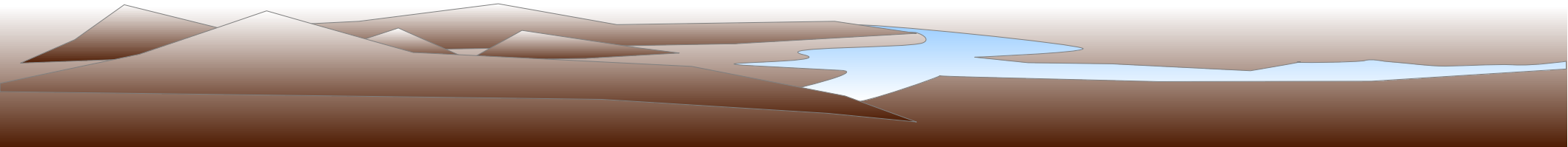


Master in Bioinformatica

Programmazione in Perl Operazioni su stringhe e liste

Andrea Sterbini
sterbini@di.uniroma1.it



Operazioni su stringhe

 Concatenare due stringhe con il “.” (punto)

```
my $unita = $stringa1 . $stringa2 ;
```

 Tagliarne un pezzo


```
my $pezzo = substr( $stringa, $inizio, $fine ) ;
```

 Trasformare un carattere in un altro (con “tr”)

```
$stringa =~ tr/<caratteri>/<rimpiazzi>/ ;
```

```
$stringa =~ tr/actgACTG/TGACTgac/ ;
```

 (torna anche il numero di caratteri trasformati)





 Cercare “qualcosa” in una stringa (con “m”)

```
$stringa =~ m/<pattern>/ ;
```






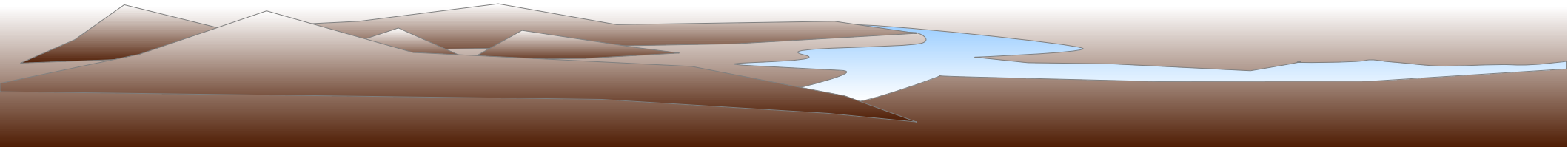
Cosa cercare in una stringa

 I pattern elementari più comuni:


-  un carattere alfanumerico corrisponde a sè stesso
-  il carattere '.' (punto) che corrisponde a qualsiasi carattere
-  [`<caratteri>`] uno tra i caratteri elencati
-  [`^<caratteri>`] un carattere DIVERSO da quelli elencati

 Pattern composti

-  `<pattern>*` **zero o più** ripetizioni del `<pattern>`
-  `<pattern>+` **una o più** ripetizioni del `<pattern>`
-  `<pattern>?` **zero o una** ripetizioni del `<pattern>`



Parentesi e caratteri speciali

 Le parentesi tonde servono a **raggruppare** ed a **estrarre** i sotto-pattern trovati

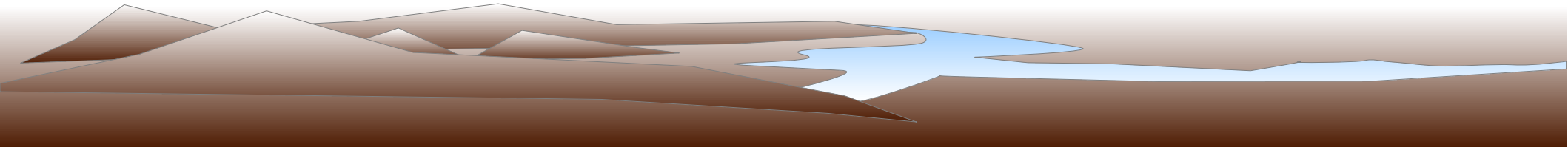
```
$stringa =~ m/ (<pattern> ) / ;  
$trovato = $1 ;
```

 Vengono create le variabili automatiche **\$<N>** (contate quante sono le parentesi aperte)

```
( (<pattern> ) + <altro> )      -> $2
```

 I caratteri speciali vanno preceduti da '****' (backslash)

```
\\      \(      \)      \[      \]      \.      \/
```



Ricerche: esempi

☕ Cercare una sequenza di almeno un A oppure T

```
$stringa =~ m/ ([AT]+) / ;
```

```
my $trovato = $1 ;
```

☕ Cercare un A seguito da qualsiasi cosa seguito da C

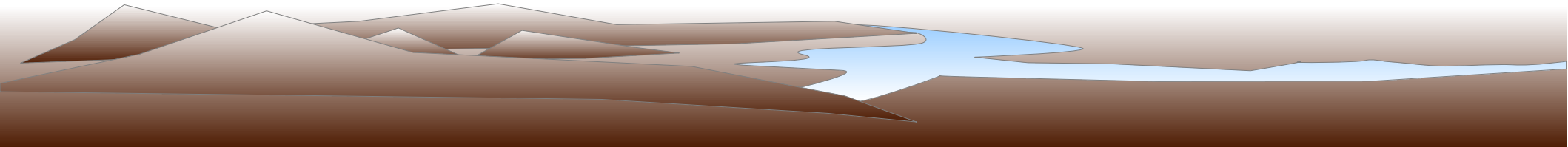
☕ Normalmente i pattern ripetuti sono “**avidì**” e cercano le corrispondenze PIU' LUNGHE possibili

```
$stringa =~ m/ (A.*C) / ;
```

```
my $trovato = $1 ;
```

☕ Se voglio la corrispondenza PIU' CORTA (uso '?')

```
$stringa =~ m/ (A.*?C) / ;
```



Cercare e sostituire

 Rimpiazzare una parte con un'altra

```
$stringa =~ s/<pattern>/<rimpiazzo>/ ;
```

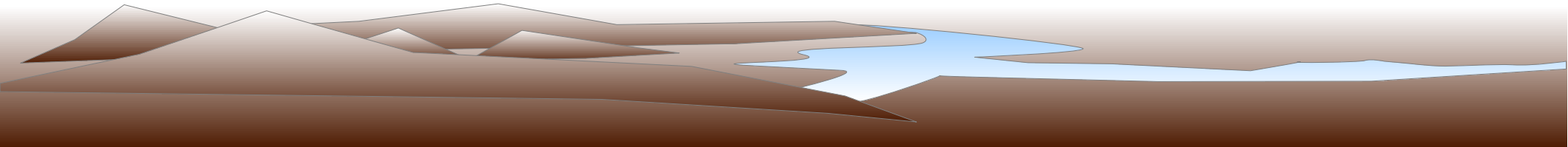
 Per rimpiazzare TUTTE le occorrenze (con “**g**”)

```
$stringa =~ s/<pattern>/<rimpiazzo>/g ;
```

 Es.: scambiare due parti

 Usate le variabili automatiche nella sostituzione

```
$stringa =~ s/([^ ]+)([^ ]+)/$2$1/ ;
```



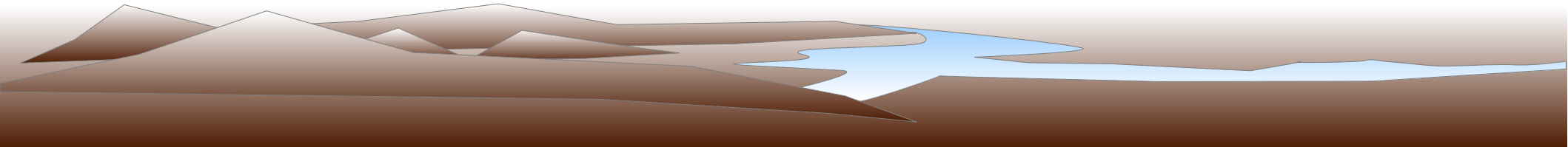
Esempio: complemento (sbagliato)

```
my $DNA = 'ACTGGTACCT' ;  
$DNA =~ s/A/T/g ;  
$DNA =~ s/C/G/g ;  
$DNA =~ s/T/A/g ;  
$DNA =~ s/G/C/g ;  
print $DNA;
```

☕ Si ottiene: **ACACCAACCA**

☕ invece che: **TGACCATGGA**

☕ PERCHE?



Esempio: complemento (corretto)

☕ Usate l'operatore **tr**

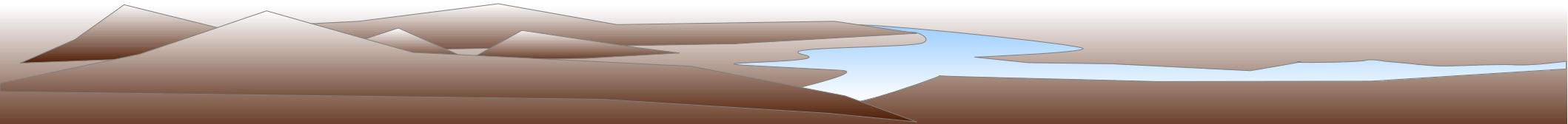
```
my $DNA = 'ACTGGTACCT' ;  
$DNA =~ tr/ACTG/TGAC/ ;  
print $DNA;
```

☕ Finalmente si ottiene: **TGACCATGGA**

☕ Con **tr** potete anche contare le **A, T, C, G**

```
my $numA = ($DNA =~ tr/A/A/ ) ;  
my $numT = ($DNA =~ tr/T/T/ ) ;
```

...



Operazioni sulle liste

☕ Estrarre un gruppo di valori successivi

```
@lista = (0, 1, 2, 3, 4, 5, 6, 7, 8) ;
```

```
@parte = @lista[3..5] ;
```

☕ Prendere un elemento

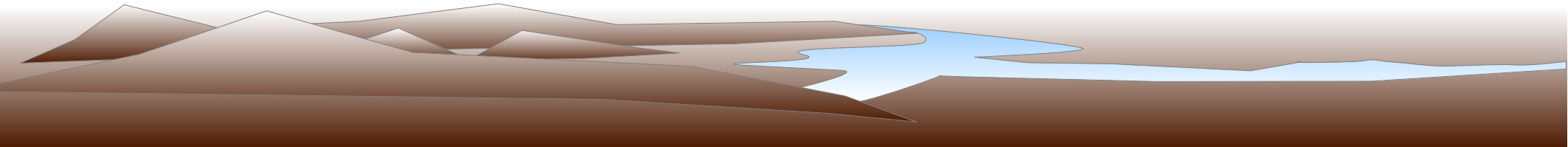
```
$quarto = $lista[3] ;
```

☕ Cambiare un elemento

```
$lista[3] = 55 ;
```

☕ Costruire una stringa di valori separati da **virgole**

```
$stringa = join(", ", @lista) ;
```



Altri esempi

☕ Come sapere il numero di elementi di una lista

```
my $numero = @lista ;
```

```
my $numero = scalar(@lista) ; # meglio
```

☕ Spezzare una stringa in corrispondenza degli **spazi**

```
@parole = split( " ", $testo ) ;
```

☕ Per elaborare gli elementi di una lista

```
my $elemento ;
```

```
foreach $elemento (@lista) {
```

```
    # codice eseguito per ogni elemento
```

```
}
```

