# Теория автоматов

## Лекция 6: Регулярные выражения

Дьулустан Никифоров

Кафедра ИТ
Северо-Восточный Федеральный Университет

Осень 2024

# Регулярные выражения: definition

- Есть алфавит $\Sigma$. Тогда мы говорим, что $R$ — это **регулярное выражение (regular expression)**, если $R$ является одним из следующих:
  - $a$ для некоторого $a \in \Sigma$,
  - $\varepsilon$,
  - $\emptyset$,
  - $(R_1 + R_2)$, где $R_1$ и $R_2$ регулярные выражения,
  - $(R_1 R_2)$, где $R_1$ и $R_2$ регулярные выражения, или
  - $(R_1^*)$, где $R_1$ регулярное выражение.

# Регулярные выражения: definition

- Есть алфавит $\Sigma$. Тогда мы говорим, что $R$ — это **регулярное выражение (regular expression)**, если $R$ является одним из следующих:
  - $a$ для некоторого $a \in \Sigma$,
  - $\varepsilon$,
  - $\emptyset$,
  - $(R_1 + R_2)$, где $R_1$ и $R_2$ регулярные выражения,
  - $(R_1 R_2)$, где $R_1$ и $R_2$ регулярные выражения, или
  - $(R_1^*)$, где $R_1$ регулярное выражение.
- $\varepsilon$ — соответствует языку, состоящему из одной пустой строки.
- $\emptyset$ — пустой язык.
- Приоритет операций в регексах:
  1. $^*$
  2. конкатенация
  3. $+$

# Регулярные выражения: примеры

Посмотрим на простые рег. выражения.

- $0^*10^*$
- $(0+1)^*001(0+1)^*$
- $((0+1)(0+1)(0+1))^*$
- $0(0+1)^*0+1(0+1)^*1+0+1$
- $(0+\varepsilon)1^*$
- $1^*\emptyset$
- $\emptyset^*$

# Регулярные выражения: примеры

Посмотрим на простые рег. выражения.

- $0^*10^* = \{w \mid w$ содержит ровно одну 1-ку$\}$
- $(0+1)^*001(0+1)^* = \{w \mid w$ содержит подстроку $001\}$
- $((0+1)(0+1)(0+1))^* = \{w \mid$ длина строки $w$ кратно трем$\}$
- $0(0+1)^*0 + 1(0+1)^*1 + 0 + 1 =$
  $\{w \mid w$ начинается и заканчивается одинаковыми символами$\}$
- $(0+\varepsilon)1^* = 01^* + 1^*$
- $1^*\emptyset = \emptyset$
- $\emptyset^* = \{\varepsilon\}$

# Регулярные выражения: примеры

Посмотрите на это:

- $R + \emptyset = ?$
- $R\varepsilon = ?$
- $R + \varepsilon = ?$
- $R\emptyset = ?$

# Регулярные выражения: примеры

Посмотрите на это:

- $R + \emptyset = R$
- $R\varepsilon = R$
- $R + \varepsilon$ может быть $\neq R$
- $R\emptyset = \emptyset$

# Регулярные выражения: примеры

Сделайте регекс для:

- Язык бинарных строк, содержащих четное количество 0-ков.
- Язык бинарных строк четной длины.
- Язык бинарных строк, начинающихся на 110.
- Язык бинарных строк, содержащих ровно три 1-ки.
- Язык бинарных строк, делящихся на 2.

# Регулярные выражения: примеры

- $1^*(01^*01^*)^*$
- $((0+1)(0+1))^*$
- $110(0+1)^*$
- $0^*10^*10^*10^*$
- $(0+1)^*0$

# Регулярные выражения: более сложные примеры

Сделайте регекс для:

- Язык бинарных строк, *не* содержащих две единицы подряд.
- Язык бинарных строк таких, что любая подстрока 00 появляется перед любой подстрокой 11.
- Язык бинарных строк, *не* содержащих подстроку 101.

# Регулярные выражения: сложные примеры

- $0^*(100^*)^*(1+\varepsilon)$
  *Объяснение:* После каждой единицы обязательно идёт либо 0, либо конец строки.

- $0^*(100^*)^*(1+\varepsilon)1^*(011^*)^*(0+\varepsilon)$
  *Объяснение:* Такую строку можно разбить на две части — в первой части нету 11, а во второй нету 00.

- $0^*(1+000^*)^*0^*$
  *Объяснение:* Чтобы избежать 101, после каждой единицы обязательно идёт либо вторая единица, либо хотя бы два нуля.

# Регулярные выражения: математические vs компьютерные

- Регексы, которые мы делали сейчас — это настоящие, теоретические регексы.
- Будем называть их **математическими** регексами, чтобы отличать от другого способа записи регексов, которые используются в компьютерах.
- Как я раньше говорил, регексы *широко* используются в мире компьютеров для эффективной и удобной обработки текстовых данных.
- Но такие регексы ипользуют немного другой способ записи, чем математические регексы. Будем называть их **компьютерными** регексами.

# Компьютерные регексы: ресурсы

По компьютерным регексам можно найти кучу обучающих и тестирующих ресурсов в интернете.

- https://regex101.com/ — на занятиях, можно пользоваться этим сайтом, чтобы тестировать свои регексы и подсматривать синтаксис;
- https://regexone.com/ — серия простых упражнений, чтобы легко освоить регексы.

Так как, регексы реализованы в стандартных библиотеках всех релевантных языков программирования $\Rightarrow$ выучив регексы, вы одновременно напрямую поднимаете свое владение программированием!

# Компьютерные регексы: синтаксис

**Якори (anchors)**:

`^` якорь начала строки

`$` якорь конца строки

- `^The` принимает строки, начинающиеся на `The`
- `end$` принимает строки, заканчивающиеся на `end`
- `^The end$` принимает строки, ровно совпадающие с `The end`
- `boom` принимает строки, содержащие `boom`

# Компьютерные регексы: синтаксис

**Квантифаеры (quantifiers)**: `*` `+` `?` `{}`

- `abc*` ab, за которой следует $\geq 0$ кол-во c
- `abc+` ab, за которой следует $\geq 1$ кол-во c
- `abc?` ab, за которой следует ноль или один c
- `abc{3}` ab, за которой следует ровно три c
- `abc{3,}` ab, за которой следует $\geq 3$ кол-во c
- `abc{3,7}` ab, за которой следует $7 \geq k \geq 3$ кол-во c
- `a(bc)+` a, за которой следует $\geq 1$ кол-во подстроки `bc`

# Компьютерные регексы: синтаксис

**ИЛИ (OR)**: `|` `[]`
**отрицание**: `^`

- `a(b|c)` a, за которой следует `b` *или* `c`
- `a[bc]` то же самое
- `ab[pqrs]` a, за которой следует `p` или `q` или `r` или `s`
- `a[^xyz]` a, за которой следует символ, который не `x` или `y` или `z`
- `[a-z]` строчная латинская буква
- `[A-Xa-x]` строчная или прописная буква от `a` до `x`.

**(!)** Оператор `|` действует на всю группу, в которой находится.

# Компьютерные регексы: синтаксис

**группы**: `()`

- `()` — делает захват (*capture*) того, что находится внутри скобок.
- `(?:)` — не делает захвата (*capture*) того, что находится внутри скобок.

# Компьютерные регексы: синтаксис

**классы символов**:

- `\d` символ-цифра
- `\w` word символ, т.е. цифра, латинская буква или _
- `\s` whitespace символ, т.е. пробел, табуляция или перевод строки
- `.` *любой* символ
- `\D` `\W` `\S` отрицания этих символов

# Компьютерные регексы: синтаксис

**symbol escape**: \
Так как некоторые символы являются частью синтаксиса регексов $\Rightarrow$ если хотим использовать такие символы, то мы должны делать *escape* с помощью \ (как в языках программирования):

- `\.` символ точки
- `\$` символ $

# Компьютерные регексы: синтаксис

- В язык компьтерных регексов есть и много других более изощренных и странных конструкций — их *нельзя* использовать!
- Некоторые особо дикие конструкции позволяют таким регексам выходить за пределы возможностей математических регексов — oh nooo
- Вам надо уметь писать компьютерные регексы и в их математическом виде!

# Компьютерные регексы: примеры

**Задача:** Регекс для слова длины не меньше 3 и не более 12, состоящей только из строчных/прописных латинских букв, цифр или _.

**Ответ:**

# Компьютерные регексы: примеры

**Задача:** Регекс для слова длины не меньше 3 и не более 12, состоящей только из строчных/прописных латинских букв, цифр или _.

**Ответ:**
Возможные решения:

```
^[a-zA-Z0-9_]{3,12}$
```

```
^\w{3,12}$
```

```
^\w\w\w\w?\w?\w?\w?\w?\w?\w?\w?\w?$
```

# Компьютерные регексы: примеры

**Задача:** Регекс для последовательности слов (каждая длины от 3 до 8), разделённых запятыми и пробелом. В конце последовательности должна стоять точка. Слов должно быть не менее трёх.

**Ответ:**

# Компьютерные регексы: примеры

**Задача:** Регекс для последовательности слов (каждая длины от 3 до 8), разделённых запятыми и пробелом. В конце последовательности должна стоять точка. Слов должно быть не менее трёх.

**Ответ:**
Возможные решения:

```
^(\w{3-8}, ){2,}\w{3-8}\.$
```

```
^(\w{3-8}, )(\w{3-8}, )(\w{3-8}, )*\w{3-8}\.$
```

# Компьютерные регексы: примеры

**Задача:** Регекс для даты в формате $dd - mm - yyyy$:

**Ответ:**

# Компьютерные регексы: примеры

**Задача:** Регекс для даты в формате $dd - mm - yyyy$:

**Ответ:**
Возможное (слабенькое) решение:

```
^(0[1-9]|[12][0-9]|3[01])(0[1-9]|1[0-2])[12][0-9]{3}$
```

- Этот регекс только проверяет, что $1 \leq dd \leq 31$, $1 \leq mm \leq 12$, $1000 \leq yyyy \leq 2999$.
- Было бы лучше, если бы регекс еще проверял $dd$ в зависимости от месяца $mm$ (разные месяцы имеют разные диапазоны).

# Компьютерные регексы: примеры

Регекс, учитывающий номер месяца при проверке номера дня:

```
^((0[1-9]|[12][0-9]|3[01])(0[13578]|1[02])
|(0[1-9]|[12][0-9]|30)(0[469]|11)
|(0[1-9]|1[0-9]|2[0-8])02)[12][0-9]{3}$
```

- Регекс предполагает, что год невисокосный — 29 февраля не принимается.
- В идеале, конечно, регекс должен выяснять високосность года, чтобы правильно проверять для февраля! Но это будет весьма сложный регекс.

# Компьютерные регексы: примеры

**Задача:** Придумайте регекс для валидных английских *имён*. Имя состоит из *first name*, опционального *middle name* (или просто инициал) и *last name* — они все разделены пробелами. First name, middle name, last name — должны начинаться на прописную латинскую букву и далее состоят из строчных латинских букв (возможно, ноль). Если middle name — инициал, то он прописная латинская буква с точкой.

Валидные имена:

- Don Quixote Doflamingo
- Luffy D. Monkey
- Eren Yeager

Невалидные имена:

- Erwin smith
- Trafalgar D. Water Law
- Kaido

**Ответ:**

# Компьютерные регексы: примеры

**Задача:** Придумайте регекс для валидных английских *имён*. Имя состоит из *first name*, опционального *middle name* (или просто инициал) и *last name* — они все разделены пробелами. First name, middle name, last name — должны начинаться на прописную латинскую букву и далее состоят из строчных латинских букв (возможно, ноль). Если middle name — инициал, то он прописная латинская буква с точкой.

Валидные имена:

- Don Quixote Doflamingo
- Luffy D. Monkey
- Eren Yeager

Невалидные имена:

- Erwin smith
- Trafalgar D. Water Law
- Kaido

**Ответ:**

```
^[A-Z][a-z]* ([A-Z][a-z]*|[A-Z]\. )?[A-Z][a-z]*$
```

# Компьютерные регексы: примеры

**Задача:** Придумайте регекс для валидных адресов электронной почты. Требования к валидному адресу:

- Имя e-mail (то, что идёт до собачки) состоит только из строчных латинских букв, цифр, дефиса или _. Может начинаться только на букву. Длина хотя бы два.
- Имена доменов e-mail (то, что идёт после собачки) состоят только из строчных латинских букв, цифр или дефиса. Могут начинаться только на букву. Длина хотя бы два.
- Должен быть хотя бы один уровень домена.

**Ответ:**

# Компьютерные регексы: примеры

**Задача:** Придумайте регекс для валидных адресов электронной почты. Требования к валидному адресу:

- Имя e-mail (то, что идёт до собачки) состоит только из строчных латинских букв, цифр, дефиса или _. Может начинаться только на букву. Длина хотя бы два.
- Имена доменов e-mail (то, что идёт после собачки) состоят только из строчных латинских букв, цифр или дефиса. Могут начинаться только на букву. Длина хотя бы два.
- Должен быть хотя бы один уровень домена.

**Ответ:**

```
^([a-z][a-z0-9-_]+@([a-z][a-z0-9-]+\.)+([a-z][a-z0-9-]+)$
```

# Компьютерные регексы: примеры

А в реальной жизни, все-таки используют более сложный регекс.
RFC 5322 Official Standard:

```
(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?
^_`{|}~-]+)*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b
\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@
(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9]
(?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:25[0-5]|2[0-4][0-9]|
[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?
[0-9][0-9]?|[a-z0-9-]*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e
-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)
\])
```

И даже этот регекс работает не на 100%.