Introdução à Programação de Computadores para Biologia

Expressões Regulares

"regex"

Aula 10

https://tttorres.github.io/introprog2021/

ENTRADA DE DADOS

Argumentos do script

Os argumentos podem ser passados para o script na própria linha de comando:

```
TatianasMacBook:~ tatiana$ perl script.pl arg1 arg2 arg3
```

No script, Perl transforma os argumentos em um array, @ARGV:

```
# A variavel @ARGV (ARGument Values)

print "$ARGV[0]\n"; #imprime o primeiro argumento
print "$ARGV[1]\n"; #imprime o segundo argumento

# ...

print "$ARGV[$#ARGV]\n"; #imprime o ultimo argumento
```

ENTRADA E SAÍDA DE DADOS

Arquivos

- Comando *open()*
- filehandle

No script:

```
open(FILEHANDLE, filename);
```

Na linha de comando:

```
Darwin:~ Tatiana$ perl files.pl ~/seq/dmel-gene.fasta
```

(11) 3091-8759

KDG 7447

EXPRESSÕES REGULARES

Buscas

Problema:

> Testar se determinado bloco de caracteres é uma placa de carro

EXPRESSÕES REGULARES

Plano A: checar cada placa

É uma placa de carro?

```
#!/usr/bin/perl

# Plano A: checar cada placa
# quase de 200 milhões de linhas

$ok = 0; # default

print "Placa\:\n";

$placa = <STDIN>;

if ( $placa eq "AAA0000" ) { $ok = 1 }
if ( $placa eq "AAA0001" ) { $ok = 1 }
if ( $placa eq "AAA0002" ) { $ok = 1 }
#  tente imaginar o que vem no meio ...
if ( $placa eq "ZZZ9999" ) { $ok = 1 }
if ( $ok == 1 ) { print "Eh uma placa de carro!" }

exit;
```

EXPRESSÕES REGULARES

Plano B: checar cada caracter

É uma placa de carro?

```
# Plano B: checar cada caracter
# cerca de 130 linhas

$ok = 0; # quantos caracteres estao corretos
$placa = <STDIN>;

$char7 = chop ( $placa );
if ( $char7 eq "0" ) { $ok++ } # mais 1..8
if ( $char7 eq "9" ) { $ok++ }
$char6 = chop ( $placa );
if ( $char6 eq "0" ) { $ok++ } # mais 1..9
# fazer para os quatro digitos e três letras
$char1 = chop ( $plate );
if ( $char1 eq "A" ) { $ok++ }
if ( $char1 eq "B" ) { $ok++ } # mais C..Y
if ( $char1 eq "Z" ) { $ok++ }
# checar se foram sete acertos
if ( $ok == 7 ) { print "Eh uma placa de carro!" }
```

EXPRESSÕES REGULARES

Novo operador =~

Utilizado para comparação e substituição

EXPRESSÕES REGULARES

Novo operador =~

Utilizado para comparação e substituição

```
# Para reconhecer um digito podemos usar a seguinte
# expressao regular:


if ( $char7 =~ m/[0123456789]/ ) {
   $ok++;
}
```

EXPRESSÕES REGULARES

Plano C: checar cada caracter com =~

É uma placa de carro?

```
# Plano C
# menos de 20 linhas

$ok = 0; # default

print "Placa\:\n";

$placa = <STDIN>;

$ch7 = chop ( $placa );
if ( $ch7 =~ m/[0123456789]/ ) { $ok++ }
$ch6 = chop ( $plate );
if ( $ch6 =~ m/[0123456789]/ ) { $ok++ }
# outros digitos
# letras
if ( $ok == 7 ) { print "Eh uma placa de carro!" }

exit;
```

EXPRESSÕES REGULARES

Novo operador =~

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como placas.pl
4. Copiar **exemplo01** da página da disciplina.
5. Alterá-lo para o resultado seguinte:

```
Darwin:~ Tatiana$ perl placas.pl

Placa:
KDG7447 #input do usuario

Possui quatro dígitos! #resposta do script
```

EXPRESSÕES REGULARES

Plano C: checar cada caracter com =~

Os últimos quatro caracteres são dígitos?

```
$ok = 0; # default

print "Placa\:\n";

$placa = <STDIN>;
chomp ( $placa );

$ch7 = chop ( $placa );
if ( $ch7 =~ m/[0123456789]/ ) { $ok++ }
$ch6 = chop ( $placa );
if ( $ch6 =~ m/[0123456789]/ ) { $ok++ }
$ch5 = chop ( $placa );
if ( $ch5 =~ m/[0123456789]/ ) { $ok++ }
$ch4 = chop ( $placa );
if ( $ch4 =~ m/[0123456789]/ ) { $ok++ }
if ( $ok == 4) { print "Possui quatro dígitos!" }

exit;
```

EXPRESSÕES REGULARES

Plano C: checar cada caracter com =~

É uma placa de carro?

```
# Plano C

$ok = 0; # default

print "Placa\:\n";

$placa = <STDIN>;
chomp ( $placa );

$ch7 = chop ( $placa );
if ( $ch7 =~ m/[0123456789]/ ) { $ok++ }
$ch6 = chop ( $placa );
if ( $ch6 =~ m/[0123456789]/ ) { $ok++ }
$ch5 = chop ( $placa );
if ( $ch5 =~ m/[0123456789]/ ) { $ok++ }
$ch4 = chop ( $placa );
if ( $ch4 =~ m/[0123456789]/ ) { $ok++ }
if ( $ok == 4) { print "Possui quatro dígitos!" }

exit;
```

EXPRESSÕES REGULARES

Plano C: checar cada caracter com =~

É uma placa de carro?

```
# Plano C: menos de 20 linhas

$ok = 0; # default

print "Placa\:\n";

$placa = <STDIN>;
chomp ( $placa );

$ch7 = chop ( $placa );
if ( $ch7 =~ m/[0123456789]/ ) { $ok++ }
$ch6 = chop ( $placa );
if ( $ch6 =~ m/[0123456789]/ ) { $ok++ }
# outros digitos
# letras
if ( $ok == 7 ) { print "Eh uma placa de carro!" }

exit;
```

EXPRESSÕES REGULARES

Intervalos de caracteres

Atalho para designar caracteres consecutivos

```
# Digitos:

if ( $char7 =~ m/[0-9]/ ) {
          $ok++
}

# Letras:

if ( $char1 =~ m/[A-Z]/ ) {
          $ok++
}
```

EXPRESSÕES REGULARES

Intervalos de caracteres

Código binário para que codificação de 128 sinais:

- 95 sinais gráficos (letras, pontuação e sinais matemáticos)
- 33 sinais de controle (não imprimíveis, ex: \n, \t)

```
0 in ASCII is 0110000
1 in ASCII is 0110001
2 in ASCII is 0110010
3 in ASCII is 0110011
4 in ASCII is 0110100
5 in ASCII is 0110101
6 in ASCII is 0110110
7 in ASCII is 0110111
8 in ASCII is 0111000
9 in ASCII is 0111001
```

*ASCII American Standard Code for Information Interchange

EXPRESSÕES REGULARES

Tabela ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.asciitable.com

EXPRESSÕES REGULARES

ASCII

Código binário para que codificação de 128 sinais:

- 95 sinais gráficos (letras, pontuação e sinais matemáticos)
- 33 sinais de controle (não imprimíveis, ex: \n, \t)

```
[A-Z]    ADJACENTES

[a-z]    ADJACENTES

[A-z]    NÃO FUNCIONA

[a-Z]    NÃO FUNCIONA
```

EXPRESSÕES REGULARES

Intervalos de caracteres

Alterar o script placas.pl usando intervalo de caracteres

```
# Plano C: menos de 20 linhas

$ok = 0; # default

print "Placa\:\n";

$placa = <STDIN>;
chomp ( $placa );

$ch7 = chop ( $placa );
if ( $ch7 =~ m/[0123456789]/ ) { $ok++ }
$ch6 = chop ( $placa );
if ( $ch6 =~ m/[0123456789]/ ) { $ok++ }
# outros digitos
# letras
if ( $ok == 7 ) { print "Eh uma placa de carro!" }

exit;
```

Alterar o script placas.pl usando intervalo de caracteres

```
# Plano D: menos de 20 linhas

$ok = 0; # default

print "Placa\:\n";
$placa = <STDIN>;

$ch7 = chop ( $placa );
if ( $ch7 =~ m/[0-9]/ ) { $ok++ }
$ch6 = chop ( $placa );
if ( $ch6 =~ m/[0-9]/ ) { $ok++ }
$ch5 = chop ( $placa );
if ( $ch5 =~ m/[0-9]/ ) { $ok++ }
$ch4 = chop ( $placa );
if ( $ch4 =~ m/[0-9]/ ) { $ok++ }
$ch3 = chop ( $placa );
if ( $ch3 =~ m/[A-Z]/ ) { $ok++ }
$ch2 = chop ( $placa );
if ( $ch2 =~ m/[A-Z]/ ) { $ok++ }
$ch1 = chop ( $placa );
if ( $ch1 =~ m/[A-Z]/ ) { $ok++ }

if ( $ok == 7 ) { print "Eh uma placa de  carro!" }

exit;
```

EXPRESSÕES REGULARES

Reconhecimento de padrões

K	D	G		7	4	4	7
L	L	L		D	D	D	D

EXPRESSÕES REGULARES

Reconhecimento de padrões

Comparação de mais de um caracter

```
# Para reconhecer um bloco de texto podemos usar a
# seguinte expressao regular:

# Exemplo: KDG7447 (LLLDDDD)

$placa =~ m/[A-Z][A-Z][A-Z][0-9][0-9][0-9][0-9]/;
#           L    L    L    D    D    D    D

exit;
```

EXPRESSÕES REGULARES

Reconhecimento de padrões

Alterar o script placas.pl usando intervalo de caracteres

```
$ok = 0; # default
print "Placa\:\n";

$placa = <STDIN>;
chomp ( $placa );

$ch7 = chop ( $placa );
if ( $ch7 =~ m/[0-9]/ ) { $ok++ }
$ch6 = chop ( $placa );
if ( $ch6 =~ m/[0-9]/ ) { $ok++ }
$ch5 = chop ( $placa );
if ( $ch5 =~ m/[0-9]/ ) { $ok++ }
$ch4 = chop ( $placa );
if ( $ch4 =~ m/[0-9]/ ) { $ok++ }
$ch3 = chop ( $placa );
if ( $ch3 =~ m/[A-Z]/ ) { $ok++ }
$ch2 = chop ( $placa );
if ( $ch2 =~ m/[A-Z]/ ) { $ok++ }
$ch1 = chop ( $placa );
if ( $ch1 =~ m/[A-Z]/ ) { $ok++ }

if ( $ok == 7 ) { print "Eh uma placa de carro!" }

exit;
```

EXPRESSÕES REGULARES

Reconhecimento de padrões

Alterar o script placas.pl usando intervalo de caracteres

```
# Plano E: 5 linhas

#!/usr/bin/perl

print "Placa\:\n";
$placa = <STDIN>;

if ( $placa =~ m/[A-Z][A-Z][A-Z][0-9][0-9][0-9][0-9]/ ) {
   print "Eh uma placa de  carro!\n";
}

exit;
```

EXPRESSÕES REGULARES

Multiplicadores

```
K D G   7 4 4 7
L * 3   D * 4 .
```

EXPRESSÕES REGULARES

Reconhecimento de padrões

Comparação de mais de um caracter

```
# Para reconhecer um bloco de texto podemos usar a
# seguinte expressao regular:

# Exemplo: KDG7447 (L*3 D*4)

$placa =~ m/[A-Z]{3}[0-9]{4}/;
#             L * 3   D * 4


exit;
```

EXPRESSÕES REGULARES

Reconhecimento de padrões

Alterar o script placas.pl usando intervalo de caracteres

```
# Plano E: 5 linhas

#!/usr/bin/perl

print "Placa\:\n";
$placa = <STDIN>;

if ( $placa =~ m/[A-Z][A-Z][A-Z][0-9][0-9][0-9][0-9]/ ) {
   print "Eh uma placa de  carro!\n";
}

exit;
```

EXPRESSÕES REGULARES

Reconhecimento de padrões

Alterar o script placas.pl usando intervalo de caracteres

```
# Plano E: 5 linhas

#!/usr/bin/perl

print "Placa\:\n";
$placa = <STDIN>;

if ( $placa =~ m/[A-Z]{3}[0-9]{4}/ ) {
   print "Eh uma placa de  carro!\n";
}


exit;
```

EXPRESSÕES REGULARES

Reconhecimento de padrões

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como telefone.pl
4. Copiar **exemplo02** da página da disciplina.
5. Copie o exemplo 2, e complete o script para reconhecer números de telefones fixos brasileiros, no seguinte formato: (11)3091-8759

```
Darwin:~ Tatiana$ perl telefone.pl

Telefone:
(11)3091-8759 #input do usuario

Eh um telefone! #resposta do script
```

EXPRESSÕES REGULARES

Reconhecimento de padrões

Script: telefone.pl

```
##!/usr/bin/perl

# formato (11)3091-8759

print "Telefone\:\n";
$tel = <STDIN>;

if ( $tel =~ /\([0-9]{2}\)[0-9]{4}\-[0-9]{4}/ ) {
        print "Eh um telefone!\n";
}

exit;
```

EXPRESSÕES REGULARES

Reconhecimento de padrões

Altere o script telefone.pl para reconhecer telefones fixos E celulares de São Paulo.

```
##!/usr/bin/perl

# formato (11)3091-8759 ou (11)93091-8759

print "Telefone\:\n";
$tel = <STDIN>;

if ( $tel =~ /\([0-9]{2}\)[0-9]{4}\-[0-9]{4}/ ) {
        print "Eh um telefone!\n";
}



exit;
```

EXPRESSÕES REGULARES

Reconhecimento de padrões

Altere o script telefone.pl para reconhecer telefones fixos E celulares de São Paulo.

```
##!/usr/bin/perl

# formato (11)3091-8759 ou (11)93091-8759

print "Telefone\:\n";
$tel = <STDIN>;

if ( $tel =~ /\([0-9]{2}\)[0-9]{4}\-[0-9]{4}/ ) {
   print "Eh um telefone!\n";
} elsif ( $tel =~ /\([0-9]{2}\)[0-9]{5}\-[0-9]{4}/ ) {
   print "Eh um telefone!\n";
}

exit;
```

EXPRESSÕES REGULARES

Flexibilidade

Altere o script telefone.pl para reconhecer telefones fixos E celulares de São Paulo.

```
##!/usr/bin/perl

# formato (11)3091-8759 ou (11)93091-8759

print "Telefone\:\n";
$tel = <STDIN>;

if ( $tel =~ /\([0-9]{2}\)[0-9]{4,5}\-[0-9]{4}/ ) {
   print "Eh um telefone!\n";
}

exit;
```

EXPRESSÕES REGULARES

Flexibilidade

Altere o script telefone.pl para reconhecer telefones fixos E celulares de São Paulo.

```
##!/usr/bin/perl

# formato (11)3091-8759 ou (11)93091-8759

print "Telefone\:\n";
$tel = <STDIN>;

if ( $tel =~ /\(\d{2}\)\d{4,5}\-\d{4}/ ) {
   print "Eh um telefone!\n";
}

exit;
```

EXPRESSÕES REGULARES

Flexibilidade

CARACTER	SIGNIFICADO
\n	Nova linha
\t	Tabulação
\w	Alfanumérico e "_"
\W	Tudo não conteplado em \w
\s	Espaço em branco
\d	Dígito
\D	Não dígito
.	Qualquer caracter, exceto \n

EXPRESSÕES REGULARES

Flexibilidade

Outros exemplos:

- Mínimo de 1 e máximo de 5 {1,5}
- Três ou mais repetições {3,}
- Menos de 6 {0,5}

EXPRESSÕES REGULARES

Flexibilidade

Quantificadores:

CARACTERES	FUNÇÃO
{n}	Exatamente "n" ocorrências
{n,}	Pelo menos "n" ocorrências
{n,m}	Mínimo de "n" e máximo de "m" ocorrências
*	{0,}
+	{1,}
?	{0,1}

EXPRESSÕES REGULARES

Flexibilidade

Altere o script telefone.pl para reconhecer telefones fixos E celulares de São Paulo com os possíveis formatos:

`(11)3091-8759` `(11)93091-8759` `(11)30918759`
`(11)930918759` `113091-8759` `1193091-8759`
`1130918759` `11930918759`

```
##!/usr/bin/perl

print "Telefone\:\n";
$tel = <STDIN>;

if ( $tel =~ /\(\d{2}\)\d{4,5}\-\d{4}/ ) {
   print "Eh um telefone!\n";
}

exit;
```

EXPRESSÕES REGULARES

Flexibilidade

Altere o script telefone.pl para reconhecer telefones fixos E celulares de São Paulo com os possíveis formatos:

`(11)3091-8759` `(11)93091-8759` `(11)30918759`
`(11)930918759` `113091-8759` `1193091-8759`
`1130918759` `11930918759`

```
##!/usr/bin/perl

print "Telefone\:\n";
$tel = <STDIN>;

if ( $tel =~ /\({0,1}\d{2}\){0,1}\d{4,5}\-{0,1}\d{4}/ ) {
   print "Eh um telefone!\n";
}

exit;
```

EXPRESSÕES REGULARES

Flexibilidade

Altere o script telefone.pl para reconhecer telefones fixos E celulares de São Paulo com os possíveis formatos:

`(11)3091-8759` `(11)93091-8759` `(11)30918759`
`(11)930918759` `113091-8759` `1193091-8759`
`1130918759` `11930918759`

```
##!/usr/bin/perl

print "Telefone\:\n";
$tel = <STDIN>;

if ( $tel =~ //\(?\d{2}\)?\d{4,5}\-?\d{4}/ ) {
   print "Eh um telefone!\n";
}

exit;
```