

МИНИСТЕРСТВО НАУКИ И ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Кафедра «Вычислительные системы и информационная безопасность»

**Методические указания к контрольной работе по дисциплине «Межплатформенное программирование»**

**для студентов 3-го курса**

**направления подготовки 09.03.02 Информационные системы и технологии**

**заочной формы обучения**

Ростов-на-Дону

2024 г.

Составитель к.т.н., доцент, доцент А.Р. Айдинян,

**УДК 681.3**

Подготовлено на кафедре «Информационная безопасность в вычислительных системах и сетях»

Методические рекомендации  
к контрольной работе по дисциплине   
«Межплатформенное программирование»

/ ДГТУ, Ростов-на-Дону, 2024, 24 с.

Методические рекомендации для студентов заочной формы обучения представляют собой комплекс рекомендаций и разъяснений, позволяющих студенту оптимальным образом организовать процесс выполнения контрольной работы.

УДК 681.3

1. **Алгоритм выбора варианта контрольной работы**

Контрольная работа заключается в выполнении заданий, приведенных в данных методических указаниях. При выполнении каждого задания необходимо привести:

— формулировку задания;

— описание алгоритма;

— листинг программы;

— введенные и полученные данные.

Номер варианта выбирается по следующему алгоритму:

Для выбора варианта необходимо взять предпоследнюю и последнюю цифры номера зачетной книжки. Номер варианта находится на пересечении соответствующей строки и столбца \*.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Последняя цифра номера зачетной книжки | | | | | | | | | | |
|  |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Предпоследняя цифра номера зачетной книжки | 0 | 1 | 2 | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 2 | 1 | 2 | 7 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 3 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 4 | 21 | 22 | 23 | 24 | 25 | 6 | 7 | 8 | 9 | 10 |
| 5 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 6 | 1 | 2 | 10 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 7 | 11 | 12 | 13 | 14 | 15 | 21 | 22 | 23 | 24 | 25 |
| 8 | 21 | 22 | 23 | 24 | 25 | 6 | 7 | 8 | 9 | 10 |
| 9 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 13 | 14 | 15 |

Например, для зачетки с номером 123456 необходимо взять номер варианта из 5-ой строки и 6-го столбца (вариант 17).

**\* Для тех, кто имеет опыт разработки программ: можно взять индивидуальное задание у преподавателя с использованием предпочтительных для работодателя сред и языков программирования для разработки программы для нужд предприятия – места работы.**

1. **Среда и язык программирования**

Для выполнения контрольной работы необходимо использовать одну из сред и языков программирования, которые позволяют разработать межплатформенную программу.

К ним относятся среды, поддерживающие библиотеки .NET, QT, а также языки C#, C++, Java, Python и т.д.

Для выполнения контрольной работы можно использовать среды Visual Studio, QT Creator, и т.д.

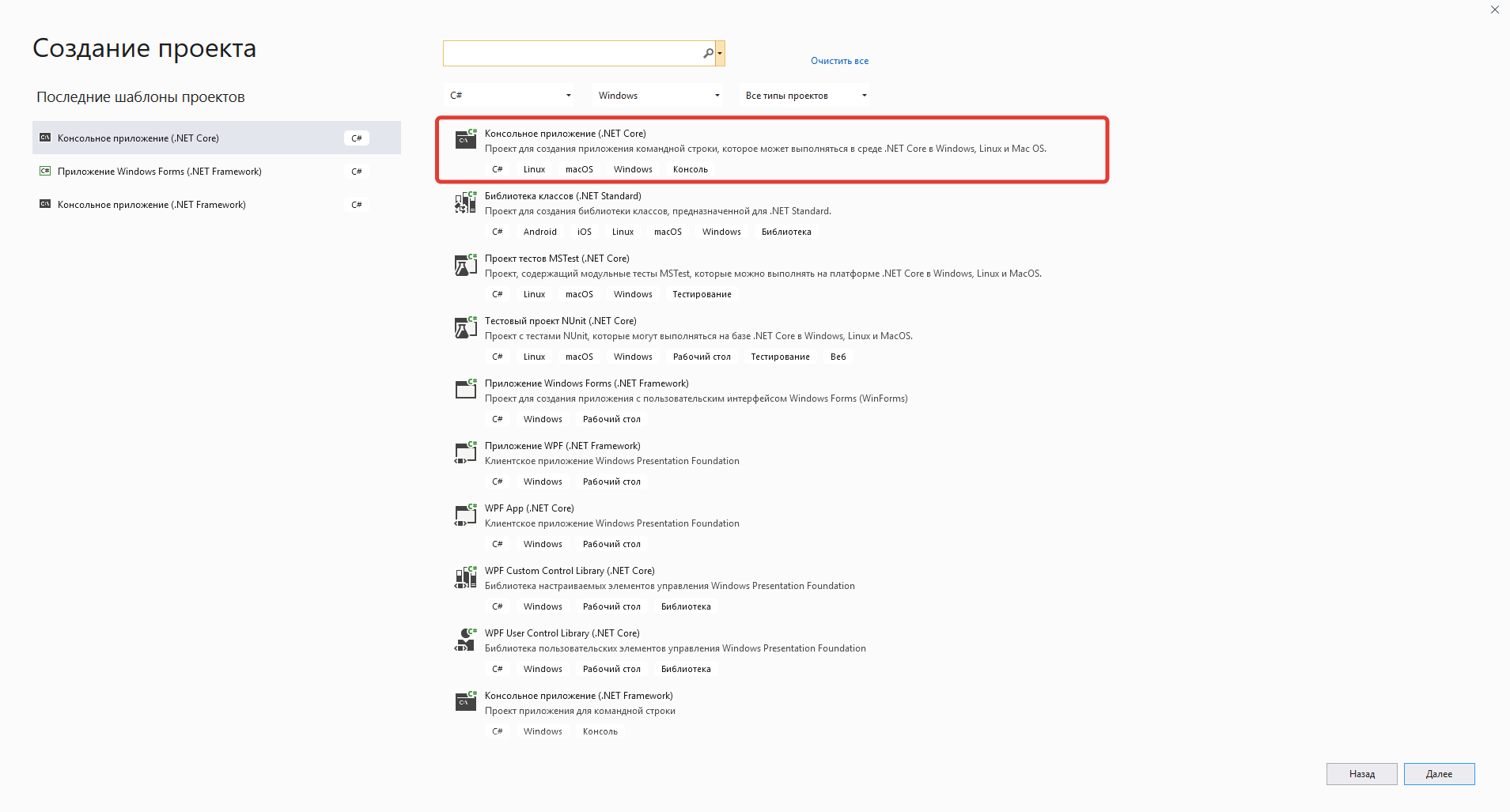
Любая современная среда программирования позволяет писать текст программы, транслировать его на понятный машине, запускать полученную программу на исполнение или в режиме отладки, отлаживать программу путем запуска по шагам или до контрольной точки, просматривая промежуточные результаты вычислений и т.д.

Некоторые среды программирования позволяют писать программы на разных языках и создавать программы разных типов. Например, можно создать программу типа «Консольное приложение» или программу с графическими окнами.

Для написания программы необходимо создать проект на требуемом языке и требуемого типа. После этого в менеджере проекта необходимо добавлять файлы проекта и писать в них текст программы. В каждой среде программирования эти этапы выглядят по-разному.

Для тех, кто имеет опыт разработки межплатформенных программ может получить индивидуальное задание у преподавателя с использованием знакомых сред и языков программирования и разработке программы в своей предметной области для нужд предприятия – места работы*.*

В версии 2019 имеется возможность создать приложение Net Core, которое совместимо с другими ОС и включает в себя только базовые возможности и позволяет создавать межплатформенные (кроссплатформенные приложения).



Несмотря на то, что никаких действий в данном коде пока не описано, его уже можно скомпилировать и запустить. Для этого нужно выбрать пункт меню «Debug» → «Start Without Debugging» или нажать сочетание клавиш Ctrl F5 (рис. 4).

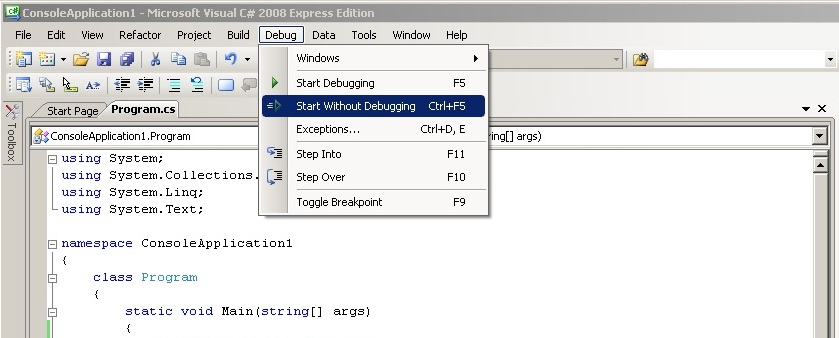


Рис. 4 — Пункт меню для запуска проекта

Среда Visual Studio удобна тем, что любые программы от Microsoft легко устанавливаются и не требуют дальнейших настроек. К тому же имеется версия абсолютно бесплатная как для домашнего использования, студентов, так и для небольших организаций.

При использовании других сред, особенно бесплатных, надо быть готовым к тому, что из настройка может занять длительное время и потребует скачивания различных модулей с различных сайтов и затем их взаимоувязки для совместной работы.

Рекомендуется скачать последнюю версию Visual Studio Community с сайта Microsoft. Ее можно использовать абсолютно бесплатно.

**Задание 1. Использование технологии LINQ для работы с коллекцией чисел**

**Цель работы**: Исследовать возможности технологии LINQ при работе с коллекцией чисел.

**Примеры использования технологии LINQ**

1) Определить отношение суммы элементов массива, расположенных до первого из минимальных элементов в массиве, к сумме элементов, расположенных после минимального.

var min = V.Min(a=>a);

var s = V.TakeWhile(a => a != min).Sum() \* 1.0 /

V.SkipWhile(a => a != min).Skip(1).Sum();

или

var s2 = V.TakeWhile(a => a != V.Min()).Sum() \* 1.0 /

V.SkipWhile(a => a != V.Min()).Skip(1).Sum();

2) Сумма элементов после последнего максимального

var s4 = V.Reverse().TakeWhile(a => a != V.Max()).Sum();

3) Произведение элементов после последнего максимального

var s5 = V.Reverse().TakeWhile(a => a != V.Max()).Aggregate((x, y) => x \* y);

4) Найти минимальный по модулю элемент массива

var min = V.Min(a => Math.Abs(a));

5) Заменить первое число больше 2 на 5 и 6.

int[] V = new int[] { -20, 41, 0};

var ss = V.TakeWhile(a => a <= 2).Concat(new int[2] { 5, 6 }).Concat(V.SkipWhile(a => a <= 2).Skip(1)).ToArray().ToArray();

Результат:

-20 5 6 0

В случае, если в исходном массиве не будет числа больше 2, то числа 5 и 6 будут добавлены в конец массива. Если необходимо избежать добавления чисел в конец массива, то предварительно необходимо проверить наличие чисел больше 2 в исходном массиве следующим образом:

int[] V = new int[] { -20, 41, 0};

if (V.Any(a=>a>2))

var ss = V.TakeWhile(a => a <= 2).Concat(new int[2] { 5, 6 }). Concat(V.SkipWhile(a => a <= 2).Skip(1)).ToArray();

6) Добавить после первого числа, которое > 1, числа 2 и 2

int [] V = new int[14] {1,1,0,3,0,-10,1,6,1,1,0,0,3,6};

var ss = V.TakeWhile(a => a <= 1).Concat(V.SkipWhile(a => a <= 1).Take(1)).Concat(new int[2] { 2, 2 }).Concat(V.SkipWhile(a => a <= 1).Skip(1));

Результат:

1, 1, 0, 3, 2, 2, 0, -10, 1, 6, 1, 1, 0, 0, 3, 6

7) Получить строку из массива строк

string[] V = new string[] { "aa", "vcv", "dd"};

var ss = V.Aggregate((x,y)=>x+ " "+ y);

или так

string[] V = new string[] { "aa", "vcv", "dd" };

var ss = V.Aggregate("", (x, y) => x + y);

8) Получить строку, в которой элементы массива перечислены через запятую

int[] V = new int[] { 4, 5, 6};

var ss = V.Aggregate("", (x, y) => x + y + ",");

Для удаления последней запятой можно использовать код

ss=ss.TrimEnd(',');

или одним выражением

var ss = V.Aggregate("", (x, y) => x + y + ",").TrimEnd(',');

**Порядок выполнения работы**

Выполнить три задания в соответствии с вариантом. Значения элементов массивов можно задать в тексте программы.

При выполнении Задания 2 для вставки элементов в массив необходимо использовать разделение массива на две части с разрывом в месте вставки и последующим объединением подстрок с помощью команды Concat. Это связано с тем, что LINQ не имеет возможность менять исходный контейнер.

**Индивидуальные задания**

**Задание 1.** Обработка элементов вектора

Дан одномерный числовой массив.

1) Определить отношение суммы элементов массива, расположенных до первого из минимальных элементов в массиве, к произведению элементов, распо­ложенных после минимального. Если по какой-либо причине вычислить отношение не удается, выдать об этом сообщение с указанием причины.

2) Определить отношение произведения элементов массива, расположенных до последнего из максимальных элементов в массиве, если он не единственный, к сумме элементов, расположенных после максимального. Если по какой-либо причине вычислить отношение не удается, выдать об этом сообщение с указанием причины.

3) Найти среднее арифметическое элементов массива, расположенных меж­ду максимальным элементом и минимальным элементом (первыми по порядку, если их несколько). Если по какой-либо причине вычислить среднее арифметиче­ское не удается, выдать об этом сообщение с указанием причины.

4) Определить количество элементов массива, расположенных между мак­симальным элементом и минимальным элементом (первыми по порядку, если их несколько). Если по какой-либо причине количество элементов определить не удается, выдать об этом сообщение с указанием причины.

5) Найти произведение всех элементов массива, предшествующих первому нулевому элементу. Если по какой-либо причине вычислить произведение не уда­ется, выдать об этом сообщение с указанием причины.

6) Найти произведение положительных элементов массива, предшествую­щих первому отрицательному элементу. Если по какой-либо причине вычислить произведение не удается, выдать об этом сообщение с указанием причины.

7) Найти среднее арифметическое отрицательных элементов массива, предшествующих первому положительному элементу. Если по какой-либо при­чине вычислить среднее арифметическое не удается, выдать об этом сообщение с указанием причины.

8) Найти произведение положительных элементов массива, следующих по­сле первого нулевого элемента. Если по какой-либо причине вычислить произве­дение не удается, выдать об этом сообщение с указанием причины.

9) Найти среднее арифметическое элементов, расположенных между пер­вым и вторым нулевыми элементами. Если по какой-либо причине вычислить среднее арифметическое элементов не удается, выдать об этом сообщение с ука­занием причины.

10) Найти наименьший по абсолютной величине элемент среди элементов, расположенных между первым и вторым нулевыми элементами. Если по какой- либо причине найти такой элемент не удается, выдать об этом сообщение с указа­нием причины.

11) Определить количество элементов массива, расположенных между мак­симальным и «центральным» элементами массива (предполагается, что число элементов - нечётное и максимальный - единственный). Если по какой-либо причине количество элементов определить не удается, выдать об этом сообщение с указанием причины.

12) Определить произведение элементов массива, расположенных между максимальным и «центральным» элементами массива (предполагается, что число элементов - нечетное и максимальный единственный). Если по какой-либо при­чине вычислить произведение не удается, выдать об этом сообщение с указанием причины.

13) Найти отношение минимального элемента массива к максимальному среди элементов, предшествующих первому нулевому элементу. Если по какой- либо причине вычислить отношение не удается, выдать об этом сообщение с ука­занием причины.

14) Найти отношение суммы отрицательных элементов массива к мини­мальному элементу массива. Если по какой-либо причине вычислить отношение не удается, выдать об этом сообщение с указанием причины.

15) Найти отношение максимального элемента массива к произведению по­ложительных элементов. Если по какой-либо причине вычислить отношение не удается, выдать об этом сообщение с указанием причины.

16) Определить отношение произведения элементов массива, расположенных до первого из минимальных элементов в массиве, к сумме элементов, располо­женных после минимального. Если по какой-либо причине вычислить отношение не удается, выдать об этом сообщение с указанием причины.

17) Определить отношение суммы элементов массива, расположенных до по­следнего из максимальных элементов в массиве, если он не единственный, к про­изведению элементов, расположенных после максимального. Если по какой-либо причине вычислить отношение не удается, выдать об этом сообщение с указанием причины.

18) Найти сумму элементов массива, расположенных между максимальным элементом и минимальным элементом (первыми по порядку, если их несколько). Если по какой-либо причине вычислить сумму не удается, выдать об этом сооб­щение с указанием причины.

19) Найти произведение элементов массива, расположенных между макси­мальным элементом и минимальным элементом (первыми по порядку, если их не­сколько). Если по какой-либо причине вычислить произведение не удается, вы­дать об этом сообщение с указанием причины.

20) Найти произведение отрицательных элементов массива, предшествую­щих первому положительному элементу. Если по какой-либо причине вычислить произведение не удается, выдать об этом сообщение с указанием причины.

21) Найти среднее арифметическое элементов массива, предшествующих первому нулевому элементу. Если по какой-либо причине вычислить среднее арифметическое не удается, выдать об этом сообщение с указанием причины.

22) Найти среднее арифметическое положительных элементов массива, предшествующих первому отрицательному элементу. Если по какой-либо причи­не вычислить среднее арифметическое не удается, выдать об этом сообщение с указанием причины.

23) Найти произведение отрицательных элементов массива, следующих после первого положительного элемента. Если по какой-либо причине вычислить произведение не удается, выдать об этом сообщение с указанием причины.

24) Найти произведение элементов, расположенных между первым и вторым нулевым элементами. Если по какой-либо причине вычислить произведение не удается, выдать об этом сообщение с указанием причины.

25) Найти наибольший по абсолютной величине элемент среди элементов, расположенных между первым и вторым нулевыми элементами массива. Если по какой-либо причине найти такой элемент не удается, выдать об этом сообщение с указанием причины.

26) Определить среднее арифметическое элементов массива, расположен­ных между минимальным и «центральным» элементами массива (предполагается, что число элементов - нечетное и минимальный - единственный). Если по какой- либо причине вычислить среднее арифметическое не удается, выдать об этом со­общение с указанием причины.

28) Найти отношение максимального элемента массива к минимальному среди элементов, предшествующих первому нулевому элементу. Если по какой- либо причине вычислить отношение не удается, выдать об этом сообщение с ука­занием причины.

29) Найти отношение произведения положительных элементов массива к максимальному элементу массива. Если по какой-либо причине вычислить отно­шение не удается, выдать об этом сообщение с указанием причины.

30) Найти отношение минимального элемента последовательности массива к сумме отрицательных элементов. Если по какой-либо причине вычислить отно­шение не удается, выдать об этом сообщение с указанием причины.

**Задание 2. Создание массива с добавлением элементов**

1) Вставить после первого положительного элемента в массиве два элемента, равных заданному значению. Если вставка элементов невозможна, выдать об этом сообщение.

2) Вставить после последнего отрицательного элемента в массиве два эле­мента, равных заданному значению. Если вставка элементов невозможна, выдать об этом сообщение.

3) Вставить после первого нулевого элемента в массиве два элемента, рав­ных заданному значению. Если вставка элементов невозможна, выдать об этом сообщение.

4) Вставить заданное значение перед элементом массива, располо­женным после первого нулевого элемента. Если вставка элементов невозможна, выдать об этом сообщение.

5) Вставить заданное значение после элемента массива, располо­женного до первого нулевого элемента. Если вставка элементов невозможна, вы­дать об этом сообщение.

6) Вставить в любой позиции массива между пер­вым и вторым нулевыми элементами, заданное значение. Если вставка элементов невозможна, выдать об этом сообщение.

7) Заменить последний из нулевых элементов в массиве на три подряд иду­щих нулевых элемента. Если такая замена невозможна, выдать об этом сообще­ние.

8) Заменить первый положительный элемент в массиве на три элемента, равных заданному значению. Если такая замена невозможна, выдать об этом со­общение.

9) Заменить последний из отрицательных элементов в массиве на три эле­мента, равных заданному значению. Если такая замена невозможна, выдать об этом сообщение.

10) После каждого элемента, имеющего значение х, вставить элемент со значением у. Если вставка элементов невозможна, выдать об этом сообщение.

11) Вставить в массив элемент с заданным значением перед элементом с за­данным номером к. Если вставка элементов невозможна, выдать об этом сообще­ние.

12) Вставить в массив элемент с заданным значением перед первым элементом с максимальным значением и перед первым элементом с минимальным значением. Если вставка элементов невозможна, выдать об этом сообщение.

13) Вставить в массив элемент с заданным значением после первого элемента с максимальным значением и перед первым элементом элемента с минимальным значением. Если вставка элементов невозможна, выдать об этом сообщение.

14) Вставить после последнего положительного элемента в массиве два элемента, равных заданному значению. Если вставка элементов невозможна, выдать об этом сообщение.

15) Вставить после первого отрицательного элемента в массиве два элемента, равных заданному значению. Если вставка элементов невозможна, выдать об этом сообщение.

16) Вставить после последнего из нулевых элементов в массиве два элемента, равных заданному значению. Если вставка элементов невозможна, выдать об этом сообщение.

17) Заменить последний из положительных элементов в массиве на три элемента, равных заданному значению. Если такая замена невозможна, выдать об этом сообщение.

18) Заменить первый нулевой элемент в массиве на три подряд идущих нулевых элемента. Если такая замена невозможна, выдать об этом сообщение.

19) Заменить первый отрицательный элемент в массиве на три элемента, равных этому отрицательному элементу. Если такая замена невозможна, выдать об этом сообщение.

20) Вставить в массив элемент с заданным значением после элемента с заданным номером к. Если вставка элементов невозможна, выдать об этом сообщение.

21) Вставить в массив элемент с заданным значением после первого эле­мента с максимальным значением и после первого элемента с минимальным зна­чением. Если вставка элементов невозможна, выдать об этом сообщение.

22) Вставить в массив элемент с заданным значением перед первым элемен­том с максимальным значением и после первого элемента с минимальным значе­нием. Если вставка элементов невозможна, выдать об этом сообщение.

23) За первым элементом с заданным значением вставить его дубликат. Ес­ли вставка элементов невозможна, выдать об этом сообщение.

24) Перед элементом с отрицательным значением вставить элемент, равный абсолютной величине этого отрицательного элемента. Если вставка элементов невозможна, выдать об этом сообщение.

25) Перед первым максимальным элементом вставить элемент, равный абсолютной величине минимального элемента. Если вставка элементов невозможна, выдать об этом сообщение.

**Задание 3. Проверка состояния вектора**

1) Проверить, есть ли в массиве нулевые элементы.

2) Проверить, есть ли в массиве положительные элементы.

3) Проверить, есть ли в массиве отрицательные элементы.

4) Проверить, есть ли в массиве элементы, равные заданному значению.

5) Проверить, есть ли в массиве элементы больше 12.

6) Проверить, содержится ли в массиве чётное количество нулевых элемен­тов.

7) Проверить, упорядочены ли элементы по возрастанию.

8) Проверить, упорядочены ли элементы по неубыванию.

9) Проверить, упорядочены ли элементы по невозрастанию их абсолютных величин.

10) Проверить, есть ли в массиве несколько элементов с максимальным зна­чением.

11) Проверить, содержатся ли в массиве положительные элементы.

12) Проверить, есть ли в массиве одинаковые элементы.

13) Проверить, есть ли в массиве элементы, равные за­данному значению.

14) Проверить, все ли элементы массива положительны.

15) Проверить, все ли элементы массива отрицательны.

16) Проверить, все ли элементы массива нулевые.

17) Проверить, все ли элементы массива имеют одинаковые значения.

18) Проверить, все ли отрицательные элементы вектора расположены перед элементом равным 0.

19)  Проверить, содержится ли в массиве нечётное количество нулевых эле­ментов.

20) Проверить, упорядочены ли элементы по убыванию.

21) Проверить, упорядочены ли элементы по невозрастанию.

22) Проверить, упорядочены ли элементы по неубыванию их абсолютных ве­личин.

23) Проверить, есть ли в массиве несколько элементов с минимальным зна­чением.

24) Проверить, все ли элементы вектора равны 0 или 1.

25) Проверить, все ли элементы в массиве равны нулю.

**Контрольные вопросы**

1. Поясните назначение методов Aggregate, All, Any, Where, Select.
2. В чем разница между записями   
   var n = m.First() и   
   var n2 = m.First(a=>a>0)?
3. Что такое лямбда-выражение?
4. Что означает запись v.Where(a=>a<0).Select(a=>a\*a)?
5. Как избежать отложенное выполнение?
6. Запишите

var res = from n in m

where n > 0 && n%2==0

select n\*2;

с использованием синтексиса методов.

Какой результат в res будет получен для m={3, -4, 5, 6, 7, -8, -10, 10}?

**Задание 2. Библиотека для работы со строками и текстовыми файлами**

Написать консольное приложение, которое:

– читает многострочный текст из текстового файла в одномерный массив или коллекцию строк;

– затем выполняет задание в соответствии с вариантом.

1. Подсчитать количество слов в каждой строке текста.

2. Подсчитать количество символов в тексте.

3. Подсчитать количество точек в тексте.

4. Подсчитать количество пробелов в тексте.

5. Удалить из текста все пробелы.

6. Удалить из текста все точки.

7. Вставить вместо каждого пробела восклицательный знак.

8. Вставить перед каждым восклицательным знаком вопросительный.

9. Определить, содержит ли текст хотя бы один восклицательный знак, и в какой строке.

10. Подсчитать количество слов в чётных строках текста.

11. Найти номер самой длинной строки текста.

12. Променять местами первую и последнюю строки текста.

13. Определить, есть ли в тексте пустые строки.

14. Определить, содержит ли текст хотя бы пару соседних одинаковых строк.

15. Найти самую короткую строку текста и заменить её фразой «С новым годом!».

16. Найти самую длинную строку текста и заменить её пустой строкой.

17. Определить количество слов в нечётных строках текста.

18. Определить количество пробелов в чётных строках текста.

19. Определить количество предложений в тексте, учитывая, что предложение заканчивается точкой, вопросительным или восклицательным знаком.

20. Поменять местами самую длинную и самую короткую строки текста.

21. Вывести на печать первое предложение текста, учитывая, что оно заканчивается точкой.

22. Определить количество пробелов в нечётных строках текста.

23. Удалить из теста все восклицательные и вопросительные знаки.

24. Определить, содержит ли текст хотя бы один вопросительный знак, и в какой строке.

25. Добавить в начало каждой строки текста её номер и пробел.

**Теоретические сведения**

Теоретические сведения по необходимым доступны в документации Майкрософт по адресам сети Интернет:

String - <https://docs.microsoft.com/ru-ru/dotnet/api/system.string?view=net-6.0>

StreamReader - <https://docs.microsoft.com/ru-ru/dotnet/api/system.io.streamreader?view=net-6.0>

StreamWriter - <https://docs.microsoft.com/ru-ru/dotnet/api/system.io.streamwriter?view=net-6.0>

List<T> - <https://docs.microsoft.com/ru-ru/dotnet/api/system.collections.generic.list-1?view=net-6.0>

**Пример программы**

Исходный файл находится на диске e:\ под именем a.txt. В нем находятся 4 строки в формате UTF8: Это наиболее распространенный формат для различных систем. Файл в этот формат можно преобразовать с помощью различных конвертаторов или набрать в программе Notepad++.

*Мороз и Солнце,*

*День чудесный.*

*Еще ты дремлешь,*

*Друг прелесный.*

Для чтения строк из файла используется класс для работы с потоковым файлом StreamReader. Каждая строка читается в динамический обобщенный список List <string>. Это связано с тем, что изначально количество строк в файле программе неизвестно, а динамический список может увеличивать свой размер по мере необходимости.

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

namespace ConsoleApp3

{

class Program

{

static void Main(string[] args)

{

//Список строк

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

//Открытие файла

System.IO.StreamReader sr = new StreamReader("e:\\a.txt");

//чтение строк файла с занесением в список, пока не завершится поток

while (!sr.EndOfStream)

{

string s = sr.ReadLine();

list.Add(s);

}

//Закрытие файла

sr.Close();

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

{

if (i % 2 == 1)

list[i] = list[i].Replace('.', '!');

//разделение строки на слова, разделенные пробелом и табуляцией

string[] words = list[i].Split(new char[] { ' ', '\t' });

Console.WriteLine("В {0} предложении количество слов ={1}", i, words.Count());

}

Console.WriteLine("Количество строк = {0}", list.Count);

Console.WriteLine("Измененный текст:");

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

{

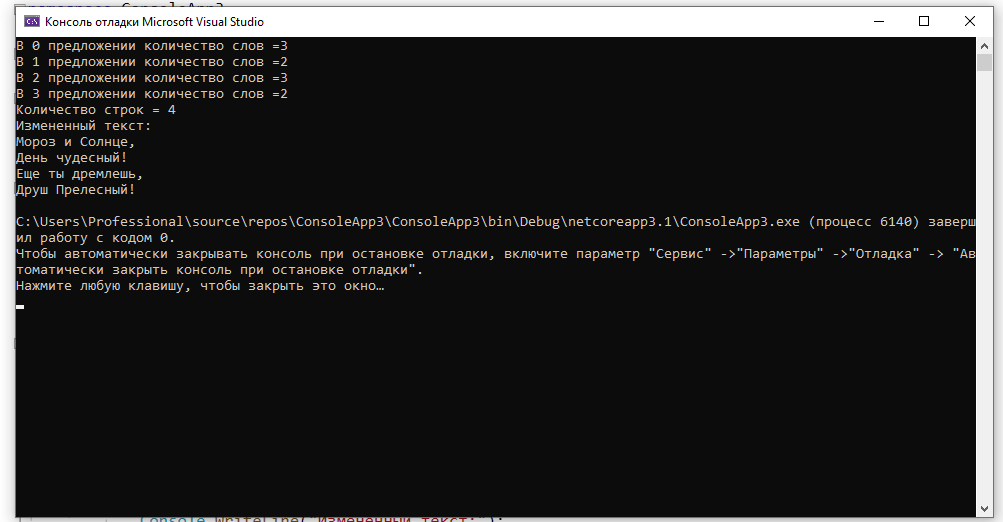
Console.WriteLine(list[i].ToString() );

}

}

}

}



# Контрольные вопросы

1. Что представляет собой строка?
2. Какие методы имеет класс String?
3. Что такое файловый поток?
4. Какие методы имеет класс StreamReader?
5. Какие методы имеет класс StreamWriter?
6. Что представляет собой класс List<T>?

**Задание 3. Регулярные выражения**

**Цель работы:** Исследовать пространство имен для работы с регулярными выражениями для анализа текстовых файлов.

**Задание для выполнения работы**

1. Создайте приложение **.NET Core**.
2. Откройте программно предварительно подготовленный текстовый файл и html-файл.
3. Программно из файла читайте строки.
4. В каждой строке с помощью регулярных выражений необходимо найти требуемую информацию, например:

— дату и время;

— номер телефона;

— имена файлов;

— имена переименовываемых файлов;

— имена удаляемых файлов;

— и т.д.

Найденную информацию вывести на экран.

**Краткие теоретические сведения**

Регулярное выражение (regular expression) — эффективный и гибкий метод по обработке больших текстов, позволяя в то же время существенно уменьшить объемы кода по сравнению с использованием стандартных операций со строками.

Поддержка регулярных выражений в .Net выполняется классами пространства имен: **System.Text.RegularExpressions.**

Основные классы для работы с регулярными выражениями:

* Regex - постоянное регулярное выражение.
* Match - предоставляет результаты очередного применения всего регулярного выражения к исходной строке.
* MatchCollection - предоставляет набор успешных сопоставлений, при итеративном применении образца регулярного выражения к строке.

Теперь посмотрим на некоторые примеры использования. Возьмем первый пример с скороговоркой "Бык тупогуб, тупогубенький бычок, у быка губа бела была тупа" и найдем в ней все слова, где встречается корень "губ":

string s = "Бык тупогуб, тупогубенький бычок, у быка губа бела была тупа";

Regex regex = new Regex(@"\w\*губ\w\*");

Так как выражение \w\* соответствует любой последовательности алфавитно-цифровых символов любой длины, то данное регулярное выражение задает все слова, содержащие "губ".

А последующий код найдет:

– первое вхождение выражения regex в строку s:

string s = "Бык тупогуб, тупогубенький бычок, у быка губа бела была тупа губа";

Regex regex = new Regex(@"\w\*губ\w\*");

Match match = regex.Match(s);

if (match.Success)

{

Console.WriteLine(match.Value.ToString()); //выведет тупогуб

}

– все вхождения выражения regex в строку s

string s = "Бык тупогуб, тупогубенький бычок, у быка губа бела была тупа губа";

Regex regex = new Regex(@"\w\*губ\w\*");

MatchCollection matchcollection = regex.Matches(s);

Console.WriteLine();

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

{

Console.WriteLine(matchcollection[i].Value);

//выведет тупогуб тупогубенький губа губа

}

Второй пример - нахождение телефонного номера в формате 111-111-1111 (три цифры, дефис, три цифры, дефис, четыре цифры:

Regex regex = new Regex(@"\d{3}-\d{3}-\d{4}");

string s = "456-435-2318";

Если мы точно знаем, сколько определенных символов должно быть, то мы можем явным образом указать их количество в фигурных скобках: \d{3} - то есть в данном случае три цифры.

Мы можем не только задать поиск по определенным типам символов - пробелы, цифры, но и задать конкретные символы, которые должны входить в регулярное выражение. Например, перепишем пример с номером телефона и явно укажем, какие символы там должны быть:

string s = "456-435-2318";

Regex regex = new Regex("[0-9]{3}-[0-9]{3}-[0-9]{4}");

В квадратных скобках задается диапазон символов, которые должны в данном месте встречаться. В итоге данный и предыдущий шаблоны телефонного номера будут эквивалентны.

Также можно задать диапазон для алфавитных символов: Regex regex = new Regex("[a-v]{5}"); - данное выражение будет соответствовать любому сочетанию пяти символов, в котором все символы находятся в диапазоне от a до v.

Можно также указать отдельные значения: Regex regex = new Regex(@"[2]\*-[0-9]{3}-\d{4}");. Это выражение будет соответствовать, например, такому номеру телефона "222-222-2222" (так как первые числа двойки)

С помощью операции | можно задать альтернативные символы, например:

Regex regex = new Regex(@"(2|3){3}-[0-9]{3}-\d{4}");

То есть первые три цифры могут содержать только двойки или тройки. Такой шаблон будет соответствовать, например, строкам "222-222-2222" и "323-435-2318". А вот строка "235-435-2318" уже не подпадает под шаблон, так как одной из трех первых цифр является цифра 5.

Итак, у нас такие символы, как \*, + и ряд других используются в качестве специальных символов. Если надо найти, строки, где содержится точка, звездочка или какой-то другой специальный символ, то в этом случае надо экранировать эти символы слешем:

Regex regex = new Regex(@"(2|3){3}\.[0-9]{3}\.\d{4}");

Этому выражению будет соответствовать, например, строка "222.222.2222".

**Пример 1**. Функция для однократного (первого вхождения) поиска соответствия регулярного выражения, заданного объектом класса Regex в строке s.

**string** *s* = "Дядя: (812)5555555 Тетя: 555-55-55 Петя: (848)222-22-22";

**Regex** *regex* = **new** **Regex**("дя|я");

**Match** *match* = *regex*.Match(s);

**if** (*match*.Success)

{

**Console**.WriteLine(*match*.Value.ToString());

}

Будет выведено: я

(первая буква я в строке

"Дядя: (812)5555555 Тетя: 555-55-55 Петя: (848)222-22-22"

)

**Пример 2**. Для поиска всех вхождений «дя» или «я» необходимо использовать код и объект класса MatchCollection. Метод Matches класса Regex принимает строку, к которой надо применить регулярные выражения, и возвращает коллекцию найденных совпадений. Каждый элемент такой коллекции представляет объект Match. Его свойство Value возвращает найденное совпадение.

Regex regex = new Regex("дя|я");

string s = "Дядя: (812)5555555 Тетя: 555-55-55 Петя: (848)222-22-22";

MatchCollection matchcollection = regex.Matches(s);

Console.WriteLine();

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

{

Console.WriteLine(matchcollection[i].Value);

}

Будет выведено:

я

дя

я

я

То есть:

"Дядя: (812)5555555 Тетя: 555-55-55 Петя: (848)222-22-22"

Пример 3. Для поиска всех трехзначных цифр можно использовать тот же самый код, но изменить регулярное выражение:

Regex regex = new Regex(@"[0-9]{3,3}");

string s = "Дядя: (812)5555555 Тетя: 555-55-55 Петя: (848)222-22-22";

MatchCollection matchcollection = regex.Matches(s);

Console.WriteLine();

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

{

Console.WriteLine(matchcollection[i].Value);

}

Будет выведено:

812

555

555

555

848

222

То есть

"Дядя: (812)5555555 Тетя: 555-55-55 Петя: (848)222-22-22"

Пример 4. Поиск одного вхождения и подвыражений в этом вхождении. Подвыражение в регулярном выражении заключается в круглые скобки и позволяет выделить часть найденной подстроки.

string s = "Дядя: (812)5555555 Тетя: 555-55-55 Петя: (848)222-22-22 Привет";

Regex regex = new Regex(@"([0-9]{3,3}).+(\d)");

Match match = regex.Match(s);

Console.WriteLine();

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

{

Console.WriteLine(match.Groups[i].Value);

}

Будет выведено

812)5555555 Тетя: 555-55-55 Петя: (848)222-22-22

812

2

Символы, используемые в регулярных выражениях приведены в таблице 1. В таблице 1 даются ссылки на код, которые приведены ниже.

**Код 1:**

**string** *s* = "Дядя: (812)5555555 Тетя: 555-55-55 Петя: (848)222-22-22";

**Regex** *regex* = **new** **Regex**("дя|я");

**Match** *match* = *regex*.Match(s);

**if** (*match*.Success)

{

**Console**.WriteLine(*match*.Value.ToString());

}

**Код 2:**

**string** *s* = "Дядя: (812)5555555 Тетя: 555-55-55 Петя: (848)222-22-22";

**Regex** regex = new Regex(@"[0-9]{3,3}");

**string** *s* = "Дядя: (812)5555555 Тетя: 555-55-55 Петя: (848)222-22-22";

**MatchCollection** matchcollection = regex.Matches(s);

**Console**.WriteLine();

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

{

**Console**.WriteLine(matchcollection[i].Value);

}

Таблица 1. Символы, используемые в регулярных выражениях

|  |  |  |
| --- | --- | --- |
| **Символ** | **Значение** | Примеры применения |
| **.** | Любой одиночный символ, кроме символа новой строки \n. | Regex("a.b");  Код 1 Строка s = "a1ba2ba3ba4b";  Результат: a1b  Код 2 Строка s = "a1ba2ba3ba4b";  Результат: a1b a2b a3b a4b |
| **\** | Определение метасимвола | Символ "." соответствует любому символу, а "\." точке .  Regex(@"a\.b"); в предыдущем примере ничего не найдет  Код 1 Строка s = "a1ba2ba3ba.b";  Результат: a.b  Код 2 Строка s = "a1ba.ba3ba.b";  Результат: a.b a.b |
| **|** | Разделение шаблона (или) | Regex("a|b");  Код 1 Строка s = "abcabcabcabc";  Результат: a  Код 2 Строка s = "abcabcabcabc";  Результат: a b a b a b a b |
| **[мн-во сим-в]** | Соответствует любому символу из данного множества (одному и только одному) | Regex("[abc]");  Код 1 Строка s = "abcdabcdabcdabcd";  Результат: a  Код 2 Строка s = "abcdabcdabcdabcd";  Результат: a b c a b c a b c a b c |
| **[^мн-во сим-в]** | Отрицание множества символов | Regex("[^abc]");  Код 1 Строка s = "abcdabcdabcdabcd";  Результат: d  Код 2 Строка s = "abcdabcdabcdabcd";  Результат: d d d d |
| **^** | Соответствует началу строки Отрицание - если первый в квадратной скобке | Regex("^abc");  Код 1 Строка s = "abcdefabcdef";  Результат: abc  Код 2 Строка s = "abcdefabcdef";  Результат: abc  Regex("^bc"); в обоих примерах ничего не найдет |
| **$** | Соответствует концу строки. Это конец строки или позиция перед символом начала новой строки. | Regex("def$");  Код 1 Строка s = "abcdefabcdef";  Результат: def  Код 2 Строка s = "abcdefabcdef";  Результат: def |
| **(....)** | Группировка элементов. | Regex("(ab)");  Код 1 Строка s = "abcdefabcdef";  Результат: ab ab  Код 2 Строка s = "abcdefabcdef";  Результат: ab ab |
| **?** | Повторение 0 или 1 раз, стоящего перед | Regex("ab?");  Код 1 Строка s = "aabbbcde";  Результат: a  Код 2 Строка s = "aabbbcde";  Результат: a ab  Regex("a?b?");  Код 1 Строка s = "aabbbcde";  Результат: a  Код 2 Строка s = "aabbbcde";  Результат: a ab b b  Regex("a?b");  Код 1 Строка s = "aabbbcde";  Результат: ab  Код 2 Строка s = "aabbbcde";  Результат: ab b b |
| **\*** | Повторение 0 или более раз стоящего перед | Regex("ab\*");  Код 1 Строка s = "aabbbcde";  Результат: a  Код 2 Строка s = "aabbbcde";  Результат: a abbb  Regex("a\*b\*");  Код 1 Строка s = "aabbbcde";  Результат: aabbb  Код 2 Строка s = "aabbbcde";  Результат: aabbb  Regex("a\*b");  Код 1 Строка s = "aabbbcde";  Результат: aab  Код 2 Строка s = "aabbbcde";  Результат: aab b b |
| **+** | Повторение 1 или более раз стоящего перед ним | Regex("ab+");  Код 1 Строка s = "aabbbcde";  Результат: abbb  Код 2 Строка s = "aabbbcde";  Результат: abbb  Regex("a+b+");  Код 1 Строка s = "aabbbcde";  Результат: aabbb  Код 2 Строка s = "aabbbcde";  Результат: aabbb  Regex("a+b");  Код 1 Строка s = "aabbbcde";  Результат: aab  Код 2 Строка s = "aabbbcde";  Результат: aab |
| **{n}** | Повторение точно n раз | Regex("ab{2}");  Код 1 Строка s = "aabbbcde";  Результат: abb  Код 2 Строка s = "aabbbcde";  Результат: abb |
| **{n,m}** | повторение от n до m раз | Regex("ab{2,4}");  Код 1 Строка s = "abbbcdeabbbbbcde";  Результат: abbb  Код 2 Строка s = "abbbcdeabbbbbcde";  Результат: abbb abbbb |
| **{n,}** | повторение n и более раз | Regex("ab{2,}");  Код 1 Строка s = "abbbcdeabbbbbcde";  Результат: abb  Код 2 Строка s = "abbbcdeabbbbbcde";  Результат: abbb abbbbb |
| **\*?** | повторение 0 или более раз, минимально возможное количество | Regex("ab\*?c");  Код 1 Строка s = "abbbcdeabbbbbcde";  Результат: abbbс  Код 2 Строка s = "abbbcdeabbbbbcde";  Результат: abbbс abbbbbс |
| **+?** | повторение 1 или более раз, минимально возможное количество | Regex("ab+?c");  Код 1 Строка s = "abbbcdeabbbbbcde";  Результат: abbbс  Код 2 Строка s = "abbbcdeabbbbbcde";  Результат: abbbс abbbbbс |
| **??** | повторение 0 или 1 раз, минимально возможное количество | Regex("ab??c");  Код 1 Строка s = "acdeabcdeabbcde";  Результат: aс  Код 2 Строка s = "acdeabcdeabbcde";  Результат: abс |
| **\w** | Слово (цифра или буква или знак подчеркивания) | Regex(@"\w");  Код 1 Строка s = "abcd\_~`!@#$%^&\*()-=+|:;\",.<>?/12345";  Результат: A  Код 2 Строка s = "abcd\_~`!@#$%^&\*()-=+|:;\",.<>?/12345";  Результат: A Ф b c d \_ 1 2 3 4 5 |
| **\W** | Не слово (не цифра и не буква) | Regex(@"\W");  Код 1 Строка s = "abcd\_~`!@#$%^&\*()-=+|:;\",.<>?/12345";  Результат: ~  Код 2 Строка s = "abcd\_~`!@#$%^&\*()-=+|:;\",.<>?/12345";  Результат: ~ ` ! @ # $ % ^ & \* ( ) - = + | : ; " , . < > ? / |
| **\d** | Десятичная цифра | Regex(@"\d");  Код 1 Строка s = "abcd12345";  Результат: 1  Код 2 Строка s = "abcd12345";  Результат: 1 1 2 3 4 5 |
| **\D** | Не десятичная цифра | Regex(@"\D");  Код 1 Строка s = "abcd12345";  Результат: a  Код 2 Строка s = "abcd12345";  Результат: a b c d \_ ~ ` ! @ # $ % ^ & \* ( ) - = + | : ; " , . < > ? / |
| **\s** | Пустое место (пробел, \f, \n, \r, \t, \v) | Regex(@"\s");  Код 1 Строка s = "ab cd";  Результат: b c  Код 2 Строка s = "ab cd";  Результат: b c |
| **\S** | Не пустое место (не пробел, не \f, не \n, не \r, не \t, не \v) | Regex(@"\S");  Код 1 Строка s = "ab c\td";  Результат: a  Код 2 Строка s = "ab c\td";  Результат: a b c d |
| **\b** | Граница слова (Символы для поиска пишутся: для начала слова - справа; для конца - слева.) | Regex(@"\bqw"); //ничего не будет найдено при Regex(@"\bwe");  Код 1 Строка s = "abcde qwerty qwerty";  Результат: qw  Код 2 Строка s = "abcde";  Результат: qw qw  Regex(@"ty\b"); //ничего не будет найдено при Regex(@"rt\b");  Код 1 Строка s = "abcde qwerty qwerty";  Результат: ty  Код 2 Строка s = "abcde qwerty qwerty";  Результат: ty ty |
| **\B** | Не граница слова (Символы для поиска пишутся: для начала слова - справа; для конца - слева.) | Regex(@"\Bwe"); //ничего не будет найдено при Regex(@"\qw");  Код 1 Строка s = "abcde qwerty qwerty";  Результат: we  Код 2 Строка s = "abcde";  Результат: we we  Regex(@"rt\B"); //ничего не будет найдено при Regex(@"ty\B");  Код 1 Строка s = "abcde qwerty qwerty";  Результат: rt  Код 2 Строка s = "abcde qwerty qwerty";  Результат: rt rt |

**Пример.** Есть строки, состоящие из шести цифр. Нужно выбрать такие, где какая-то цифра идёт подряд четыре или более раз?

Например: 222217, 777701, 588888, 945555, 111101, 000034.

string pattern = @"0{4,}|1{4,}|2{4,}|3{4,}|4{4,}|5{4,}|6{4,}|7{4,}|8{4,}|9{4,}";

var data = new string[]

{

"334442 ",

"123456",

"334444",

"111122",

"333334",

"675443"

};

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

{

if (Regex.IsMatch(data[i], pattern))

{

Console.WriteLine(data[i]);

}

}

Вывод:

334444

111122

333334

**Именованные группы**

Именованные группы также как и неименованные группы задаются круглыми скобками, но содержат знак вопроса и имя группы. Например, неименованная группа с одной цифрой имеет вид (\d), а именованная (?<dig>\d), где dig – имя группы.

Поиск 4-х повторяющихся цифр в числах можно реализовать более короткой записью с помощью именованных групп. В этом случае регулярное выражение записывается более кратко:

string pattern = @"(?'dig'\d)\k'dig'{3}".

Здесь dig – это название группы, состоящей из единственной любой цифры, которая сперва должна сама определиться при поиске, а потом сразу после, когда мы ссылаемся на неё с помощью \k<dig>, должна определиться ещё 3 раза подряд, чтобы шаблон совпал.

В заданном списке найти шестизначные числа с симметрично расположенными цифрами.

string pattern = @"^(?'d1'\d)(?'d2'\d)(?'d3'\d)\k'd3'\k'd2'\k'd1'$";

var data = new string[]

{

"334433 ",

"123321",

"14444441",

"111122",

"333334",

"675443"

};

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

{

if (Regex.IsMatch(data[i], pattern))

{

Console.WriteLine(data[i]);

}

}

Вывод:

334433

123321

В этом примере создается 3 именованные группы с именами d1, d2, d3.

На сайте <https://regex101.com> можно протестировать регулярные выражения.

**Контрольные вопросы**

1. Что такое регулярные выражения?
2. Что означают символы в регулярных выражениях \d, \D, [1-5],   
   [a-n4-6], [^а-я], /b, /B, e.c, d\*, \d+ a\.v ?
3. Для чего предназначены классы **Regex**, **Match**, **MatchCollection**?
4. Какой класс используется для поиска всех вхождений регулярного выражения?