Einführung in die Computerlinguistik – Einführung in Perl (1)

Dozentin: Wiebke Petersen

26.11.2009

Compiler

"Ein Compiler (auch Übersetzer oder Kompilierer genannt) ist ein Computerprogramm, das ein in einer Quellsprache geschriebenes Programm – genannt Quellprogramm – in ein semantisch äquivalentes Programm einer Zielsprache (Zielprogramm) umwandelt.
Üblicherweise handelt es sich dabei um die Übersetzung eines von einem Programmierer in einer Programmiersprache geschriebenen Quelltextes in Assemblersprache, Bytecode oder Maschinensprache.
Das Übersetzen eines Quellprogramms in ein Zielprogramm durch einen Compiler wird auch als Kompilierung bezeichnet." (Aus `http://de.wikipedia.org/wiki/Compiler`)

Interpreter

"Ein Interpreter (im Sinne der Softwaretechnik) ist ein Computerprogramm, das einen Programm-Quellcode im Gegensatz zu Assemblern oder Compilern nicht in eine auf dem System direkt ausführbare Datei umwandelt, sondern den Quellcode einliest, analysiert und ausführt. Die Analyse des Quellcodes erfolgt also zur Laufzeit des Programms." (Aus `http://de.wikipedia.org/wiki/Interpreter`)

Perl – Geschichte und mehr

- **Perl** – **p**ractical **e**xtraction and **r**eport **l**anguage
- 1987 entwickelt von Larry Wall
- Ziel: Verknüpfung klassischer Programmierbefehle (Schleifen, if-then-else, ...) und regulärer Ausdrücke
- Perl steht zwischen reinen Interpreter- und reinen Compiler-Sprachen: Programme werden vorkompiliert in Bytecode, der dann interpretiert wird. Beides wird von sogenannten Perl-Interpretern gemacht.
- Perl-Interpreter (freeware) für Windows: active perl `http://www.activestate.com/activeperl/`
- kurzes Tutorial: `http://www.ssw.uni-linz.ac.at/Teaching/Lectures/Sem/2000/Katzmayr/index.html`
- empfehlenswertes Buch
  Schwartz et. al.: *Einführung in Perl*. O'Reilly, 5. Auflage, 2009.

Schreiben von Perl-Programmen

- Perl-Programme können mit jedem beliebigen Texteditor geschrieben werden
- Speichern mit Endung "`.pl`" (Beispiel "`script.pl`")
- Ausführen von der Kommandozeile mit "`perl script.pl`"

Schreibe ein "Hello world!" Programm

```
1  #!perl -w
2  # a very simple program printing "Hello world!"
3
4  print "Hello world!";
```

- Kommentarzeichen: "#" (alles was in der Zeile wird nicht als zum Programm gehörig interpretiert)
- Kommentare sind wichtig, sie helfen den Code zu verstehen
- Gewöhnen Sie sich an, sorgfältig zu kommentieren!
- "`print`" ist eine Anweisung
- "`"Hello world!"`" ist das Argument von "`print`"
- jede Anweisung bekommt eine eigene Zeile!
- jede Anweisungszeile wird mit Semikolon abgeschlossen!

Ausführen eines Programms (unter Windows)

- Öffnen Sie ein Dos-Fenster:
  - Start-> Ausführen -> cmd.exe
- Wechseln Sie zu dem Ordner in dem das Perlprogramm liegt (Beispiel: `D:\Perl\my_scripts`):
  - wechseln zum Directory D: "`D:`"
  - wechseln in den Unterordner "`my_scripts`":
    `cd \Perl\my_scripts`
- Ausführen des Programms (Beispiel: "`world.pl`"):
  `perl world.pl`

Übungseinheit (1)

1. Erstellen Sie einen Unterordner für Ihre Perl-Programme
2. Öffnen Sie einen Texteditor (z.B. Textpad)
3. Schreiben Sie ein "Hello world!" Programm
4. Nicht vergessen zu speichern (Unter Textpad unbedingt "Zeichensatz=utf-8" wählen)
5. Testen Sie das Programm
6. Sollte Ihr Programm einmal nicht von selbst stoppen, so können Sie es jederzeit mit `Strg c` abbrechen (Steuerung/control C)

Variablen

```
1  #!perl -w
2  # variable declaration:
3  my $fruit1 = "apples";
4  my $fruit2 = "plums";
5  my $amount1 = 5;
6  my $amount2 = 10;
7  my $sum= $amount1 + $amount2;
8  # creating output:
9  print "$amount1 $fruit1 and $amount2 $fruit2 are $sum fruits\n";
```

Erst mit Variablen werden Programme variabel!

- Variablen werden mit dem Dollarzeichen markiert
- Varibalennamen bestehen aus Buchstaben, Ziffern und dem Unterstrich (keine Leerzeichen)
- Groß- und Kleinschreibung wird unterschieden!
- Verwenden Sie *bedeutungsvolle* Bezeichnungen
- Variablen können Werte zugeordnet werden:
  - Zahlen
  - Strings/Zeichenketten (in Anführungszeichen eingeschlossen)

Ändern von Variabelbelegungen

```
1  #!perl -w
2  # variable declaration:
3  my $fruit1 = "apples";
4  my $fruit2 = "plums";
5  my $amount1 = 5;
6  my $amount2 = 10;
7  my $sum= $amount1 + $amount2;
8  # creating output:
9  print "$amount1 $fruit1 and $amount2 $fruit2 are $sum fruits\n";
10 print "The juice could be called $fruit1"."-$fruit2-juice.\n";
```

- numerische Operationen: $a=$a+4, $a=3-$a $a=$a*4, $a=3/$a
- Konkatenation von Strings: "Haus"."bau", "Haus".$b

Basis In- und Output

Output

- **print** "text␣$a"
- **print** "text".$a+$b."text"
- \n: neue Zeile
- \t: Tabulator

Input

- $a=<STDIN> liest eine Eingabe von der Standardeingabe (Tastatur) und schreibt sie in die Variable $a
- **chomp**($a) entfernt das Newline-Zeichen aus $a

```
1  #! perl -w
2  # Der User muss wissen, was eingegeben werden soll
3  print "What␣is␣your␣name?\n";
4  $name=<STDIN>;
5  chomp $name;
6  print "Hello␣$name!\n";
```

Übungseinheit (2)

1. Schreiben Sie ein Programm, das den User nach dem Radius eines Kreises fragt und den Umfang und die Fläche des Kreises berechnet und ausgibt.
2. Testen Sie das Programm
3. Wenn Sie noch Zeit haben, erweitern Sie das Früchteprogramm um eine Eingabe durch den User

Kontrollstrukturen: if – else

```
1  #! perl -w
2  print "Please enter password: ";
3  $password = <STDIN>;
4  chomp $password;
5  if ($password == 42) {
6  print "Correct password! Welcome.";
7  } else {
8  print "Wrong password! Access denied.";
9  }
```

= ist der Belegungsoperator

== ist der numerische Vergleichsoperator (6 == 3*2)

eq ist der String-Vergleichsoperator ("momo" eq "mo"."mo")

Kontrollstrukturen: while

```
1  #! perl -w
2  print "Please␣enter␣password:␣";
3  $password = <>;
4  chomp($password);
5  while ($password != 42) {
6  print "Access␣denied.\n";
7  print "Please␣enter␣password:␣";
8  $password = <>;
9  chomp($password);
10 }
11 print "Correct␣password!␣Welcome.";
```

!= ist der Ungleich-Vergleichsoperator
(weitere Vergleichsoperatoren: < kleiner, <= kleiner oder gleich, > größer, >= größer oder gleich)

Übungseinheit (3)

1. Erweitern Sie ihr Kreisberechnungsprogramm: Verzichten Sie bei negativen Werten für den Radius auf die Berechnung und generieren Sie statt dessen eine Fehlermeldung für den User. (Zwei Programme, eins mit if/else, eins mit while)
2. Wenn Sie Zeit haben, können Sie den User auch fragen, ob er ein Quadrat oder einen Kreis berechnen möchte.