SANDATA | IT-Trainingszentrum GmbH
Die IT-Gruppe

Perl - Seminar

Grundkurs

Dr.sc.nat. Michael J.M. Wagner*

Revision 208

*michael@wagnertech.de

Inhaltsverzeichnis

1 Einführung

Was ist Perl

- Programmiersprache aus dem UNIX-Bereich
- Ursprünglicher Einsatz: Bearbeiten und Auswerten großer Datenmengen (Log-Dateien)
- „Practical Extraction and Reporting Language"
- Nach Erfindung des WWW: Standard für CGI1[1]-Programme
- Heute ist Perl für fast alle Betriebssysteme verfügbar
- Syntax basiert auf C, ist aber reicher (komplizierter?)
- Skriptsprache → Portabilität

Entwicklung und Geschichte von Perl

- 1987 von Larry Wall
- Best of C, Basic, UNIX-Shell, awk, sed
- Zitat von Larry Wall

  > Der Entwickler Larry Wall bezeichnet Perl als „... Interpretersprache, die optimiert ist, willkürliche Textdateien zu durchduchen, aus ihnen Informationen zu filtern und Berichte aus diesen Informationen zu erstellen. Außerdem eigent sich Perl sehr gut für fast alle Aufgaben der Systemsteuerung. Die Sprache soll praktischen Nutzen bringen und nicht unbedingt schön sein."[2]

- Open-Source Entwicklergemeinde

Perl installieren

Für diesen Kurs ist eine virtuelle Maschine (LinuxMint) vorbereitet, die Eclipse mit Perl-Addon enthält.

- Eclipse ist eine Standardentwicklungsumgebung
- Perl ist auch als UNIX-Interpreter installiert

Aufgabe:

- Virtuelle Maschine starten
- Eclipse starten

[1] Common Gateway Interface
[2] Teich: S. 7.

- Eclipse-Projekt anlegen
- `HalloWelt.pl` anlegen (Code findet sich am „Schreibtisch", Kapitel 2)
- Start über Eclipse und Kommandozeile

2 Programmieren mit Perl

Anweisungen, Blöcke und Kommentare

- Anweisungen werden mit „;" abgeschlossen
- Tolerante Syntax (z.B. `print` mit und ohne Klammern)
- Anweisungsblöcke mit geschweiften Klammern
- `#` als Kommentarzeichen
- Blockkommentare über `=pdo / =cut`

She-Bang, die erste Zeile

- Eingeschränkte Bedeutung in Windowssystemen
- Optionen für Perl-Interpreter [ST: S. 19]

Regeln für Bezeichner

- Buchstaben, Ziffern, Unterstrich
- Sensibel auf Groß-/Kleinschreibung

Ausgabe von Daten

- Syntax der Print-Anweisung [ST: S. 20]
- „`\n`" muss explizit angegeben werden
- Heredoc [ST: S. 22]

Aufgabe:

Beispieldatei `Kapitel_03/Heredoc.pl` in den *workspace* kopieren und ausführen.

3 Einfache Sprachelemente

Variablen

- Skalare Variablen (Zahlen, Zeichenketten, Referenzen): `$skalar`
- Listen, Arrays: `@array`
- Assoziative Arrays (Hashes): `%hash`
- Wertzuweisung erfolgt über „=" (auch transitiv möglich)
- Perl-Standard: Variablen müssen *nicht* deklariert werden
- Programmierstandard: Variablen müssen deklariert werden → `use strict;`
- Variablendeklaration erfolgt mit `my`. Die Sichtbarketi der Variable ist dann auf den Block geschränkt (wie bei C, Java, ...).

Datentypen

- Keine Typisierung durch den Programmierer (außer Skalar, Array, Hash)
- Interne Typisierung: Zahlen/Zeichenketten
- Implizite Umwandlung

Operatoren

- Arithmetische Operatoren [ST: S. 35]
- Vergleichsoperatoren (numerisch oder alphanumerisch) [ST: S. 36/37]
- Logische Operatoren [ST: S. 37]
- Zeichenkettenoperatoren (Konkatenation mit „.")
- Zuweisungsoperatoren (+=, ...)

4 Kontrollstrukturen

Anweisungen zur Bedingungsauswahl

- `if / if` (nachgestellt) [ST: S. 44]
- `unless / unless` (nachgestellt) [ST: S. 47]
- Ternärer Vergleichsoperator [ST: S. 47]
- `if - elsif - else` [ST: S. 49]

Bedingte Wiederholungsschleifen

- `while`-Schleife [ST: S. 51]
- `until`-Schleife: wie while, nur umgekehrte Logik
- `do - while`: wie while, nur mit Prüfung am Ende des Anweisungsblock
- `do - until` [ST]

Zählergesteuerte Wiederholung

- `for`-Syntax, wie in C [ST: S. 54]

Aufgabe (Buch, Kapitel 5):

- `AddZahlen`
- `Punkte`

5 Listen und Datenfelder

Daten in Feldern speichern

- Arten von Datenfeldern: Array, Hash [ST: S. 58]

Arrays verwenden

- Definition über runde Klammer: `('Mo', 'Di', 'Mi', 'Do', 'Fr')`
- Alternative: mit `qw`: `qw(Mo Di Mi Do Fr)`
- Variablenname beginnt mit `@`: `@array_var = qw(Mo Di Mi Do Fr);`
- Zugriff auf Einzelelement über `$` und eckige Klammer: `$array_var[3]`
  Das ist der String 'Do'
- Anzahl der Elemente eines Arrays: `$#array_var + 1` (`$#array_var` ist der höchste Index)
- Funktionen für Arrays [ST: S. 64]

Hashes verwenden

- Hashes bestehen aus Schlüssel-Wert-Paaren
- Definition über runde Klammer: `(Mo=>'Montag', Di=>'Dienstag', Mi=>'Mittwoch')`
- Variablenname beginnt mit `%`: `%hash_var`
- Zugriff auf Einzelelement über `$` und geschweifte Klammer: `$hash_var{Mi}`

  Das ist der String `Mittwoch`
- Der Zugriff auf ein Einzelelement kann sowohl lesend als auch schreibend erfolgen.
- Weitere Elemente können durch Zuweisung hinzugefügt werden: `$hash_var{Do} = 'Donnerstag';`
- Array mit Schlüsselwerten extrahieren: `keys(%hash_var)`
- Array mit Wert-Werten extrahieren: `values(%hash_var)`
- Element löschen: `delete $hash_var{key}`

Arrays und Hashes in Schleifen durchlaufen

- `foreach`-Schleife bei Arrays und Hashes → Buch

Aufgabe (Buch, Kapitel 6):

- `Uebung3`
  Hinweis: `defined` prüft, ob eine Variable definiert/ein Eintrag im Hash vorhanden ist.

6 Unterprogramme und Funktionen

Unterprogramme in Perl

- Definition mit `sub`: `sub unterprog { ... }`
- Variablenübergabe als Stellungsparameter. Die Stellungsparameter stehen in der gerufenen Routine in einem Array mit Namen `@_` zur Verfügung. Für den Zugriff darauf gibt es verschiedenste Praktiken:
  - Zugriff auf Parameterarray über Index: `my $param = $_[0];`
  - Zugriff über Arrayfunktion `shift`: `my $param = shift;`
  - Abbildung des Arrays auf Einzelvariablen: `(my $a, my $b, my $c) = @_;`

  Achtung: Befindet sich unter den übergebenen Parametern ein Array geht das in `@_` auf.

- Funktionen geben Wert zurück: `return $ret_val;`
- Bei der Verwendung von Unterprogrammen ist ein `use strict`, zusammen mit lokalen Variablen dringend empfohlen. Schlechtes Beispiel: Buch, `Volumen.pl`

Referenzen verwenden

- Referenzieren über `\`: `$var_ref = \$var;`
- Dereferenzieren über `$/@/%`: `$$var_ref, @$array_ref, %$hash_ref`

  Hier erhalte ich die ursprüngliche Variable/Array/Hash
- Dereferenzieren über `->` (nur bei Array, Hash)

  Hier erhalte ich ein Element: `$array_ref->[3], $hash_ref->{Mi}`
- Arrayreferenz definieren: `$arr_ref = ['Mo', 'Di', 'Mi'];`
- Hashreferenz definieren: `$hash_ref = {Mo=>'Montag', Di=>'Dienstag'}`
- Anwendungen
  - Variablenübergabe an Funktionen [ST]
  - Bildung von mehrdimensionalen Arrays (Array von Arrayreferenzen)

Vordefinierte Perl-Funktionen

- → Buch [ST: S. 84]
- Es gibt zahllose Perl-Module, die weitere Funktionen zur Verfügung stellen.
- Die Verwendung eines Perl-Moduls muss mit `use` angekündigt werden.

Aufgabe (Buch, Kapitel 7):

- `Uebung3`

7 Datei und Verzeichnisfunktionen

Mit Dateien arbeiten

- Dateien öffnen → Buch
- Dateien schließen: `close(DATEIHANDLE);`

Textdateien lesen

- Zeilenweise Verarbeitung → Buch
- Kompletter Dateiinhalt in einer Arrayvariablen: `@zeilen = <DATEIHANDLE>;`

In Dateien schreiben

- In Datei schreiben: `print DATEIHANDLE 'Inhalt';`

Statusinformationen über Dateien ermitteln

- Dateiprüfoperationen [ST: S. 96]

Funktionen des Dateisystems / Mit Verzeichnissen arbeiten

Viele Betriebssystemfunktionen, wie Dateien umbenennen, löschen, etc. sind direkt in Perl verfügbar [ST: S. 100]

> Aufgabe:
>
> Erstellen Sie eine Funktion, die eine beliebige Anzahl von Paramtern übernimmt und diese an eine Datei anhängt:
>
> - Aufruf: `daten_anhangen("medien.csv", "A001", "Goethe", "Faust", "B", 345, 0);`
> - Ergebnis: Datei `medien.csv` enthält: `A001,Goethe,Faust,B,345,0`

8 Zeichenketten

8.1 Funktionen mit Zeichenketten

Im Kapitel 6 wurden bereits die wichtigsten Funktionen mit Zeichenketten gezeigt.

8.2 Reguläre Ausdrücke

Suchen und Ersetzen mit regulären Ausdrücken

- Zweck → Buch

Mustererkennung mit regulären Ausdrücken

- Einfache Suchmuster → Buch

Einige Zeichen haben innerhalb regulärer Ausdrücke eine spezielle Bedeutung und müssen bei der Verwendung mit einem *backslash* gekennzeichnst werden:

`/`	leitet einen regulären Ausfruck ein
`.`	steht für ein beliebiges Zeichen
`+ * ? {}`	kennzeichnen Kardinalitäten
`^ $`	verankern ein Suchmuster am Anfang/Ende
`|`	kennzeichnet eine Auswahl
`()`	gruppiert Treffer
`[]`	definiert Buchstabenmengen

Optionen für reguläre Ausdrücke

- Alternative Zeichen werden mit eckigen Klammern ausgedrückt → Buch
- Zeichenausschluss mit Dach: `[^x]`: kein `x`
- Vordefinierte Zeichenklassen → Buch
- Wiederholungen → Buch
- Zeilenanfang: `^`, Zeilenende: `$`
- Alternative Ausdrücke werden mit `|` getrennt.

Speichern von Übereinstimmungen

- Sollen Muster gespeichert werden, so werden sie in runde Klammern geschrieben → Buch
- Die Wiederholoperatoren (`+`,`*`) verhalten sich *greedy* (gierig), d.h. sie suchen die maximale Zeichenmenge, die mit dem Suchmuster verträglich ist.
- Nicht-gierige Suche mit `?` → Buch
- Leider funktioniert das nicht in allen Regex-Implementierungen. Alternative: `[^x]*`

Ersetzen von Textteilen mit regulären Ausdrücken

- Syntax: `$var =~ s/Suchmuster/Ersetzung/[option];`
- Mögliche Optionen → Buch
- Beispiele → Buch
- Falls `/` im Suchmuster vorkommt, kann der Suchausdruck mit einem anderen Zeichen eingeleitet werden: `s!Suchmuster!Ersetzung!` → Buch

- Soll eine im Suchmuster gefundene Übereinstimmung im Ersetzungstext wiederverwendet werden:
  - Im Suchmuster mit runden Klammern „speichern"
  - Im Ersetzungstext mit `\1` „abrufen"
  - Beispiel → Buch

Aufgaben:

- Buch Kapitel 10: `Ubung2`
- Lösen Sie das Problem von Kapitel 9, Aufgabe 4 unter Verwendung regulärer Ausdrücke.

9 Objektorientierung

- Kaum eigene Sytax (kein `class` o.ä.), nur `new` (wird beim Anlegen eines Objekts aufgerufen)
- Ansatz:
  - Nehme eine Hashreferenz
  - Packe Funktionen in ein Paket (Paketname = Klassenname)
  - Vererbung wird über ein entsprechendes use-Statement realisiert: `use base ('Basisklasse');`
  - Verbinde die Hashreferenz mit dem Modul: `bless($object, 'Klasse');`
- Besonderheiten:
  - Ein Klassenmodul enthält eine Funktion `new` (den Konstuktor), die `bless` enthält und das Objekt (=Hashreferenz) zurückgibt.
  - `$obj->func($var1, $var2)` ruft die Funktion `func` aus dem zur Hashreferenz `$obj` „geblessten" Modul auf. `$obj` wird dabei als erster Parameter (`$_[0]`), `$var1` als zweiter Parameter (`$_[1]`), etc. übergeben.
  - In der Funktion `func` wird der erste Parameter üblicherweise der Variablen `$self` übergeben: `$self = $_[0];`

Die Objektinstanz liegt als als Hashreferenz vor. Damit ist es möglich, auf jedes Element der Instanz direkt zuzugreifen (`obj->{elem}`). Dies widerspricht aber der Idee, dass die Elemente eines Objekts (*members*) dessen Implementierungsgeheimnis sind. In „richtigen" objektorientierten Sprachen wird auf die Elemente eines Objets nur prozedural (über „Methoden") zugegfriffen. Die elementaren Zugriffsmethoden sind dabei die *getter* und *setter*.

Um zu prüfen, ob eine Hashreferenz zu einer bestimmten Klasse gehört, gibt es folgende Möglichkeiten:

- `blessed $obj` liefert den Klassennamen zurück.
- `$obj->isa("Kasse")` liefert `1`, falls `$obj` zur Klassenhierarchie von `Klasse` gehört.

Aufgabe:

Erstellen Sie eine Klasse `Medium` mit folgenden Eigenschaften:

- Der Konstruktor soll aus einer kommaseparierten Liste der Form `Signatur,Autor,Titel,Typ,Seitenzahl,Spieldauer` die Daten in das Objekt übernehmen.
  - Zerlegen Sie den übergebenen String (`split`)
  - Definieren Sie die Hashreferenz (`$self = { ... }`)
  - `bless($self);`
  - Rückgabe der Hashreferenz (`return $self`)
- Fügen Sie einen *getter* (`getSignatur()`) hinzu.
- Fügen Sie eine Funktion `hatSignatur($sig)` hinzu, die prüft, ob das Medium die Signatur `$sig` hat. Rückgabe: `0/1`
- Schreiben Sie ein Hauptprogramm, das
  - ein Medium anlegt
  - in einer Schleife vom Benutzer Signaturen abfragt: *Signatur eingeben:*
  - Prüft, ob die Signatur stimmt
  - Die Schleife ist beendet, wenn der Benutzer die richtige Signatur angegeben hat.

10 Fehlerbehandlung

Zur Fehlerbehandlung gibt es verschiedene Ansätze:

- Rückgabewerte vom Typ `int`
- Rückgabewerte eines speziellen Fehlertyps
- Exceptions
- Eigene Routine zur Ermittlung des Fehlerstatus

Vorteil der Rückgabewerte:

- Einfachere lokale Fehlerbehandlung

Vorteil der Exceptions:

- Fehlerbehandlung stört nicht den logischen Programmfluss.
- Fehlerbehandlung wird nicht vergessen.

Empfehlung:

- Zu erwartende (oft fachliche) Fehler werden auf Rückgabewerte abgebildet.
- Unerwartete Fehler (oft technische Fehler, Logikfehler) werden über Exceptions behandelt.

Perl hat kein echtes Konzept zur Ausnahmebehandlung. Es gibt allerdings den Befehl `die "Fehlergrund";`, der die Programmbearbeitung abbricht.

Umgekeht kann mit der Ausführung eines Programmteils innerhalb eines `eval`-Blockes gerade ein solcher Abbruch verhindert werden. Ein `die` führt dann lediglich zum Verlassen dieses Blockes.[3]

```
sub foo{
  # test something
  if ( ... ){
    die "something happend";
  }
}

eval{
  # do something
  foo();
};
if ($@) {
  # something happend
  print $@;
}
```

11 CGI-Programmierung

Perl wird gerne auch für Web-Programme benutzt. Dazu müssen die Skripte vom Webserver aufgerufen werden. Ein sehr früher Standard dafür ist das *common gateway interface* CGI.

Perl als CGI-Anwendung einrichten: [ST: S. 146]

Datenübergabe an CGI-Programme: [ST: S. 148]

> Aufgabe:
>
> Kapitel 12, Aufgabe 2

12 Datenbanken

Interaktive Webprogramme oder betriebliche Anwendungen benötigen für ihre Datenhaltung Datenbanken. Eine im Web-Umfeld weit verbreitete Datenbank ist MySQL.

Datenbanken und Tabellen erstellen: [ST: S. 192]

[3] https://www.perl.com/pub/2002/11/14/exception.html/ (13.6.2018)

Verbindung zum Datenbankserver herstellen: [ST: S. 194]

Abfragen erstellen: [ST: S. 197]

Abfrageergebnis ermitteln: [ST: S. 201.]

Aufgabe:

Ergänzen Sie die Bibliotheksanwendung:

- Legen Sie in der Datenbank eine Tabelle mit folgenden Feldern an:

```
signatur   CHAR(20) NOT NULL PRIMARY KEY,
autor      VARCHAR(150) NOT NULL,
titel      VARCHAR(150) NOT NULL,
typ        CHAR(1) NOT NULL,
seitenzahl INT NULL,
spieldauer INT NULL
```

- Ergänzen Sie die Klassen `Buch` und `CD` um eine Methode `save`, die
  - prüft, ob die Signatur in der Datenbank vorhanden ist.
    Diese Funktion können Sie in die Mutterklasse verlagern.
  - Wenn ja, einen `UPDATE` durchführt.
  - Wenn nein, die Instanz in die Datenbank einfügt.
- Legen Sie im Hauptprogramm Instanzen von `Buch` und `CD` an und rufen Sie die `save`-Methode.

Anmerkung: In größeren Programmen werden OR-Mapper (*object-relational mapper*) verwendet, um die Anwendungsklassen frei von SQL-Code zu halten.

13 Socket-Programmierung

Über Sockets wird unter UNIX/im Internet die Kommunikation zwischen Prozessen abgebildet. Für Perl steht unter UNIX das Modul `IO::Socket::UNIX` zur Verfügung, mit sich die Socketkommunikation einfach implementieren lässt. Als Kommunikationspartner stehen stets Client und Server zur Verfügung. Die Kommunikation kann synchron/verbindungsorientiert oder asynchron/-verbindungslos sein.

Definition des Servers:

```
use IO::Socket::UNIX;
my $server = IO::Socket::UNIX->new(
    Type => SOCK_STREAM,    # synchron
    Local => $SOCK_PATH,    # innerhalb des Servers
    Listen => 1,
);
die "Can't create socket: $!" unless $server;
```

Lesen/Schreiben von Daten am Socket:

```
while (my $conn = $server->accept())
{
        chomp (my $line = <$conn>);
        # Daten stehen in $line zur Verf"ugung
        print $conn "$result\n"; # Antwort
}
```

Definition des Clients:

```
use IO::Socket::UNIX;
my $client = IO::Socket::UNIX->new(
        Type => SOCK_STREAM,
        Peer => $SOCK_PATH,
);
die "Can't create socket: $!" unless $client;
```

Schreiben/Lesen von Daten am Socket:

```
$client->send("$data\n");
chomp (my $ans = <$client>);
```

Für die Kommunikation mit dem Internet steht die Klasse `IO::Socket::INET` zur Verfügung, die in ähnlicher Weise gnutzt wird.

Aufgabe:

Ergänzen Sie obiges Beispiel in der Weise, dass

- der Client den Anwender nach seinem Namen fragt,
- der Client den Namen an den Server schickt,
- der Server „Hallo" vor den Namen setzt und an den Client zurück gibt,
- der Client die Serverantwort ausgibt.

14 Weitere Aufgaben

Funktionsaufruf

Aufgabe:

Mit den nächsten Übungen soll Schritt für Schritt eine Bücherverwaltung für eine Bücherei erstellt werden. Führen Sie folgende Schritte aus:

- Legen Sie ein neues Projekt *Buecherei* an.
- Legen Sie die Datei `Medienverwaltung.pm` an.

- Schreiben Sie eine Funktion `addMedium`, die sechs Parameter nimmt:

  ```
  addMedium($signatur, $autor, $titel, $typ, $seitenzahl, $spieldauer)
  ```

- `addMedium` soll über die Verwendung von `daten_anhangen` die Daten an die Datei `medien.csv` kommasepariert anhängen.
- Rufen Sie die Prozedur aus dem Hauptprogramm (mit konstanten Werten) auf.

Datei / Regex

Aufgabe: Schreiben Sie im Modul `Medienverwaltung.pm` eine Funktion `isSignatureInFile($signatur)`, die

- die Datei `medien.csv` Zeile für Zeile ausliest,
- über einen regulären Ausdruck jede Zeile prüft, ob sie mit der übergebenen Signatur beginnt,
- als Ergebnis `1` rurückgibt, falls die Signatur bereits in der Datei vorhanden ist, sonst `0`.
- Testen Sie die Funktion.

Fehlerbehandlung

Aufgabe:

Ergänzen Sie die Medienverwaltung:

- In der Funktion `addMedium` um den Aufruf von `isSignatureInFile`. Falls die Signatur schon vorhanden ist, geben Sie einen entsprechenden Integer-Fehlercode an das Hauptprogramm zurück
- Werten Sie im Hauptprogramm die Fehlercodes aus.

Falls die Eingabedatei in `isSignatureInFile` nicht vorhanden ist, wirft das System eine Ausnahme. Damit lässt in diesem Fall über `addMedium` die Datei gar nicht mehr anlegen. Eine Möglichkeit wäre, das Öffnen der Datei in einen `eval`-Block zu packen. Dies gilt aber als schlechter Stil. Besser ist es, erst keine Ausnahmen zu provozieren:

- Ergänzen Sie die Funktion `isSignatureInFile` um eine Prüfung, ob die Datei vorhanden ist. Falls nein, geben Sie `0` („nicht vorganden") zurück.
- Schreiben Sie eine weitere Funktion `check_format`, die
  - `1` zurück gibt, falls die Datei nicht vorhanden ist oder alle Zeilen 5 Kommas enthalten.
  - `0` zurück gibt, falls ein einer Zeile keine 5 Kommas enthalten sind.

    Die „Kommaexistenz" prüfen Sie über einen reglären Ausdruck.

- Rufen Sie `check_format` in `addMedium` auf. Falls `0` zurückgegeben wird, brechen Sie die Bearbeitung mit `die` ab.
- Packen Sie im Hauptprogramm alle Aufrufe in einen `eval`-Block und bearbeiten Sie die entsprechenden Ausnahmen.

Objektorientierung

Aufgabe:

Ergänzen Sie die Medienverwaltung:

- Übernehmen Sie die Klasse `Medium` in das Bücherei-Projekt
- Ergänzen Sie `Medium` um eine Methode `format()`, die die Bestandteile kommasepariert zurück gibt.
- Schreiben Sie im Modul `Medienverwaltung` eine Funktion `addObj($fname, $obj)`, die die Datei `fname` zum Anhängen öffnet und das Ergebnis von `$obj->format()` anhängt.
- Prüfen Sie in `addMedium`, ob es sich beim ersten Parameter um eine Instanz von `Medium` handelt.
  - Falls ja, holen Sie sich über den *getter* die Signatur und rufen damit `isSignaturInFile`. Falls noch nicht vorhanden, fügen Sie das Medium über `addObj` der Datei hinzu.
  - Falls nein, verzweigen Sie in den bestehenden Code.
- Legen Sie im Hauptprogramm eine Instanz von Medium an und testen Sie damit den Code.

Vererbung

Aufgabe:

Ergänzen Sie die Bibliotheksanwendung:

- Legen Sie ein Modul `MediumHira.pm` an. Leiten Sie von der Klasse `MediumBase` die Klassen `Buch` und `CD` ab.
- Verteilen Sie die Attribute sinnvoll auf die Klassen. Schreiben Sie Konstruktoren (`new`-Methode), die die entsprechenden Attribute als Paramter übernimmt.
- Ergänzen Sie jede Klasse um eine Methode `format`, die den Datensatz als Zeichenkette in dem Format zurückgibt, wie er in der Eingabedatei erwartet wird.
- Testen Sie den Code durch direkte Verwendung im Hauptprogramm.

- Ergänzen Sie die Medienverwalatung um eine Funktion

  `addMediumHira(mediumBase)`,

  die nach einer Duplikatsprüfung die Instanz über `addObj` an die Datei anhängt.
- Verwenden Sie die Methode im Hauptprogramm, indem Sie einmal ein Buch, einmal eine CD übergeben.

CGI-Programmierung

Aufgabe:

- Schreiben Sie eine HTML-Seite `add_medium.html`, die ein Formular mit Eigabefeldern für die sechs Parameter von `Medium` enthält.
- Beim Absenden soll über einen `POST`-Request das CGI-Programm `add_medium.cgi` gerufen werden. Dieses packt die Daten aus, legt eine Instanz von Buch/CD an und ruft `addMediumHira` (s. Aufgabe „Vererbung").
- Je nach Ergebnis von `addMediumHira` soll eine entsprechende Information („Datensatz angelegt" / „Datensatz konnte nicht angelegt werden: ...") dem Benutzer angezeigt werden.

15 Quellen

Teich, Peter	Perl 5, Herdt Verlag, 1. Auflage, 2005
ST	Steyer, Ralph; Teich, Peter: Perl 5, Herdt Verlag, 2. Auflage, 2016