



# PERL & PYTHON

**Course Code: CS466**

**No. of Credit Hours: 2 credits (LEC)**

Lecturer: Dang, Tran Huu Minh

Mobile/ Zalo: 0918.763.367

Email: [tranhminhdang@dtu.edu.vn](mailto:tranhminhdang@dtu.edu.vn)



## GIỚI THIỆU THÔNG TIN SLIDE

Chủ đề:

# Data Structures in Perl

Thời lượng: 120 phút

Lecturer: Dang, Tran Huu Minh

Mobile/ Zalo: 0918.763.367

Email: [tranhminhdang@dtu.edu.vn](mailto:tranhminhdang@dtu.edu.vn)

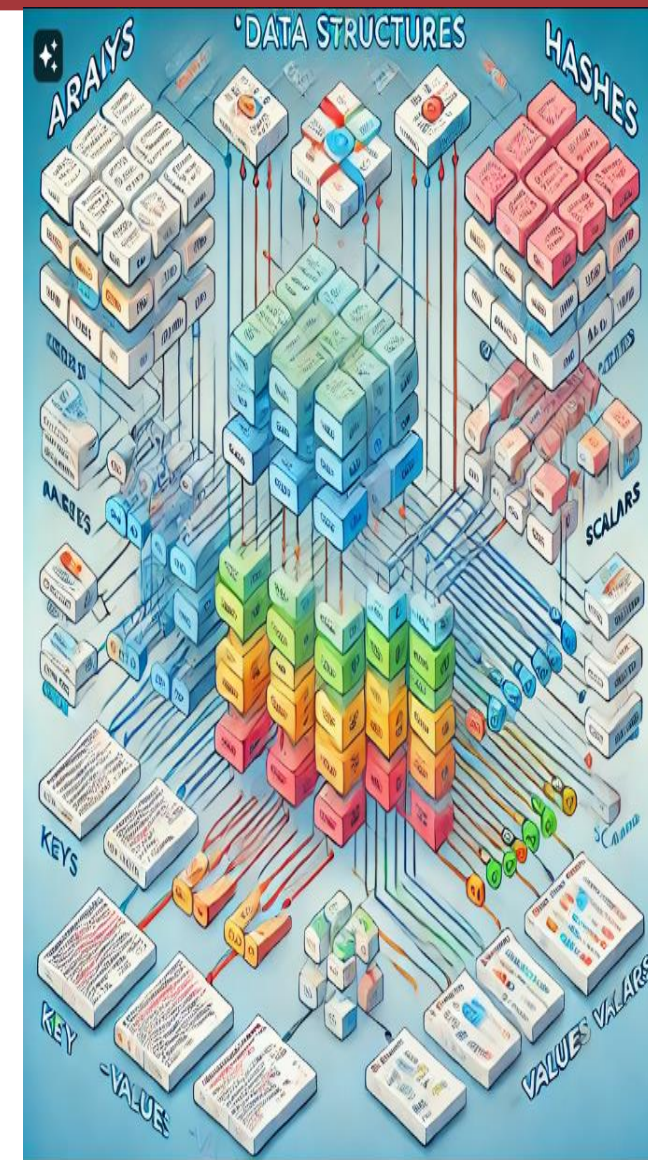




## ❑ MỤC TIÊU:

Sau khi tìm hiểu và nghiên cứu, sinh viên sẽ có được những khả năng sau:

- 01 Hiểu rõ về các cấu trúc dữ liệu cơ bản trong Perl
- 02 Hiểu và sử dụng mảng (array), hash (associative array), danh sách liên kết (linked list), chuỗi (string), tệp (file) trong Perl.
- 03 Hiểu được cách chuyển đổi giữa chúng, tìm kiếm, sắp xếp và lọc dữ liệu.
- 04 *Vận dụng kiến thức để giải quyết các bài toán lập trình cụ thể*





# Reference materials

## Textbooks:

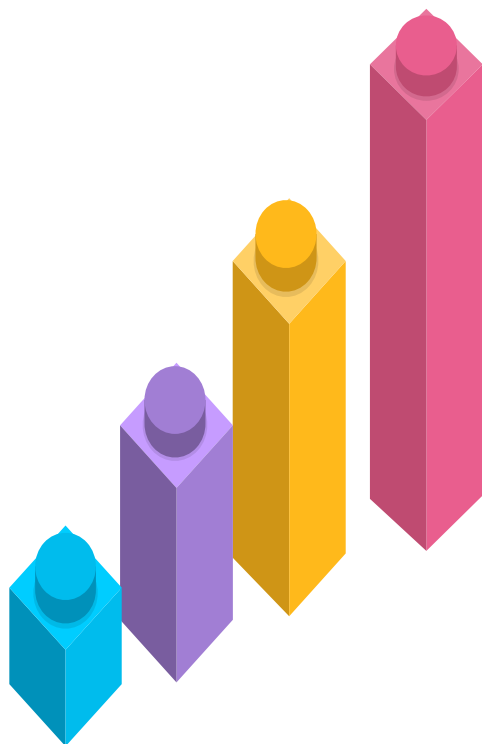
1. Randal L. Schwartz, Tom Phoenix, Brian D Foy (2021). *Learning Perl: Making Easy Things Easy and Hard Things Possible, 8th Edition*. O'Reilly Media.
2. Eric Chou (2023). *Mastering Python Networking, Fourth Edition*. Packt Publishing.

## Reference Materials:

1. Florian Dedov (2020). *The Python Bible 7 in 1: Volumes One To Seven (Beginner, Intermediate, Data Science, Machine Learning, Finance, Neural Networks, Computer Vision)*. Independently published.



## NỘI DUNG



Arrays, Hashes, Scalars



Regular Expressions And Strings



Files



Conclusion





# Arrays in Perl

Mảng được sắp xếp theo thứ tự danh sách các phần tử.

### 1 Ordered collections

Lưu trữ một chuỗi các yếu tố.

### 3 Dynamic sizing

Phát triển hoặc thu nhỏ khi cần thiết.

### 2 Indexed access

### 4 Versatile





## Creating Arrays in Perl

Sử dụng biểu tượng @ để tạo mảng.

**Directly**

Use: `@array = (1, 2, 3);`

**Using qw**

Use: `@array = qw(one two three);`

## Accessing Array Elements

Sử dụng các chỉ mục bắt đầu từ 0.

Ví dụ:

```
@array = (1, 2, 3);
```

```
$element = $array[1];
```

```
print $element; # Outputs 2
```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```



# Array Operations

Thực hiện các hành động trên các phần tử mảng.

1

## Indexing

Truy cập các yếu tố riêng lẻ theo chỉ mục của chúng.

2

## Slicing

Trích xuất một phần của mảng.

3

## Appending

Thêm các phần tử mới vào cuối mảng.

## Array Operations

> spekrtin;  
Orryper unista laoney:

> spekriion,  
Ornype unista laoresy

### aclding

>= direo (f00l aarg);  
Proestug(l);  
Orivpre-unisal laonee:  
|

Ixe - drating),  
Axperiroa()  
Oryppe unista lkonse:

### slicing

>=>> (Ipraper),  
XUB;  
Inryper-unista lkonse:

### iniding

x=> = 90a(07) =ast);  
Movuc:le;  
Oryppe untista lkonse:

### rending

> dlat Opertions attp()  
irre= alto,  
Onrvppe untista lkonse:

### ||

Pritstiltg (beagreal),  
attp();  
| rryper distal laonse:





# Looping Through Arrays



Lopf 2

Xử lý từng phần tử trong mảng.



Apuna > 7

1

For Loop

Use: `for (my $i = 0; $i < @array; $i++)`

2

Foreach Loop

Use: `foreach my $element (@array)`

2. hes neluci to prase wayed: 3

4. hes neluci to prase waJue: 2

next: leaah leauf 4

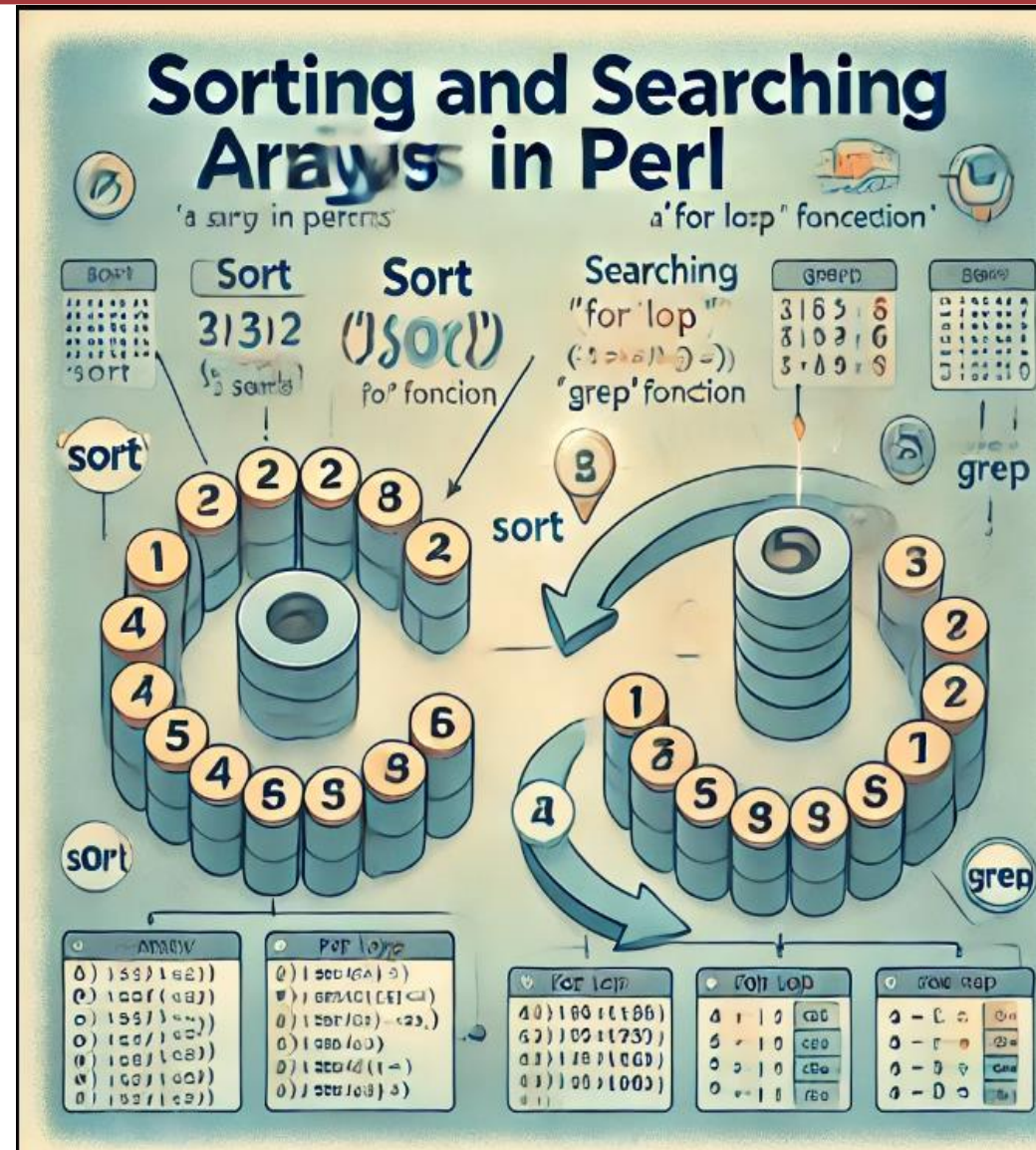


# Sorting and Searching Arrays

Đặt hàng các yếu tố và tìm các yếu tố cụ thể.

**Sort** Sử dụng "sắp xếp" để sắp xếp các phần tử theo thứ tự tăng dần.

**Search** Sử dụng "grep" để tìm các phần tử phù hợp với một điều kiện nhất định.





# Multidimensional Arrays

## Mảng chứa các mảng khác.



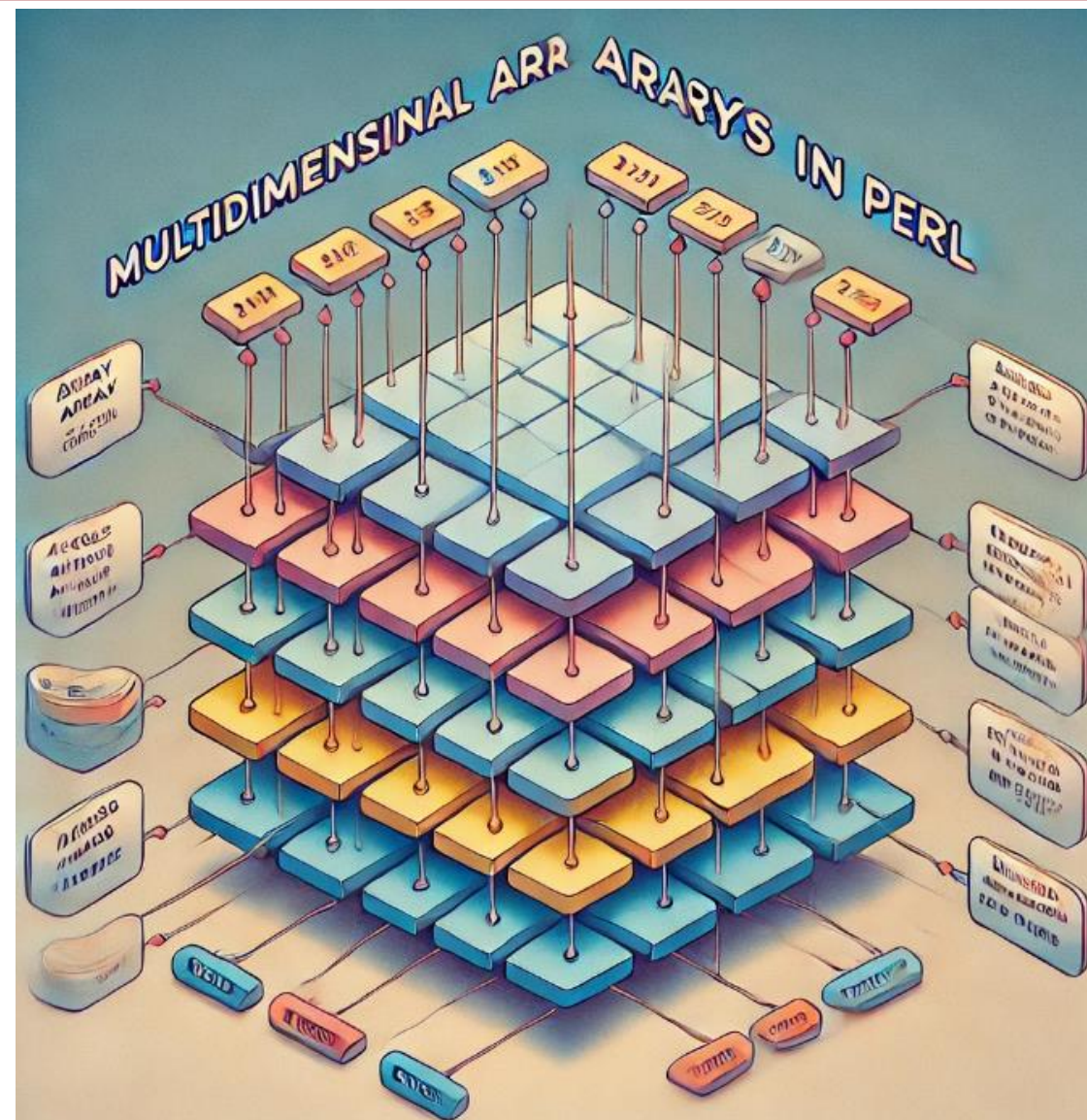
# Structure

Sắp xếp dữ liệu  
theo hàng và cột.



# Accessing

Sử dụng nhiều chỉ mục để truy cập các phần tử.





```
Perl functions - catfial methods
remell  Pretures  Lealls  Serves
1 Perl array: intencionl ipenther:
2
3   cvover unce,ararte:
6   perric(Patectionl perlntetaley;
9   erectve tercoratanlicnte(anciod:
6   cerra/latestacteer) = proript colospager. (pencid);
11   perl prates ecall
14   irnificators tevilal netraplacraad:
15   cescll- mnteck
70   inttecl.oter perflerante: eary;
90   artent/Patect soposial cast, reafous/c.arteny.5secpler.
100   intermaater.ragen.-perreate BMy ints/Colcalut.
129   cescll redfrees tatp- erenplc;
125   prrratllane/Cotel lasch = caget:
217   pertil.Aaver porcass: WNR
377   perric;Fntectiony igrtaClonem: arient:
397   ▼ perric/Patocll lanced ApericalDecto):
873   parril/Racolly.,usee/lnviet:/ eaclypletran(chomes: (Paxes(Rolal, → Intle90);
817   infreactions.(onAwalesples; add the caffeme);
224   ▼ perrico.osset(Dietapacme) {
274   arreval cetteglane: = Inteid.4nction);
284   arsancaliota);
329   petsica: restatle; Mosel -acid, recvide:,
298   prmpetllard ates coprizated (ahter; forciare car offfecne);
335   aer pectraiaclly frenciic.
127   cortresate Apert determination tesef with @plat(icate; ← sectinstictl);
238   coseratstaracly.asers./aosdertyforcoasing.
309   te vervians correctner luptal Methods:
931   /rule
294   intections: and fesal = crater(Dover.epersionsalify to cotelectapping,
915   intections: and.(Dersp.otraztlome(priorore.: ear detert integranen),
397   for aditentanwing lo asoletles.
397   stepules.
128   berll(innaters Mettentas, ide E bbiteing'; and...worrid interints.Astaltas
399   interteritall by the farther bespired, artirgaion, or: ureracty netipolcally,
207   fortact irtecriing recenicy:
817   cctack lande.
295   tarts ecclal renteciate:
916   capteners:
390   firtter and arrayl intercallia))
```

# Array Functions and Methods

Perform operations on entire arrays.

1 `@array = reverse(@array);`

Reverse the order of elements.

2 `@array = sort(@array);`

Sort the array elements.

3 `push(@array, "new")`

Add an element to the end.



# Practical Applications

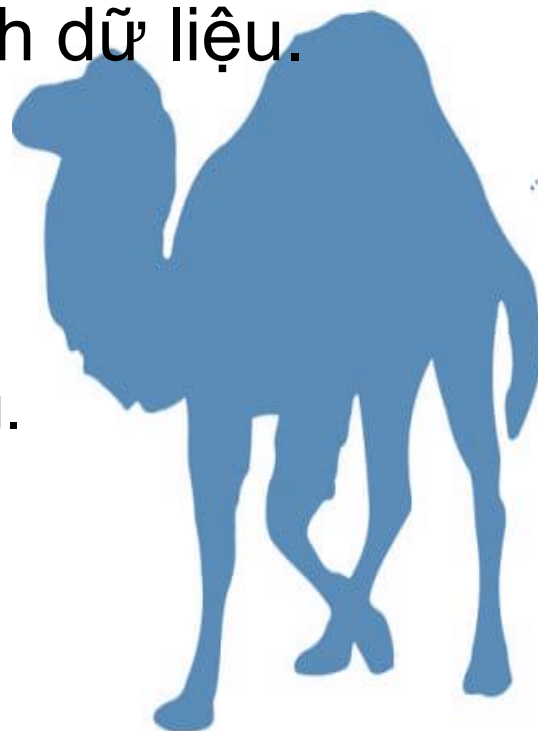
Lưu trữ, thao tác và phân tích dữ liệu.

## Web Development

Lưu trữ dữ liệu người dùng,  
xử lý yêu cầu và hiển thị nội dung.

## Data Analysis

Phân tích các bộ dữ liệu lớn, xác  
định các mẫu và tạo thông tin  
chuyên sâu.

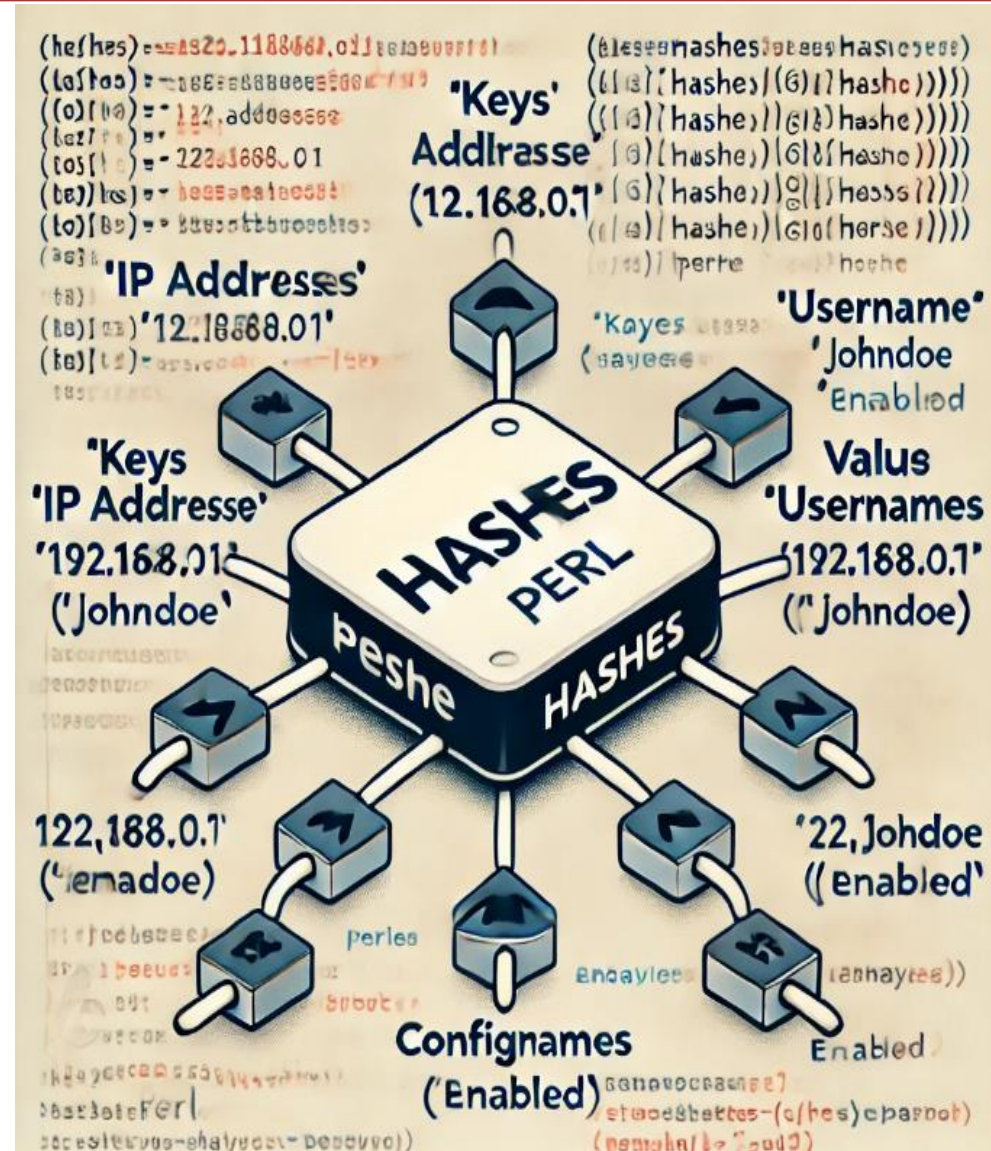






# Hashes in Perl

Hashes là cấu trúc dữ liệu cơ bản trong Perl. Chúng là các mảng kết hợp, có nghĩa là các phần tử được truy cập bằng cách sử dụng các cặp khóa-giá trị. Chúng rất linh hoạt và được sử dụng rộng rãi trong lập trình Perl.





Để tạo hàm băm, bạn gán nó cho một biến bằng dấu ngoặc nhọn {}. Các cặp khóa-giá trị được phân tách bằng dấu phẩy và được gán với toán tử =>.

Ví dụ: %my\_hash = ('key1' => 'value1', 'key2' => 'value2');

### Literal Creation

Chỉ định trực tiếp các cặp khóa-giá trị trong niềng răng xoắn.

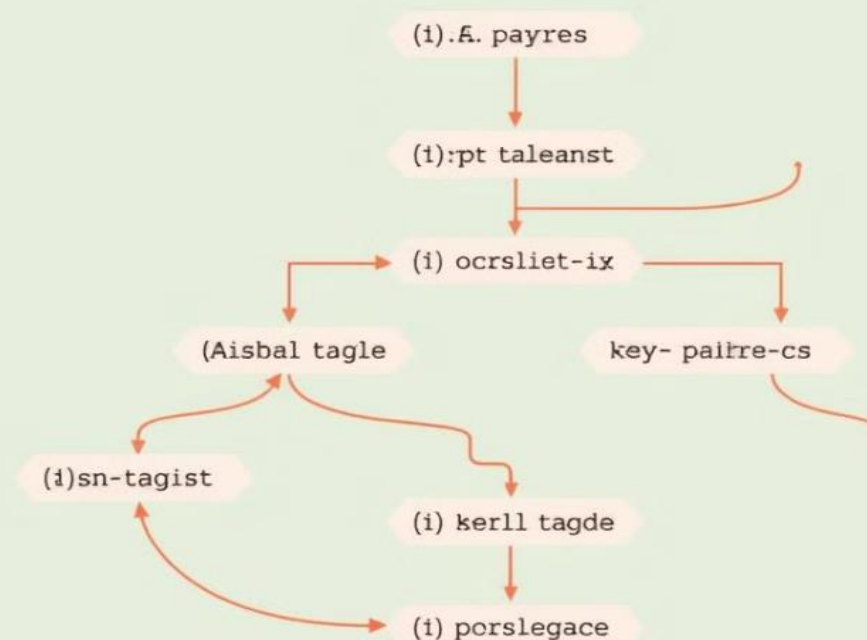
### Using the 'keys' function

Tạo khóa dựa trên các giá trị hiện có hoặc các biến khác.

### Hash Slices

Tạo hàm băm phụ với các khóa cụ thể cho các trường hợp sử dụng cụ thể.

## Creating a Hash





# Accessing Hash Elements

Truy cập các phần tử trong hàm băm được thực hiện bằng cách sử dụng khóa trong dấu ngoặc vuông [].

Ví dụ: `$value = $my_hash['key1'];`

Thao tác này sẽ truy xuất giá trị được liên kết với khóa 'key1'.

## Direct Access

Truy cập giá trị trực tiếp bằng cách sử dụng khóa trong dấu ngoặc vuông.

## Exists Operator

Kiểm tra xem khóa có tồn tại trong hàm băm hay không bằng toán tử 'tồn tại'.

## Default Value

Sử dụng toán tử '//' để cung cấp giá trị mặc định nếu không tìm thấy khóa.



# Modifying Hash Elements

- Sửa đổi các phần tử băm bằng cách gán một giá trị mới cho khóa hiện có.
- Ví dụ: `$my_hash['key1'] = 'new_value';`
- Điều này thay thế giá trị cũ được liên kết với 'key1' bằng 'new\_value'.

## 1 Direct Assignment

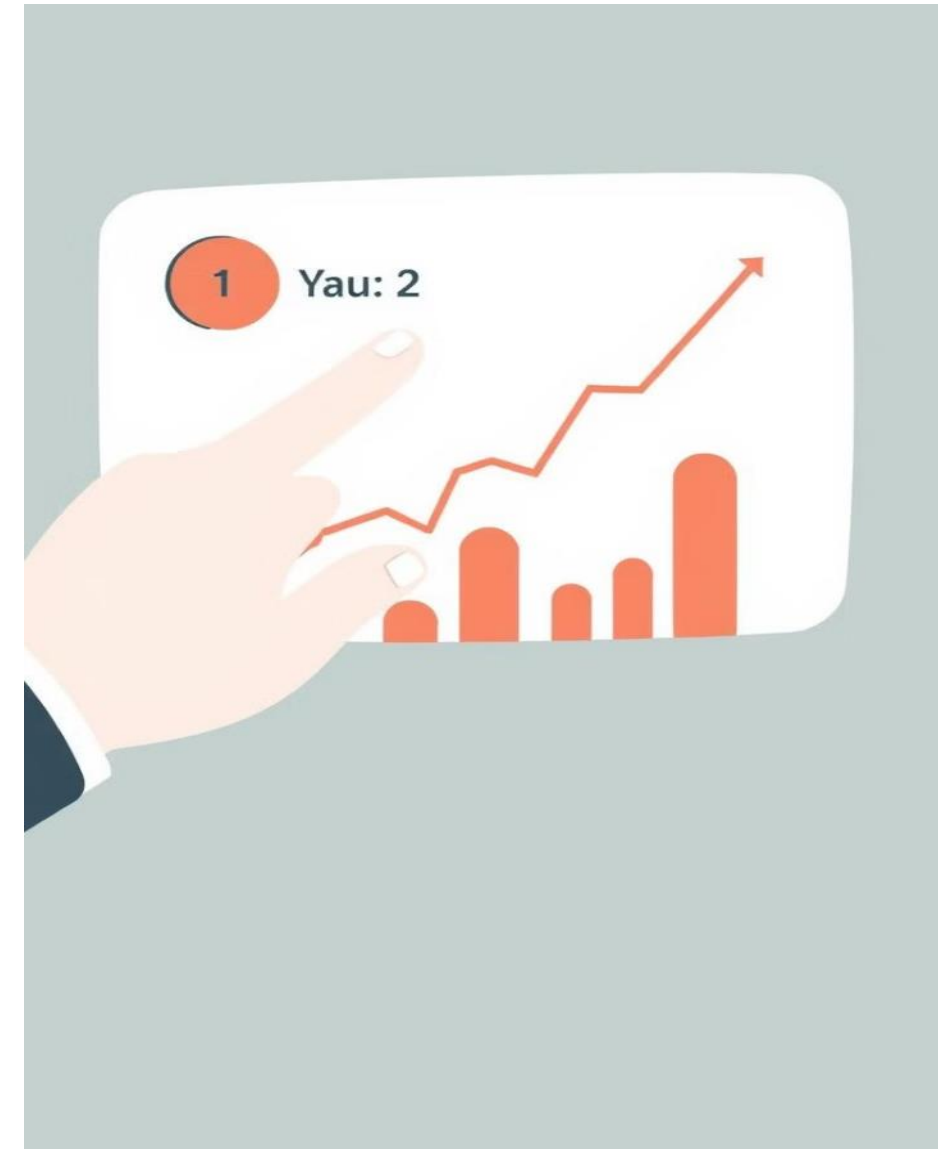
Gán một giá trị mới cho khóa hiện có.

## 2 Using the 'delete' operator

Xóa một phần tử khỏi hàm băm.

## 3 Hash Slices

Sửa đổi nhiều yếu tố cùng một lúc.





## Deleting Hash Elements

- Để xóa phần tử khỏi hàm băm, hãy sử dụng từ khóa xóa theo sau là tên băm và khóa trong dấu ngoặc vuông.
- Ví dụ: `delete $my_hash['key1'];`
- *Thao tác này sẽ loại bỏ phần tử được liên kết với khóa 'key1'.*

1

### Identify Target

Xác định cặp khóa-giá trị cụ thể cần xóa.

2

### Use delete Keyword

Sử dụng từ khóa 'xóa' theo sau là tên băm và khóa.

3

### Confirm Deletion

Xác minh rằng phần tử đã được loại bỏ thành công.





## Iterating Over a Hash

- Bạn có thể lặp qua một hàm băm bằng cách sử dụng vòng lặp 'foreach'.
- Ví dụ: `foreach my $key (keys %my_hash) { print "$key => $my_hash{$key}\n"; }`
- Thao tác này lặp qua từng khóa trong hàm băm và in cặp khóa-giá trị.

1

### Loop Initialization

Khởi tạo vòng lặp 'foreach' để lặp qua các khóa băm.

2

### Key Iteration

Lặp qua từng khóa trong hàm băm bằng hàm 'keys'.

3

### Value Access

Truy cập giá trị tương ứng được liên kết với mỗi khóa.



# Hash Functions and Methods

- Perl cung cấp các hàm và phương thức tích hợp để làm việc với các hàm băm.
- Ví dụ, `keys()`, `values()`, `exists()`, `delete()`, and more.
- Các chức năng và phương thức này có thể được sử dụng cho các tác vụ thao tác băm khác nhau.

<b>keys()</b>	Returns a list of keys in a hash.
<b>values()</b>	Returns a list of values in a hash.
<b>exists()</b>	Checks if a key exists in a hash.
<b>delete()</b>	Removes an element from a hash.

## HASH TABLE

Lookup)

(Hash)

Insertion →

```
Hasst :: let3  
huolt :: let3  
1es5t :: let3
```

hiylps

→ deletion



# Sorting Hashes

Hashes vốn không được sắp xếp. Để sắp xếp hàm băm, bạn cần sử dụng các hàm hoạt động trên danh sách, chẳng hạn như `sort()`.

Ví dụ: `@sorted_keys = sort keys %my_hash;`

*Điều này sắp xếp các phím theo thứ tự bảng chữ cái.*



## Sorting by Key

Sắp xếp các phần tử băm dựa trên khóa của chúng.



## Sorting by Value

Sắp xếp các phần tử băm dựa trên giá trị của chúng.



## Custom Sorting

Áp dụng logic sắp xếp tùy chỉnh bằng cách sử dụng hàm so sánh.



# Hash References

Tham chiếu băm là con trỏ đến các đối tượng băm. Chúng cho phép bạn truyền hàm băm làm đối số cho các hàm hoặc lưu trữ chúng trong các cấu trúc dữ liệu khác.

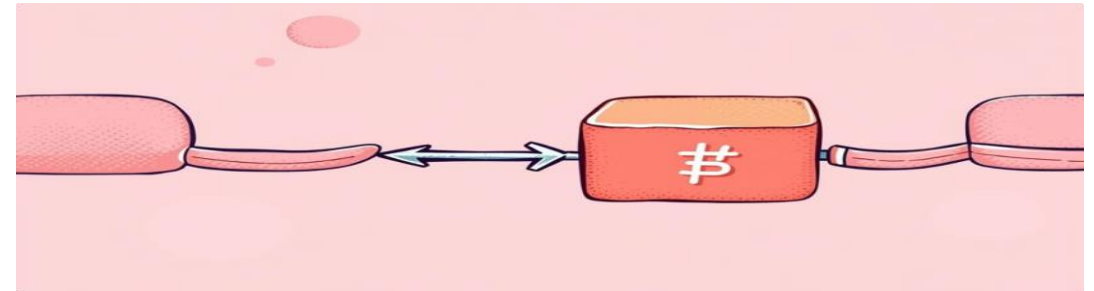
Ví dụ: `$hash_ref = \%my_hash;`

*Điều này tạo ra một tham chiếu đến hàm băm %my\_hash.*



## Pointer to Hash

Tham chiếu băm hoạt động như con trỏ đến đối tượng băm thực tế trong bộ nhớ.



## Indirect Access

Truy cập các phần tử trong hàm băm bằng cách sử dụng tham chiếu băm và khóa.



# Use Cases for Hashes in Perl

Băm rất cần thiết cho các tác vụ khác nhau trong lập trình Perl. Chúng thường được sử dụng để lưu trữ cấu hình, ánh xạ dữ liệu, tạo bảng tra cứu, xây dựng từ điển và triển khai cấu trúc dữ liệu.

## 1 Configuration Files

Lưu trữ các thiết lập và tham số ứng dụng.

## 2 Data Mapping

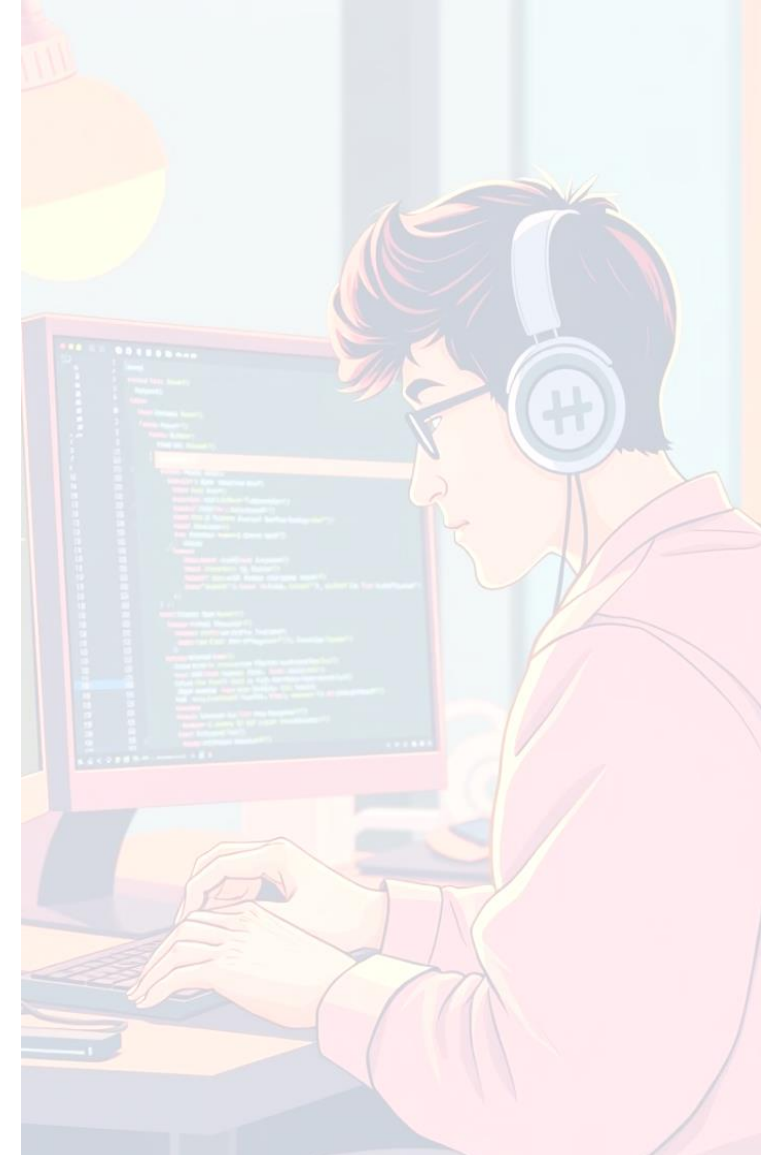
Ánh xạ các giá trị từ nguồn dữ liệu này sang nguồn dữ liệu khác.

## 3 Lookup Tables

Tìm kiếm thông tin hiệu quả dựa trên các khóa cụ thể.

## 4 Data Structures

Triển khai các cấu trúc dữ liệu phức tạp như đồ thị hoặc cây.







# Scalars in Perl

Vô hướng là các khối xây dựng cơ bản của dữ liệu trong Perl. Chúng đại diện cho các giá trị đơn, chẳng hạn như số, chuỗi hoặc giá trị boolean. Hiểu vô hướng là điều cần thiết cho bất kỳ lập trình viên Perl nào.





## Definition of Scalars

Vô hướng là một giá trị dữ liệu duy nhất có thể là số, chuỗi hoặc boolean. Vô hướng là kiểu dữ liệu đơn giản nhất trong Perl và là nền tảng cho ngôn ngữ.

### Numbers

Số đại diện cho các giá trị số, chẳng hạn như số nguyên hoặc số dấu phẩy động.  
Ví dụ, 10, 3.14, and -5 are all numbers.

### Strings

Chuỗi đại diện cho dữ liệu văn bản, chẳng hạn như từ hoặc câu.  
Ví dụ, "Hello world" and "Perl is great" are both strings.

### Booleans

Booleans đại diện cho các giá trị sự thật, đúng hoặc sai. Trong Perl, true được biểu diễn bằng 1 và false được biểu thị bằng 0.



# Scalar Variables

Các biến vô hướng trong Perl được sử dụng để lưu trữ các giá trị đơn lẻ. Chúng được khai báo bằng ký hiệu đô la (\$) theo sau là tên.

## 1 Naming Conventions

Tên biến trong Perl có thể bắt đầu bằng một chữ cái hoặc dấu gạch dưới và có thể chứa các chữ cái, số và dấu gạch dưới. Hãy sử dụng tên có ý nghĩa.

## 2 Scope

Phạm vi biến xác định nơi một biến có thể truy cập được. Các biến cục bộ chỉ hiển thị trong khối hiện tại, trong khi các biến toàn cục có thể truy cập được trong toàn bộ chương trình.

## 3 Assignment

Để gán giá trị cho biến vô hướng, hãy dùng toán tử gán (=). Ví dụ, `$name = "John"` assigns the string "John" to the variable `$name`.



# Scalar Operators

Toán tử vô hướng thực hiện các hoạt động trên các giá trị vô hướng. Chúng được sử dụng để thao tác dữ liệu và tạo ra các giá trị mới.

Operator	Description	Ví dụ
+	Addition	<code>\$sum = 5 + 3; # \$sum = 8</code>
-	Subtraction	<code>\$difference = 10 - 4; # \$difference = 6</code>
*	Multiplication	<code>\$product = 2 * 6; # \$product = 12</code>
/	Division	<code>\$quotient = 15 / 3; # \$quotient = 5</code>
	Exponentiation	<code>\$power = 2 3; # \$power = 8</code>
<code>==</code>	Equality	<code>\$result = (5 == 5); # \$result is true</code>
<code>!=</code>	Inequality	<code>\$result = (10 != 5); # \$result is true</code>
<code>&lt;</code>	Less than	<code>\$result = (3 &lt; 5); # \$result is true</code>
<code>&gt;</code>	Greater than	<code>\$result = (8 &gt; 2); # \$result is true</code>
<code>&lt;=</code>	Less than or equal to	<code>\$result = (5 &lt;= 5); # \$result is true</code>
<code>&gt;=</code>	Greater than or equal to	<code>\$result = (10 &gt;= 5); # \$result is true</code>
.	String concatenation	<code>\$greeting = "Hello" . " world!"; # \$greeting is "Hello world!"</code>



## Scalar Literals

Chữ vô hướng là các giá trị không đổi đại diện cho dữ liệu cụ thể.

Chúng được ghi trực tiếp vào mã và giá trị của chúng không thể thay đổi.



## # Numeric Literals

- Chữ số đại diện cho các giá trị số, chẳng hạn như số nguyên và số dấu phẩy động.
- Ví dụ, 10, 3.14, and -5 are all numeric literals.

## T String Literals

- Chuỗi chữ đại diện cho dữ liệu văn bản. Chúng được đặt trong dấu ngoặc kép đơn hoặc kép.
- Ví dụ, "Hello world" and 'Perl is great' are both string literals.

## ✓ Boolean Literals

- Nghĩa đen Boolean đại diện cho các giá trị sự thật: đúng hoặc sai. Trong Perl, 1 đại diện cho true và 0 đại diện cho false. Bạn cũng có thể sử dụng các từ khóa "true" và "false" để thể hiện các giá trị boolean.





# Scalar Interpolation

Nội suy vô hướng là một tính năng mạnh mẽ của Perl cho phép bạn nhúng các biến trực tiếp vào chuỗi.

1

## Double Quotes

Khi một chuỗi chữ được đặt trong dấu ngoặc kép, Perl sẽ thay thế các biến trong chuỗi bằng các giá trị hiện tại của chúng.

2

## Variable Substitution

Các biến được thể hiện bằng ký hiệu đô la (\$) theo sau là tên của chúng. Ví dụ, "\$name" will be replaced with the value stored in the variable \$name.

3

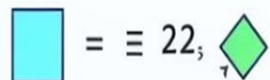
## String Concatenation

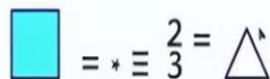
Nếu một chuỗi chứa nhiều biến, Perl sẽ nối các giá trị của chúng thành một chuỗi duy nhất. Điều này tạo ra các chuỗi năng động và linh hoạt.



# Scalar Functions

Hàm vô hướng là các hàm tích hợp hoạt động trên các giá trị vô hướng và trả về kết quả vô hướng. Chúng được sử dụng để thực hiện các tác vụ phổ biến và thao tác dữ liệu.


$$\square = \equiv 22, \diamond$$


$$\square = * \equiv \frac{2}{3} = \triangle$$

## 1 Length

Hàm length (length()) trả về số ký tự trong một chuỗi.

## 2 Conversion

Các hàm như int() và float() chuyển đổi giá trị giữa các kiểu số khác nhau, trong khi các hàm như chr() và ord() chuyển đổi giữa các ký tự và giá trị ASCII của chúng.

## 3 Comparison

Các hàm như lc() và uc() lần lượt chuyển đổi chuỗi thành chữ thường và chữ hoa.

## 4 String Manipulation

Các hàm như substr(), index() và reverse() thao tác với các chuỗi con bằng cách trích xuất các chuỗi con, tìm vị trí ký tự hoặc đảo ngược thứ tự của các ký tự.



# Scalar Input and Output

Perl cung cấp các cách để lấy đầu vào từ người dùng và hiển thị đầu ra cho bảng điều khiển hoặc các tệp.

1

## Input

Hàm < đọc một dòng văn bản từ đầu vào tiêu chuẩn (thường là bàn phím).

2

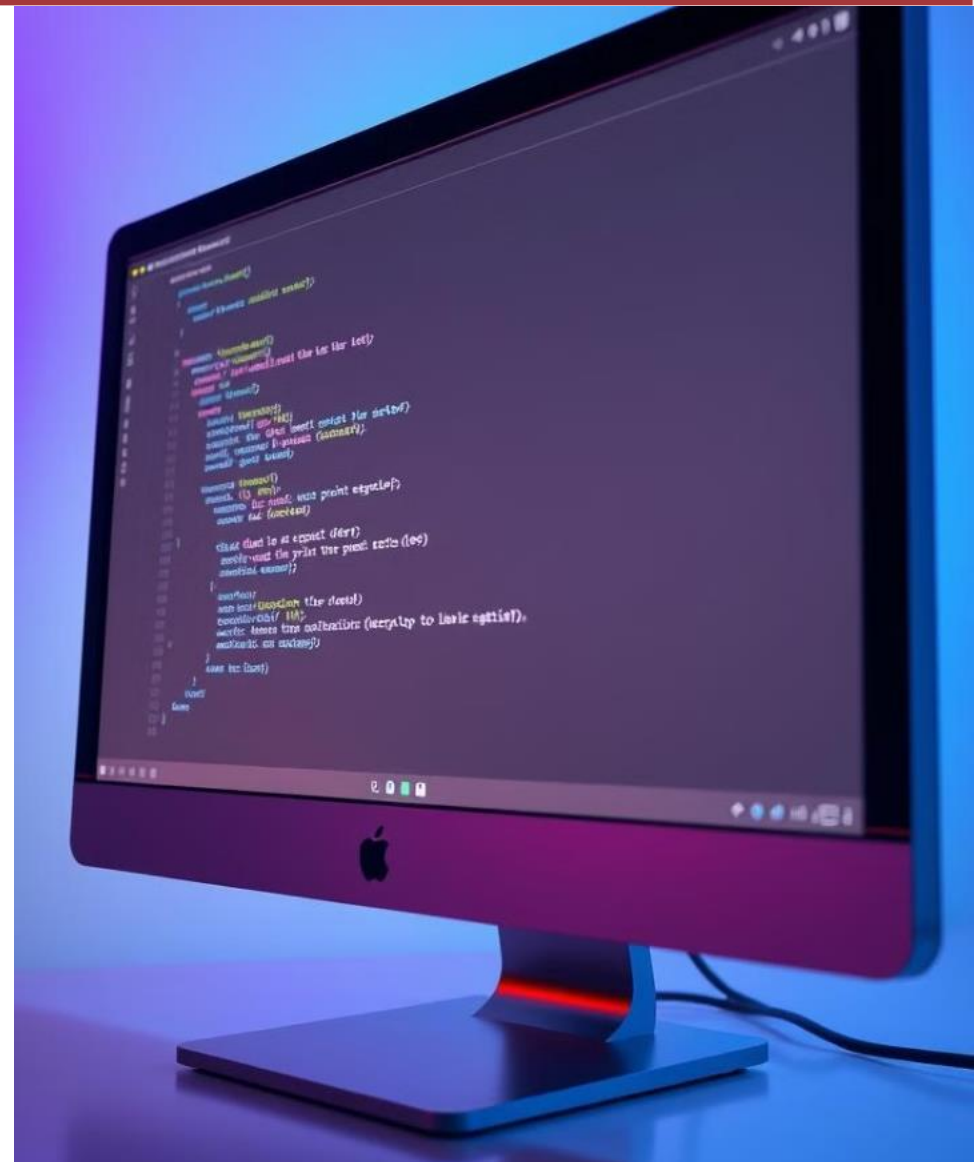
## Output

Chức năng in ghi văn bản vào đầu ra tiêu chuẩn (thường là bảng điều khiển).

3

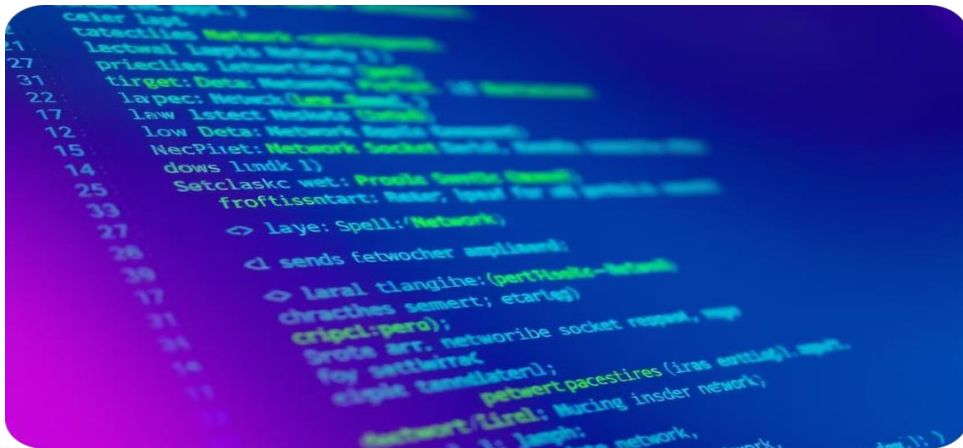
## File Handling

Perl cung cấp các hàm như `open()`, `read()` và `write()` để tương tác với các tệp và đọc/ghi dữ liệu.



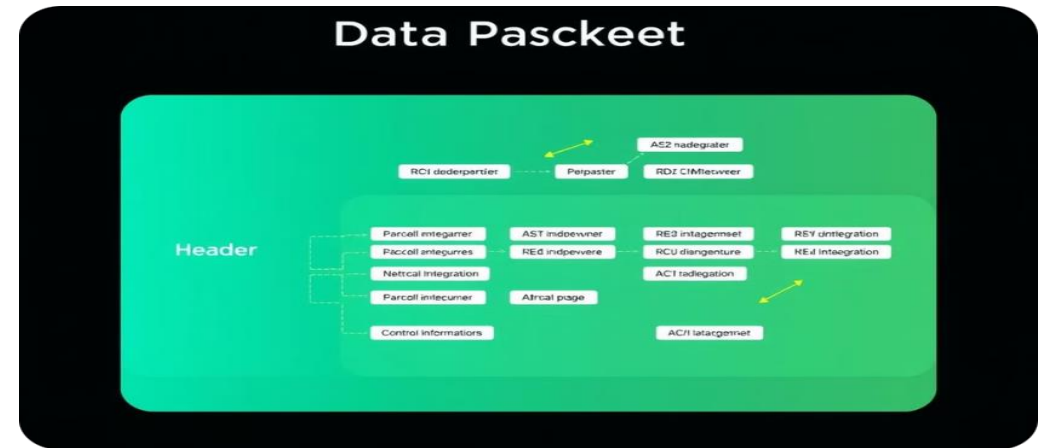
# Scalars for Network

Vô hướng đóng một vai trò quan trọng trong lập trình mạng trong Perl. Chúng được sử dụng để đại diện cho địa chỉ mạng, số cổng và dữ liệu được truyền đi.



# Socket Programming

Perl cung cấp các mô-đun như Socket và IO::Socket để tạo socket và tương tác với các dịch vụ mạng.



# Data Representation

Dữ liệu mạng thường được biểu diễn dưới dạng một chuỗi byte, có thể được lưu trữ và thao tác dưới dạng vô hướng trong Perl.





**Perl**  
Programming languages

# Chuỗi và Biểu thức chính quy trong Perl

Perl là một ngôn ngữ lập trình mạnh mẽ được biết đến với khả năng xử lý văn bản và dữ liệu. Chuỗi và Biểu thức chính quy đóng vai trò quan trọng trong Perl, cho phép chúng ta thao tác và phân tích văn bản một cách hiệu quả.



# Cú pháp và cách sử dụng Chuỗi trong Perl

Trong Perl, chuỗi được bao quanh bởi dấu nháy đơn (') hoặc dấu nháy kép ("). Dấu nháy kép cho phép nội suy biến, trong khi dấu nháy đơn không.

### Khai báo chuỗi

```
$string1 = 'Hello world!';  
$string2 = "This is a string  
with interpolation:  
$variable";
```

### Kết hợp chuỗi

```
$combined_string =  
$string1 . " " . $string2;
```

### Truy cập ký tự

```
$character =  
substr($string, 0, 1); # Lấy  
ký tự đầu tiên
```



# Các phép toán và hàm xử lý Chuỗi

Perl cung cấp nhiều toán tử và hàm để thao tác với chuỗi.

### 1 Kết hợp chuỗi

Toán tử "." dùng để nối chuỗi.

### 2 So sánh chuỗi

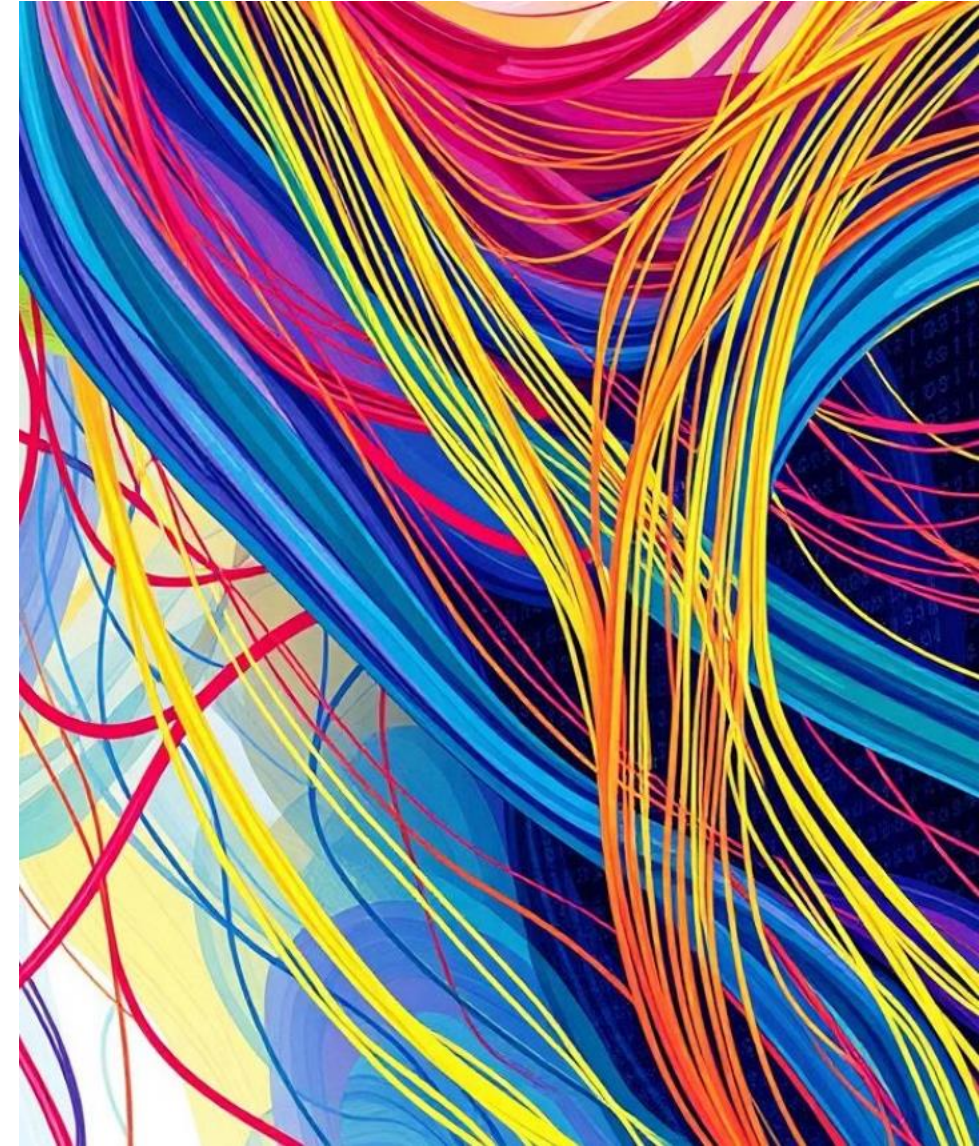
Toán tử "eq", "ne", "lt", "gt", "le", "ge" để so sánh chuỗi.

### 3 Hàm xử lý chuỗi

Hàm như "length", "substr", "index", "reverse", "uc", "lc" để thao tác với chuỗi.

### 4 Thay thế chuỗi

Sử dụng hàm "tr" hoặc "s" để thay thế ký tự trong chuỗi.





# Khái niệm và cú pháp của Biểu thức chính quy

Biểu thức chính quy (Regular Expression) là một chuỗi ký tự đặc biệt cho phép mô tả một mẫu văn bản.

### Cú pháp cơ bản

Biểu thức chính quy được bao quanh bởi dấu gạch chéo nghiêng (/). Ví dụ: /abc/.

### Ký tự đặc biệt

Các ký tự như "." (bất kỳ ký tự nào), "\*" (0 hoặc nhiều lần), "+" (1 hoặc nhiều lần), "?" (0 hoặc 1 lần), "|" (hoặc).

### Nhóm

Sử dụng dấu ngoặc đơn "(" và ")" để nhóm các ký tự.

### Lớp ký tự

Sử dụng dấu ngoặc vuông "[" và "]" để xác định một tập hợp các ký tự.





## Các toán tử và ký hiệu thường dùng trong Biểu thức chính quy

```
repeat: castcr the (ihspretines2015.fff);
ff-); to =(pc);
ratet(ouritecr the (fl]5)rt:5.t-4);
separt: ff);]
intt(ert 6p: 10; "fy),/o))";
pepect(ouritecr the (fl]3ratls(45=13; t..7);
{
rect: bitecc lize 0);
> ablitef (f
>
pact: b eplize (ite s1),iff);
tseplize "Suyet od=tl);
ett / tpwife (lfet
st: c sp lize "11
recet: roscprlize (f ystins. 5; ternk);
pest: roscplize 3);
recet: roscplize (itestring: 13))
penect: catsternt. (ffs if));
revect: bitetwite[];
```

Biểu thức chính quy sử dụng nhiều toán tử và ký hiệu để tạo ra các mẫu phức tạp.

Toán tử	Mô tả	Ví dụ
.	Bất kỳ ký tự nào	/a.c/
*	0 hoặc nhiều lần	/a*b/
+	1 hoặc nhiều lần	/a+b/
?	0 hoặc 1 lần	/a?b/
	Hoặc	/a b/
^	Bắt đầu chuỗi	/^abc/
\$	Kết thúc chuỗi	/abc\$/



### Các toán tử và ký hiệu thường dùng trong Biểu thức chính quy

```
repeat: castcr the (ihspretines2015.fff);  
ff-); to =(pc);  
ratet(ouwritecr the (fl]5)rt:5.t-4);  
separt: ff);]  
intt(ert 6p: 10; "fy),/o))";  
pepect(ouwritecr the (fl]3ratls(45=13; t..7);  
{  
rect: bitecc lize 0);  
> ablitef (f  
>  
pact: b seplize (ite s1),iffi);  
tseplize "Suyen od=tl);  
ett y tpwife (lfet  
st: c seplize "11  
recet: roscprlize (f ystins. 5; ternk);  
pest: roscplize 3);  
recet: roscplize (itestring: 13))  
penect: catsternt. (ffs if));  
revect: bitetwite[];
```

- `\d` : Khớp với một chữ số (0-9).
- `\w` : Khớp với một ký tự chữ cái hoặc chữ số (chữ cái, chữ số, và dấu gạch dưới).
- `\s` : Khớp với một ký tự khoảng trắng (bao gồm khoảng trắng, tab, hoặc xuống dòng).
- `[abc]` : Khớp với một trong các ký tự a, b, hoặc c.
- `[^abc]` : Khớp với bất kỳ ký tự nào không phải là a, b, hoặc c.
- `(abc)` : Nhóm các ký tự lại thành một mẫu để khớp.
- `{n,m}` : Khớp với số lượng ký tự trong khoảng từ n đến m.



# Ứng dụng Biểu thức chính quy trong việc xử lý và tìm kiếm Chuỗi

Biểu thức chính quy rất hữu ích trong việc tìm kiếm, thay thế và thao tác với chuỗi.

1

### Tìm kiếm

Sử dụng hàm "match" hoặc "grep" để tìm kiếm chuỗi phù hợp với mẫu.

2

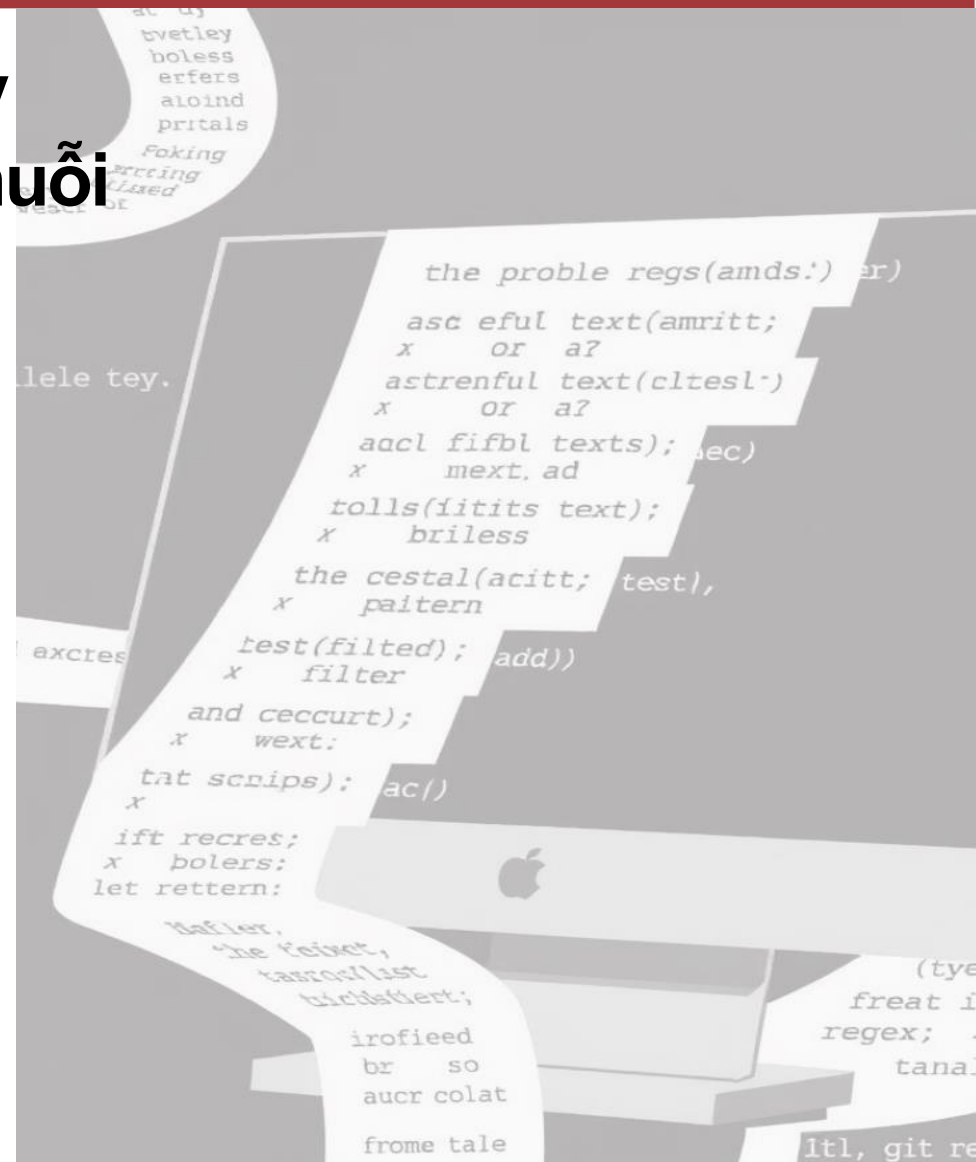
### Thay thế

Sử dụng hàm "s" để thay thế các chuỗi phù hợp với mẫu bằng chuỗi khác.

3

### Phân tích

Sử dụng hàm "split" để chia chuỗi thành các phần dựa trên mẫu.





# Một số ví dụ và tình huống sử dụng Chuỗi và Biểu thức chính quy

Biểu thức chính quy được sử dụng rộng rãi trong nhiều tình huống như kiểm tra dữ liệu, trích xuất thông tin, phân tích văn bản, v.v.



### Kiểm tra địa chỉ email

Sử dụng biểu thức chính quy để kiểm tra xem địa chỉ email có hợp lệ hay không.



### Kiểm tra số điện thoại

Sử dụng biểu thức chính quy để kiểm tra xem số điện thoại có hợp lệ hay không.



### Phân tích URL

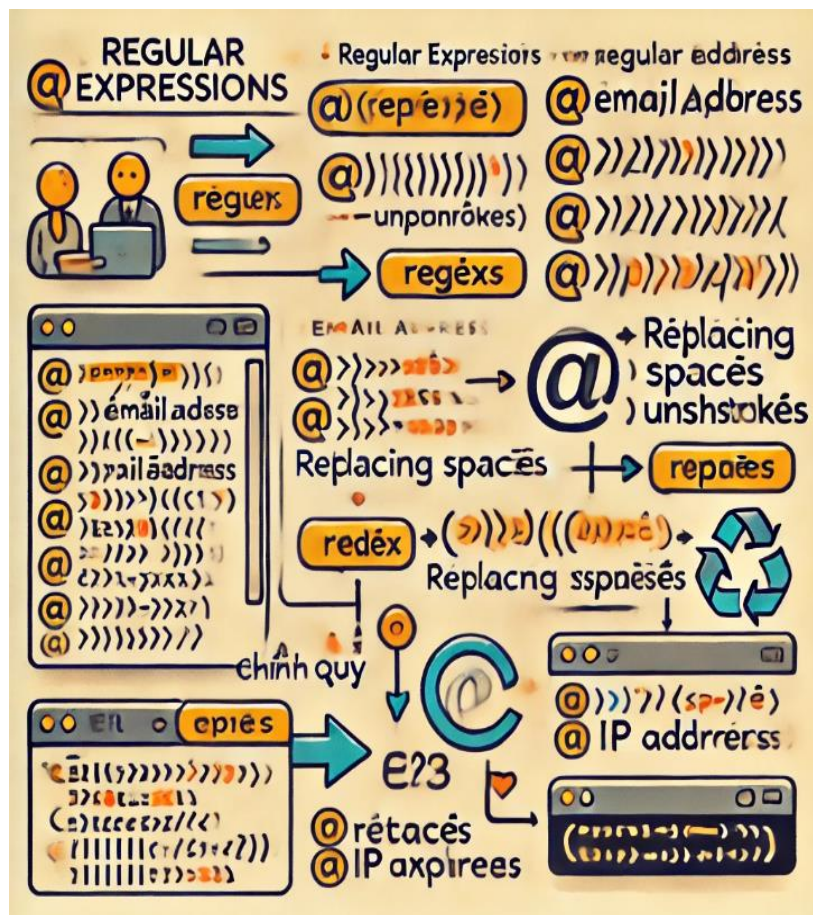
Sử dụng biểu thức chính quy để trích xuất các thành phần của URL, như tên miền, đường dẫn, v.v.



### Tìm kiếm và thay thế

Sử dụng biểu thức chính quy để tìm kiếm và thay thế các chuỗi văn bản.

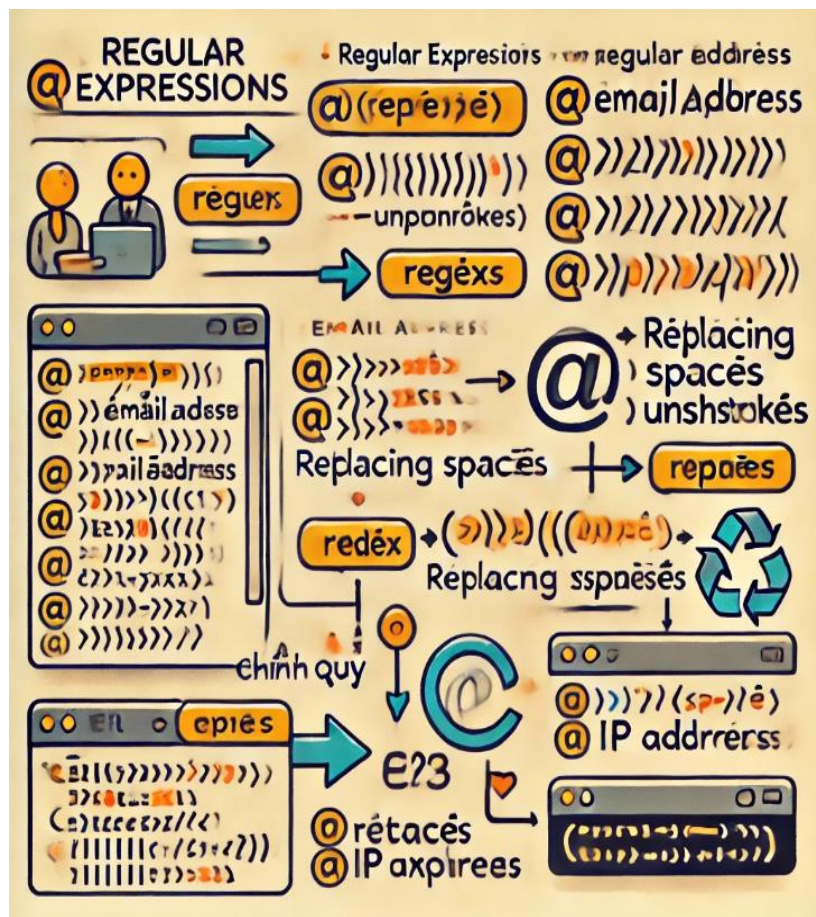




## EXAMPLE

### So khớp (Match): `=~`

- Cú pháp: `if ($string =~ /pattern/)`
- Dùng để kiểm tra xem chuỗi `$string` có khớp với mẫu pattern hay không.
- Nếu khớp, biểu thức trả về true.

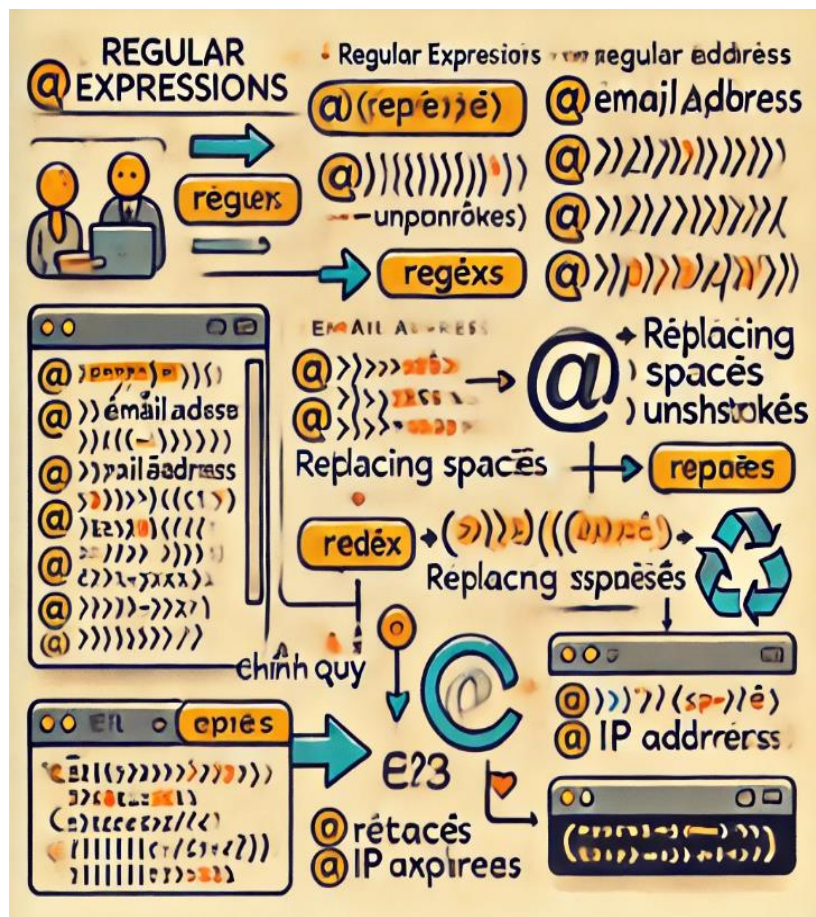


# EXAMPLE

## Không khớp (Not Match): !~

- Cú pháp: `if ($string !~ /pattern/)`
- Kiểm tra chuỗi \$string không khớp với mẫu pattern.

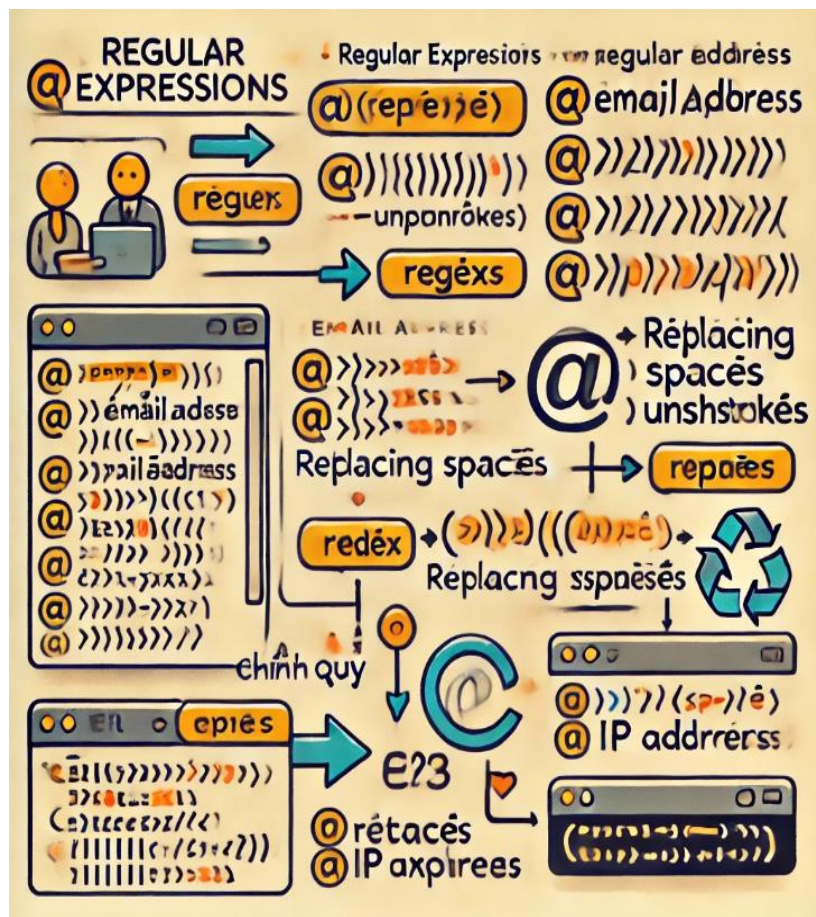




# EXAMPLE

## Thay thế (Substitution): s///

- Cú pháp: `$string =~ s/pattern/replacement/`
- Thay thế tất cả các phần khớp với pattern trong `$string` bằng chuỗi `replacement`.



# EXAMPLE

## Dịch vụ tìm kiếm toàn bộ (Global Match): g

- Cú pháp: `$string =~ s/pattern/replacement/g`
- Thay thế tất cả các phần khớp trong chuỗi, không chỉ phần đầu tiên.



## EXAMPLE

### Kiểm tra định dạng địa chỉ email

```
my $email = "example@gmail.com";  
if ($email =~ /^[\\w\\.]+\\@[\\w\\.]+\\. [a-zA-Z]{2,6}$/) {  
    print "Email hợp lệ\\n";  
} else {  
    print "Email không hợp lệ\\n";  
}
```

- `^[\\w\\.]+`: Bắt đầu với một hoặc nhiều ký tự chữ cái, số, dấu chấm hoặc gạch nối.
- `\\@`: Dấu @.
- `[\\w\\.]+`: Một hoặc nhiều ký tự chữ cái, số, dấu chấm hoặc gạch nối.
- `\\.`: Dấu chấm.
- `[a-zA-Z]{2,6}$`: Kết thúc bằng phần tên miền có 2 đến 6 ký tự chữ cái.





## EXAMPLE

Thay thế tất cả các khoảng trắng bằng dấu gạch dưới

```
my $text = "Xin chào thế giới";  
$text =~ s/\s+/_/g;  
print $text; # Output: Xin_chào_thế_giới
```

- \s+: Khớp với một hoặc nhiều ký tự khoảng trắng.
- g: Thay thế tất cả các khoảng trắng bằng dấu gạch dưới.



## EXAMPLE

### Tách chuỗi thành các phần tử bằng dấu phẩy

```
my $line = "Device1, Router, 192.168.0.1, Active";  
my @fields = split /\s*/, $line;  
print join("\n", @fields);
```

- `\s*`: Khớp với dấu phẩy và một hoặc nhiều khoảng trắng sau dấu phẩy.
- `split`: Chia chuỗi thành các phần tử dựa trên biểu thức chính quy.



## EXAMPLE

### Kiểm tra định dạng địa chỉ IP

```
my $ip = "192.168.1.1";  
if ($ip =~ /^(\d{1,3}\.){3}\d{1,3}$/) {  
    print "Địa chỉ IP hợp lệ\n";  
} else {  
    print "Địa chỉ IP không hợp lệ\n";  
}
```

- `\d{1,3}`: Khớp với một số từ 1 đến 3 chữ số.
- `(\d{1,3}\.){3}`: Khớp với 3 nhóm số, mỗi nhóm kết thúc bằng dấu chấm.
- `\d{1,3}$`: Khớp với nhóm số cuối cùng.



## EXAMPLE

### Kiểm tra chuỗi bắt đầu và kết thúc bằng các ký tự cụ thể

```
my $string = "Hello world!";  
if ($string =~ /^Hello.*!$/ ) {  
    print "Chuỗi bắt đầu với 'Hello' và kết thúc với '!\n';  
} else {  
    print "Chuỗi không khớp\n";  
}
```

- ^Hello: Khớp với chuỗi bắt đầu bằng "Hello".
- .\*: Khớp với bất kỳ ký tự nào hoặc không có ký tự nào giữa "Hello" và dấu !.
- !\$: Khớp với dấu chấm than ở cuối chuỗi.



## EXAMPLE

### Đếm số lần xuất hiện của một từ trong chuỗi

```
my $text = "Perl is great. Perl is fun. I love Perl.";
my $count = () = $text =~ /Perl/g;
print "Từ 'Perl' xuất hiện $count lần\n";
```

- /Perl/g: Tìm tất cả các lần xuất hiện của từ "Perl".
- () =: Đây là mẹo để đếm số lần khớp của một biểu thức chính quy.



# File Handling in Perl

Trong Perl, làm việc với các tệp liên quan đến các tác vụ khác nhau như đọc từ tệp, ghi vào tệp và nối dữ liệu vào tệp. Perl cung cấp một cách đơn giản và mạnh mẽ để xử lý các tệp bằng các chức năng tích hợp. Dưới đây là bảng phân tích các thao tác tệp phổ biến nhất trong Perl





## Opening a File

Bạn có thể mở một tệp để đọc, viết hoặc nối thêm bằng chức năng "mở". Cú pháp chung là:

```
open(FILEHANDLE, "mode", "filename") or die "Error message";
```

"FILEHANDLE": Một tham chiếu đến các tệp tin, được sử dụng cho các hoạt động của tệp tin.

- "mode": Chỉ định cách mở tệp. Các chế độ phổ biến là:
  - ""r"" or ""<"": Mở để đọc (mặc định).
  - ""w"" or "">"": Mở để viết (tạo một tệp mới hoặc cắt bớt tệp hiện có).
  - ""a"" or "">>"": Open for appending (adds content to the end of the file).
- ""filename"": Tên của tệp sẽ được mở.
- "die "Error message"": Thoát khỏi chương trình và in thông báo lỗi nếu không thể mở tệp.



# Reading from Files

Đọc từ các tệp trong Perl có thể đạt được bằng cách sử dụng các kỹ thuật khác nhau. Toán tử '<' có thể đọc toàn bộ tệp thành một biến, trong khi hàm 'read' cho phép bạn đọc một số byte cụ thể.

### Line-by-Line

Sử dụng một vòng lặp để đọc từng dòng của tệp, xử lý nó khi cần thiết.

### Entire File

Đọc toàn bộ tệp thành một biến bằng toán tử '<'.

### Specific Bytes

Chức năng 'read' cho phép bạn đọc một số byte xác định.





# Writing to Files

Ghi vào các tệp trong Perl được thực hiện bằng chức năng 'in' hoặc chức năng 'ghi'. Các chức năng này lấy xử lý tệp và dữ liệu sẽ được ghi.

### 1 Print Function

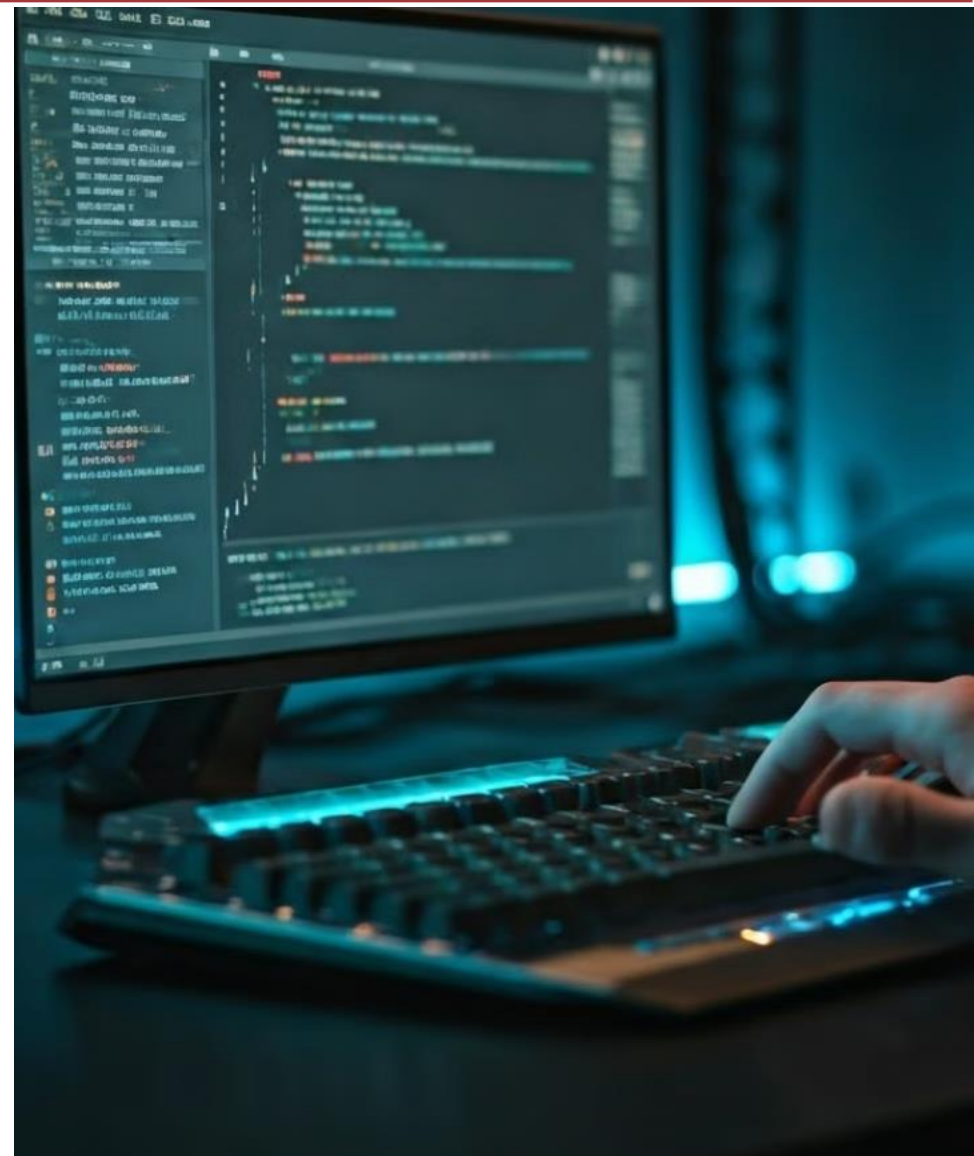
Chức năng 'print' gửi dữ liệu đến tay cầm tệp.

### 2 Write Function

Hàm 'write' ghi một số byte cụ thể vào tệp.

### 3 Append Mode

Sử dụng toán tử '>>' sẽ nối thêm dữ liệu mới vào cuối tệp.





### File Modes and Permissions

Chế độ tệp xác định cách một tệp có thể được truy cập và sửa đổi. Họ kiểm soát các quyền đọc, ghi và thực thi cho chủ sở hữu, nhóm và những người khác.

Mode	Description
r	Read-only
w	Write-only, create if not exists
a	Append, create if not exists
r+	Read and write
w+	Read and write, create if not exists
a+	Read and append, create if not exists





Lỗi tệp có thể xảy ra trong quá trình hoạt động của tệp. Perl cung cấp các cơ chế để xử lý các lỗi này một cách duyên dáng. Chức năng 'die' có thể chấm dứt tập lệnh, trong khi chức năng 'cảnh báo' cung cấp thông báo cảnh báo.

1

### Error Checking

Sử dụng câu lệnh 'if' để kiểm tra lỗi sau khi thao tác tệp.

2

### Error Handling

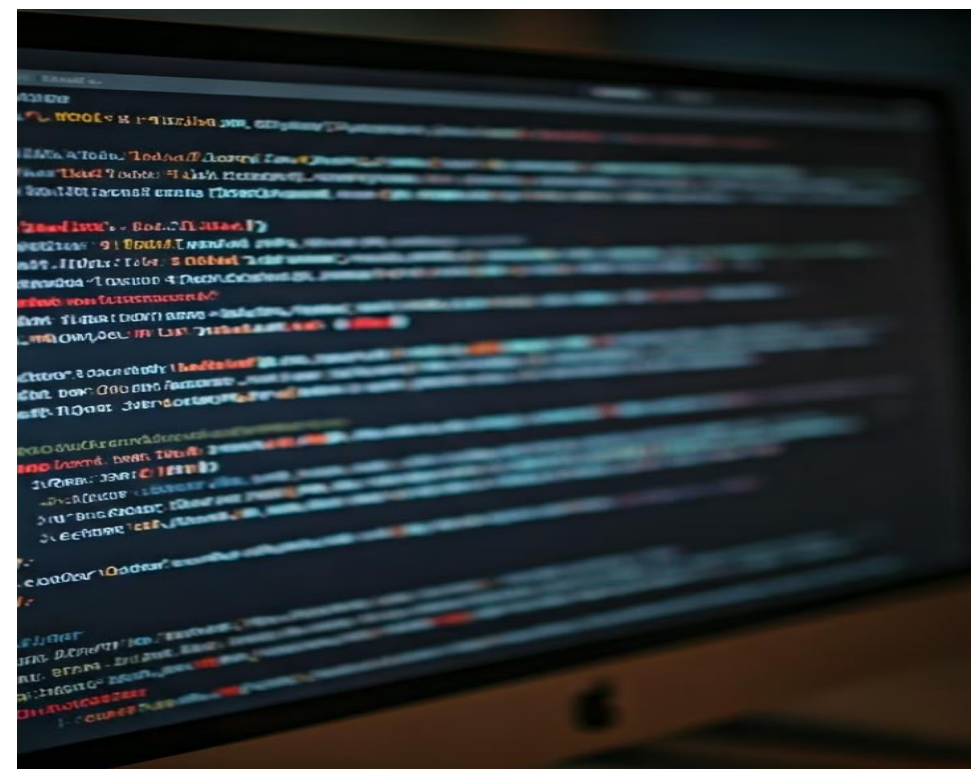
Sử dụng chức năng 'die' để chấm dứt tập lệnh khi có lỗi.

3

### Error Logging

Sử dụng chức năng 'cảnh báo' để cung cấp thông báo cảnh báo.

## Handling File Errors





# Seeking and Positioning in Files

Tìm kiếm và định vị trong tệp cho phép bạn di chuyển con trỏ tệp đến một vị trí cụ thể trong tệp. Chức năng "Seek" được sử dụng để đặt con trỏ đến vị trí mong muốn.

1

### File Pointer

Con trỏ tệp cho biết vị trí read/write hiện tại trong tệp.

2

### Seek Function

Chức năng "seek" di chuyển con trỏ tệp đến một vị trí được chỉ định.

3

### Tell Function

Hàm "tell" trả về vị trí hiện tại của con trỏ tệp.





### Deleting and Renaming Files

Xóa và đổi tên tệp trong Perl có thể được thực hiện bằng chức năng "hủy liên kết" để xóa và chức năng "đổi tên" để đổi tên.



Unlink

Chức năng "hủy liên kết" sẽ xóa một tệp khỏi hệ thống tệp.



Rename

Chức năng "đổi tên" thay đổi tên hoặc vị trí của tệp.



### Directories and Paths

Perl cung cấp các chức năng để quản lý các thư mục, bao gồm tạo, xóa và điều hướng cấu trúc thư mục. Chức năng "mkdir" tạo ra một thư mục mới, trong khi chức năng "rmdir" loại bỏ một thư mục trống.



#### Directory Structure

Các thư mục được tổ chức theo thứ bậc, tạo thành một cấu trúc giống như cây.



#### Path Manipulation

Perl cung cấp các chức năng để thao tác đường dẫn tệp, chẳng hạn như kết hợp đường dẫn và trích xuất các thành phần.



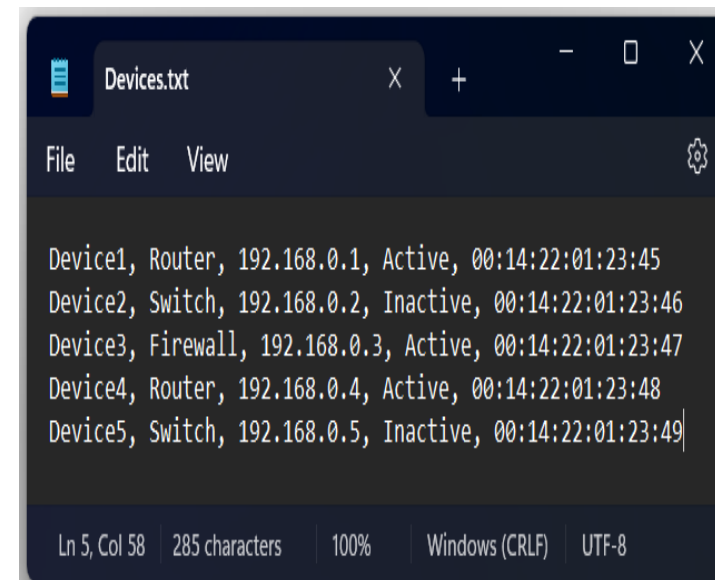
### EXAMPLE

- ❑ **Mô tả:** File mẫu (devices.txt ) chứa thông tin về các thiết bị mạng. Mỗi dòng có thể đại diện cho một thiết bị với các thuộc tính như tên, loại thiết bị, địa chỉ IP, trạng thái, và địa chỉ MAC, được phân cách bằng dấu phẩy.

Yêu cầu hãy viết Script:

1. Đọc file trong Perl
2. Trích xuất thông tin cụ thể (ví dụ: Tên thiết bị, IP, Trạng thái)
3. Lọc thiết bị theo trạng thái (ví dụ: Các thiết bị đang hoạt động - Active)
4. Đếm số lượng các loại thiết bị (ví dụ: Router, Switch)
5. Trích xuất thiết bị có IP thuộc dải nhất định (ví dụ: 192.168.0.x)
6. Ghi dữ liệu đã lọc ra file mới (ví dụ: Chỉ các thiết bị đang hoạt động)

#### File mẫu (devices.txt)



```
Devices.txt
File Edit View
Device1, Router, 192.168.0.1, Active, 00:14:22:01:23:45
Device2, Switch, 192.168.0.2, Inactive, 00:14:22:01:23:46
Device3, Firewall, 192.168.0.3, Active, 00:14:22:01:23:47
Device4, Router, 192.168.0.4, Active, 00:14:22:01:23:48
Device5, Switch, 192.168.0.5, Inactive, 00:14:22:01:23:49
Ln 5, Col 58 285 characters 100% Windows (CRLF) UTF-8
```





### 1. Đọc file

```
open(my $fh, "<", "devices.txt") or die "Không thể mở file: $!";  
my @devices = <$fh>;  
close($fh);
```

Lệnh này sẽ đọc toàn bộ các dòng từ file vào mảng @devices.



### 2. Trích xuất thông tin cụ thể

```
foreach my $device (@devices) {  
    chomp($device); # Loại bỏ ký tự xuống dòng  
    my ($name, $type, $ip, $status, $mac) = split /, /, $device;  
    print "Thiết bị: $name, IP: $ip, Trạng thái: $status\n";  
}
```

Lệnh này sẽ in ra tên thiết bị, địa chỉ IP, và trạng thái của từng thiết bị.



### 3. Lọc thiết bị theo trạng thái

(ví dụ: Các thiết bị đang hoạt động - Active)

```
foreach my $device (@devices) {  
    chomp($device);  
    my ($name, $type, $ip, $status, $mac) = split /, /, $device;  
    if ($status eq "Active") {  
        print "Thiết bị đang hoạt động: $name, IP: $ip\n";  
    }  
}
```

Lệnh này sẽ chỉ hiển thị các thiết bị đang hoạt động.



### 4. Đếm số lượng các loại thiết bị

(ví dụ: Router, Switch)

```
my %device_count;
foreach my $device (@devices) {
    chomp($device);
    my ($name, $type, $ip, $status, $mac) = split /, /, $device;
    $device_count{$type}++;
}
# In ra số lượng của từng loại thiết bị
foreach my $type (keys %device_count) {
    print "Số lượng $type: $device_count{$type}\n";
}
```



### 5. Trích xuất thiết bị có IP thuộc dải nhất định

(ví dụ: 192.168.0.x)

```
foreach my $device (@devices) {  
    chomp($device);  
    my ($name, $type, $ip, $status, $mac) = split /, /, $device;  
    if ($ip =~ /^192\.168\.0\../) {  
        print "Thiết bị thuộc dải 192.168.0.x: $name, IP: $ip\n";  
    }  
}
```

Để trích xuất các thiết bị thuộc dải địa chỉ IP cụ thể, sử dụng biểu thức chính quy (regular expressions)





### 6. Ghi dữ liệu đã lọc ra file mới

(ví dụ: Chỉ các thiết bị đang hoạt động)

```
open(my $out, ">", "active_devices.txt") or die "Không thể mở  
file: $!";  
foreach my $device (@devices) {  
    chomp($device);  
    my ($name, $type, $ip, $status, $mac) = split /, /, $device;  
    if ($status eq "Active") {  
        print $out "$device\n"; # Ghi thiết bị đang hoạt động vào file  
    }  
}  
close($out);
```



# Q&A

## Data Structures in Perl



# Conclusion

1. Tính linh hoạt của các loại dữ liệu trong Perl
2. So sánh Array và Hash trong Perl
3. Quản lý bộ nhớ với cấu trúc dữ liệu trong Perl
4. Cấu trúc dữ liệu tùy chỉnh trong Perl
5. Hiệu suất của cấu trúc dữ liệu trong Perl

... ftte attel  
atteel werituarrenn.

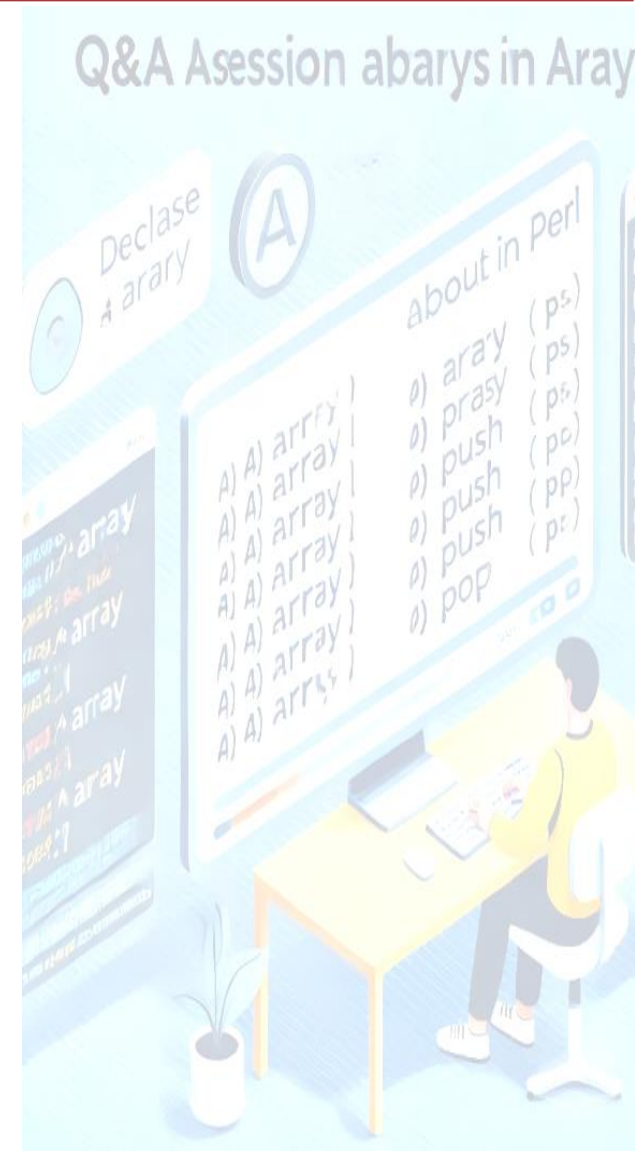




## Conclusion

### Tính linh hoạt của các loại dữ liệu trong Perl

**Question 1:** "Perl hỗ trợ việc sử dụng và chuyển đổi giữa các loại dữ liệu khác nhau (vô hướng, mảng, băm) như thế nào? Những tính năng nào của Perl giúp quản lý và thao tác dữ liệu một cách linh hoạt?"



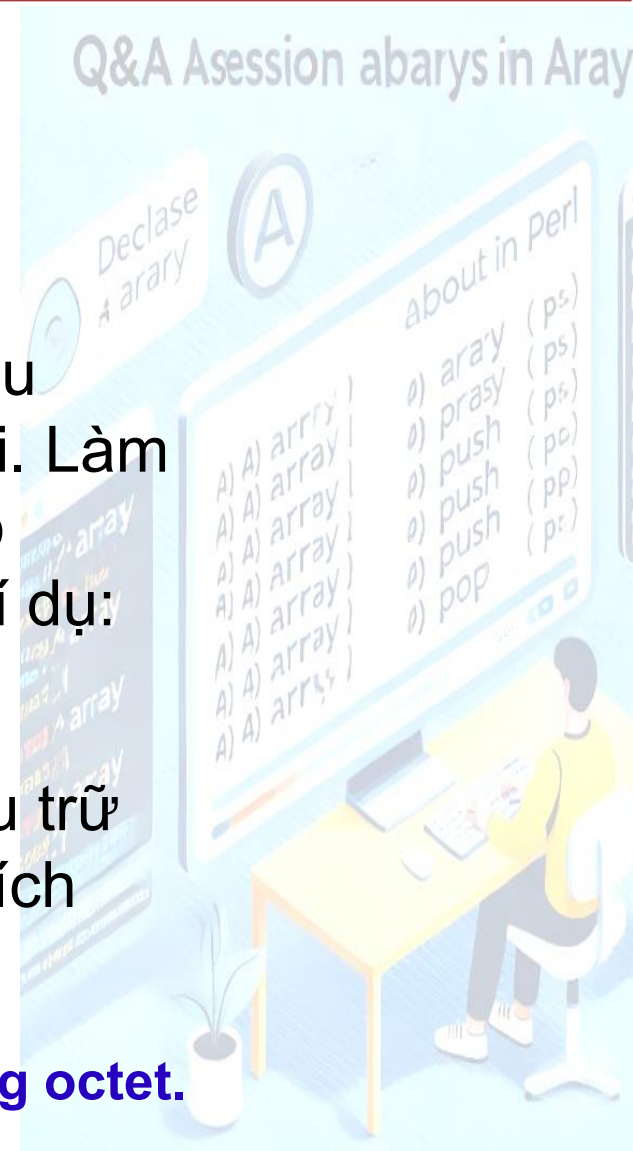
# Conclusion

## 11. Using Scalars to Store IP Addresses

**Discussion:** Địa chỉ IP của các thiết bị mạng là những mẫu thông tin quan trọng thường được lưu trữ dưới dạng chuỗi. Làm thế nào bạn có thể lưu trữ một địa chỉ IP trong một biến vô hướng và phân tích cú pháp nó để trích xuất từng phần (ví dụ: 192.168.1.1 thành 192, 168, 1 và 1)?

**Solution:** Trong Perl, bạn có thể sử dụng vô hướng để lưu trữ địa chỉ IP dưới dạng chuỗi và sử dụng hàm split để phân tích cú pháp địa chỉ IP.

**Bài tập: Hãy thử phân tích cú pháp bất kỳ địa chỉ IP nào và in từng octet.**





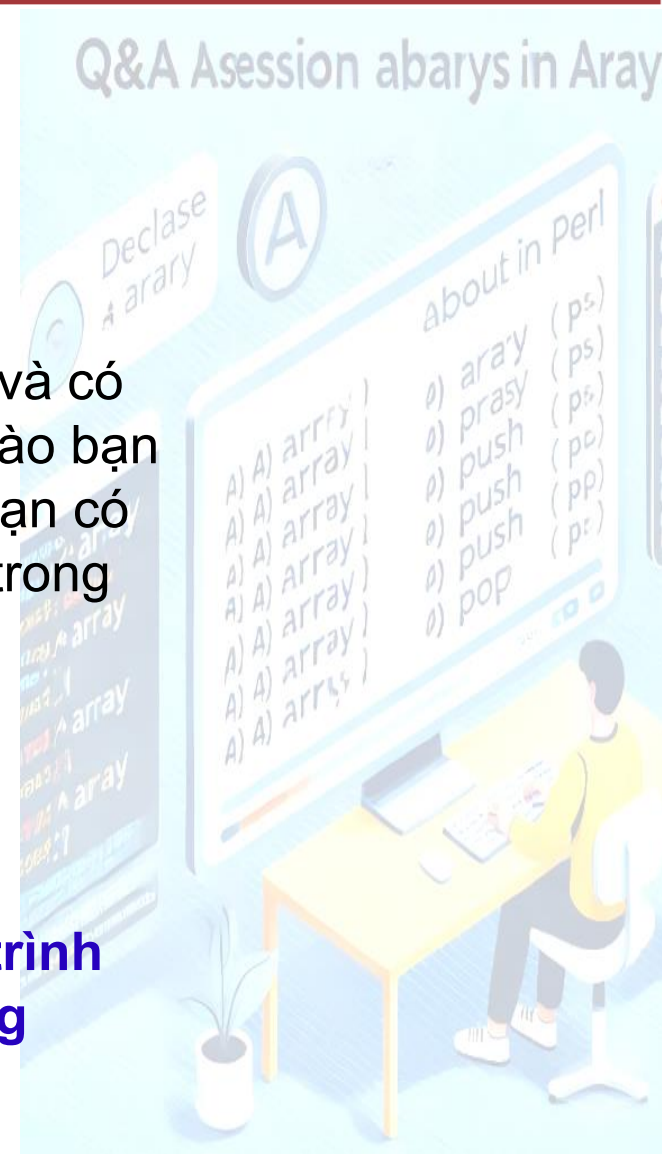
# Conclusion

## 12. Sử dụng mảng để quản lý bảng định tuyến

**Discussion:** Một bảng định tuyến thường chứa nhiều tuyến đường và có thể được quản lý hiệu quả bằng cách sử dụng các mảng. Làm thế nào bạn có thể sử dụng một mảng để lưu trữ các tuyến này và làm thế nào bạn có thể kiểm tra xem một địa chỉ IP cụ thể có thuộc về bất kỳ tuyến nào trong bảng định tuyến hay không?

**Solution:** Một mảng có thể chứa một danh sách các địa chỉ IP hoặc mạng con. Bạn có thể lặp lại thông qua mảng để kiểm tra từng tuyến đường.

**Exercise: Tạo một mảng các tuyến đường và viết một chương trình để tìm xem một địa chỉ IP cụ thể có thuộc về bất kỳ tuyến đường nào hay không.**



# Conclusion

### 13. Quản lý thông tin mạng phức tạp với hàm băm

**Discussion: Trong mạng, một thiết bị thường có nhiều thuộc tính như tên máy chủ, địa chỉ IP và địa chỉ MAC. Làm thế nào bạn có thể sử dụng một hàm băm để lưu trữ thông tin này và truy xuất nó một cách nhanh chóng?**

**Solution:** Một hàm băm có thể ánh xạ các thuộc tính (khóa) với giá trị của chúng. Truy cập dữ liệu nhanh chóng bằng cách sử dụng khóa.

**Exercise: Tạo hàm băm để lưu trữ thông tin thiết bị và truy xuất từng giá trị.**

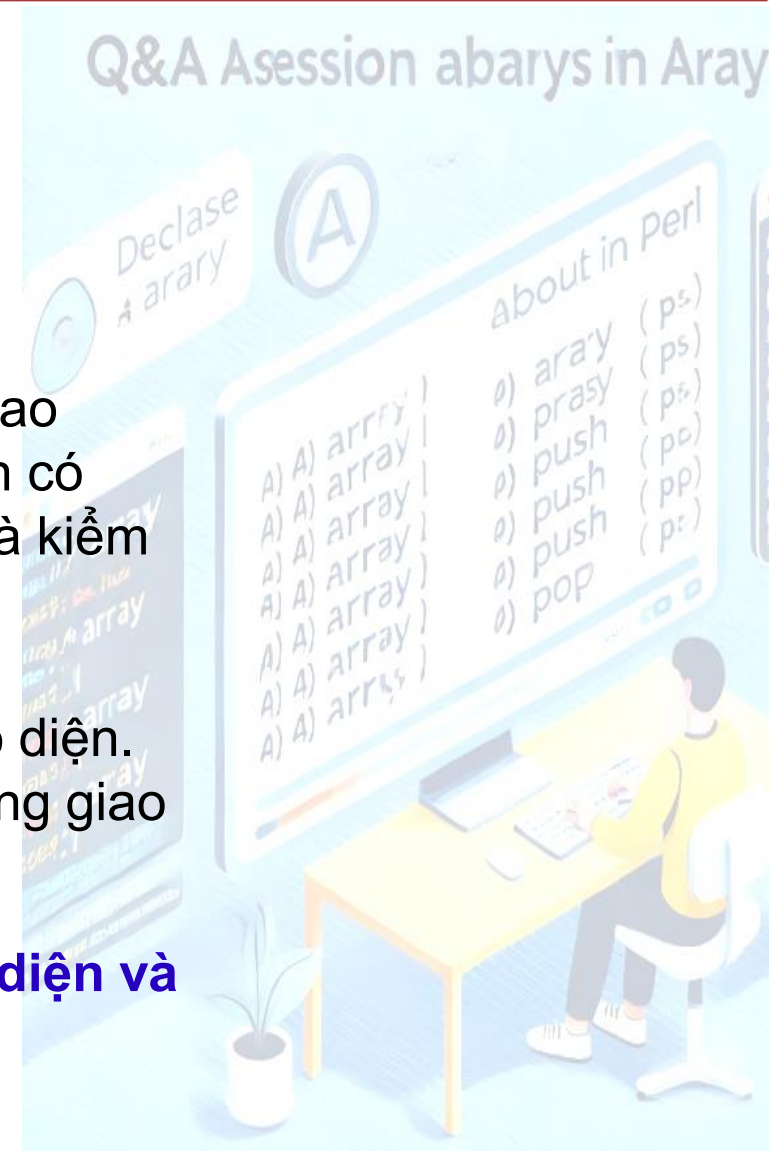
# Conclusion

## 14. Sử dụng mạng để giám sát nhiều giao diện mạng

**Discussion:** Trong một thiết bị mạng có nhiều giao diện, mỗi giao diện có thể có trạng thái hoạt động khác nhau. Làm thế nào bạn có thể sử dụng mangle để lưu trữ trạng thái của các giao diện này và kiểm tra giao diện nào lên hoặc xuống?

**Solution:** Sử dụng một mảng để lưu trữ trạng thái của các giao diện. Bạn có thể lặp lại thông qua mảng để kiểm tra trạng thái của từng giao diện.

**Exercise: Tạo một mảng để lưu trữ trạng thái của các giao diện và kiểm tra giao diện nào đang hoạt động.**



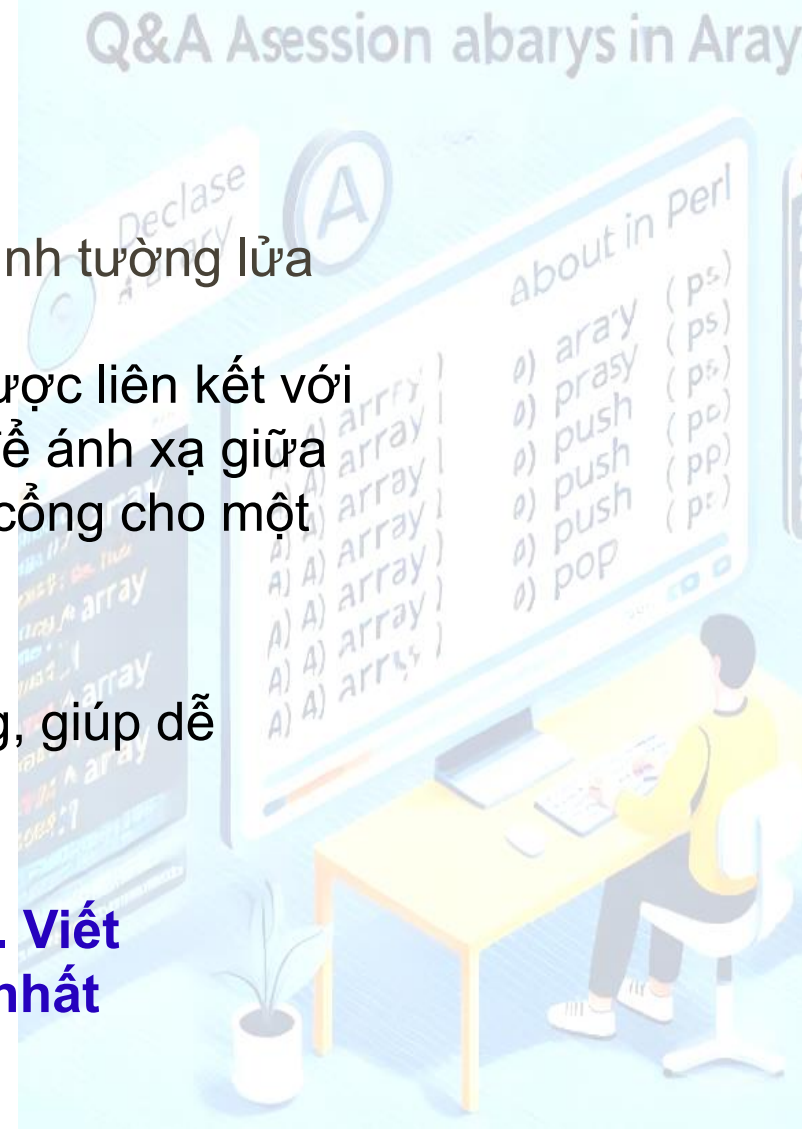
# Conclusion

## 15. Sử dụng hàm băm để ánh xạ dịch vụ đến các cổng trong cấu hình tường lửa

**Discussion:** Trong cấu hình tường lửa, mỗi dịch vụ thường được liên kết với một số cổng. Làm thế nào bạn có thể sử dụng một hàm băm để ánh xạ giữa tên dịch vụ và số cổng, và làm thế nào bạn có thể kiểm tra số cổng cho một dịch vụ cụ thể?

**Solution:** Một hàm băm có thể ánh xạ tên dịch vụ đến số cổng, giúp dễ dàng quản lý các dịch vụ khác nhau.

**Exercise: Tạo hàm băm để ánh xạ các dịch vụ tới số cổng. Viết một chương trình để kiểm tra số cổng cho bất kỳ dịch vụ nhất định nào.**







## References

- [1] Lê Văn Cường, *Python cơ bản và ứng dụng*, NXB Đại học Quốc gia Hà Nội, 2019
- [2] Randal L. Schwartz, Tom Phoenix & Brian D Foy (2008). *Learning Perl, 5<sup>th</sup> Edition*. O'Reilly Media.
- [3] Mark Lutz (2013). *Learning Python, 5<sup>th</sup> Edition*. O'Reilly Media.Stein,
- [4] Lincoln D. (2001). *Network Programming with Perl*. Addison-Wesley.
- [5] Rhodes, Brandon & Goerzen, John (2014). *Foundations of Python Network Programming, 3rd Edition*. U.S.: Apress.

## Website & Youtube

[6] [www.python.org](http://www.python.org)

[7] [www.perl.org](http://www.perl.org)

[8] <https://www.youtube.com/c/TheNetNinja/playlists>

[9] <https://www.youtube.com/watch?v=T11QYVfZoD0>

[10] [https://www.youtube.com/watch?v=xbT7Pvh\\_5LQ](https://www.youtube.com/watch?v=xbT7Pvh_5LQ)





# THANK YOU!

