt32(Console.ReadLine());

Console.Write("y3=");

y3 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine($"result = {d(f(x1, y1, x2, y2), f(x2, y2, x3, y3), f(x3, y3, x1, y1))}");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

double f(double x1, double y1, double x2, double y2)

{

return Math.Pow((Math.Pow((x2 - x1), 2) + Math.Pow((y2 - y1), 2)), 1 / 2);

}

double d(double a, double b, double c)

{

return a + b + c;

}

Таблица 3.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| x1=1  y1=2  x2=3  y2=4  x3=5  y3=6 | result = 3 |

Анализ результатов:

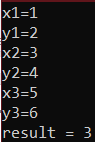


Рисунок 3.1 – Результаты работы программы

Источник: собственная разработка

Задание №2 и задание №3. Постройте таблицу значений функции y = f(x) для c шагом h. Перегрузите метод f так, чтобы его сигнатура соответствовала виду static void f (double x, out double y).

Листинг программы:

class Programm

{

public static void Main(string[] args)

{

try

{

Console.Write("введит a = ");

var a = double.Parse(Console.ReadLine());

Console.Write("введит b = ");

var b = double.Parse(Console.ReadLine());

Console.Write("введит h = ");

var h = double.Parse(Console.ReadLine());

double y;

for (double i = a; i <= b; i += h)

{

Task(i, out y);

Console.WriteLine($"f{Math.Round(i, 1)} = {y}");

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

static double Task(double x)

{

double y = 0;

if (x > 0)

{

y = x;

}

else if (x >= -1 && x <= 0)

{

y = 0;

}

else

{

y = Math.Pow(x, 2);

}

return y;

}

static void Task(double x, out double y)

{

y = 0;

if (x > 0)

{

y = x;

}

else if (x >= -1 && x <= 0)

{

y = 0;

}

else

{

y = Math.Pow(x, 2);

}

}

}

Таблица 3.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| a = 10  b = 20  h = 1 | f10 = 10  f11 = 11  f12 = 12  f13 = 13  f14 = 14  f15 = 15  f16 = 16  f17 = 17  f18 = 18  f19 = 19  f20 = 20 |

Анализ результатов:

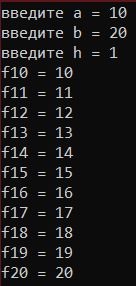


Рисунок 3.2 – Результаты работы программы

Источник: собственная разработка

Задание №4. Описать класс, реализующий тип данных «вещественная матрица». Реализовать вычитание строки заданного номера из всех остальных строк, кроме данной строки.

Листинг программы:

Console.WriteLine("Введите размер матрицы");

int n = Convert.ToInt32(Console.ReadLine());

int m = Convert.ToInt32(Console.ReadLine());

Matrix matrix = new Matrix();

double[,] a = new double[n, m];

matrix.m = m;

matrix.n = n;

Console.WriteLine("Введите элементы матрицы");

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

a[i, j] = Convert.ToDouble(Console.ReadLine());

}

}

Console.WriteLine("Введите строку");

int b = Convert.ToInt32(Console.ReadLine());

matrix.PrintMatrix(a, b);

matrix.Task(a, b, n, m);

Console.ReadLine();

class Matrix

{

public int n;

public int m;

public void PrintMatrix(double[,] a, int b)

{

Console.WriteLine("Ваша матрица");

for (int i = 0; i < n; i++) // вывод нашей матрицы

{

for (int j = 0; j < m; j++)

{

Console.Write("{0}{1}", a[i, j], "\t");

}

Console.WriteLine();

}

}

public void Task(double[,] a, int b, int n)

{

double[,] c = new double[n, m];

Console.WriteLine("Исходная матрица");

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++) // вычитание элементов в строке

{

c[i, j] = a[i, j] - a[b, j];

}

}

for (int j = 0; j < m; j++) // присвоение элементам исходного значения (до вычитания)

{

c[b, j] = c[b, j] + a[b, j];

}

for (int i = 0; i < n; i++) //вывод полученной матрицы

{

for (int j = 0; j < m; j++)

{

Console.Write("{0}{1}", c[i, j], "\t");

}

Console.WriteLine();

}

}

public void Task(double[,] a, int b, int n, int m) //перегрузка методов, беру методы, ибо оператора нормального нет

{

double[,] c = new double[n, m];

Console.WriteLine("Исходная матрица");

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++) // вычитание элементов в строке

{

c[i, j] = a[i, j] - a[b, j];

}

}

for (int j = 0; j < m; j++) // присвоение элементам исходного значения (до вычитания)

{

c[b, j] = c[b, j] + a[b, j];

}

for (int i = 0; i < n; i++) //вывод полученной матрицы

{

for (int j = 0; j < m; j++)

{

Console.Write("{0}{1}", c[i, j], "\t");

}

Console.WriteLine();

}

}

}

Таблица 3.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Введите размер матрицы  2  2  Введите элементы матрицы  1  2  3  4  Введите строку  1 | Ваша матрица  1 2  3 4  Исходная матрица  -2 -2  3 4 |

Анализ результатов:

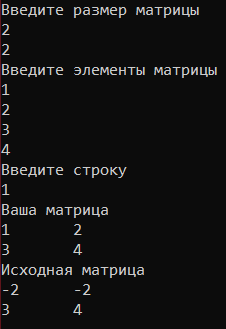


Рисунок 3.3 – Результаты работы программы

Источник: собственная разработка

Задание №5. Описать класс «аэропорт», содержащий закрытый массив самолетов. Обеспечить следующие возможности: вывод информации о самолете по номеру рейса с помощью индекса; вывод информации о самолетах, отправляющихся в течение часа после введенного с клавиатуры времени; вывод информации о самолетах, отправляющихся в заданный пункт назначения; перегруженную операцию сравнения, выполняющую сравнение времени отправления двух самолетов.

Информация должна быть отсортирована по времени отправления.

Листинг программы:

try

{

Aeroport aeroport = new Aeroport();

Aeroplane plane = new Aeroplane("Москва", 999999, new DateTime(2011, 5, 26, 20, 0, 0, 0));

Aeroplane plane1 = new Aeroplane("Киев", 999998, new DateTime(2011, 5, 26, 20, 10, 0, 0));

Aeroplane plane2 = new Aeroplane("Минск", 999997, new DateTime(2011, 5, 26, 20, 20, 0, 0));

Aeroplane plane3 = new Aeroplane("Париж", 999996, new DateTime(2011, 5, 26, 22, 0, 0, 0));

aeroport.AddAeroplaneRange(plane, plane1, plane2, plane3);

aeroport.GetInforation(0);//информация о рейсе

aeroport.ThisHour(new DateTime(2011, 5, 26, 20, 0, 0, 0));//в течение часа

aeroport.TripFromPointName("Париж");//информация о рейсе в Париж

Console.WriteLine("-------------------------------------------");

aeroport.InfoSort();

Console.ReadLine();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

//Класс самолет

class Aeroplane

{

private string \_pointName;//пункт назначения

private int \_number; //номер рейса

private DateTime \_time; //время вылета

public Aeroplane(string Pointname, int Number, DateTime Time)

{

\_pointName = Pointname;

\_number = Number;

\_time = Time;

}

public int Number

{

get { return \_number; }

set { \_number = value; }

}

public string PointName

{

get { return \_pointName; }

set { \_pointName = value; }

}

public DateTime TimeToFly

{

get { return \_time; }

set { \_time = value; }

}

public static bool operator >(Aeroplane plane1, Aeroplane plane2) //перегрузка операторов

{

return plane1.TimeToFly > plane2.TimeToFly;

}

public static bool operator <(Aeroplane plane1, Aeroplane plane2) //перегрузка операторов

{

return plane1.TimeToFly < plane2.TimeToFly;

}

public override string ToString()

{

return $"{\_pointName},{\_number},{\_time}";

}

}

class Aeroport

{

List<Aeroplane> planes; // принимаем массив самолетов

public Aeroport()

{

planes = new List<Aeroplane>(); // инициализируем переменную с массивом

}

public void AddAeroplane(Aeroplane plane) // добавление самолетов

{

planes.Add(plane);

}

public void AddAeroplaneRange(params Aeroplane[] Planes)//параметр массив, который будем заполнять // добавление нескольких самолетов одновременно

{

planes.AddRange(Planes);

}

public void GetInforation(int index) // получение информации по индексу

{

//var isMore = planes[0] > planes[1]; // проверка перегрузки

//Console.WriteLine(isMore);

if (index > planes.Count - 1)

{

throw new Exception("Рейса не существует");

}

Console.WriteLine("Номер рейса: " + planes[index].Number);

Console.WriteLine("Пункт назначения: " + planes[index].PointName);

Console.WriteLine("Время вылета: " + planes[index].TimeToFly);

}

public void InfoSort() //сортировка по времени

{

var temp = planes[0];

for (int i = 0; i < planes.Count - 1; i++)

{

for (int j = i + 1; j < planes.Count; j++)

{

if (planes[i].TimeToFly > planes[j].TimeToFly)

{

temp = planes[i];

planes[i] = planes[j];

planes[j] = temp;

}

}

}

Console.WriteLine("Заданный пункт назначения, Номер рейса, Дата и время отправления соответственно (сортировка по времени):");

for (int i = 0; i < planes.Count; i++)

{

Console.WriteLine(planes[i]);

}

}

public void ThisHour(DateTime time) // получение данных о самолетах, которые вылетят в течении часа

{

DateTime time1 = new DateTime(time.Year, time.Month, time.Day, time.Hour + 1,

time.Minute, time.Second, DateTimeKind.Local);

Console.WriteLine("\nВ течение часа вылетят самолеты:");

for (int i = 0; i < planes.Count; i++)

if (planes[i].TimeToFly > time && planes[i].TimeToFly < time1)

GetInforation(i);

Console.WriteLine("Список окончен\n");

}

public void TripFromPointName(string PointName) // получение самолетов, которые летят в определенную точку

{

Console.WriteLine("В " + PointName + " летят следующие самолеты: ");

for (int i = 0; i < planes.Count; i++)

if (planes[i].PointName == PointName)

{

GetInforation(i);

}

Console.WriteLine("Список окончен\n");

}

}

Таблица 3.4 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Номер рейса: 999999  Пункт назначения: Москва  Время вылета: 26.05.2011 20:00:00  В течение часа вылетят самолеты:  Номер рейса: 999998  Пункт назначения: Киев  Время вылета: 26.05.2011 20:10:00  Номер рейса: 999997  Пункт назначения: Минск  Время вылета: 26.05.2011 20:20:00  Список окончен  В Париж летят следующие самолеты:  Номер рейса: 999996  Пункт назначения: Париж  Время вылета: 26.05.2011 22:00:00  Список окончен  -------------------------------------------  Заданный пункт назначения, Номер рейса, Дата и время отправления соответственно (сортировка по времени):  Москва,999999,26.05.2011 20:00:00  Киев,999998,26.05.2011 20:10:00  Минск,999997,26.05.2011 20:20:00  Париж,999996,26.05.2011 22:00:00 |

Анализ результатов:

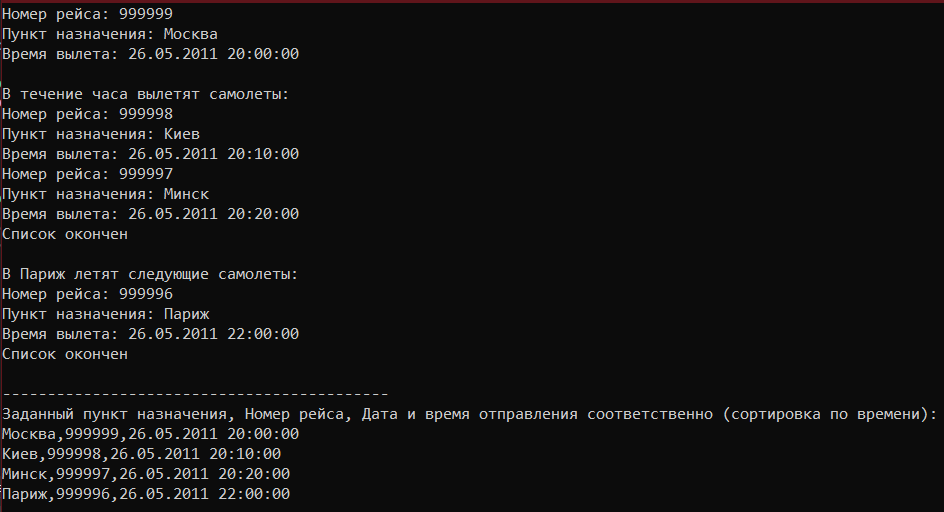


Рисунок 3.4 – Результаты работы программы

Источник: собственная разработка