1. **Процедуры и функции – методы класса**

Задание 1. Решение простейших задач. Разработать метод F(x), который в двухзначном числе меняет цифры местами, а остальные числа оставляет без изменения.

Листинг программы:

namespace Task\_1

{

class Task\_1

{

public static void Main(string[] args)

{

Console.Write("Введите число: ");

int num = int.Parse(Console.ReadLine());

Console.WriteLine($"После функции: {F(num)}");

}

public static int F(int num)

{

int result = 0;

if (num > 9 && num < 100)

{

result = num / 10 + num % 10 \* 10;

}

else

{

result = num;

}

return result;

}

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 23 | 32 |

Анализ результатов:



Рисунок 1.1 – Результат работы программы

Источник: собственная разработка

Задание 2. Постройте таблицу значений функции y = F(x) для x ∈ [a, b] с шагом h.

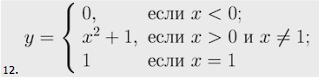


Рисунок 2.1 – Функция

Источник: собственная разработка

Листинг программы:

namespace Task\_2

{

class Task\_2

{

public static void Main(string[] args)

{

Console.Write("Введите начальное значение: ");

double start = double.Parse(Console.ReadLine());

Console.Write("Введите конечное значение: ");

double finish = double.Parse(Console.ReadLine());

Console.Write("Введите шаг: ");

double step = double.Parse(Console.ReadLine());

Program pr = new Program();

foreach (double num in pr.F(start, finish, step))

{

Console.WriteLine($"F(x)= {num}");

}

}

}

class Program

{

public List<double> F(double start, double finish, double h)

{

List<double> list = new List<double>();

for (double i = start; i <= finish; i += h)

{

if (i < 0)

{

list.Add(0);

}

else if (i > 0 && i != 1)

{

list.Add(Math.Pow(i, 2) +1);

}

else

{

list.Add(1);

}

}

return list;

}

}

}

Таблица 2.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| -4,8 5 1,8 | 0 0 0 1,8099... 8,2900… 21,25 |

Анализ результатов:

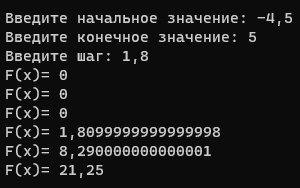


Рисунок 2.2 – Результат работы программы

Источник: собственная разработка

Задание 3. Перегрузите метод F из предыдущего раздела так, чтобы его сигнатура соответствовала виду static void f (double x, out double y).

Листинг программы:

namespace Task\_3

{

class Task\_3

{

public static void Main(string[] args)

{

Console.Write("Введите начальное значение: ");

double start = double.Parse(Console.ReadLine());

Console.Write("Введите конечное значение: ");

double finish = double.Parse(Console.ReadLine());

Console.Write("Введите шаг: ");

double step = double.Parse(Console.ReadLine());

Program pr = new Program();

for (double i = start; i <= finish; i += step)

{

double result = 0;

pr.F(i, out result);

Console.WriteLine($"F(x)= {result}");

}

}

}

class Program

{

public void F(double x, out double y)

{

if (x < 0)

{

y =0;

}

else if (x > 0 && x != 1)

{

y = (Math.Pow(x, 2) + 1);

}

else

{

y = 1;

}

}

}

}

Таблица 3.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| -4 5 1 | 0 0 0 0 1 1 5 10 17 26 |

Анализ результатов:

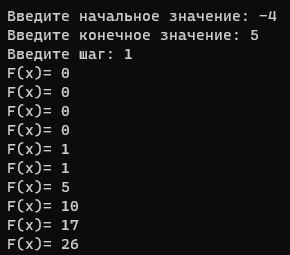


Рисунок 3.1 – Результат работы программы

Источник: собственная разработка

Задание 4. Перегрузка операций. Создайте проект, в котором опишите класс для решения задачи Вашего варианта. Разрабатываемый класс должен содержать следующие элементы: скрытые и открытые поля, конструкторы, перегруженные операции. В программе должна выполняться проверка всех разработанных элементов класса.

Описать класс для работы с двумерным массивом целых чисел. Реализовать возможность нахождения числа, полученного перемножением положительных элементов массива, меньших 10.

Листинг программы:

using System.Data;

using System.Drawing;

namespace Task\_4

{

class Task\_4

{

public static void Main(string[] args)

{

Console.Write("Введите размерность двумерного массива:\nСтрок= ");

int row = int.Parse(Console.ReadLine());

Console.Write("Колонок= ");

int colum = int.Parse(Console.ReadLine());

Console.WriteLine("Action One");

Array array = new Array(row, colum);

InputNumInMas(array, row, colum);

Console.WriteLine($"Перемножением положительных элементов массива, меньших 10: {array.ArrayAction()}");

Console.WriteLine("\nAction Two");

Array ar = new Array();

ar.CreateArrayMas(row, colum);

InputNumInMas(ar, row, colum);

int[,] mas = ar.GetArray();

Console.WriteLine($"Перемножением положительных элементов массива, меньших 10: {ar.ArrayAction(mas)}");

}

private static void InputNumInMas(Array array, int row, int colum)

{

Console.WriteLine("Введите значения массива: ");

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

{

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

{

Console.Write(i + "," + j + "=");

int num = int.Parse(Console.ReadLine());

array.SetNumArray(num, i, j);

}

}

}

}

public class Array

{

private int[,] masDuo;

public Array()

{

}

public Array(int row, int colum)

{

masDuo = new int[row, colum];

}

public void SetNumArray(int x, int i, int j)

{

masDuo[i, j] = x;

}

public void CreateArrayMas(int row, int colum)

{

masDuo = new int[row, colum];

}

public int [,] GetArray()

{

return masDuo;

}

public int ArrayAction()

{

int result = 0;

foreach (var i in masDuo)

{

if (i >= 0 && i < 10)

{

result += i;

}

}

return result;

}

public int ArrayAction(int[,] mas)

{

int result = 0;

foreach (var i in mas)

{

if (i >= 0 && i < 10)

{

result += i;

}

}

return result;

}

}

}

Таблица 4.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 2 4 (2 -5 -3 -6 -4 -6 -2 0) (2 2 2 5 45 7 -100 23) | 2 18 |

Анализ результатов:

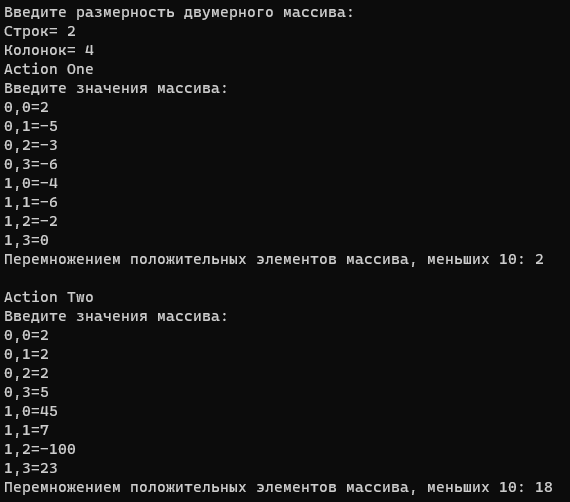


Рисунок 4.1 – Результат работы программы

Источник: собственная разработка

Задание 5. Каждый разрабатываемый класс должен, как правило, содержать следующие элементы: скрытые поля, конструкторы с параметрами и без параметров, методы; свойства, перегруженные операции. В программе должна выполняться проверка всех разработанных элементов класса.

Описать класс «вокзал», содержащий закрытый массив поездов.

Обеспечить следующие возможности:

 вывод информации о поезде по номеру с помощью индекса;

 вывод информации о поездах, отправляющихся после введенного с клавиатуры времени;

 перегруженную операцию сравнения, выполняющую сравнение времени отправления двух поездов;

 вывод информации о поездах, отправляющихся в заданный пункт назначения.

Информация должна быть отсортирована по времени отправления.

Написать программу, демонстрирующую все разработанные элементы классов.

Листинг программы:

namespace Task\_5

{

class Task\_5

{

public static void Main(string[] args)

{

RailwayStation rs = new RailwayStation();

AddTrain(rs);

bool flag = true;

while (flag)

{

Console.WriteLine("Выберите действие:\n" +

"1-информация о поезде по номеру\n" +

"2-информации о поездах, отправляющихся после введенного времени\n" +

"3-информации о поездах, отправляющихся в заданный пункт назначения\n" +

"0-закрыть");

int point = int.Parse(Console.ReadLine());

if (point >= 0 && point <= 3)

{

switch (point)

{

case 1:

ShowInfoTrainNum(rs);

break;

case 2:

ShowLastTimeInfoTrain(rs);

break;

case 3:

ShowFinishPunkt(rs);

break;

case 0:

flag = false;

break;

}

}

}

}

public static void ShowInfoTrainNum(RailwayStation rs)

{

int num = 0;

bool flag = true;

while (flag)

{

try

{

Console.Write("Введите индекс: ");

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

flag = false;

}

catch

{

Console.WriteLine("Неверный формат");

}

}

Console.WriteLine(rs[num].ToString());

}

public static void ShowLastTimeInfoTrain(RailwayStation rs)

{

List<string> train; ;

rs.LastTimeInfoTrain(out train);

foreach (var tr in train)

{

Console.WriteLine(tr);

}

}

public static void ShowFinishPunkt(RailwayStation rs)

{

Console.Write("Введите пункт назначения: ");

string punkt = Console.ReadLine();

List<Train> list = new List<Train>(rs.ShowFinishPunct(punkt));

foreach (var tr in list)

{

Console.WriteLine(tr.ToString());

}

}

public static void AddTrain(RailwayStation rs)

{

for (int i = 0; i < 3; i++)

{

int num = 0;

bool flag = true;

while (flag)

{

try

{

Console.Write("Введите номер поезда: ");

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

flag = false;

}

catch

{

Console.WriteLine("Неверный формат номера");

}

}

Console.Write("Введите конечный пункт назначения: ");

string finishPunct = Console.ReadLine();

rs.Add(finishPunct, num);

}

}

}

class RailwayStation

{

private List<Train> trains = new List<Train>();

public Train this[int index] //пользовательский индексатор для класса

{

get

{

return trains[index];

}

set

{

if (index >= 0 && value is Train)

{

trains.Add(value);

trains.Sort();

}

}

}

public void Add(string finishPunct, int num) //метод добавления поезда в коллекцию

{

Train train = new Train(finishPunct, num);

trains.Add(train);

trains.Sort();

}

public void LastTimeInfoTrain(out List<string> train)

{

train = new List<string>();

Train tr = new Train("null", 1);

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

{

if (tr < this[i])

{

train.Add(this[i].ToString());

}

}

}

public List<Train> ShowFinishPunct(string punkt) //метод для вывода ин-ии о поездах, отправляющихся в заданный пункт назначения

{

List<Train> list = new List<Train>();

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

{

if (trains[i].FinishPunct == punkt)

{

list.Add(trains[i]);

}

}

return list;

}

}

class Train : IComparable

{

public Train(string finishPunct, int namber)

{

this.finishPunct = finishPunct;

this.namber = namber;

SetTime();

}

public Train(string finishPunct, int namber, DateTime time)

{

this.finishPunct = finishPunct;

this.namber = namber;

this.time = time;

}

String finishPunct;

public String FinishPunct

{

get { return finishPunct; }

set { finishPunct = value; }

}

int namber;

public int Namber

{

get { return namber; }

set

{

if (value > 0)

namber = value;

}

}

DateTime time;

public DateTime Time

{

get { return time; }

set { time = value; }

}

void SetTime()

{

while (true)

{

int hour = 0;

int minute = 0;

Console.Write("Введите время отправления поезда [hour.minute]: ");

string t = Console.ReadLine();

string h = "";

string m = "";

int q;

bool flag = false;

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

{

if (!flag)

{

if (Int32.TryParse(Convert.ToString(t[i]), out q))

{

h = h + t[i];

}

else

{

flag = true;

continue;

}

}

if (flag)

{

if (Int32.TryParse(Convert.ToString(t[i]), out q))

{

m = m + t[i];

}

}

}

try

{

hour = Int32.Parse(h);

minute = Int32.Parse(m);

Time = new DateTime(2023, 12, 12, hour, minute, 0);

return;

}

catch

{

Console.WriteLine("Введен неверный формат времени");

}

}

}

public int CompareTo(object input)

{

if (input is Train)

{

Train p1 = (Train)input;

if (this.Time > p1.Time)

return 1;

else if (this.Time < p1.Time)

return -1;

else

return 0;

}

return 0;

}

public string ToString()

{

string info = $"Поезд №{namber} следует в пункт назначения {finishPunct}. Время отправления: {time.Hour}.{time.Minute}!";

return info;

}

public static bool operator >(Train p1, Train p2)

{

if (p1.Time > p2.Time)

return true;

return false;

}

public static bool operator <(Train p1, Train p2)

{

if (p1.Time < p2.Time)

return true;

return false;

}

public static bool operator ==(Train p1, Train p2)

{

if (p1.Time == p2.Time)

return true;

return false;

}

public static bool operator !=(Train p1, Train p2)

{

if (p1.Time != p2.Time)

return true;

return false;

}

}

}

Таблица 5.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 456 Grodno 12.10  45 Minsk 7.04  8732 Riga 10.22  1 2 3 | Информация о поездах |

Анализ результатов:

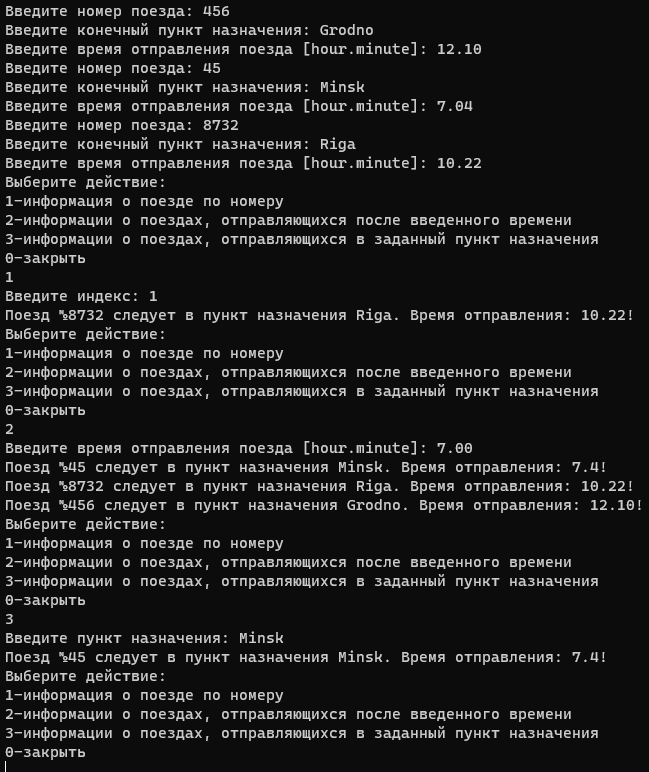


Рисунок 5.1 – Результат работы программы

Источник: собственная разработка