Практическая работа № 2.  
МЕТОДЫ

**Цель практической работы**

Закрепление теоретических знаний по обработке текстов с помощью регулярных выражений.

**Постановка задачи**

**Метод –** блок кода, содержащий ряд инструкций.

[тип возвращаемого значения] [Имя]([аргументы])

{

// Тело метода

// Напримр:

float a = 10.2f;

float b = 3.4f;

return a+b;

}

При передаче аргументов в метод они копируются. Если это тип значения, то копируются сами значения, если тип ссылочный, то копируется ссылка.

**Пример.**

float getSum(float a, float b)

{

return a + b;

}

**void** – специальное слово, означающее, что метод **не возвращает значения**;

**return** – специальное слово, **завершающее** выполнение метода и **возвращающее** результат (при его наличии).

**Пример** создания методов.

var var1 = Convert.ToDouble(Console.ReadLine());

var var2 = Convert.ToDouble(Console.ReadLine());

WriteSum(var1, var2);

var var3 = GetSum(var1, var2);

void WriteSum(double a, double b)

{

Console.WriteLine(a + b);

}

double GetSum(double a, double b)

{

return a + b;

}

**Пример создания метода для решения квадратного уравнения:**

calculateSquareEquation();

Console.WriteLine();

calculateSquareEquation();

void calculateSquareEquation()

{

Console.WriteLine("Введеите через пробел коэффициенты A, B, C квадратного уравнения Ax^2+Bx+C=0");

var input = Console.ReadLine().Split(" ");

var A = double.Parse(input[0]);

var B = double.Parse(input[1]);

var C = double.Parse(input[2]);

var D = B \* B - 4 \* A \* C;

var x1 = (-B + Math.Sqrt(D)) / (2 \* A);

var x2 = (-B - Math.Sqrt(D)) / (2 \* A);

Console.WriteLine($"Корни уравнения {A}x^2+{B}x+{C}=0");

Console.WriteLine($"x1={x1}\tx2={x2}");

}

var a1 = 5;

var arr1 = new int[5];

Console.Write(a1);

Console.Write(arr1[0]);

a1 = Method1(a1);

Method2(arr1);

Console.Write(a1);

Console.Write(arr1[0]);

int Method1(int a1)

{

a1 = 10;

return 15;

}

void Method2(int[] arr)

{

arr[0] = 11;

}

**Рекурсия – вызов метода из самого метода.**

Console.WriteLine(Factorial(5));

double Factorial(double d)

{

if (d <= 1)

return 1;

return d \* Factorial(d - 1);

}

Факториал натурального числа n определяется как произведение всех натуральных чисел от 1 до n включительно.

Любой рекурсивный алгоритм можно переделать в не рекурсивный, например, с помощью бесконечных циклов или специальных коллекций данных.

**ref** – ключевое слово, указывает, что значение передается по ссылке.

Используется, когда необходимо изменить значение переменной внутри метода (если тип значение).

**out** – инициирует передачу аргументов по ссылке.

Используется, когда нужно вернуть несколько значений из метода.

var a1 = 5;

Console.Write($"{a1} ");

Method1(ref a1);

Method2(out int a2);

Console.Write($"{a1} ");

Console.Write($"{a2} ");

void Method1(ref int a)

{

a = 10;

}

void Method2(out int a)

{

a = 7;

}

**params** – ключевое слово, означающее, что метод принимает переменное число аргументов (одного типа). Используется для удобства написания кода.

Method1(1, 2, 3, 4, 5);

void Method1(params int[] arguments)

{

foreach (var argument in arguments)

{

Console.Write($"{argument} ");

}

}

**Задание на практическую работу**

1. Разработать консольное приложение для решения задач согласно варианту.
2. Предусмотреть понятный интерфейс ввода и вывода данных на консоль.
3. Реализовать меню для выбора задания, обеспечивающее возможность многократного выполнения различных заданий.
4. После завершения выполнения выбранного задания пользователь должен возвращаться в меню для повторного выбора.
5. Завершение работы программы должно осуществляться при явном выборе соответствующего пункта меню (например, 'Выход').

**Задание 1.**

Витя работает недалеко от одной из станций кольцевой линии метро, а живет рядом с другой станцией той же линии. Требуется выяснить, мимо какого наименьшего количества промежуточных станций необходимо проехать Вите по кольцу, чтобы добраться с работы домой.

**Входные данные**

Заданы три числа: сначала N – общее количество станций кольцевой линии, а затем i и j – номера станции, на которой Витя садится, и станции, на которой он должен выйти. Станции пронумерованы подряд натуральными числами 1, 2, 3, …, N (1-я станция – соседняя с N-й), N не превосходит 100. Числа i и j не совпадают. Все числа разделены пробелом.

**Выходные данные**

Требуется вывести минимальное количество промежуточных станций (не считая станции посадки и высадки), которые необходимо проехать Вите.

**Примеры**

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT** | **OUTPUT** |
| 1 | 100 5 6 | 0 |
| 2 | 10 1 9 | 1 |

**Задание 2.**

Напишите программу, которая выводит элемент из строки Y и столбца X матрицы размера N×M, которая заполнена следующим образом:

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **1** | **2** | **3** |
| **7** | **6** | **5** | **4** |
| **8** | **9** | **10** | **11** |

**Входные данные**

Натуральные числа N, M, Y, X (Y ≤ N ≤ 50; X ≤ M ≤ 50). N - количество строк матрицы, M - количество столбцов матрицы, Y и X - номера строки и столбца искомого элемента.

**Выходные данные**

Выведите искомый элемент.

**Примеры**

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT** | **OUTPUT** |
| 1 | 3 4 2 3 | 5 |
| 2 | 5 2 3 1 | 4 |
| 3 | 3 4 3 2 | 9 |

**Задание 3.**

Клетки шахматной доски пронумерованы числами от 1 до 64 по строкам слева направо и снизу-вверх. Напишите программу, которая по заданному номеру клетки определяет номера всех клеток, имеющих с ней общую сторону.

**Входные данные**

Одно целое число от 1 до 64.

**Выходные данные**

Выведите через пробел в порядке возрастания номера всех клеток, имеющих с заданной общую сторону.

**Примеры**

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT** | **OUTPUT** |
| 1 | 2 | 1 3 10 |
| 2 | 64 | 56 63 |

**Пример разработки приложения**

**Задание 1.**

Витя работает недалеко от одной из станций кольцевой линии метро, а живет рядом с другой станцией той же линии. Требуется выяснить, мимо какого наименьшего количества промежуточных станций необходимо проехать Вите по кольцу, чтобы добраться с работы домой.

**Входные данные**

Заданы три числа: сначала N – общее количество станций кольцевой линии, а затем i и j – номера станции, на которой Витя садится, и станции, на которой он должен выйти. Станции пронумерованы подряд натуральными числами 1, 2, 3, …, N (1-я станция – соседняя с N-й), N не превосходит 100. Числа i и j не совпадают. Все числа разделены пробелом.

**Выходные данные**

Требуется вывести минимальное количество промежуточных станций (не считая станции посадки и высадки), которые необходимо проехать Вите.

**Примеры**

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT** | **OUTPUT** |
| 1 | 100 5 6 | 0 |
| 2 | 10 1 9 | 1 |

**Шаг 1: Понимание задачи**

Нам нужно определить минимальное количество промежуточных станций, которые Витя проедет, перемещаясь от станции i до станции j на кольцевой линии метро. Кольцевая линия означает, что станции расположены по кругу, и можно двигаться в двух направлениях: по часовой стрелке и против часовой стрелки. Необходимо найти направление, в котором промежуточных станций будет меньше.

**Шаг 2: Анализ входных данных**

Входные данные:

* N — общее количество станций (кольцо, поэтому станция 1 соседствует со станцией N).
* i — станция посадки (начальная станция).
* j — станция выхода (конечная станция).
* Условия: i ≠ j, N ≤ 100.

**Шаг 3: Определение направлений движения**

На кольцевой линии есть два пути от i до j:

1. **По часовой стрелке** (прямое направление):
   * Если i < j, количество промежуточных станций: j - i - 1.
   * Если i > j, количество промежуточных станций: (N - i) + j - 1.
2. **Против часовой стрелки** (обратное направление):
   * Если i < j, количество промежуточных станций: (i - 1) + (N - j) - 1.
   * Если i > j, количество промежуточных станций: i - j - 1.

Но можно упростить:

* Прямое расстояние: (j - i + N) % N - 1.
* Обратное расстояние: (i - j + N) % N - 1.  
  Минимальное из этих двух значений и будет ответом.

**Шаг 4: Реализация логики**

1. Вычислить расстояние по часовой стрелке: distance1 = (j - i + N) % N.
2. Вычислить расстояние против часовой стрелки: distance2 = (i - j + N) % N.
3. Минимальное количество промежуточных станций: min(distance1, distance2) - 1.

**Шаг 5: Написание кода на C#**

while(true)

{

var task = Console.ReadLine();

switch (task)

{

case "1": RunTask1(); break;

}

}

void RunTask1()

{

Console.WriteLine("""

Задание 1.

Заданы три числа: сначала N – общее количество станций кольцевой линии,

а затем i и j – номера станции, на которой Витя садится, и станции,

на которой он должен выйти. Стан-ции пронумерованы подряд натуральными числами

1, 2, 3, …, N (1-я станция – соседняя с N-й), N не превосходит 100.

Числа i и j не совпадают. Все числа разделены пробелом.

""");

Console.WriteLine("Введите N, i, j через пробел:");

string[] input = Console.ReadLine().Split();

int N = int.Parse(input[0]);

int i = int.Parse(input[1]);

int j = int.Parse(input[2]);

// Вычисляем расстояние в обоих направлениях

int distanceClockwise = (j - i + N) % N;

int distanceCounterClockwise = (i - j + N) % N;

// Находим минимальное расстояние

int minDistance = Math.Min(distanceClockwise, distanceCounterClockwise);

// Количество промежуточных станций (без учёта i и j)

int intermediateStations = minDistance - 1;

Console.WriteLine(intermediateStations);

}

**Шаг 6: Пояснение кода**

1. **Чтение входных данных**: Считываем строку, разбиваем её на части и преобразуем в числа N, i, j.
2. **Вычисление расстояний**:
   * distanceClockwise — расстояние от i до j по часовой стрелке.
   * distanceCounterClockwise — расстояние от i до j против часовой стрелки.
3. **Определение минимального расстояния**: Используем Math.Min для выбора меньшего из двух расстояний.
4. **Вычисление промежуточных станций**: Вычитаем 1, так как нам нужны только промежуточные станции (без учёта i и j).

**Шаг 7: Тестирование**

Проверим программу на примерах из условия:

1. **Вход**: 100 5 6 → **Ожидаемый вывод**: 0.
   * По часовой стрелке: (6 - 5 + 100) % 100 = 1 → промежуточных станций: 1 - 1 = 0.
   * Против часовой стрелки: (5 - 6 + 100) % 100 = 99 → 99 - 1 = 98.
   * Минимум: 0.
2. **Вход**: 10 1 9 → **Ожидаемый вывод**: 1.
   * По часовой стрелке: (9 - 1 + 10) % 10 = 8 → 8 - 1 = 7.
   * Против часовой стрелки: (1 - 9 + 10) % 10 = 2 → 2 - 1 = 1.
   * Минимум: 1.

Дополнительные тесты:

* 5 1 3 → 1 (по часовой стрелке: 2 - 1 = 1, против: 3 - 1 = 2 → минимум 1).
* 5 4 1 → 1 (по часовой стрелке: (1 - 4 + 5) % 5 = 2 → 1, против: (4 - 1 + 5) % 5 = 3 → 2 → минимум 1).

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2022 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Open Office (Свободно распространяемое программное обеспечение).
4. Браузер (Свободно распространяемое программное обеспечение).

# Список литературы

* + - 1. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 285 c.
      2. Биллиг, В. A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В. A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c. — 978-5-4487-0145-0. — Режим доступа: http://www.iprbookshop.ru/72339.html
      3. Павловская, Т. А. Программирование на языке высокого уровня C# [Электронный ресурс] / Т. А. Павловская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 245 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73713.html
      4. Агапов, В. П. Основы программирования на языке С# [Электронный ресурс] : учебное пособие / В. П. Агапов. — Электрон. текстовые данные. — М. : Московский государственный строительный университет, ЭБС АСВ, 2012. — 128 c. — 978-5-7264-0576-6. — Режим доступа: http://www.iprbookshop.ru/16366.html
      5. Медведев, М. А. Программирование на СИ# [Электронный ресурс] : учебное пособие / М. А. Медведев, А. Н. Медведев ; под ред. А. В. Присяжный. — Электрон. текстовые данные. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2015. — 64 c. — 978-5-7996-1561-1. — Режим доступа: http://www.iprbookshop.ru/69667.html
      6. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual С# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2011.— 180 c
      7. Уйманова Н.А. Основы объектно-ориентированного программирования [Электронный ресурс]: практикум/ Уйманова Н.А., Таспаева М.Г.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017.— 156 c.— Режим доступа: http://www.iprbookshop.ru/78808.html.— ЭБС «IPRbooks»
      8. Новиков П.В. Объектно-ориентированное программирование [Электронный ресурс]: учебно-методическое пособие к лабораторным работам/ Новиков П.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2017.— 124 c.— Режим доступа: http://www.iprbookshop.ru/64650.html.— ЭБС «IPRbooks»