Universität Stuttgart – Institut für Computerphysik

Klausur

Computergrundlagen WS 2016/2017

Dr. Maria Fyta Dr. Jens Smiatek Johannes Zeman
Julian Michalowsky Dr. Frank Uhlig Patrick Kreissl
Kai Szuttor Frank Maier Evangelos Ribeiro Tzaras

7. März 2017

Name	
Vorname	
Matrikelnummer	

Hinweise

- Die Maximalpunktzahl ist 100.
- Der verfügbare freie Platz gibt einen Hinweis darauf, welchen Umfang die Lösung haben sollte.
- Die Klausur ist zu umfangreich um alle Themengebiete abdecken zu können. In der Regel wird es nicht möglich sein, alle Aufgaben vollständig zu bearbeiten. Bearbeiten Sie deswegen zuerst die Themengebiete, die Ihnen besonders liegen!
- Falls der Platz nicht ausreichen sollte, verwenden Sie zusätzliche Blätter. Beschriften Sie diese unbedingt mit Ihrem Namen und Ihrer Matrikelnummer!
- Einige Fragen ähneln den Fragen aus vorigen Klausuren, sind aber *nicht* identisch! Lesen Sie die Fragen deshalb bitte *genau* durch!

Viel Erfolg!

1 Unixgrundlagen (25 Punkte)

Aufgabe 1: (1 Punkt)

Beschreiben Sie was der Befehl `man man` bewirkt.

Antwort:

> Er zeigt die manpage zum Befehl man an.

Aufgabe 2: (1 Punkt)

Mit welchem Unix-Befehl kann man die Anzahl der Wörter in der Datei "gpl.txt" erhalten?

Antwort:

> `wc -w gpl.txt` oder einfach `wc gpl.txt`

Aufgabe 3: (2 Punkte)

Im aktuellen Verzeichnis befindet sich eine Datei "Filename with spaces.txt". Sie möchten den Inhalt der Datei in Ihrem Terminal mit Hilfe des `cat`-Befehls ausgeben und erhalten folgende Fehlermeldungen:

```
$ cat Filename with spaces.txt
cat: Filename: No such file or directory
cat: with: No such file or directory
cat: spaces.txt: No such file or directory
```

Nennen Sie 2 Möglichkeiten die Datei mit dem `cat`-Befehl erfolgreich auszugeben.

Antwort:

> `cat Filename\ with\ spaces.txt` oder
> `cat "Filename␣with␣spaces.txt"`

Aufgabe 4: (1 Punkt)

Was macht der folgende Shell-Befehl?

```
find ~ -size +100M
```

Antwort:

> Er listet alle Dateien im Heimatverzeichnis und in dessen Unterverzeichnissen auf, welche eine Dateigröße von mindestens 100 MiB aufweisen.

Aufgabe 5: (1 Punkt)
Beschreiben Sie die *Gesamtfunktion* des folgenden Shell-Befehls.

```
cd ~/Documents && ls -1 *.pdf | grep -v "Kontoauszug"
```

Antwort:

> Listet (wenn das Verzeichnis existiert) alle .pdf Dateien auf, die nicht "Kontoauszug" im Namen enthalten.

Aufgabe 6: (1 Punkt)
Mit welchem Shell-Befehl kann man zählen, in wievielen Zeilen das Wort *copyright* in der Datei "gpl.txt" vorkommt? Groß- und Kleinschreibung soll dabei *nicht* beachtet werden!
Antwort:

> `grep -i "copyright"gpl.txt | wc -l` oder
> `grep -ic copyright gpl.txt`

Aufgabe 7: (1 Punkt)
Was ist der Unterschied zwischen einem Terminal und einer Shell?
Antwort:

> Das Terminal ist die Schnittstelle der Shell und befasst sich mit der Ein- und Ausgabe, während die Shell die eigentlichen Befehle verarbeitet.

Aufgabe 8: (1 Punkt)
Was macht der Befehl `kate notizen.txt &` ?
Antwort:

> Startet den Editor kate im Hintergrund um die Datei "notizen.txt" zu bearbeiten.

Aufgabe 9: (2 Punkte)
Nennen Sie 2 Vorteile eines Kommandozeileninterfaces gegenüber grafischen Oberflächen.
Antwort:

> Man kann verschiedene Befehle miteinander kombinieren (Redirection und Pipes) um komplexe Aufgaben zu lösen. Grafische Oberflächen bieten meist im Vergleich zu CLI Varianten nur eine eingeschränkte Bedienung/Funktionalität. Automatisierung von Abläufen.

Aufgabe 10: (2 Punkte)
Nennen Sie 4 verschiedene Dienste, die im Internet verwendet werden.
Antwort:

> WWW(HTTP, HTTPS), Email(STMP, IMAP, POP3), SSH (SSH), Dateiübetragung(FTP, SFTP), Chat:IRC (IRC), XMPP/Jabber (XMPP), Instant-Messaging (proprietäre Protokolle wie zB ICQ, Skype) Peer-2-Peer (Torrent, emule, kazaa), Namensaufloesung (DNS), etc.

Aufgabe 11: (4 Punkte)

Markieren und korrigieren Sie die 4 Fehler im folgenden bash-Skript.

```
1  #!/bin/sh
2
3  BACK=~/.config/xfce4/desktop/backdrop.list #Leerzeichen vor =
4  echo "# unknown script" > $BACK
5
6  for dir in /home/user/Pictures/*; do #$ zu viel, Variable wird Wert zugewie
7    if test -d "$dir" -a -f "$dir/.background"; then #; then fehlt, anstelle
8      find "$dir" -mindepth 1 -maxdepth 1  >> $BACK #>> anstelle von >>>
9    fi
10 done
```

Aufgabe 12: (4 Punkte)

Beschreiben Sie zeilenweise, was das oben korrigierte bash-Skript macht.

Antwort:

1. Speichert den Pfad einer Datei in die Variable BACK ab 2. Schreibt in die Datei eine Ausgabe 3. geht die Auflistung in Verzeichnis Pictures durch 4. Ist der Eintag ein Verzeichnis und enthült dieses eine versteckte Datei namens .background 5. so füge den Inhalt dieses Verzeichnisses auf erster Ebene an die BACK

An einem Institutscomputer hat Benutzer `zeman` folgenden Dialog in der Shell:

```
> groups zeman kai cgl16-001
zeman : icp cgl video cpp pc
kai : icp asm sysguru stud video
cgl16-001 : cgl
> ls -la
total 8
drwxr-xrwx+ 5 zeman cgl 76  Nov 2 13:55 .
drwxr--r-x+ 3 zeman icp 17  Nov 2 12:01 ..
-rw-r-----  1 zeman cgl 441 Nov 2 12:01 bar.txt
dr-xrwxr-x+ 2 zeman cgl 6   Nov 2 12:01 cglstuff
-r-xrwxr--  1 zeman cgl 260 Nov 2 12:01 foo.txt
-rw-------  1 zeman icp 42  Nov 2 12:01 script.sh
```

Aufgabe 13: (2 Punkte)

Welche der Benutzer `zeman`, `kai` und `cgl16-001` können welche der folgenden Befehle erfolgreich ausführen?

```
cp bar.txt tender.txt
mv foo.txt fighters.txt
```

Antwort:

cp: read on file, write on directory level: zeman mv: read and write on directory: zeman und kai

Aufgabe 14: (2 Punkte)

Welchen Befehl muss `zeman` ausfüren, damit `zeman` die Datei `script.sh` ausfüren kann, aber niemand anderes?

Antwort:

`chmod u+x script.sh`

2 Python (20 Punkte)

Aufgabe 15: (2 Punkte)

Führt man den folgenden Python-Dreizeiler aus, so wird der Rückgabewert `False` auf der Standardausgabe ausgegeben. Warum? Welche Zahlenwerte enthalten die Variablen `a` und `b`?

```
a = 4*3/2
b = 3/2*4
print(a == b)
```

Antwort:

Integerdivision in Python ignoriert Nachkommastellen. Nachdem die Operatoren `*` und `/` gleichberechtigt sind, wird von vorne nach hinten gerechnet, d. h. $a = (4 \cdot 3)/2 = (12)/2 = 6$ und $b = (3/2) \cdot 4 = (1) \cdot 4 = 4$. Beide Variablen enthalten also verschiedene Werte, logischer Vergleich (`a == b`) ergibt daher `False` (Ungleichheit der Variablen).

Aufgabe 16: (4 Punkte)

Wenn sich N Leute treffen, ist die Wahrscheinlichkeit, dass *keiner* von ihnen am gleichen Tag Geburtstag hat wie einer der übrigen Anwesenden (unter Vernachlässigung von Schaltjahren, Zwillingen usw., sowie der Annahme, dass jeder Tag gleich wahrscheinlich ist):

$$P_{\text{notBd}} = \frac{365}{365} \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \ldots \cdot \frac{365 - (N-1)}{365}.$$

Schreiben Sie eine Python-Funktion, die für eine beliebige Anzahl Menschen $N > 0$ berechnet, wie hoch die Wahrscheinlichkeit ist, dass mindestens zwei von ihnen am gleichen Tag Geburtstag haben.

Antwort:

```
def p_same_birthday(n):
    p_different_bd = 1.
    for i in range(n):
        p_different_bd *= (365. - i)/365.
    return 1. - p_different_bd
```

Aufgabe 17: (5 Punkte)

Das untenstehende Python-Skript in dieser Aufgabe kann mit einer beliebigen natürliche Zahl $n > 0$ als Parameter aufgerufen werden. Ausgehend von $a_0 = n$ soll nach dem einfachen Bildungsgesetz

$$a_{n+1} = \begin{cases} a_n/2 & \text{falls } a_n \text{ gerade,} \\ 3a_n + 1 & \text{falls } a_n \text{ ungerade,} \end{cases} \tag{1}$$

eine Zahlenreihe konstruiert werden, wobei abgebrochen wird, sobald $a_n = 1$. Leider enthält das Skript 4 Syntax-Fehler. Korrigieren Sie diese.
Was geschieht, wenn man das korrigierte Skript mit dem Parameter $n = 0$ aufruft ("`python ./scriptname.py 0`")?

```
import sys

n = int(sys.argv[1])
collatz = []

while n != 1
    collatz.append(n)
    if n % 2 = 0:
        n /= 2
    else:
        n *= 3
        n++

collatz.append(1)

print(Collatz)
```

Antwort:

Syntaxfehler:

- Zuweisungs- statt Vergleichsoperator in `if`-Bedingung
- Doppelpunkt nach `while <cond>`
- C-Syntax `n++` vs. Python `n += 1`
- `Collatz` vs. `collatz`

Ruft man das Skript mit $n = 0$ auf, landet man in einer Endlosschleift, da `n != 1` sowie `0 % 2 == 0`

Hintergrundinformation: Die sogenannte "Collatz-Vermutung" besagt, dass jede Zahlenfolge, die nach obiger Definition gebildet wird, in den Zyklus 4, 2, 1 mündet – unabhängig davon, mit welcher natürlichen Zahl $n > 0$ begonnen wird. Die Vermutung konnte bis heute nicht bewiesen/widerlegt werden und gehört damit zu den ungelösten Problemen der Mathematik.

Aufgabe 18: (3 Punkte)

Welchen Zweck erfüllt die folgende Python-Funktion?

```
def function(z):
    r = True
    if z <= 1:
        r = False
    else:
        for i in range(2, z):
            if z % i == 0:
                r = False
                break
    return r
```

Was würde der folgende Aufruf obiger Funktion ausgeben?

```
print(function(7))
```

Antwort:

`function(z)` testet zunächst, ob $z > 1$ ist, und prüft dann für alle natürlichen Zahlen $i \in \{2, \ldots, z-1\}$ ob z ohne Rest durch diese teilbar ist. Kurz: Es wird getestet, ob z eine Primzahl ist und der entsprechende Wahrheitswert (`True`, wenn prim – `False`, wenn nicht) zurückgegeben.
Somit gibt der Aufruf `print(function(7))` den Rückgabewert `True` auf der Standardausgabe aus (7 ist eine Primzahl).

Aufgabe 19: (1 Punkt)

Die Funktion `mean` im Paket `numpy` kann genutzt werden, um den Mittelwert von (`numpy`-)Arrays zu berechnen. Sei `data` ein eindimensionales `numpy`-Array. Schreiben Sie ein Pythonprogramm, welches das `numpy`-Paket einbindet und den Mittelwert von `data` auf der Standardausgabe des Terminals ausgibt.

Antwort:

```
import numpy
print(numpy.mean(data))
```

Aufgabe 20: (1 Punkt)

Welchen prinzipiellen Vorteil hat es, `numpy`-Funktionen zu verwenden?

Antwort:

Das `numpy`-Paket is in C geschrieben und daher im Allgemeinen schneller als "pure Python". (Andere Antworten sind ebenfalls möglich, z.B. Ersparnis von Schreibarbeit ;-D)

Aufgabe 21: (4 Punkte)

Die Sequenz der Fibonacci-Zahlen ist wie folgt definiert:

$$\mathrm{fib}(n) = \begin{cases} 1 & \text{falls } n \leq 1, \\ \mathrm{fib}(n-1) + \mathrm{fib}(n-2) & \text{sonst.} \end{cases} \tag{2}$$

Als sogenannten "goldenen Schnitt" bezeichnet man das (irrationale) Teilungsverhältnis $\Phi = 1{,}6180339887\ldots$ Bildet man das Verhältnis zweier aufeinanderfolgender Fibonacci-Zahlen, so strebt das Ergebnis für ansteigende Werte von n gegen Φ:

$$\lim_{n\to\infty} \frac{\mathrm{fib}(n)}{\mathrm{fib}(n-1)} = \Phi. \tag{3}$$

Sei `fib(n)` eine bereits vorhandene Python-Funktion, die Gleichung (2) implementiert. Ergänzen Sie in folgendem Skript die `while`-Schleifte so, dass der Wert der Variable `n` jeweils um eins erhöht wird, bis sich zwei aufeinanderfolgende Approximationen von Φ (über den Quotienten in Gleichung (3)) um weniger als `max_deviation` unterscheiden. Hinweis: Berechnen Sie die ersten drei Quotienten, um zu erkennen, welche Abbruchbedingung für die `while`-Schleife nötig ist.

```
max_deviation = 1e-7
n = 2
previous = 1.*fib(n-1)/fib(n-2)
current = 1.*fib(n)/fib(n-1)

while ...


print("Genauigkeit erreicht bei n = {0}".format(n))
```

Antwort:

```
while abs(current - previous) >= max_deviation:
    n += 1
    previous = current
    current = 1.*fib(n)/fib(n-1)
```

3 C (10 Punkte)

Aufgabe 22: (2 Punkte)

Nennen Sie 2 konzeptionelle (d.h. nicht syntaktische) Unterschiede zwischen den Programmiersprachen PYTHON und C.

Antwort:

- interpretierte vs. kompilierte Sprache
- dynamische vs. statische Typen
- keine / explizite Speicherverwaltung
- objektorientierte Programmierung and mehr intrinsische Datentypen/-

Aufgabe 23: (4 Punkte)

Beschreiben Sie stichpunktartig die Funktionsweise des folgenden Programmcodes. Was wird ausgegeben, wenn er ausgeführt wird?

```
#include <stdio.h>

void multiply_1 ( double result, double a, double b ) {
  result = a * b;
}

void multiply_2 ( double *result, double a, double b ) {
  *result = a * b;
}

int main() {
  double result = 0.;

  multiply_1 ( result, 10, 23 );
  printf("%f\n", result);

  multiply_2 ( &result, 10, 23 );
  printf("%f\n", result);
}
```

Antwort:

- deklariert zwei Funktionen multiply_1/2, die als argument ein double/pointer, und zwei weitere doubles nehmen
- definiert main funktion, die eine Variable deklariert, und multiply_1/2 ausführt und den Wert in Result ausgibt.
- Ausgabe ist 0. und 230.

Aufgabe 24: (4 Punkte)

Finden und korrigieren Sie die 4 Fehler im untenstehenden C-Programmcode. Die Funktion `bubble_sort` sei in einer Bibliothek vorhanden, die bei Kompilierung verlinkt wird.

```
#include <stdio.h>

void bubble_sort(long [], long)

int main()
{
  long n = 10;
  long array[10] = { 23, 12, 55, 24, 174, 49, 31, 2, 10, 124 };
  long c, d, swap;

  buble_sort(array, n);

  printf("Sorted␣list␣in␣ascending␣order:\n");

  for ( c = 0 ; c < N ; c++ )
     printf("%ld\n", array[c]);

  return 0;
```

Antwort:

```
#include <stdio.h>

void bubble_sort(long [], long); // <-- missing ;

int main()
{
  long n = 10;
  long array[10] = { 23, 12, 55, 24, 174, 49, 31, 2, 10, 124 };
  long c, d, swap;

  bubble_sort(array, n); // <-- wrong function name

  printf("Sorted␣list␣in␣ascending␣order:\n");

  for ( c = 0 ; c < n ; c++ ) // <-- undeclared variable N
     printf("%ld\n", array[c]);

  return 0;
} // <-- missing braces
```

4 Algorithmen und Datenstrukturen (20 Punkte)

Aufgabe 25: (3 Punkte)

Beschreiben Sie den Zweck des folgenden Programms. Was wäre seine Ausgabe, wenn er mit einem aktuellen Compiler kompiliert und auf einem handelsüblichen Rechner mit x64-Architektur ausgeführt würde? Beachten Sie dabei, dass sich `int` und `unsigned int` dadurch voneinander unterscheiden, dass bei ersterem ein Bit für das Vorzeichen der Zahl reserviert ist.

```
#include <stdio.h>

int main () {
  signed int test = 1;
  unsigned int cnt = 1;

  while (test >= 0) {
    test *= 2;
    cnt++;
  }

  printf("%i\n", cnt);

  return 0;
}
```

Antwort:

- Bestimmung der Größe des Typs INT in Bits.
- Ausgabe: 32 (da INT und nicht LONG Datentyp)

Aufgabe 26: (2 Punkte)

Warum können in der Programmiersprache Python Berechnungen, die Arrays aus dem `numpy` Paket verwenden, im Allgemeinen schneller ausgeführt werden als Berechnungen, die herkömmliche Python Datencontainer (z. B. Listen) verwenden?

Antwort:

Explizite for-Loops können umgangen werden. Vektorisierte Berechnungen sind durch effizientere Arbeitsspeicherzugriffe (benachbarte Array-Elemente liegen auch im Arbeitsspeicher "nebeneinander") deutlich schneller. Hinzu kommt, dass viele mathematische Operationen auf `numpy` Arrays in C implementiert sind.

Aufgabe 27: (3 Punkte)

Beschreiben Sie den "Radix sort"-Algorithmus.

Antwort:

Radixsort ist ein Sortierverfahren für natürliche Zahlen, dass stellenweise vorgeht (Details für Interessierte: `http://de.wikipedia.org/wiki/Radixsort`). Bei diesem Verfahren werden die Zahlen zunächst nur entsprechend der niedrigsten Ziffer sortiert. Dazu werden zehn Listen, den zehn möglichen Ziffern entsprechend, erstellt, an die die Elemente in der Reihenfolge ihres Erscheinens angehängt werden. Anschließend werden diese Listen aneinandergehängt, so dass die mit einer Null endenden Zahlen zuerst kommen, dann die mit Eins endenden usw. Nun wird mit der nächstgrößeren Ziffer fortgefahren, also 10 neue Listen gebildet, an die die Elemente angehängt werden. Dabei ist wichtig, dass in der Erscheinensreihenfolge angehängt wird, da so die Sortierung der niedrigsten Stelle erhalten bleibt. Das Verfahren endet, wenn die größte vorhandene Ziffer in der Liste erreicht ist.

Aufgabe 28: (2 Punkte)

Implementieren Sie in einer Programmiersprache Ihrer Wahl (Python, C, eindeutiger Pseudocode) eine Funktion, die die Fakultät einer gegebenen natürlichen Zahl berechnet und zurückgibt. Nutzen Sie hierbei einen rekursiven Algorithmus (d.h. eine Funktion, die sich selbst aufruft).

Antwort:

```
def factorial(x):
  if x == 0:
    return 1
  else:
    return x * factorial(x - 1)
```

Aufgabe 29: (10 Punkte)

In der Mathematik und Physik hat die Exponentialfunktion eine große Bedeutung. Sie ist durch folgende Reihe definiert:

$$e^z = \sum_{k=0}^{\infty} \frac{z^k}{k!}. \tag{4}$$

Auch die trigonometrischen Funktionen lassen sich mit hilfe der Exponentialfunktion berechnen, wie zum Beispiel die Cosinus-Funktion:

$$\cos(x) = 0.5\left(\exp(ix) + \exp(-ix)\right). \tag{5}$$

Implementieren Sie in einer Programmiersprache Ihrer Wahl (Python, C, eindeutiger Pseudocode) eine Funktion `cos(x)`, die den Cosinus einer Zahl `x` berechnet und zurückgibt. Hierbei soll die Exponentialfunktion beliebig genau genähert werden können. Nehmen Sie desweiteren an, dass die Funktion zur Berechnung der Fakultät bereits implementiert ist.

Hinweis: Teilen Sie das Problem in einzelne Unterfunktionen auf.

Antwort:

```
def exponential(z, accuracy):
    err = 2.0 * accuracy
    res = 0.0
    old_res = 0.0
    k = 0
    while err > accuracy:
        old_res = res
        res += x ** k / factorial(k)
        k += 1
        err = abs(res - old_res)
    return res

def cos(x, accuracy):
    return 0.5 * (exponential(x*1.0j, accuracy) +
           exponential(-x*1.0j, accuracy))
```

5 LaTeX (15 Punkte)

Aufgabe 30: (5 Punkte)

Schreiben Sie den Teil eines LaTeX-Codes, der die folgende Tabelle erzeugt.
Anmerkung: Es soll lediglich die Tabelle erzeugt werden; ihre Positionierung im Dokument ist nicht Teil der Aufgabe.

Student	**Matrikelnummer**	**e-Mail**
Alice	*2709684*	alice@amail.com
Bob	*3994810*	bob@bmail.com

Antwort:

```
        \begin{tabular}{l|rr}
            \textbf{Student} & \textbf{Matrikelnummer}
& \textbf{e-Mail} \\\hline
            Alice            & \textit{2709684}
& alice@amail.com \\
            Bob              & \textit{3994810}
& bob@bmail.com
        \end{tabular}
```

Aufgabe 31: (3 Punkte)

Ergänzen Sie den folgenden Code so, dass die eingebettete Grafik eine Bildunterschrift erhält. Überlegen Sie sich, welche LaTeX Umgebung dafür geeignet ist. Sorgen Sie außerdem dafür, dass die Grafik horizontal zentriert im Dokument erscheint.
Anmerkung: Der Inhalt der Bildunterschrift ist unerheblich.

```
%%% Hier soll eine Grafik zentriert eingebettet werden %%%




\includegraphics[width=15cm]{./graphs/graph01}




%%%
```

Aufgabe 32: (1 Punkt)
Nennen Sie zwei LaTeX-Umgebungen, die für Aufzählungen geeignet sind. Erklären Sie kurz den Unterschied zwischen beiden Befehlen.
Antwort:

1. itemize: Liste ohne Nummerierung
2. enumerate: Liste mit Nummerierung

Aufgabe 33: (1 Punkt)
Nennen Sie jeweils eine konkrete Möglichkeit, die dargestellte Schrift in einem Dokument gegenüber der Standardgröße zu vergrößern und zu verkleinern.
Antwort:

`\Huge` und `tiny`

Aufgabe 34: (1 Punkt)
Sie möchten einen Brief in LaTeXschreiben. Nennen Sie eine geeignete Dokumentklasse.
Antwort:

`letter` oder `dinbrief`

Aufgabe 35: (1 Punkt)
Nennen Sie eine LaTeX-Umgebung, die das erstellen nummerierter Gleichungen erlaubt.
Antwort:

`equation` oder `align`

Aufgabe 36: (1 Punkt)
Beschreiben Sie eine Möglichkeit, Fußnoten in LaTeX zu erzeugen.
Antwort:

Mit `\footnote` oder `\footnotemark` und `\footnotetext`. Beispiel:

```
Hier entsteht eine Fu{\ss}note.\footnote{Und hier steht der F
```

Aufgabe 37: (2 Punkte)
Schreiben Sie den Teil eines LaTeX-Codes, der die folgende Ausgabe erzeugt:

$$\frac{z_1}{z_2} = \frac{|z_1|}{|z_2|} \cdot \exp(i(\phi_1 - \phi_2))$$

Antwort:

```
$\frac{z_1}{z_2} = \frac{|z_1|}{|z_2|} \cdot \exp (i (\phi_1
```

6 Visualisierung (10 Punkte)

Aufgabe 38: (1 Punkt)

Nennen Sie je ein Dateiformat, welches Bilder als Pixel- bzw. als Vektorgrafik speichert.

Antwort:

> PNG oder JPEG sind Pixelgrafikformate, EPS oder SVG Vektorgrafikformate.

Aufgabe 39: (1 Punkt)

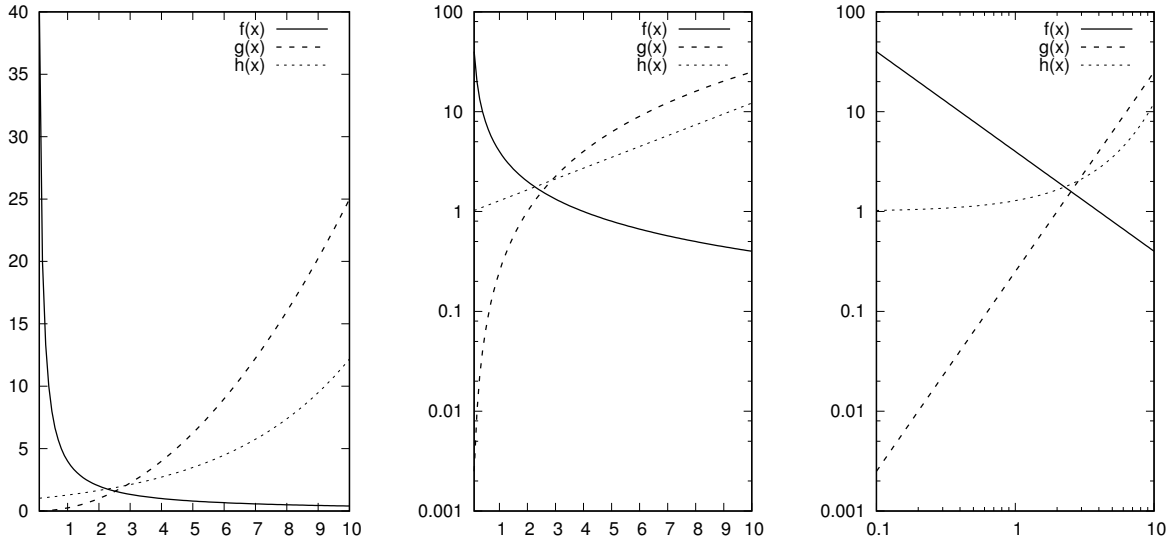
Was ist der Unterschied zwischen einer Vektorgrafik und einer Pixelgrafik? Wann ist eine Vektorgrafik zu bevorzugen?

Antwort:

> In Vektorgrafiken werden die Objekte über ihre Eigenschaften, z. B. Eckpunkte (Polygone), Parameter von Bezierkurven, etc. gespeichert. Pixelgrafiken hingegen haben eine feste Grösse und speichern nur die Farbinformationen eines jeden Pixels. Vektorgrafiken sind in der Regel vorzuziehen, wenn Bilder unterschiedlicher Auflösung scharfe Bilder präsentieren sollen.

Aufgabe 40: (3 Punkte)

Im folgenden sind drei Funktionen auf 2D-Wiedergabefläche unabhängig voneinander dargestellt. Geben Sie zu jeder Kurve eine Funktion an, die diese bestmöglich beschreibt. Parameter sind nicht verlangt! Beispiel: Beschreiben Sie eine vermeintlich logarithmische Funktion $f_1(x)$ nur mit $f_1(x) \propto \log(x)$.



Antwort:

> $f(x) \propto \frac{1}{x}$, $g(x) \propto x^2$, $h(x) \propto \exp(x)$

Aufgabe 41: (3 Punkte)

Beschreiben Sie, was die folgenden Gnuplot-Kommandos machen:

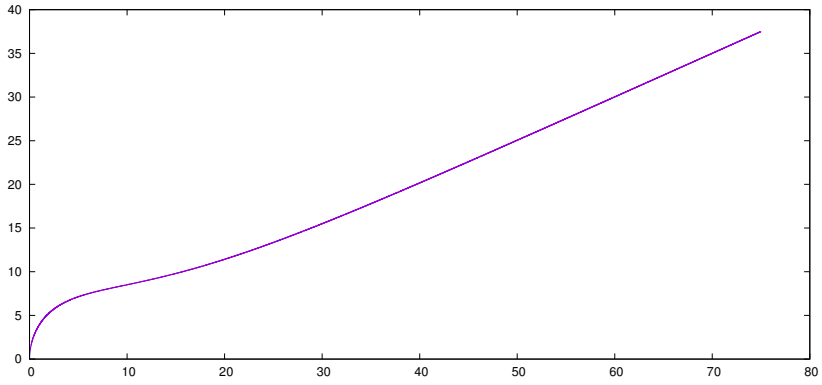
```
set xrange [0:20]
set xlabel "Zeit t [ms]"
set ylabel "Spannung U [V]"
set terminal postscript eps enhanced color
set output "daten.eps"
plot "data.dat" u 1:2:3 w yerrorbars w lp
```

Antwort:

- X-Achse geht von 0 bis 20
- X-Achsenbeschriftung
- Y-Achsenbeschriftung
- Ausgabe als farbiges EPS-Bild
- Festlegen des Dateinamens
- Plottet die Daten der Datei *data.dat* mit Y-Fehlerbalken und mit durch Linien verbundene Punkte

Aufgabe 42: (2 Punkte)

Der folgende Plot wurde durch den Gnuplot-Befehl `plot "data.dat" u 1:2` erstellt.



Wie man sieht, existiert ein lineares Regime im Langzeitverhalten der Kurve ($x > 40$). Beschreiben Sie, wie (mit welchen Kommandos) ein Fit für die Gerade gefunden werden kann. Geben Sie den Befehl an, mit dem die Daten zusammen mit dem Fit geplottet werden können. Geben Sie ausserdem den zwei Kurven einen Titel.

Antwort:

```
f(x)=a*x+b
fit [40:] f(x) 'data.dat' u 1:2 via a,b
plot 'data.dat' u 1:2 t "Data", f(x) t "Fit"
```