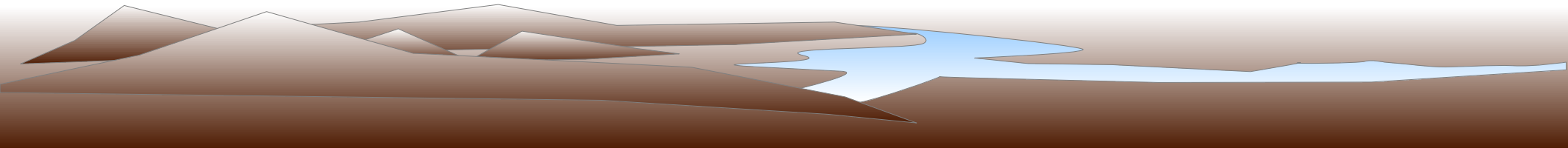
Master in Bioinformatica

Programmazione in Perl
Strutture dati

Andrea Sterbini
sterbini@di.uniroma1.it



Tipi elementari

- Finora abbiamo visto i tipi di dato “elementari”
  - **Valori scalari**: numeri interi, decimali, stringhe
  - **Liste/vettori**: sequenze di elementi scalari
  - **Hash-tables**: insieme di coppie
    chiave (stringa) => valore (scalare)
- Come fare a inserire strutture dentro altre strutture?
  - Basta trasformarle in un valore scalare: il **riferimento**
  - **Riferimento**: un numero che corrisponde ad una struttura dati in memoria
  - NON E' UN PUNTATORE! (indirizzo in memoria)

Conversione dato -> riferimento

- Per convertire una struttura in memoria nel suo riferimento si usa l'operatore '\' (backslash)

```
my @altezze = ( 150, 165, 184, 190);
my $rifAltezze = \@altezze;
my %eta = (andrea => 44, carlo => 35);
my $rifEta = \%eta;
my $dna = 'ACTGGTACT';
my $rifDna = \$dna;
```

Conversione riferimento -> dato

- Per convertire un riferimento nel dato originale si usano gli operatori '**$**', '**%**' e '**@**'

```
my @altezze = @$rifAltezze ;

my %eta = %$rifEta ;

my $dna = $$rifDna ;
```

- Per evitare errori conviene usare le parentesi graffe

```
my @altezze = @{$rifAltezze} ;

my %eta = %{$rifEta} ;

my $dna = ${$rifDna} ;
```

Esempio: una agenda telefonica

- Una voce contiene: nome, cognome, numero di telefono. Viene spontaneo usare una hash-table

```
my %voce = ( nome => 'Andrea', cognome =>
  'Sterbini', telefono => '0649918538' );
```

- Vogliamo trovare le voci dell'agenda per cognome-nome. Viene spontaneo usare ancora una hash-table

```
my %agenda = (
    Sterbini-Andrea => ... ,
    Rossi-Carlo     => ... )
```

- Il valore associato a ciascun cognome-nome sarà il riferimento alla hash-table della voce con i dettagli.

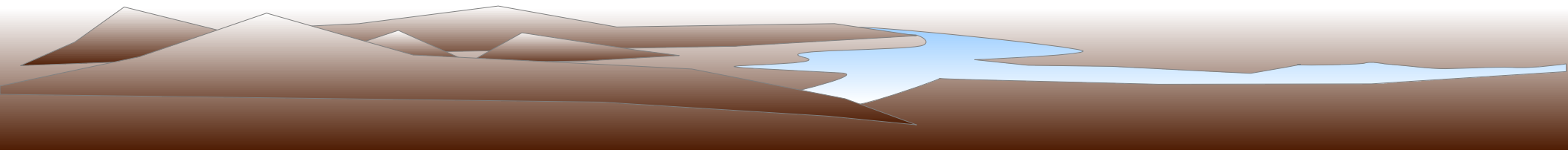
Intermezzo: abbreviazioni

- Per comodità possiamo usare due abbreviazioni:
  - Un riferimento ad una lista/vettore si ottiene con: `[...]`

```
my $rifVettore = [1, 2, 3, 4, 5, 6] ;
```

  - Un riferimento ad una hash-table si ottiene con: `{...}`

```
my $rifHashTable = {
      nome       => 'Andrea',
      cognome    => 'Sterbini',
      telefono   => '0649918538',
  } ;
```



Es.: agenda telefonica (continua)

- Usiamo le abbreviazioni per scrivere alcuni dati nell'agenda

```
my %agenda = (
    'Sterbini-Andrea' => {
          nome       => 'Andrea',
          cognome    => 'Sterbini' ,
          telefono   => '0649918538' },
    'Rossi-Paolo' => {
          nome       => 'Paolo',
          cognome    => 'Rossi' ,
          telefono   => '1234567890' },
  );
```

Come usare l'agenda

- Vediamo se una voce è presente con **exists**:

```
exists $agenda{'Sterbini-Andrea'};
```

- Leggiamo il numero di telefono

```
my $tel =
  $agenda{'Sterbini-Andrea'}{telefono};
```

- Cambiamo un numero di telefono

```
$agenda{'Rossi-Paolo'}{telefono} =
  '0123456789';
```

... usare l'agenda

- Inseriamo una nuova voce

```
$agenda{'Agnelli-Gianni'} = {
  cognome  => 'Agnelli',
  nome     => 'Gianni',
  telefono => '022222222' };
```

- Eliminiamo una voce

```
$agenda{'Sterbini-Andrea'} = undef;
```

- Difetti (pregi?) di questa implementazione
  - Non ammette due persone con lo stesso cognome-nome
  - Come fare a cercare una voce per solo cognome?

Agenda 1: indicizzata per cognome

- Complichiamo la struttura (aggiungiamo un livello)
  - Usiamo il solo cognome come chiave
  - Ad ogni cognome corrisponderanno più voci: usiamo una hash-table indicizzata sui loro nomi

```
my %agenda = (
    Sterbini => {
        Andrea => {
            nome => ...,
            cognome => ...,
            telefono => ...
        },
        Elena => { ... },
    },
    ...
);
```

Usiamo la nuova agenda

- Esiste una certa voce?

```
exists $agenda{'Sterbini'} &&
exists $agenda{'Sterbini'}{'Andrea'}
```

- Leggo il numero di telefono

```
my $tel = $agenda{'Sterbini'}{'Andrea'}{'telefono'};
```

- Aggiungo una voce

```
$agenda{'Sterbini'}{'Andrea'} = {
    nome => 'Andrea', cognome => 'Sterbini',
    telefono => '0649918538' };
```

- Elimino una voce

```
$agenda{'Sterbini'}{'Andrea'} = undef;
```

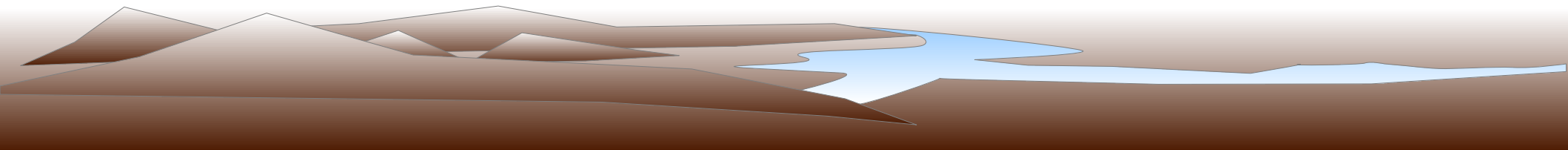
Agenda 2: proviamo con un vettore

- L'implementazione precedente:
  - è molto veloce nell'accedere i singoli elementi
  - ma non permette facilmente di ordinare gli elementi
  - o di fare ricerche su altri campi
  - o di avere più persone con stesso nome-cognome
- Proviamo a semplificarne la struttura
  - Ogni voce rimane una hash-table
  - L'agenda è un vettore di (riferimenti a) voci
- Chiaramente l'accesso agli elementi sarà + complicato

Agenda 2

- Iniziamo a metterci un po' di dati

```
my @agenda = (
        { nome      => 'Andrea',
          cognome   => 'Sterbini' ,
          telefono  => '0649918538' },
        { nome      => 'Paolo',
          cognome   => 'Rossi' ,
          telefono  => '1234567890' },
        ...
);
```



Agenda 2: troviamo un valore

- Dobbiamo cercare l'elemento scorrendo il vettore

```
my $nomeC      = 'Andrea';
my $cognomeC   = 'Sterbini';
my $voce       = undef;
my $trovata    = 0; # 0 corrisponde a FALSO
foreach $voce (@agenda) {
  if ($$voce{nome} eq $nomeC &&
      $$voce{cognome} eq $cognomeC) {
    $trovata = 1;
    print 'Trovato: ' . $$voce{telefono};
    last;
  }
}
print 'voce non trovata' if !$trovata;
```