**Отчёт**

1. Текст был разбит по символам, потому что регулярное выражение @"\S" соответствует любому символу, кроме пробельного (\S), и применяется к всей строке (обозначено знаком $) с помощью операции жадности (\*). Таким образом, регулярное выражение ищет и выделяет все символы строки.
2. Результат выполнения программы:

фыидо

fjbld33

2олт

Текст был разбит по словам, потому что регулярное выражение @"\w+" соответствует одному или нескольким последовательным символам слова (\w).

1. Результат выполнения программы

2олт

Осталось только одно слово, потому что регулярное выражение @"\b[0-9]+\w+\b" соответствует слову, начинающемуся с цифры (\b[0-9]+) и заканчивающемуся буквой (\w+). Единственным словом, удовлетворяющим этому условию, является «2олт».

1. Регулярное выражение для слов с цифрой

Regex re = new Regex(@"\w\*\d+\w\*");

1. Результат выполнения программы

fjbld33

Осталось только одно слово, потому что регулярное выражение @"\w+3\b" соответствует слову, содержащему цифру «3». Единственным словом, удовлетворяющим этому условию, является «fjbld33».

1. Результат выполнения программы

33

Совпало только “33”, потому что регулярное выражение @"(\w)\1" соответствует символу (\w), а затем тому же символу снова (обратная ссылка \1). Единственной парой символов, удовлетворяющих этому условию, является “33”.

1. Регулярное выражение для слов со сдвоенными символами

Regex re = new Regex(@"\b(\w)+(\1+)+\w\*\b");

1. Результат выполнения программы

как, Привет дела? меня У все хорошо!

Регулярное выражение @"(\W\*)(\w+)(\W+)(\w+)(\W+)(\w+)(\W\*)" разделяет входную строку на 7 групп:

* Группа 1: символы, предшествующие первому слову
* Группа 2: первое слово
* Группа 3: символы между первым и вторым словом
* Группа 4: второе слово
* Группа 5: символы между вторым и третьим словом
* Группа 6: третье слово
* Группа 7: символы, следующие за третьим словом

Метод Replace заменяет входную строку строкой, в которой группы замещаются значениями, указанными во втором параметре.

1. Регулярное выражение для слов с латинскими буквами и цифрами

Regex re = new Regex(@"\b[a-zA-Z0-9]+\b");

1. Регулярное выражение для слов с хотя бы одной латинской буквой

Regex re = new Regex(@"\b[a-zA-Z]+\w\*\b");

k) Чтение файла CreateDB.sql в строку

l) Регулярное выражение для списков параметров в круглых скобках

Regex regex = new Regex(@"\(\s\*(?!\d+(,\s\*\d+)\*\s\*)\w+(\s\*,\s\*\w+)\*\s\*\)");

m) Проверка регулярного выражения

3 совпадения

n) Ответы на вопросы

a) Символ @ в начале строкового литерала в C# означает, что строка является строкой необработанного текста, и символы в ней не обрабатываются как специальные символы. Без использования символа @ литерал @"(\W\*)(\w+)(\W+)(\w+)(\W+)(\w+)(\W\*)" будет выглядеть как: "(\W\*)(\w+)(\W+)(\w+)(\W+)(\w+)(\W\*)"

b) Шаблоны:

\w\*: Соответствует нулю или нескольким символам слова (\w)

\W\*: Соответствует нулю или нескольким символам, отличным от слова (\W)

\w+: Соответствует одному или нескольким символам слова (\w)

\W+: Соответствует одному или нескольким символам, отличным от слова (\W)

c) Шаблон [^<>]\* соответствует любому количеству символов, кроме < и >.

d) Именованная группа создается с помощью синтаксиса (?<имя\_группы>...).

e) Использование:

* Исчислители: не использованы
* Классы символов: \w, \W, [a-zA-Z0-9], [<>]
* Обычные символы: (, ), #
* Обратные ссылки: \1 в выражении @"(\w)\1"
* Подстановки: использованы в выражении re.Replace(input, "$1$4$3$2$5$6$7")

Алгоритм преобразования файла CreateDB.sql в скрипт MySQL:

Шаг 1. Удалить строки GO и SET

Шаг 2. Преобразовать конструкции [].[<имя>] и [<имя>] в <имя>

Шаг 3. Преобразовать IDENTITY(1,1) в AUTO\_INCREMENT

Шаг 4. Преобразовать timestamp в timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

Шаг 5. Преобразовать конструкции PRIMARY KEY CLUSTERED (…) WITH (…) ON [PRIMARY] в PRIMARY KEY (…)

Шаг 6. Преобразовать конструкции PRIMARY KEY NONCLUSTERED (…) WITH (…) ON [PRIMARY] в PRIMARY KEY (…)

Шаг 7. Преобразовать конструкции UNIQUE NONCLUSTERED (…) WITH (…) ON [PRIMARY] в UNIQUE KEY (…)

Шаг 8. Преобразовать оставшиеся конструкции ON [PRIMARY] в ;

Шаг 9. Преобразовать типы данных (uniqueidentifier => char(32), xml, ntext, nvarchar(max) => text, image, varbinary(max) => BLOB)

Шаг 10. Удалить пустые строки

Шаг 11. Сохранить результат в файл