Perl Übungsaufgaben

zu

Perl Einführung

Aufgaben I : Strings

Ausgabe und Skalare Variablen : Strings

Serie 1

1. Schreiben Sie das "Hello World" Programm.
2. Die Ausgabe soll "Hello World" lauten (incl. Anführungszeichen), wie ist der Quelltext zu modifizieren ?
3. Weisen Sie einen String einer skalaren Variablen zu und lassen Sie sie ausgeben. Was ist die Eingabe, Verarbeitung und Ausgabe ? Welchen Vorteil bringt die Verwendung von Variablen ?
4. Definieren Sie mehrere skalare Variablen und lassen Sie sie ausgeben
5. Geben Sie die Strings in der Form *$str1 = mein_erster_string* aus. Dafür gibt es mehrere Lösungen, testen Sie die Ihnen bekannten.
6. Verwenden Sie Steuerzeichen, um die Ausgabe zu formatieren.
7. Lassen Sie sich Strings, die nur kleine Buchstaben enthalten, in Großbuchstaben ausgeben.
8. Verwenden Sie vor der Ausgabe die Konkatenation von Strings.

Serie 2

1. Lassen Sie sich außer den Strings auch noch deren Länge ausgeben.
2. Ermitteln Sie den String, der im Wort "Geltungsbereich" ab Position 8 enthalten ist.
3. Ermitteln Sie die Position des ersten Auftretens eines Strings innerhalb eines anderen. Bsp.: "ach" steht in "Apache" ab Pos. 2
4. Verwenden Sie den Wiederholungsoperator zum Formatieren, z.B. für die Erstellung einer Kopf-/Fußzeile aus dem "*"-Zeichen.
5. Verwenden Sie Ihnen bekannte Funktionen, um einen gesamten String groß- bzw. kleingeschrieben ausgeben zu lassen.

Serie 3

Geben Sie sich Variablen vor, die (mindestens) Namen, Vornamen und Wohnort mehrerer Personen enthalten.

1. Geben Sie mit *printf* die formatierten Daten aus.
2. Geben Sie die Daten in einem Satz aus (Frau VORNAME NAME wohnt in WOHNORT und ist ... und hat ...)
3. Übergeben Sie dem Programm einen Parameter, der für die Ausgabe verwertet wird. Z.B. Name des Sachbearbeiters, der die Adreßliste gerade bearbeitet, eine Versionsnummer o.ä.
4. Erweitern Sie Ihr Script um eigene Funktionalitäten.

Lösungen zu Aufgaben I

Serie 1

```
1. + 2. #!/usr/bin/perl
print "Hello World\n";
print "\"Hello World\"\n";
print q("Hello World" \n);
print q("Hello World" "\n");

3. #!/usr/bin/perl
$my_var = "Hello World";
$cr = "\n";

print $my_var,$cr;
print "$my_var$cr";
print '$my_var$cr';
print qq($my_var $cr);
print q($my_var $cr);

4. + 5. #!/usr/bin/perl
$v1 = "Herr Willibald Wunz";
$v2 = "Frau Lilli Lose";

print "\$v1 = $v1\n\$v2 = $v2\n";
print q($v1 = ), $v1, "\n";

6. #!/usr/bin/perl
$v1 = "Willibald Wunz";
$v2 = "Lilli Lose";
$v3 = "Hugo Hager";

print "Teilnehmerliste : \n\t$v1\n\t$v2\n\t$v3";

7. #!/usr/bin/perl
$v1 = "Willibald Wunz";
$v2 = "Lilli Lose";
$v3 = "Hugo Hager";
print "Teilnehmerliste : \n\t$v1\n\t", uc($v2), "\n\t$v3";

8. #!/usr/bin/perl
$v1 = "Willibald Wunz";$v1_ort = "Kiel";
$v2 = "Lilli Lose";$v2_ort = "Heiligenhafen";
$v3 = "Hugo Hager";$v3_ort = "Wien";

$ww = $v1.$v2_ort;
$v2 .= $v2_ort;
$hh = $v3." aus ".$v3_ort;
print "Teilnehmerliste : \n\t$ww\n\t$v2\n\t$hh";
```

Serie 2

```
1. #!/usr/bin/perl
$v1 = "Willibald Wunz";$v1_ort = "Kiel";
$v2 = "Lilli Lose";$v2_ort = "Heiligenhafen";
$v3 = "Hugo Hager";$v3_ort = "Wien";

$ww = $v1.$v2_ort;
$v2 .= $v2_ort;
$hh = $v3." aus ".$v3_ort;

print "\n$ww    : ", length($ww);
print "\n$v2    : ", length($v2);
print "\n$hh    : ", length($hh);


2. #!/usr/bin/perl
$str = "Geltungsbereich";
$str_ab_pos_acht = substr($str,8);
print $str_ab_pos_acht,"\n";

$str_pos_elf = substr($str,11,2);
print $str_pos_elf,"\n";
```

```
3. #!/usr/bin/perl
$str = "Apache";
$pos = index($str,"ach");
print $pos;
print "\nDer Suchstring befindet sich an Position $pos\n";

4. #!/usr/bin/perl
$s1 = "Matthias";
$s2 = "Walter";
$s3 = "Herbert";
$format = "*" x 10;
print $format,"\n";
print $format,"\n";
print "$s1\n$s2\n$s3\n";
print $format,"\n";
```

Serie 3

```
1. #!/usr/bin/perl
$v1 = "Herbert";$n1 = "Hastig";$w1 = "Hamburg";
$v2 = "Walter";$n2 = "Wichtig";$w2 = "Wien";
$format = "-" x 38;
printf("%12s %12s %12s\n","Vorname","Nachname","Wohnort");
print "$format\n";
printf("%12s %12s %12s\n",$v1,$n1,$w1);
printf("%12s %12s %12s\n",$v2,$n2,$w2);

2. #!/usr/bin/perl
$v1 = "Herbert";$n1 = "Hastig";$w1 = "Hamburg";
$v2 = "Walter";$n2 = "Wichtig";$w2 = "Wien";
printf("Herr %12s %12s wohnt in %12s\n",$v1,$n1,$w1);
printf("Herr %12s %12s wohnt in %12s\n",$v2,$n2,$w2);

3. #!/usr/bin/perl
$bearbeiter = $ARGV[0];
$zeit = localtime;
$v1 = "Herbert";$n1 = "Hastig";
$w1 = "Hamburg";$v2 = "Walter";
$n2 = "Wichtig";$w2 = "Wien";
$format = "-" x 38;
printf("%12s %12s %12s\n","Vorname","Nachname","Wohnort");
print "$format\n";
printf("%12s %12s %12s\n",$v1,$n1,$w1);
printf("%12s %12s %12s\n",$v2,$n2,$w2);
printf("\n\nListe bearbeitet von %10s am %20s\n",$bearbeiter,$zeit);

4. #!/usr/bin/perl
$bearbeiter = $ARGV[0];
$zeit = localtime;
$v1 = "Herbert";$n1 = "Hastig";
$w1 = "Hamburg";$v2 = "Walter";
$n2 = "Wichtig";$w2 = "Wien";
print "Geben Sie einen Vornamen ein :\n";
$v3 = <STDIN>;
chomp $v3;
print "Geben Sie einen Nachnamen ein :\n";
$n3 = <STDIN>;
chomp $n3;
print "Geben Sie einen Wohnort ein :\n";
chomp($w3 = <STDIN>);
$format = "-" x 38;
$ff = "\n" x 10;
printf("$ff%12s %12s %12s\n","Vorname","Nachname","Wohnort");
print "$format\n";
printf("%12s %12s %12s\n",$v1,$n1,$w1);
printf("%12s %12s %12s\n",$v2,$n2,$w2);
printf("%12s %12s %12s\n",$v3,$n3,$w3);
printf("\n\nListe bearbeitet von %10s am %20s\n",$bearbeiter,$zeit);
```

Aufgaben II : Bedingte Verzweigung

Kontrollstrukturen : Bedingte Verzweigung mit *if*

Serie 1

Welche Ausdrücke ergeben *false*, welche *true* ?

1. ("yes" eq "YES")
2. ("NO" eq "No")
3. ("Alfred" eq "\ualfred")
4. ("HUGO" eq uc("hugo"))
5. ("NO" ne "No")
6. ("hello" ne lc("HELLO"))

Bemerkung : Die Strings können auch in einfachen Anführungszeichen stehen.

Verifizieren Sie die Lösung durch den Schnelltest ***perl -e "Ausdruck"*** in der Kommandozeile. Welcher Rückgabewert steht für *true* ?

Serie 2

1. "Sichern" Sie Ihr Programm gegen unbefugtes Ausführen: Fragen Sie einen Wert ab (z.B. Paßwort oder Name). Lassen Sie das Programm mit einer Meldung abbrechen, wenn die Eingabe nicht korrekt war.
2. Ermöglichen Sie bei falscher Eingabe einen zweiten Versuch.
3. Machen Sie die Überprüfung unabhängig von Groß- oder Kleinschreibung: Ein Paßwort "Geheim" wird mit einer Eingabe "geheim" verglichen und die Ausführung des Programms erlaubt werden, wenn nur die Buchstabenfolge korrekt ist. Dabei soll Groß- oder Kleinschreibung keine Rolle spielen.
4. Lassen Sie von bestimmten Datensätzen (z.B. aus I.3.2) nur bestimmte Datensätze ausgeben.

Versuchen Sie, den Programmablauf graphisch darzustellen.

Serie 3 OPTIONAL

1. Verwenden Sie die >> ? : <<-Notation für eine Abfrage: Wenn eine Bedingung erfüllt ist, soll "ja" ausgegeben werden, sonst "nein".
2. Verwenden Sie die Kurzform der *if*-Abfrage: Abhängig von einem Programmparameter soll eine bestimmte Variable gesetzt werden. Beispiel: Eingabe "V" ==> $str =

Lösungen zu Aufgaben_II

Serie 2

```
##########################################
#               II.2.1                   #
##########################################
$pass = "GE";
chomp($in = <STDIN>);

if ( $in eq $pass ) {
       print "Passwort korrekt";
}
else {
       print "no Sir!\n";
       exit 0;
}

print "ab hier wird das Programm fortgesetzt...\n";

##########################################
# filename    d:/perl/work/II/II.2.1a.pl  #
##########################################

$pass = "GE";

if ( chomp($in=<STDIN>) eq $pass ) {
       print "Passwort korrekt";
}
else {
       print "no Sir!\n";
       exit 0;
}

print "ab hier wird das Programm fortgesetzt...\n";

##########################################
# filename    d:/perl/work/II/II.2.2.pl   #
##########################################

$pass = "GE";

print "Geben Sie Ihr Passwort ein :\n";
chomp($in = <STDIN>);

if ( $in eq $pass ) {
       print "Passwort korrekt\n";
}
else {
       print "Zweiter Versuch!\n";
       chomp($in = <STDIN>);
        if ( $in eq $pass ) {
              print "Passwort korrekt\n";
       }
       else {
              print "no Sir!\n";
              exit 0;
       }
}

print "ab hier wird das Programm fortgesetzt...\n";
```

```
##########################################
# filename    d:/perl/work/II/II.2.3.pl   #
##########################################

$pass = "GE";

print "Geben Sie Ihr Passwort ein :\n";
chomp($in = <STDIN>);

if ( lc($in) eq lc($pass) ) {
        print "Passwort korrekt\n";
}
else {
        print "no Sir !\n";
        exit 0;
}

print "ab hier wird das Programm fortgesetzt...\n";

##########################################
# filename    d:/perl/work/II/II.2.4.pl   #
##########################################
$sp1 = "Vorname";
$sp2 = "Nachname";
$sp3 = "Wohnort";

$v1 = "Herbert";
$n1 = "Hastig";
$w1 = "Hamburg";

$v2 = "Walter";
$n2 = "Wichtig";
$w2 = "Wien";

$v3 = "Frieda";
$n3 = "Froelich";
$w3 = "Freiburg";
#
# Suchbegriff einlesen
#
print "Welcher Datensatz soll angezeigt werden (Nachname)?\n";
$in = <STDIN>;
chomp($in);
#
# Tabellenkopf
#
printf("\n\n\n%12s %12s %12s\n",$sp1,$sp2,$sp3);
print "-" x 38,"\n";
#
# "Suche" des Datensatzes
#
if ($in eq "Hastig") {
        printf("%12s %12s %12s\n",$v1,$n1,$w1);
} elsif ($in eq "Wichtig") {
        printf("%12s %12s %12s\n",$v2,$n2,$w2);
} elsif ($in eq "Froelich") {
        printf("%12s %12s %12s\n",$v3,$n3,$w3);
}

else {
                print "Haben wir nicht";
}
print "\n\n\n";
```

Serie 3

```
( 7 == 8 ) ? print ”ja“ : print ”nein“;
$erg = ( ”Test“ eq ”test“) ? ”gleich“ : ”ungleich“;
$zahl = 7 if ( ”$antwort“ eq ”ok“)
```

Aufgaben III : Zahlen (Optional)

Ausgabe und Skalare Variablen : Zahlen

Serie 1

Arbeiten Sie nur mit ganzen Zahlen

1. Eine Zahl wird eingelesen und zu 5 addiert.
   Die Ausgabe hat folgendes Format (Eingabe 7) : **5 + 7 = 12**
2. Zwei Zahlen werden abgefragt und die Summe ausgegeben.
3. Testen Sie Ihr Programm kurz mit den anderen Rechenoperationen
4. Die einegebene Zahl soll maximal zweistellig sein. Überprüfen Sie die Eingabe und reagieren Sie entsprechend. Eine solche Überprüfung nennt man Plausibilitätskontrolle (ugs. Plausi)

Serie 2

1. Erstellen Sie eine Tabelle, die für mehrere Zahlen die Hälfte, ein Drittel und das Quadrat dieser Zahl ausgibt. dieser Zahl ausgibt.

Zahl	Zahl/2	Zahl/3	Zahl*Zahl
2	1	0.666666	4
...	...	...	...
...	...	...	...

Serie 3

1. Schreiben Sie ein Programm, das Ihnen eine DM-Eingabe in Euro konvertiert. Hinweis : 1 EUR = 1.95583 DEM
2. Auch die Konvertierung von Euro in DM soll möglich sein.
3. Schreiben Sie ein Programm, das Ihnen die Auswahl einer Rechenoperation (Addition, Subtraktion, Multiplikation, Division) ermöglicht (Menu), dann zwei Zahlen einliest, die ausgewählte Rechenoperation durchführt und das Ergebnis ausgibt. Gehen Sie systematisch an die Aufgabe : Stellen Sie die Aufgabe in eigenen Worten (aufschreiben). Zerlegen Sie die Aufgabe in Teilaufgaben. Klären Sie Fragen. Erzeugen Sie sich einen Pseudocode. Visualisieren Sie den Programmablauf Schreiben Sie sich auf, welche Sprachbestandteile von Perl sie benötigen. Kennen Sie diese Funktionen, Ausdrücke etc., oder kann man mit ihnen noch mehr machen ? Schreiben sie sich auf, welche Variablen benötigt werden, überlegen Sie sich eine sinnvolle Bezeichnung der Variablen. Entwerfen Sie den Quellcode grob auf Papier, bevor Sie die Aufgabe realisieren.
4. Testen Sie das Programm und ergänzen es um sinnvolle Kommentare.

Aufgaben IV : Schleifen

Kontrollstrukturen : Schleifen mit *for* und *while*

Serie 1

1. Lassen Sie sich die Quadratzahlen von 1 bis 20 ausgeben.
   Realisieren Sie die Aufgabe einmal mit *for*, einmal mit *while*.
2. Start und Endwert der Tabelle sollen variabel gehalten werden.
3. OPTIONAL : Lassen Sie sich das kleine 1x1 in Tabellenform ausgeben
4. OPTIONAL : Betrachten Sie III.2.1. Erweitern Sie das Programm, so daß die Tabelle für beliebige Werte ausgegeben wird.

Serie 2

1. Überprüfen Sie eine Eingabe auf Richtigkeit und fragen Sie *solange* nach, bis eine richtige Eingabe erfolgt.
2. OPTIONAL : Betrachten Sie III.3.1 (Eurokonverter). Geben Sie dem User die Möglichkeit, solange eine neue Konvertierung durchführen zu lassen, bis er eine bestimmte Eingabe macht.
3. OPTIONAL : Betrachten Sie II.2.4 (Bearbeitung der Adressdatenbank). Dem User soll eine wiederholte Abfrage von Datensätzen ermöglicht werden, solange die Eingabe nicht "0" lautet.

Serie 3

1. OPTIONAL : Nach Auswahl einer Rechenoperation kann ein User zwei Operanden eingeben, das Ergebnis wird ausgegeben. Dem User wird angeboten, dieselbe Operation, aber mit anderen Zahlen durchzuführen. Überlegen Sie sich einen Lösungsansatz.
   Gibt es im Programm ähnliche/identische Programmblöcke ?

2. Programmieren Sie das Spiel HiLo : Eine Eingabe wird mit einer Zufallszahl verglichen. Es erscheint eine Meldung, ob die Zahl zu groß oder zu klein war und die Abfrage beginnt von vorne. Wird die Zahl richtig geraten, so wird die Anzahl der Versuche ausgegeben.
   Hinweis : mit *$zufall = rand(10)* erhalten Sie eine Zufallszahl zwischen 0 und 10,
   wobei 0 <= $zufall < 10
   Veranschaulichen Sie sich zunächst den Programmablauf durch einen Ablaufplan, *bevor* Sie programmieren !

3. OPTIONAL : Durch Eingabeumlenkung auf Shellebene kann z.B. eine Datei auf STDIN gelenkt werden. Ein Perlscript kann dann diese Daten in einer Schleife verwerten. Lenken sie z.B. die /etc/services auf ein Script, das Ihnen diese Datei mit Zeilennummerierung

ausgibt. "Suchen" Sie nach einem bestimmten Eintrag: lassen Sie sich nur eine Zeile mit einem bestimmten Eintrag ausgeben.

Lösungen zu Aufgaben IV : Schleifen

```
#!/usr/bin/perl
#
# Aufgaben IV.1.1
#
print "While oder For ? (w/f) : ";
chomp($wof = <STDIN>);

if ($wof eq "w") {
        $i = 1;
        while ($i <= 20) {
                printf("%5d%5d\n",$i,$i*$i);
                $i++;
        }
}
else {
        for ($i=1;$i<=20;$i++) {
                printf("%5d%5d\n",$i,$i*$i);
        }
}               print "While oder For ? (w/f) : ";
chomp($wof = <STDIN>);

if ($wof eq "w") {
        $i = 1;
        while ($i <= 20) {
                printf("%5d%5d\n",$i,$i*$i);
                $i++;
        }
}
else {
        for ($i=1;$i<=20;$i++) {
                printf("%5d%5d\n",$i,$i*$i);
        }
}

#!/usr/bin/perl
#
# Aufgaben IV.1.2
#
print "While oder For ? (w/f) : ";chomp($wof = <STDIN>);
print "Startwert : ";chomp($min = <STDIN>);
print "Endwert   : ";chomp($max = <STDIN>);

if ($wof eq "w") {
        while ($min <= $max) {
                printf("%5d%5d\n",$min,$min*$min);
                $min++;
        }
}
else {
        for ($i=$min;$i<=$max;$i++) {
                printf("%5d%5d\n",$i,$i*$i);
        }
} print "While oder For ? (w/f) : ";
chomp($wof = <STDIN>);
print "Startwert : ";
chomp($min = <STDIN>);
print "Endwert   : ";
chomp($max = <STDIN>);

if ($wof eq "w") {
        while ($min <= $max) {
                printf("%5d%5d\n",$min,$min*$min);
                $min++;
        }
}
else {
        for ($i=$min;$i<=$max;$i++) {
```

```
               printf("%5d%5d\n",$i,$i*$i);  }
}


#!/usr/bin/perl
#
# Aufgaben IV.1.3


print "Startwert : ";chomp($min = <STDIN>);
print "\nEndwert   : ";chomp($max = <STDIN>);

$lf = "\n" x 20;

printf("$lf%10s%10s%10s%10s\n","Zahl","Zahl/2","Zahl/3","Zahl*Zahl");
print "-" x 44,"\n";
for ($i=$min;$i <= $max;$i++) {
   printf("%10d%10.2f%10f%10d\n",$i,$i/2,$i/3,$i*$i);
}

#!/usr/bin/perl
#
# Aufgaben IV.1.3
print "Startwert : ";chomp($min = <STDIN>);
print "\nEndwert   : ";chomp($max = <STDIN>);

$lf = "\n" x 20;printf("$lf%10s%10s%10s%10s\n","Zahl","Zahl/2","Zahl/3","Zahl*Zahl");
print "-" x 44,"\n";

for ($i=$min;$i <= $max;$i++) {
printf("%10d%10.2f%10f%10d\n",$i,$i/2,$i/3,$i*$i);
}#!/usr/bin/perl
#
# Aufgaben IV.2.1
$pass = "GEHEIM";
print "\nEingabe : ";
chomp($in = <STDIN>);
while  ($in ne $pass){
       print "\nEingabe : ";
       chomp($in = <STDIN>);
} ;
print "OK\n";$pass = "GEHEIM";
print "\nEingabe : ";
chomp($in = <STDIN>);
while  ($in ne $pass){

print "\nEingabe : ";

chomp($in = <STDIN>);

} ;

print "OK\n";

#!/usr/bin/perl
#
# Aufgaben IV.2.2
#
do {
    print "\n" x 25;
    print "Konvertierung DEM ==> EUR\n\n";
    print "\tIhr Wert in DEM : ";
    chomp ($in=<STDIN>);

    printf("\tIhr Wert in EUR : %-10.2f\n",$in/1.95583);

    print "\nNochmal ? (y/n) ";
    chomp($nochmal=<STDIN>);
} while ( $nochmal eq "y" );

#!/usr/bin/perl
#
# Aufgaben IV.2.3
#
```

```
$sp1 = "Vorname";$sp2 = "Nachname";$sp3 = "Wohnort";
$v1 = "Herbert";$n1 = "Hastig";$w1 = "Hamburg";
$v2 = "Walter";$n2 = "Wichtig";$w2 = "Wien";
$v3 = "Frieda";$n3 = "Froelich";$w3 = "Freiburg";
#
$nochmal = "y";
# Suchbegriff einlesen

while ($nochmal eq "y") {

    print "Welcher Datensatz soll angezeigt werden (Nachname)?\n";

   $in = <STDIN>;
    chomp($in);
    #
    # Tabellenkopf
    #
    printf("\n\n\n%12s %12s %12s\n",$sp1,$sp2,$sp3);
    print "-" x 38,"\n";
    #
    # "Suche" des Datensatzes
    #
    if ($in eq "Hastig") {
       printf("%12s %12s %12s\n",$v1,$n1,$w1);
    }
    elsif ($in eq "Wichtig") {
       printf("%12s %12s %12s\n",$v2,$n2,$w2);
    }
    elsif ($in eq "Froelich") {
       printf("%12s %12s %12s\n",$v3,$n3,$w3);
    }

    else { print "Haben wir nicht !\n";
    }

    print "\n\nNochmal ? (y/n) : ";
    chomp($nochmal = <STDIN>);
}
print "\n\n\n";

#!/usr/bin/perl
#
# Aufgaben IV.3.1
#
print "\n" x 25;

print "Willkommen !\n\n";
print "Waehlen Sie eine Rechenoperation :\n\n";
printf("%10s%2s%20s\n"," ",1,"Addition");
printf("%10s%2s%20s\n"," ",2,"Subtraktion");
printf("%10s%2s%20s\n"," ",3,"Multiplikation");
printf("%10s%2s%20s\n"," ",4,"Division");
printf("%10s%2s%20s\n"," ",5,"Abbruch");

print "Ihre Wahl : ";
chomp( $wahl = <STDIN> );

die "Abbruch erwuenscht ! bye...\n" if $wahl == 5;

if ( $wahl == 1 ) {
    do {
       print "Erster  Operator :";chomp( $z1 = <STDIN> );
       print "Zweiter Operator :";chomp( $z2 = <STDIN> );
        print $z1 + $z2;
        print "\nNochmal ? (y/n) : ";chomp( $noch = <STDIN> );
    } while ( $noch eq "y");
}
elsif ( $wahl == 2 ) {
    do {
       print "Erster  Operator :";chomp( $z1 = <STDIN> );
       print "Zweiter Operator :";chomp( $z2 = <STDIN> );
       print $z1 - $z2;
        print "\nNochmal ? (y/n) : ";chomp( $noch = <STDIN> );
    } while ( $noch eq "y");
```

```
}
elsif ( $wahl == 3 ) {
    do {
        print "Erster  Operator :";chomp( $z1 = <STDIN> );
        print "Zweiter Operator :";chomp( $z2 = <STDIN> );
        print $z1 * $z2;
         print "\nNochmal ? (y/n) : ";chomp( $noch = <STDIN> );
    } while ( $noch eq "y");
}
elsif ( $wahl == 4 ) {
    do {
        print "Erster  Operator :";chomp( $z1 = <STDIN> );
        print "Zweiter Operator :";chomp( $z2 = <STDIN> );
        print $z1 / $z2;
         print "\nNochmal ? (y/n) : ";chomp( $noch = <STDIN> );
    } while ( $noch eq "y");
}
else { print "Leider die falsche Wahl !\n"  }
```

```
#!/usr/bin/perl
#############################################################
# Programm : HiLo
#
# Datei    : IV/IV.3.2.pl
#
# Funktion : Eine Zahl 0<=ZAHL<10 soll geraten werden.
#            Die Anzahl der Versuche wird ausgegeben.
#
# Programmierer : Michael Kalinka
# email         : michael.kalinka@gmx.de
#
# Version  : 0.1
#
# Status   : lauffaehig, stabil, ausbaufaehig
#
# Datum    : 23.4.2001
#
#############################################################
### Solange Spiel beginnen, bis Abbruch erwuenscht wird
do {
        ### Initialisierung der Variablen
        $zufall = int(rand(10));
        $count = 0;
        $in = 1000;

        ### Solange raten, bis die Loesung gefunden wird

        do {
                ### Einlesen der geratenen Zahl
                chomp($in = <STDIN>);

                ### Zaehlen der Versuche
                $count++;

                ### Pruefen der Eingabe
                if ($in > $zufall){
                        print "Zu hoch\n";
                }
                elsif ($in < $zufall) {
                        print "Zu niedrig\n";
                }
        } while ($zufall != $in);

        ### Ausgabe der Anzahl von Versuchen $count
        print "Gewonnen nach $count Versuchen!\n";

        ### Spielwiederholung ermoeglichen
        print "Nochmal ? (y/n) : ";
        chomp( $nochmal = <STDIN> );
} while ($nochmal eq "y");
```

Aufgaben V : Listen und Arrays

Serie 1

1. Lesen Sie Namen von STDIN ein und lassen Sie sie sortiert ausgeben.
2. OPTIONAL : Lassen Sie sich nur die Namen mit "A" am Anfang ausgeben.
   Hinweis : `if ( $str =~ /^A/ ) {...}` , *wenn am Anfang ein A ...*
3. Lesen Sie über STDIN Daten ein und erzeugen Sie Datensätze.
   Hinweis : Datensatz z.B. in der Form ”Name,Vorname,Wohnort“
4. Erzeugen Sie eine Ausgabe von Datensätzen in Tabellenformat.

Serie 2

1. Definieren Sie sich Kommaseparierte Datensätze (”Name,Vorname,Wohnort,...“).
   Der Name sei eindeutig.
   Lassen Sie sich die Datensätze z.B. nach dem Nachnamen sortiert ausgeben:
   Zuerst erzeugen Sie ein Array mit den sortierten Nachnamen,
   danach (!) durchlaufen Sie diese Liste und suchen den entsprechenden Datensatz, um ihn auszugeben.
2. Lesen Sie eine Datensatzdatei ein und wandeln den Datensatztrenner um (Komma zu Semikolon) und speichern die Datei.
   Hinweis :
   ```
   ### Datei zum Lesen öffnen ”<$filename“,
   open FH,“<$filename“ or die ”Cannot open $filename\n“;
   chomp ( @lines = <FH> );
   close FH;
   ...
   ### zum Schreiben öffnen ”>$outfile“
   open OUT,“>$outfile“ or die ”Cannot open $filename\n“;
   print OUT ”text zum Schreiben in die Datei\n“;
   ...
   close OUT;
   ```
3. OPTIONAL : Lesen Sie Datensätze aus einer Datei ein. Erweitern Sie den Datensatz um zwei Felder, z.B. Mailadresse und Telephonnummer. Sichern Sie die Daten in einer Datei.
4. OPTIONAL : Ermitteln Sie aus der /etc/passwd alle Usernamen und deren Homeverzichnisse, deren UserID größer 500 ist. In welchen Gruppen sind diese User ?

Serie 3 OPTIONAL

1. Lesen Sie Datensätze aus einer Datei ein. Geben Sie dem Anwender die Möglichkeit, nach bestimmten Feldern zu suchen.
2. Schreiben Sie sich ein Script zur Wartung der Datensatzdatei. Es soll die Eingabe von

weiteren Datensätzen ermöglichen und diese an das Ende der Datei anfügen.

3. Überprüfen Sie, ob sich ein User auf Ihrem System unter anderer Userkennung angemeldet hat.

Lösungen zu Aufgaben V

Serie 1

```
r/bin/perl
#
# VI.1.1.pl
#
print "Namenseingabe ( Abbruch : ^D )\n";
push(@names,$_) while <>;
#chomp @names;
print sort @names;
```

```
#!/usr/bin/perl
#
# V.1.2.pl
#
print "Namenseingabe ( Abbruch : ^D )\n";
push(@names,$_) while <>;
#chomp @names;
foreach (sort @names) { print if /^A/ };
```

```
#!/usr/bin/perl

# V.1.3.pl
#
$data_sep = ":";
do {
    print "Nachname :\n";chomp($nn = <>);
    print "Vorname  :\n";chomp($vn = <>);
    push(@data,join($data_sep,$vn,$nn));

    print "Weiter : [RETURN] Stop : n";
    chomp($weiter=<>);
} while ($weiter ne "n");

print join("\n",@data);
```

```
#!/usr/bin/perl
#
# V.1.4.pl
#
$data_sep = ":";

do {
    print "Nachname :\n";chomp($nn = <>);
    print "Vorname  :\n";chomp($vn = <>);
    push(@data,join($data_sep,$vn,$nn));
print "Weiter=[RETURN] Stop=n : ";chomp($weiter=<>);
} while ($weiter ne "n");

foreach $dataset (@data) {
    foreach (split($data_sep,$dataset)) { printf "%10s",$_ }
    print "\n";
}
```

Serie 2

```
#!/usr/bin/perl
# Aufgaben V.2.1.pl
#

$sep = ","; # Datensatz-Feldtrenner
$index = 1; # Dort steht der Name
```

```
open (DB,"<db.txt") || die "Cannot $!\n";
chomp(@file = <DB>);
close DB;

# Liste der Nachnamen erzeugen
foreach (@file) {
    push(@nn,(split($sep,$_))[$index]);
}


 Suche und Ausgabe
#
 Durchlaufen aller sortierten Nachnamen
foreach $nn (sort @nn) {
    # Durchlaufen aller Datensaetze
    foreach (@file) {
        # Trefferausgabe
       print $_,"\n" if (split($sep,$_))[$index] eq $nn
    }
}


#!/usr/bin/perl
#
# Aufgaben V.2.2.pl
#
$db_in_file,$db_in_sep) = ("db.txt",",");
($db_out_file,$db_out_sep) = ("db_out.txt",":");
open (DBIN,"<$db_in_file") || die "Cannot open $db_in_file$!\n";
@file = <DBIN>; # hier ohne chomp
close DBIN;

open (DBOUT,">$db_out_file") || die "Cannot open $db_out_file $!\n";
foreach (@file) {
    @items = split($db_in_sep,$_);
    print DBOUT join($db_out_sep,@items);
}
close DBOUT;


#!/usr/bin/perl -w
#
# Aufgaben V.2.3.pl
#
($db_in_file,$db_in_sep) = ("db.txt",",");
($db_out_file,$db_out_sep) = ("db_out_new.txt",":");

open (DBIN,"<$db_in_file") || die "Cannot open $db_in_file$!\n";
chomp(@file = <DBIN>);
close DBIN;


open (DBOUT,">$db_out_file") || die "Cannot open $db_out_file $!\n";
foreach (@file) {
    @items = split($db_in_sep,$_);
    print "mail for $items[1]: ";chomp($mail=<>);
    print "mob  for $items[1]: ";chomp($mob=<>);
    splice(@items,3,0,($mail,$mob));
    print DBOUT join($db_out_sep,@items),"\n";
}
close DBOUT;


#!/usr/bin/perl
#
# Aufgaben V.2.4a.pl
#
($pw_file,$pw_sep) = ("/etc/passwd",":");
$gr_file = "/etc/group";
$uid_min = 500;

pen (PW,"<$pw_file") || die "Cannot open $pw_file $!\n";
chomp(@file = <PW>);
```

```
close PW;

open (GROUP,"<$gr_file") || die "Cannot open $gr_file $!\n";
chomp(@group = <GROUP>);
close GROUP;

foreach (@file) {
    @pw_items = split($pw_sep,$_);
    if ($pw_items[2] >= $uid_min) {
       $groups = &string_of_groups($pw_items[0],@group);
       printf "%-10s%-20s%s\n",$pw_items[0],$pw_items[5],$groups;
    }
}

print "done ...\n";

sub string_of_groups {
    my $user = shift;
    my @group = @_;
    my $gr_sep = ":";

    foreach (@group) {
        @colums = split($gr_sep,$_);
        if ( index($colums[3],$user) != -1 ) {
            push (@groups_of_user,$colums[0]);
       }
    }

    return join(" ",@groups_of_user);
}



#!/usr/bin/perl -w
#
# Aufgaben V.2.4.pl
#
($db_in_file,$db_in_sep) = ("/etc/passwd",":");
$uid_min = 500;

open (DBIN,"<$db_in_file") || die "Cannot open $db_in_file$!\n";
chomp(@file = <DBIN>);
close DBIN;

foreach (@file) {
    @items = split($db_in_sep,$_);
    if ($items[2] >= $uid_min) {
       printf "%-10s%-20s\n",$items[0],$items[5];
    }
}
print "done ...\n";
```