Perl入門

ITのチカラで研究を支援

amelieff

アメリエフ株式会社

本講義にあたって

- テキストが穴埋めになっています
  - 埋めて完成させてください
- クイズがたくさんあります
  - めざせ全問正解！（賞品は特にありませんが…）
- 実習もたくさんあります
  - とにかく書いてみるのが理解の早道です
  - どうしても難しい場合は「pl_answer」ディレクトリに解答例がありますのでコピーして実行してください

本講義にあたって

- クイズの解答など、お手元の資料には入っていないページがあります

ページ右上に
 がついているスライドは
配布資料にはありません

本講義の内容

- プロローグ
- Perlとは
- 変数
- 配列
- 引数
- ハッシュ
- 条件付き処理
- 繰り返し処理
- ファイル入出力
- シバン
- 正規表現
- エピローグ

プロローグ

- あなたは解析担当者です
- シェルスクリプトを使いこなして毎日効率的に解析しています
- 突然、一本の電話がかかってきました

この間はどうも！今度は
ソフトCの結果をDというソフトに
かけてもらえるかな？

プロローグ

- あなたはソフトDについて調べました

（Cの出力フォーマットとDの入力フォーマットが違う！）
（これではCの結果をDに入力できない…）

（Dを実行するのは無理だ！）

バイオインフォのすばらしいソフトウェアがたくさん公開されています
入出力するファイルのフォーマットが共通化されてきてはいますが
ソフト独自仕様になっていて他との互換性がないことがよくあります

プロローグ

- その時です

諦めないで！

Perlを使えばフォーマットを
変換できるかもしれないよ！

Perlとは

- オープンソースのプログラミング言語の一つです
- 高速な処理には向きませんが、比較的手軽に書けることと、「 テキスト処理 」が得意なところから、バイオインフォマティクス業界でよく使われています

Perlのゆるさ

- Perlは同じ処理をいろいろな書き方で書ける言語です
  - 解答例は一例です
  - 「こう書くともっと良いのでは？」というスクリプトが書けた方は、積極的に教えてください
  - 様々な解を皆でシェアしましょう

例）クイズは弊社社員で手分けして考えましたのでいろいろな書き方が出てきます

シェルスクリプトとの比較

- Perlのほうが複雑な処理に向きます

	シェル スクリプト	Perl
コマンド連続実行	○	○
変数・条件付き処理・繰り返し処理	○	○
ファイル読み込み	○	○
正規表現・複雑な計算・複雑な処理	△	○

Perlを使うとよい場面

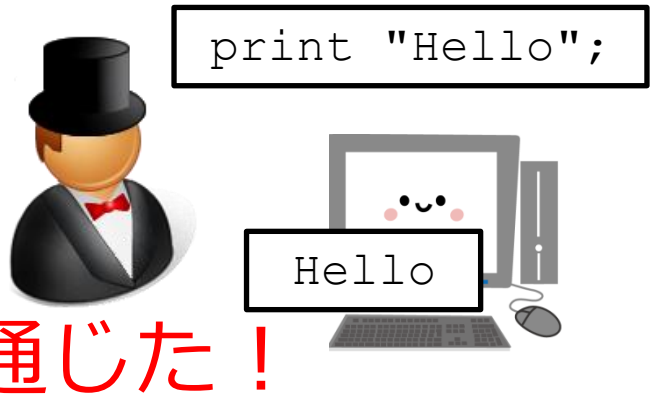
- ファイルフォーマットの変換
  - 例）「 BAMフォーマットをBEDフォーマットに変換 」
    （一般的なフォーマット間であれば大体変換スクリプトがありますが…）
- 結果ファイルの独自解析
  - 例）「 異なるソフトの出力結果をマージ 」

まずは日常会話から

- 「英語が苦手なのに、来月海外の学会に行くことになってしまった！」
  - ネイティブに負けないくらいの英語力を身に着けよう→無謀 ×
  - とりあえず学会参加に最低限必要な英語力を身に着けよう→現実的 ○

まずは日常会話から

- Perlも「言語」なので同じです
- 解析に必要な「日常会話」をとりあえず喋って（＝書いて）みましょう



まずは日常会話から

- Perlでは複雑なプログラムを書くこともできます…が
- 本講義ではバイオの解析を行うのに必要最低限な部分のみを紹介します
- こんな方を想定しています
  - とりあえずPerlの雰囲気を知りたい
  - 人が書いたPerlを読めるようになりたい

Perlスクリプトの作成と実行

1. テキストエディタ（vi, gedit等）で実行内容をファイルに書いて保存

シェルスクリプト資料の末尾

テキストエディタの使いかたは別紙をご覧ください

Perlスクリプトファイルは拡張子を「.pl」にします

2. perlコマンドで実行

```
$ perl Perlスクリプトファイル名
```

実習環境

- 仮想環境を起動します
- ホームディレクトリの下のamelieff/perlディレクトリに移動
- 実習はすべてここで行います

```
$ cd
$ cd amelieff/perl
```

実習環境

- テストデータ
  - mirbaseからダウンロードしたmiRNA配列
  - hairpin.fa（miRNA前駆体配列）
    - ftp://mirbase.org/pub/mirbase/CURRENT/hairpin.fa.zip
  - mature.fa（成熟miRNA配列）
    - ftp://mirbase.org/pub/mirbase/CURRENT/mature.fa.zip

```
$ cp ../sh/*.fa .
```

シェルスクリプトのデータを
ここにコピーします

達成目標

- 以下の作業をPerlスクリプトで実行できるようになりましょう
  - –hairpin.faの特定のRNAの配列を切り出して別のファイルに書き出す

Perlの記載方法

- 値を出力するにはprintを実行します
- 文字列はダブルクォートかシングルクォートで囲みます
- 行の末尾に;をつけます

```
print "Hello!";
```

- 全角記号や全角空白文字は使えません

実習１

- 次のPerlスクリプト・perl1.plを書いて実行してみましょう
  - Hello!と出力するPerlスクリプトです

```
$ gedit perl1.pl
```

perl1.shにこの1行を書いて保存します

```
print "Hello!";
```

```
$ perl perl1.pl
```

質問

- では、Bye!と出力するにはスクリプトをどう変更すればよいでしょう？

解答

- 実行内容を変えればいいですね

```
print "Hello!";
```

```
print "Bye!";
```

- ここで「変数」を使うとスマートです

変数

- シェルスクリプト同様、Perlでも「変数」を使うことができます
  - 「my $変数名=値;」と書くと、変数に値を代入できます
  - 「$変数」と書くと、変数に入っている値を呼び出すことができます

変数

- 「my」の話
  - 最初に変数が出てくるときにはmyをつけると覚えてください
  - myで定義した変数は、定義したスコープ内でのみ有効です
  - スコープについては後でご紹介します

実習2

- 次のPerlスクリプト・perl2.plを書いて実行してみましょう

```
$ cp perl1.pl perl2.pl
$ gedit perl2.pl
```

perl2.plを以下のように変更して保存します

```
my $message="Bye!";
print $message;
```

代入の=の前後に
空白が入ってもOKです

```
my $message = "Bye!";
```

```
$ perl perl2.pl
```

CentOS

```
[USER@SERVER ~]$ perl pl_answer/perl1.pl
Hello![USER@SERVER ~]$
```

不満

Ubuntu 一見改行されているように見えますが、末尾の%が出なくなるようにしましょう

```
admin1409@BioLinux1409[perl] perl pl_answer/perl1.pl                [ 5:24PM]
Hello!%
admin1409@BioLinux1409[perl]                                        [ 5:24PM]
```

Perlのprintはシェルスクリプトのechoとは違って最後が改行されないんだ！

改行したい場合は明示的に改行コードを書く必要があります

```
my $message="Bye!";
print $message, "¥n";
```

¥n：改行コード

¥は**バックスラッシュ**です
Linux上では右のように表示されます
テキスト中は¥で表記します

```
my $message="Bye!";
print $message, "\n";
```

クイズ

- スクリプトの実行結果はどうなりますか？

Q1a.pl

```
my $str = "Amelieff's blog";
print "$str¥n";
```

```
$ perl Q1a.pl
```

A

```
Amelieff's blog
```

C

```
Amelieff s blog
```

B

```
Amelieff
's blog
```

D

```
Amelieff
s blog
```

クイズ

- スクリプトの実行結果はどうなりますか？

```
$ perl Q1a.pl
```

正解は、**A**！！

Q1a.pl

```
my $str = "Amelieff's blog";
print "$str¥n";
```

難易度：★★

クイズ

- スクリプトの実行結果はどうなりますか？

Q1.pl

```
my $str = "Amelieff's blog";
print '$str¥n';
print "$str¥n";
```

```
$ perl Q1.pl
```

A

```
Amelieff's blogAmelieff's blog
```

B

```
$str¥n
Amelieff's blog
```

C

```
Amelieff's blog
Amelieff's blog
```

D

```
$str¥nAmelieff's blog
```

クイズ

- スクリプトの実行結果はどうなりますか？

Q1.pl

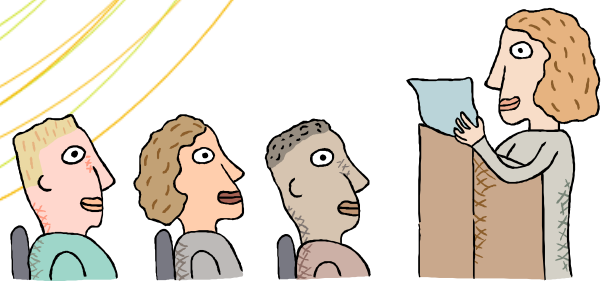
```
my $str = "Amelieff's blog";
print '$str¥n';
print "$str¥n";
```

```
$ perl Q1.pl
```

正解は、**D**！！

```
my $str = "Amelieff's blog";
print '$str¥n';      →シングルクォート内の変数は展開されない
print "$str¥n";      →ダブルクォート内の変数は展開される
```

値がたくさんある時



- あなたは小学校の先生です
- クラス40名のテストの平均点をPerlで計算してみようと思います

```
my $seito1 = 65;
my $seito2 = 90;
my $seito3 = 78;
   :
my $seito40 = 70;
```

入力するだけで大変！

値がたくさんある時

- 「 配列 」を使うとたくさんの値をまとめて扱うことができます

変数（単体）

配列（複数）

複数の値を
まとめて扱えるので便利

配列

- 配列は複数の値を1つの名前でまとめたものです

- 配列に値を入れるには

  my @配列名＝(値,値,...);

  のようにします

```
my @seito = (65,90,78, ..., 70);
```

配 列

- 配列から値を取り出すには
  $配列名[数字] と書きます

↑これを「 添字 」と呼びます

```
my @seito = (65,90,78, ..., 70);
print $seito[1], "¥n";
```

配列

- 添字は0から始まります
  - 先頭の値＝添字0
  - 2番目の値＝添字1

:

```
my @seito = (65,90,78, ..., 70);
print $seito[1], "¥n";
```

添字1＝2番目の値＝90

実習３

- 次のPerlスクリプト・perl3.plを書いて実行してみましょう

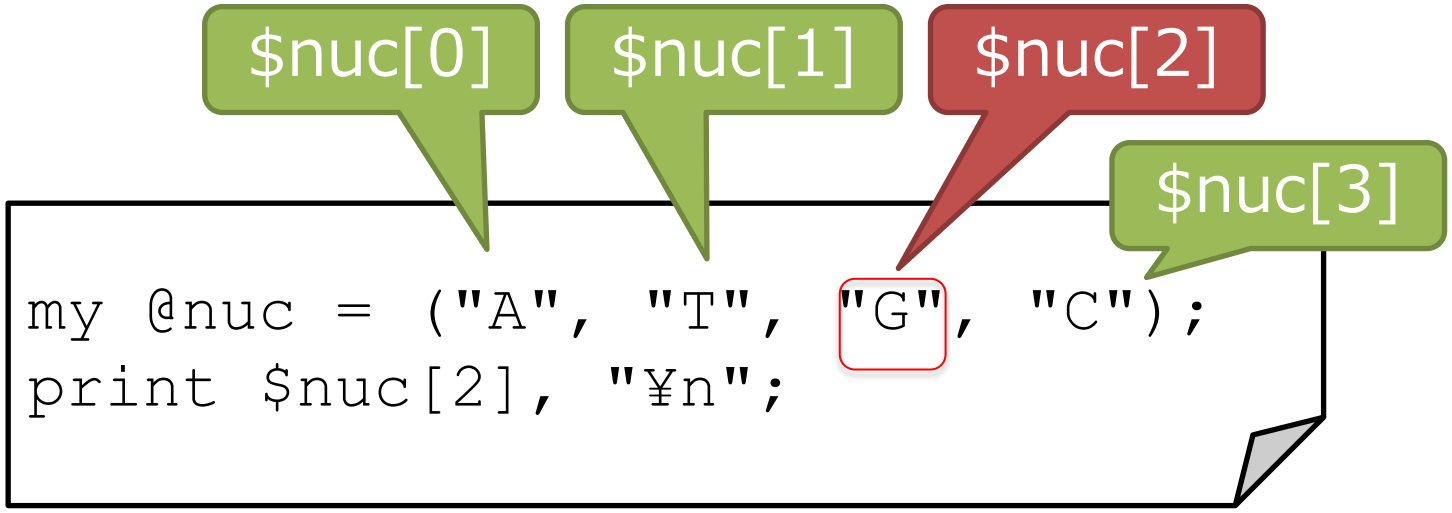
```
$ gedit perl3.pl
```

```
my @nuc = ("A", "T", "G", "C");
print $nuc[2], "¥n";
```

```
$ perl perl3.pl
```

実習３・解答

- 添字が2の値＝「G」が出力されます



クイズ

- スクリプトの実行結果はどうなりますか？

Q2a.pl

```
my @pig = ("boo", "foo", "woo");

my $str = $pig[2] . $pig[1] . $pig[0];
 # . は変数を連結して文字列にします

print $str, "¥n";
```

```
$ perl Q2a.pl
```

A boowoofoo

B woofooboo

C boofoowoo

D foowooboo

難易度：★

クイズ

- スクリプトの実行結果はどうなりますか？

Q2a.pl

```
my @pig = ("boo", "foo", "woo");

my $str = $pig[2] . $pig[1] . $pig[0];
 # . は変数を連結して文字列にします

print $str, "¥n";
```

```
$ perl Q2a.pl
```

正解は、**B**！！

B

```
woofooboo
```

クイズ

- スクリプトの実行結果はどうなりますか？

Q2.pl

難易度：★★

```
my @kana = qw/え め や も と ど さ が ま/;
  # qw/a b c/ は ('a', 'b', 'c')と同じです
my $ana;
$ana = $kana[4] . $kana[3] . $kana[5];
$ana = $kana[6] . $kana[7] . $kana[0];
$ana = $kana[8] . $kana[2] . $kana[5];

print "あ" . $ana . "り" . "¥n";
```

```
$ perl Q2.pl
```

A あさがえり

B あともどさがえまやどり

C あまやどり

D あともどり

クイズ

- スクリプトの実行結果はどうなりますか？

Q2.pl

```
my @kana = qw/え め や も と ど さ が ま/;
  # qw/a b c/ は ('a', 'b', 'c')と同じです
my $ana;
$ana = $kana[4] . $kana[3] . $kana[5];
$ana = $kana[6] . $kana[7] . $kana[0];
$ana = $kana[8] . $kana[2] . $kana[5];

print "あ" . $ana . "り" . "¥n";
```

```
$ perl Q2.pl
```

正解は、**C**！！

変数の値が上書きされるためです

配列に値を追加する

- 以下のような方法があります
  ① 配列[添字]=値
  ② push関数を使う

```
my @nuc = ("A", "T", "G", "C");
$nuc[4] = "N";
push @nuc, "a";
```

5番目に「N」が
6番目に「a」が
追加されます

配列の要素数を調べる

- 以下のような方法があります
  ① 配列を数値に変換する
  ② $#配列 + 1 を計算する

```
my $seito_su_1 = int(@seito);
my $seito_su_2 = $#seito + 1;
```

どちらも値は40になる

配列と文字列を相互に変換する

~~配列の要素数を調べる~~

- 配列を結合して文字列にする
  - join（"結合に使う文字", 配列）

```
my @array1  = ('Are', 'you', 'fine?');
my $string1 = join('-', @array1);
print $string1;
```

「Are-you-~~file~~fine?」と表示されます

- 文字列を分割して配列にする
  - split（/分割に使う文字/, 文字列）

```
my $string2 = 'Tokyo,Japan';
my @arrary2 = split(/,/, $string2);
print $array2[0], "¥n";
```

「Tokyo」と表示されます

引 数

Perlでも「引数」が使えるの？
値はどうやって受け取るの？

- Perlでも引数が使えます
- Perlでは、引数を@ARGVという名前の専用の配列で受け取ります
  - $ARGV[0]に1番目の引数が、$ARGV[1]に2番目の引数が（以下同様）入ります

実習４

- 次のPerlスクリプト・perl4.plを書いて実行してみましょう

```
$ gedit perl4.pl
```

```
print "Num: ", int(@ARGV), "¥n";
print "3rd: ", $ARGV[2], "¥n";
```

```
$ perl perl4.pl Pink Red Blue
```

値は何でもいいので、引数を3つ以上指定して実行してください

実習４・解答

```
print "Num: ", int(@ARGV), "¥n";
print "3rd: ", $ARGV[2], "¥n";
```

```
$ perl perl4.pl Pink Red Blue
Num: 3
3rd: Blue
```

- 引数が3未満だと$ARGV[2]は未定義になります

```
$ perl perl4.pl Pink Red
Num: 2
3rd:
```

値がたくさんあって各データに名前をつけたい時

- ふたたび、あなたは小学校の先生です
- クラス40名の誕生日をPerlで管理したいと思います

Aさんは5月10日、
Bさんは2月28日、...

```
my @birth = ("0510", "0228", ... );
```

配列では駄目だ！日付が誰の誕生日かわからない

値がたくさんあって
各データに名前をつけたい時

- 「 ハッシュ 」を使うと、各データの名前（キー）と値を対で入力できます

–例）Aさん　→　5月10日
キー　　値

ハッシュ

- ハッシュに値を入れるには、
  my %ハッシュ名=(キー=>値);
  のようにします

```
my %birth = ("A"=>"0510", "B"=>"0228",
 ... );
```

Aさんは5月10日、
Bさんは2月28日、...

ハッシュ

- ハッシュから値を取り出すには、
  $ハッシュ名{キー}　と書きます

```
my %birth = ("A"=>"0510", "B"=>"0228",
  ... );

print $birth{"B"}, "¥n";
```

0228が出力されます

実習5

- 次のPerlスクリプト・perl5.plを書いて実行してみましょう

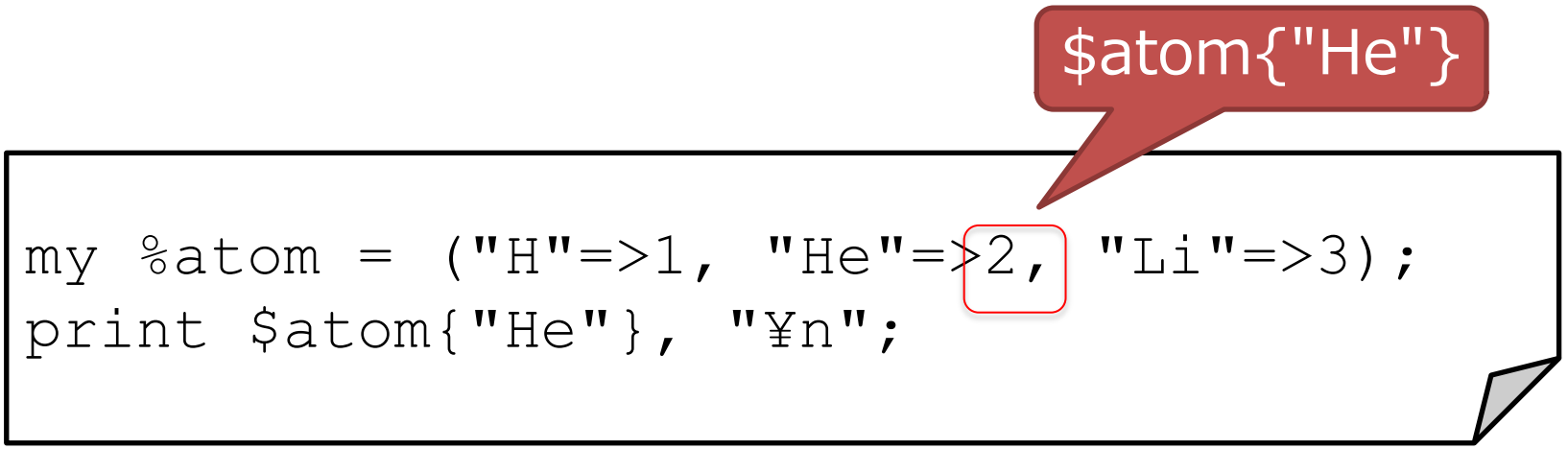
```
$ gedit perl5.pl
```

```
my %atom = ("H"=>1, "He"=>2, "Li"=>3);
print $atom{"He"}, "¥n";
```

```
$ perl perl5.pl
```

実習５・解答

- キー「He」の値＝「2」が出力されます



ハッシュにキーと値を追加する

- $ハッシュ名{キー}＝値 を実行します

```
my %atom = ("H"=>1, "He"=>2, "Li"=>3);
$atom{"Fe"} = 26;
$atom{"Ca"} = 20;
```

キー「Ca」の値として20、
キー「Fe」の値として26が
入力されます

配列と違ってハッシュは添字でアクセスしないため
最初に入力する順番には意味がありません

配列とハッシュの違い

- 配列のイメージ

配列は、複数のデータに
端から順にアクセスしたい
場合に向く

- ハッシュのイメージ

ハッシュは、該当データに
ピンポイントにアクセス
したい場合に向く

質問

シェルスクリプトのように、
Perlでも条件付き処理や
繰り返し処理が行えるの？

- Perlでも条件付き処理や繰り返し処理が可能です
- シェルスクリプトと書き方が似ていますが微妙に異なるので混乱しないようにしましょう

条件付き処理

- if-elsif-else 構文を使います

```
if(条件1){
  ～条件1を満たした時の処理～
}
elsif(条件2){
  ～条件1は満たさなかったが、
     条件2を満たした時の処理～
}
else{
   ～どの条件も満たさなかった
     時の処理～
}
```

条件付き処理

- 違う点を探してみましょう

Perl

```
if(条件1){
 ～条件1を満たした時の処理～
}
elsif(条件2){
 ～条件1は満たさなかったが、
    条件2を満たした時の処理～
}
else{
   ～どの条件も満たさなかった
    時の処理～
}
```

シェルスクリプト

```
if   [ 条件1 ]
then
   ～条件1を満たした時の処理～
elif   [ 条件2 ]
then
   ～条件1は満たさなかったが、
    条件2を満たした時の処理～
else
   ～どの条件も満たさなかった
    時の処理～
fi
```

条件付き処理

シェルスクリプトの
比較演算子と混同しない
ようにしましょう

- Perlの比較演算子
  - 数値の比較演算子

A == B	A＝Bなら
A != B	A≠Bなら
A < B	A＜Bなら
A <= B	A≦Bなら
A >= B	A≧Bなら
A > B	A＞Bなら

  - 文字列の比較演算子

A eq B	AとBが 同じなら
A ne B	AとBが 異なれば

条件付き処理

- 複数の条件を指定

	Perl	【参考】Shell
条件1　AND　条件2	条件1 && 条件2	条件1 -a 条件2
条件1　OR　条件2	条件1 || 条件2	条件1 -o 条件2
条件1　でなければ	! 条件1	! 条件1

実習6

- 次のPerlスクリプト・perl6.plを書いて引数に数字を与えて実行しましょう

```
my $i = $ARGV[0];
if($i >= 10){
  print "$i is equal to or larger than 10¥n";
}
else{
  print "$i is smaller than 10¥n";
}
```

```
$ perl perl6.pl 6
$ perl perl6.pl 11
```

実習６・解答

- 引数の値により結果が変わります

```
$ perl perl6.pl 6
6 is smaller than 10

$ perl perl6.pl 11
11 is equal to or larger than 10
```

クイズ

- スクリプトの実行結果はどうなりますか？

Q3.pl

難易度：★

```
my $time=13;
if ($time < 12 ){
    print "おはよう¥n";
}
elsif ($time <18 ){
    print "こんにちは¥n";
}
else {
    print "こんばんは¥n";
}
```

```
$ perl Q3.pl
```

A

おはよう

B

こんにちは

C

こんばんは

D

エラーになる

クイズ

- スクリプトの実行結果はどうなりますか？

Q3.pl

```
my $time=13;
if ($time < 12 ){
    print "おはよう¥n";
}
elsif ($time <18 ){
    print "こんにちは¥n";
}
else {
    print "こんばんは¥n";
}
```

```
$ perl Q3.pl
```

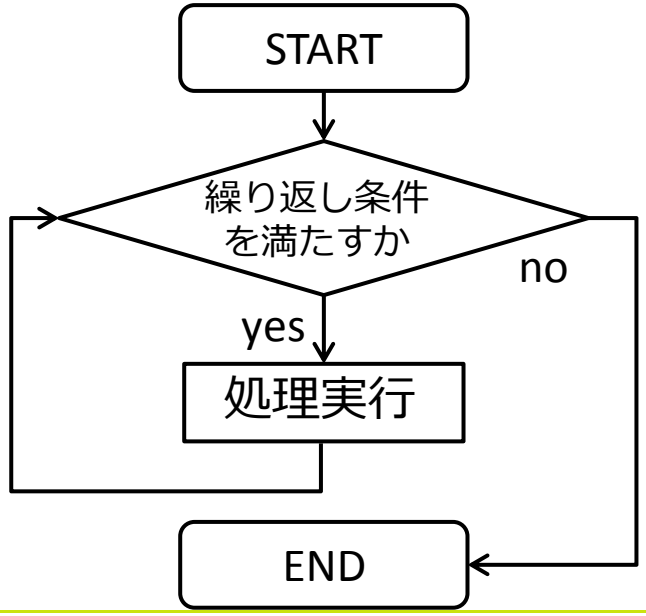
正解は、**B**！！

1. ifの条件を満たさないので、if文は実行されません。
2. elsifの条件を満たすのでelsif文が実行されます。
3. elsif文が実行されたので、else文は実行されません。

繰り返し処理

- Perlの繰り返し処理にはforeach, for, whileなどがあります

whileの例



繰り返し処理・foreach

- 配列の各要素に対して繰り返す

```
foreach $変数(@配列){処理内容}
```

```
my @gene_arr = ("Oct4","Sox2","Kif4","c-Myc");

foreach my $gene(@gene_arr){
  print $gene, "¥n";
}
```

@gene_arrの先頭から
値を一つずつ取り出して
変数$geneに入れてprint

出力結果

```
Oct4
Sox2
Kif4
c-Myc
```

繰り返し処理・for

- 変数の値の変化に応じて繰り返す

  for(変数の初期値; 繰り返し条件; 変数増分){処理内容}

```
my @gene_arr = ("Oct4","Sox2","Kif4","c-Myc");

for(my $i=0; $i<length(@gene_arr); $i=$i+1){
  print $gene_arr[$i], "¥n";
}
```

int

$iが@gene_arrの要素数より小さい間、@gene_arrの各値をprint

出力結果

```
Oct4
Sox2
Kif4
c-Myc
```

繰り返し処理・while

- 条件を満たす間繰り返す

  while(繰り返し条件){処理内容}

```
my @gene_arr = ("Oct4","Sox2","Kif4","c-Myc");

my $i=0;
while($i<length(@gene_arr)){
  print $gene_arr[$i], "¥n";
  $i = $i + 1;
}
```

int

この行がないと
無限ループになります

出力結果

```
Oct4
Sox2
Kif4
c-Myc
```

実習７

- 次のPerlスクリプト・perl7.plを書いて実行してみましょう

```
$ cp perl3.pl perl7.pl
$ gedit perl7.pl
```

perl7.plを以下のように変更して保存します

```
my @nuc = ("A", "T", "G", "C");
my $i = int(@nuc);
while($i>0){
  $i = $i - 1;
  print $nuc[$i], "¥n";
}
```

余裕のある方は
同じ処理をforでも
書いてみてください

実習７・解答

- 添字3→2→1→0の順に@nucの値がprintされます

```
$ perl perl7.pl
C
G
T
A
```

```
my @nuc = ("A", "T", "G", "C");
my $i = int(@nuc);
while($i>0){
   $i = $i - 1;
   print $nuc[$i], "¥n";
}
```

実習７・解答

- 違う書き方もできます

```
my @nuc = ("A", "T", "G", "C");
my $i = int(@nuc);
while($i>0){
  $i = $i - 1;
  print $nuc[$i], "¥n";
}
```

「 $#nuc + 1 」

$i--;

$i−−は$i=$i−1と
$i++は$i=$i+1と
同じ意味になります

実習７・解答

- 同じ処理をfor, foreachで書いた場合

```
for(my $i=int(@nuc); $i>0; $i=$i-1){
  $i = $i - 1;
  print $nuc[$i], "¥n";
}
```

```
foreach my $base(reverse @nuc){
  print $base, "¥n";
}
```

reverse:
配列を逆順にする

繰り返し処理でハッシュにアクセス

- foreachを使う場合

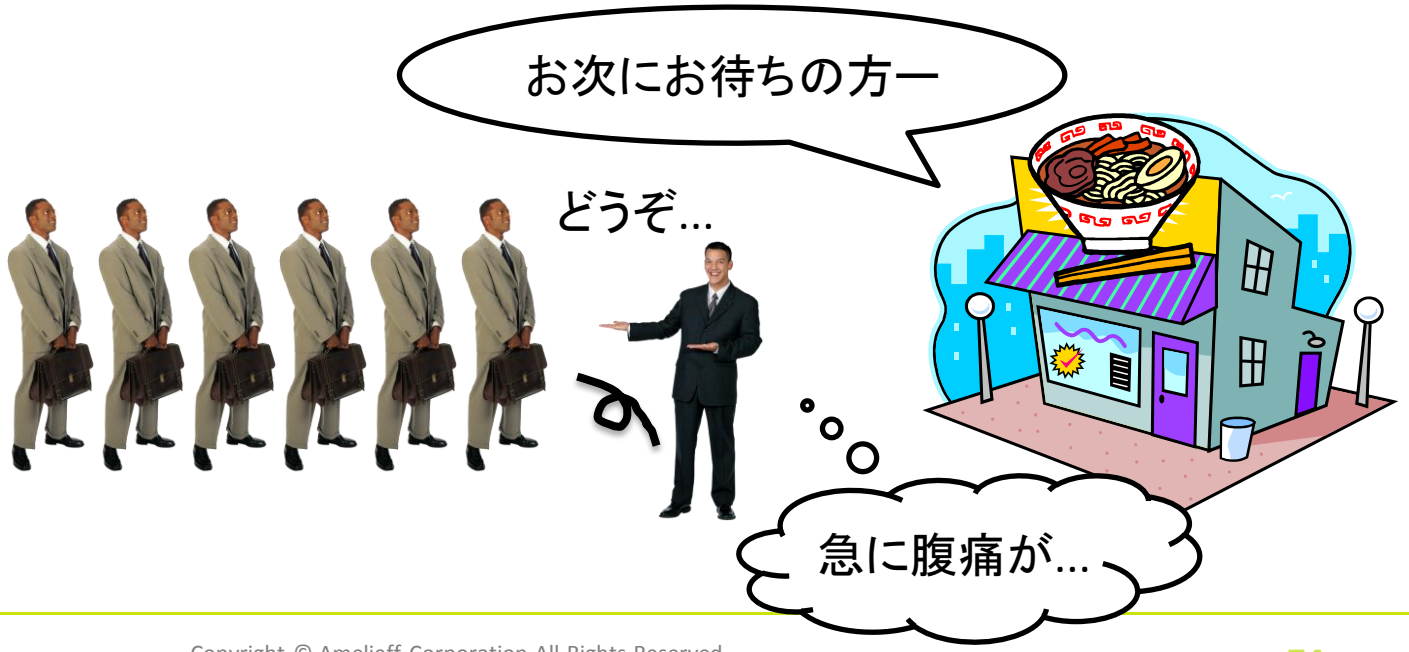
```
my %atom = ("H"=>1, "He"=>2, "Li"=>3);
foreach my $key(keys %atom){
  $val = $atom{$key};
  print $key, ":", $val, "¥n";
}
```

- whileを使う場合

```
my %atom = ("H"=>1, "He"=>2, "Li"=>3);
while(my ($key, $val) = each %atom){
  print $key, ":", $val, "¥n";
}
```

繰り返し処理

- 繰り返し処理の中で次の要素にスキップするにはnextを使います



繰り返し処理

- 繰り返し処理の中で次の要素にスキップするにはnextを使います

```
for(my $i=0; $i<5; $i++){
  if($i==2){
    next;
  }
  print $i, "¥n";
}
print "END¥n";
```

出力結果

```
0
1
3
4
END
```

繰り返し処理

- 繰り返し処理自体を中止するには lastを使います

繰り返し処理

- 繰り返し処理自体を中止するにはlastを使います

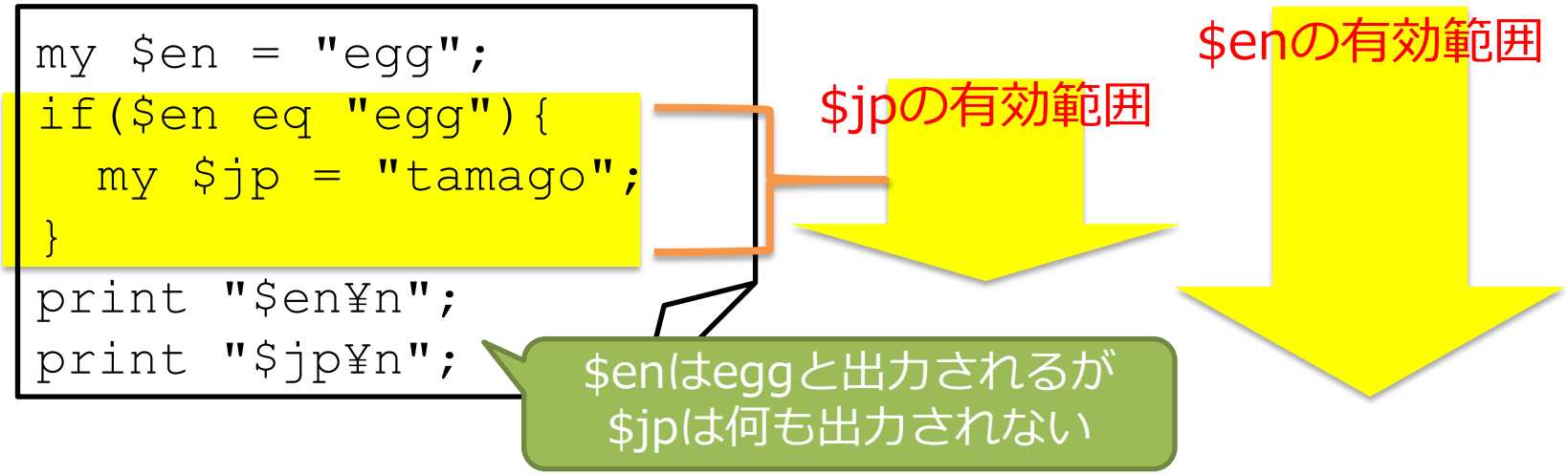
```
for(my $i=0; $i<5; $i++){
  if($i==2){
    last;
  }
  print $i, "¥n";
}
print "END¥n";
```

出力結果

```
0
1
END
```

スコープ

- {}で囲んだ範囲をスコープと呼びます
- スコープ内で定義した変数はそのスコープでのみ有効です



不満

配列やハッシュを使っても
大量のデータを手入力するのは大変！
ファイルから読み込めるといいのに

- データをファイルから読み込んだり、ファイルに書き出したりできます
- ファイル入出力には「 ファイルハンドル 」を使います

ファイル入出力

- ファイルから1行ずつ読み込んで処理するには次のように書きます

```
my $file = "input.txt";

open my $fh, "<", $file or die;
while(<$fh>){
  chomp;
  if($_ eq "abc"){
    print "$_¥n";
  }
}
close  $fh;
```

変数fileにファイル名を入力

ファイルハンドル$fhを読込用で開く

$fhから1行ずつ読み込んで変数$_に入れる

$_末尾の改行コードを除去する

$_が"abc"なら$_を改行コードを付与して出力

ファイルハンドル$fhを閉じる

ファイル入出力

- ファイルに書き出すには次のように書きます

```
my $file = "output.txt";

open my $fh, ">", $file or die;
print $fh "test¥n";
close  $fh;
```

変数fileにファイル名を入力

$fhを書出用で開く

$fhに文字列"test"を改行をつけて出力

$fhを閉じる

バグを見つけやすくする

- Perlは制約が緩い言語のため、バグを見つけにくいことがあります
- 以下を記述すると変数がちゃんと定義されているかチェックされるようになります

```
use strict;
use warnings;
```

実習８　1/2

- 次のPerlスクリプト・perl8-1, perl8-2.plを書いて実行してみましょう

```
$ cp perl2.pl perl8-1.pl
$ gedit perl8-1.pl
```

perl8-1.plを以下のように変更して保存します

```
$message="Bye!";
print $message, "¥n";
```

myを削除
printに改行コードを追加

```
$ perl perl8-1.pl
```


次ページにつづく

実習８　２/２

- 次のPerlスクリプト・perl8-1, perl8-2.plを書いて実行してみましょう

```
$ cp perl8-1.pl perl8-2.pl
$ gedit perl8-2.pl
```

perl8-2.plを以下のように変更して保存します

```
use strict;
use warnings;
$message="Bye!";
print $message, "¥n";
```

use ~ を追記

```
$ perl perl8-2.pl
```

エラーが出るようになります

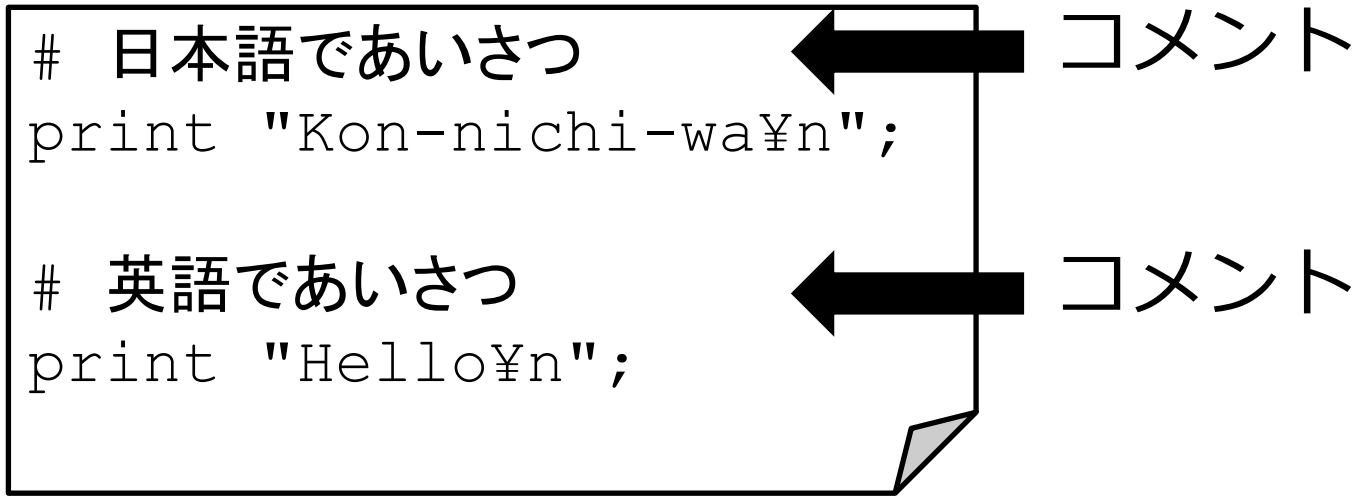
実習８・解答

- perl8-1

```
$ perl perl8-1.pl
Hello
$ perl perl8-2.pl
Global symbol "$message" requires explicit package
name at perl8.pl line 3.
Global symbol "$message" requires explicit package
name at perl8.pl line 4.
Execution of perl8.pl aborted due to compilation
errors.
```

コメント

- #で始まる行はコメント扱いとなり、処理に影響しません



シバン

- スクリプトの1行目に以下を記述するとこのファイルがPerlスクリプトであることが明示的になります

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

- これにより、perlコマンドなしでも実行できるようになります

```
$ chmod a+x perl9.pl
$ ./perl9.pl
```

chmod a+x：実行権限をつける

正規表現

- 文字列のパターンを表現する方法
  –Perlでは、正規表現を//で定義する

```
my $str = "bioinfo";
if($str =~ /info/){
 print "マッチ¥n";
}
else{
  print "マッチしません¥n ";
}
```

変数$strに「info」という
文字列が含まれていれば
「マッチ」と出力、
そうでなければ
「マッチしません」と出力

いろいろなマッチングパターン

正規表現	文字列	マッチしたか？
/bioinfo/	bioinformatics	Yes
/bioinfo/	informatics	No
/^bioinfo/	bioinformatics	Yes
/bioinfo$/	bioinformatics	No
/[atgc]/	bioinformatics	Yes
/[a-z]/	bioinformatics	Yes
/bio./	bioinfo	Yes
/bio./	bio	No
/bi*o/	biiio, bio, bo	Yes
/bi*o/	aio	No
/bi+o/	biiio,bio	Yes
/bi+o/	bo, dio	No

^	行頭を表す
$	行末を表す
[atgc]	atgcのいずれか
[a-z]	小文字の英字のいずれか
.	任意の一文字
a*	aの0回以上の繰返し
a+	aの1回以上の繰返し

マッチした箇所の取り出し

- 正規表現パターン中の()で囲った箇所を$1, $2, …で取り出すことができます

```
my $str = "Amelieff";

if($str =~ /(Ame*)li(ef+)/)
  print $1, "¥n";
  print $2, "¥n";
}
```

```
Ame
eff
```

1番目の()にマッチした文字列が$1に
2番目の()にマッチした文字列が$2に
…（以下同じ）入る

正規表現を用いた置換

- ~s/正規表現パターン/置換文字列/オプション

```
my $str1 = "genome,proteome";
$str1 =~ s/ome/omics/;
print "$str1¥n";
```

```
genomics,proteome
```

変数$str1中に最初に登場した「ome」という文字列が「omics」に置き換わる

```
my $str2 = "genome,proteome";
$str2 =~ s/ome/omics/g;
print "$str2¥n";
```

```
genomics,proteomics
```

変数$str2中に登場した全ての「ome」という文字列が「omics」に置き換わる

gオプションをつけると
マッチしたすべてについて置換

正規表現を用いた置換

- ~s/正規表現パターン/置換文字列/オプション

```
my $str3 = "I sing a song.";
$str3 =~ s/I/You/;
print "$str3¥n";
```

```
You sing a song.
```

変数$str3中に登場した全ての「I」という文字列が「You」に置き換わる

```
my $str4 = "I sing a song.";
$str4 =~ s/I/You/i;
print "$str4¥n";
```

```
You sYoung a song.
```

変数$str4中に登場した全ての「Iまたはi」という文字列が「You」に置き換わる

iオプションをつけると
大文字個別を区別せず置換

難易度：★

クイズ

- スクリプトの実行結果はどうなりますか？

```
$ perl Q4a.pl
```

Q4a.pl

```
my $org = "Gallus gallus gallus";
$org =~ s/gallus/gorilla/g;
print $org,"¥n";
```

A

```
Gallus gallus gallus
```

C

```
Gorilla gorilla gorilla
```

B

```
Gallus gorilla gorilla
```

D

```
gorilla gorilla gorilla
```

クイズ

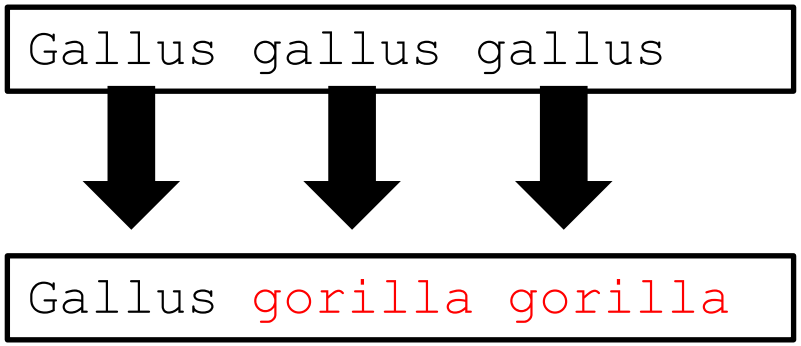
- スクリプトの実行結果はどうなりますか？

```
$ perl Q4a.pl
```

Q4a.pl

```
my $org = "Gallus gallus gallus";
$org =~ s/gallus/gorilla/g;
print $org,"¥n";
```

正解は、**B**！！



- iがついていないので、大文字小文字が区別されます（gallus→マッチ、Gallus→マッチしない）
- gがついているので、全てのgallusがgorillaに置換されます

クイズ

- スクリプトの実行結果はどうなりますか？

```
$ perl Q4.pl
```

Q4.pl

```
my $gene = "hg19;chr12;KRAS";
$gene =~ s/*;//g;
print $gene,"¥n";
```

A

```
chr12;KRAS
```

B

```
KRAS
```

C

```
chr12hg19KRAS
```

D

エラーになる

クイズ

- スクリプトの実行結果はどうなりますか？

```
$ perl Q4.pl
```

Q4.pl

```
my $gene = "chr12;hg19;KRAS";
$gene =~ s/*;//g;
print $gene,"¥n";
```

正解は、**D**！！

```
Quantifier follows nothing in
regex; marked by <-- HERE in m/*
<-- HERE ;/ at quiz.pl line 2.
```

Perlの正規表現では「*」は「直前の文字の0回以上の繰り返し」を意味するため、「*」の前には何らかの文字が必要です

エピローグ

- あなたはPerlを使って、Cの結果ファイルを、Dへ入力できるフォーマットに変換し、無事Dを実行することができました

最終課題

必要スキル
- 変数
- 配列
- 条件付き処理
- 繰り返し処理
- ファイル入出力
- 正規表現

- 次のPerlスクリプト・perl9.plを書いて実行してみましょう

【基本】hairpin.faを読み込んでそのまま出力する
  - hairpin.faを読込用で開いて1行ずつ読み込んで改行コードを削除する
  - 上記の内容に改行コード（¥n）を付けて出力

余裕のある方は
次のページへ

最終課題

必要スキル
- 変数
- 配列
- 条件付き処理
- 繰り返し処理
- ファイル入出力
- 正規表現

- perl9.plを以下のように書き換えます

【やや難】hairpin.faの塩基配列行を1行にする
  - hairpin.faを読込用で開いて1行ずつ読み込んで改行コードを削除する
  - 各行がID行かそれ以外か調べる
    - ID行なら末尾に改行コードを付加して出力する
    - ID行以外（配列行）ならそのまま出力する
    - 2回目以降のID行は頭にも改行コードをつけて出力

余裕のある方は
次のページへ

最終課題

必要スキル
- 変数
- 配列
- ハッシュ
- 条件付き処理
- 繰り返し処理
- ファイル入出力
- 正規表現

- 前ページと同じ処理を、ハッシュを使って書いてみましょう・perl10.pl

【目的】hairpin.faの塩基配列行を1行にする
  - 変数$id, ハッシュ%id2seqを定義しておく
  - hairpin.faを読込用で開いて1行ずつ読み込む
  - 各行がID行かそれ以外か調べる
    - ID行なら変数$idにID部分を入力する
    - ID行以外（配列）なら$id2seq{$id}に配列を追記
  - %id2seqの各key,valueをコンマ区切りで出力

まだ余裕のある方は次のページへ

最終課題

必要スキル
- 変数
- 配列
- 引数
- ハッシュ
- 条件付き処理
- 繰り返し処理
- ファイル入出力
- 正規表現

- perl11.pl

【応用】hairpin.faからmir-25だけ抜き出した新しいFastaファイル hairpin_mir25.fa を作成する

  - 変数$flag, 変数$id, ハッシュ%id2seqを定義しておく
  - 1番目の引数で受け取ったファイルをファイルハンドル$finで読込用で開く
  - 2番目の引数で受け取ったファイルをファイルハンドル$fotで書込用で開く
  - $fiの各行がID行かそれ以外か調べる
    - ID行なら一旦$flag=0にする
      - ID行にmir-25が含まれれば$flag=1にし、変数$idにID部分を入力する（※ただしmir-250やmir-251などは含まないようにする）
    - ID行以外（配列）でかつ$flag=1なら$id2seq{$id}に配列を追記
  - %id2seqの各key,valueを$foutにコンマ区切りで出力する

クイズ

- スクリプトの実行結果はどうなりますか？

```
$ perl Q5.pl
```

Q5.pl

難易度：★ ★ ★

```
my @arr = (1..14);
 # m..nはmからnまでの1刻みの整数を示す
my $total = 0;
foreach my $el (@arr){
  if ($el == 5){
    next;
  }
  $total += $el;
}
print("$total¥n");
```

A

100

C

105

B

114

D

120

クイズ

- スクリプトの実行結果はどうなりますか？

```
$ perl Q5.pl
```

Q5.pl

```
my @arr = (1..14);
 # m..nはmからnまでの1刻みの整数を示す
my $total = 0;
foreach my $el (@arr){
  if ($el == 5){
    next;
  }
  $total += $el;
}
print("$total¥n");
```

正解は、**A**！！

1+2+・・・+14=105ですが、ここではif文を使って$elが5のときの繰り返し処理をスキップしているので、105-5で正解は100です