Dokumentation zum Paket *EXnSOL*

Joachim Wirth

18.01.2024

Inhaltsverzeichnis

2 Einführung

2.1 Über EXnSOL

Das Paket EXnSOL ist eine Sammlung von Makros und Umgebungen zum Erstellen von schriftlichen Leistungsüberprüfungen (Tests, …) oder von Übungen (Arbeitsblätter, …). *EXnSOL*, von »EX« für englisch ***ex****am* und ***ex****ercise*, das »n« als Kurzform von *»**an**d«* und *»SOL«* für ***sol****ution*. Es erlaubt die Kennzeichnung von Aufgaben mit Anforderungsbereichen, wie sie in Deutschland an Schulen praktiziert werden. Es gibt Umgebungen, die auch für Verbatim[1]-Inhalte geeignet sind.

2.2 Lizenz

2.3 Begriffsdefinitionen für diese Dokumentation

EX Von »*EX*« wie »*examen*« oder »*exercise*«

SOL Von »*SOL*« für »*solution*«.

Toggle Auf Deutsch heißt »*toggle*« Kippschalter. Gemeint ist eine boolesche Variable mit den zwei möglichen Zuständen, WAHR (1) oder FALSCH (0). Der Begriff stammt aus dem Paket *etoolbox*, das für EXnSOL eingesetzt wird. Mehr dazu in der etoolbox-Doku von Philipp Lehmann.

AFB »AFB« ist die Abkürzung für »Anforderungsbereiche«. Diese sind im deutschen Bildungssystem gebräuchlich. (AFB I = Reproduktion, AFB II = Transfer und Organisation von Wissen, AFB III = Urteilsfindung und Problemlösung).

[1] Verbatim bildet einen Quelltext exakt ab, damit lässt sich z.B. Programmiercode wiedergeben.

3 Voreinstellungen für die LaTeX-Datei

Die Makros und Umgebungen des Pakets EXnSOL können mit verschiedenen Dokumentenklassen kombiniert werden. Erprobt sind die Dokumentenklassen *article* oder *scrartcl*.

Das Paket wird mit **\usepackage [options]{ EXnSOL }** in der Präambel der Datei geladen.

Mögliche Optionen:

AFB Oder *afb*. Diese Option zeigt die AFB-Symbole im Druck an.

1.1 Gliederung von Aufgaben (1., 2., …) und Unteraufgaben (1.1, 1.2, …) im Dezimalsystem.

a) Unteraufgaben werden mit a), b), … gelabelt.

hidePoints Verhindert die Ausgabe der Punkte.

SOL Schaltet den Ausdruck auf die Kontrollversion um. Ohne diese Option ist die Prüfungsversion aktiv und die Lösungen fehlen im Druck. Automatisch wird dadurch der Druck von Lösungen an- und der Druck von Aufgabenergänzungen abgeschaltet.

4 Die Umschaltung von der Prüfungsversion zur Kontrollversion

Wie bei dem Klassiker *exam* von P. Hirschhorn ist es beim Paket EXnSOL von grundlegender Bedeutung, dass aus einer LaTeX-Quelldatei gewöhnlich zwei verschiedene Druckversionen generiert werden:

A) Die *Prüfungsversion*, ***solmode***=*FALSE*, zur Prüfung oder Übung, zur Bearbeitung von Schülern oder Studenten.

B) Die *Kontrollversion*, ***solmode*** = *TRUE*, entweder als Korrekturhilfe für den Lehrer, oder zur Selbstkontrolle der Lernenden. Für kompakte Kontrollversionen können auch Inhalte der Prüfungsversion ausgeblendet werden.

4.1 Wichtige Schalter

Umgesetzt wird die Zielsetzung, aus einer LaTeX-Quell-datei verschiedene Druckversionen zu kompilieren, mit Hilfe von booleschen Variablen, die den jeweiligen Status anzeigen, auch *Flags* genannt. Aus dem Paket *etoolbox* wurden dafür nützliche Makros übernommen. Die LaTeX-Flags heißen hier *Toggles* (»Kippschalter«). Mit dem Befehl **\toggletrue{<*toggle*>}** wird der Statusindikator auf 1 (TRUE) und mit **\togglefalse {<*toggle*>}** auf 0 (FALSE) gesetzt. Das kann sowohl in der Präambel, nach dem Einbinden der Pakete und vor \begin{*document*}... erfolgen, als auch im Dokumentenhauptteil. Für weitere Informationen zu den *Toggles* bitte die Original-Dokumentation »The etoolbox Package« von P. Lehmann und J. Wrigth (2020) heranziehen.

4.1.1 Wichtige Statusindikatoren (*Toggles*)

Im Folgenden eine kurze Auflistung der wichtigsten EXnSOL-***Toggles***:

solmode Übergeordneter Statusindikator. *FALSE* =die Prüfungsversion ist aktiv, Aufgabentexte werden angezeigt, Ergänzungen werden gezeigt. *TRUE* = die Kontrollversion ist aktiv, Aufgabentexte und Ergänzungen können unterdrückt werden.

showsol *FALSE* = Lösungen werden nicht angezeigt, *TRUE* = Lösungen werden angezeigt.

showsupple *FALSE* = Ergänzungen zur den Aufgaben werden nicht angezeigt, *TRUE* = Ergänzungen werden angezeigt.

showtasktext *FALSE* = Aufgabentexte werden nicht angezeigt (die Nummern schon!), *TRUE* = Aufgabentexte werden angezeigt.

showAFB *FALSE* = Anforderungsbereiche (AFB) werden nicht angezeigt, *TRUE* = AFB wird angezeigt.

showPoints *FALSE* = Aufgabenpunkte werden nicht angezeigt, *TRUE* = Punkte werden angezeigt.

Der Haupt-Statusindikator ***solmode*** lässt sich bereits als Option *[SOL]* bei der Paketeinbindung aktivieren. Oder über den Befehl **\toggletrue {*solmode*}**. Sein Initialzustand, ohne die Aktivierung per Option, ist *FALSE*,

das heißt die Lösungen werden aus- und die Aufgabenergänzungen (*supplements*) gedruckt. Zusätzlich gibt es noch die Möglichkeit über vordefinierte Makros umzuschalten.

4.2 Toggles überprüfen

0 (false)

Toggle	Meaning	Value
solmode	Output Mode with solutions.	0 (false)
showsol	Shows the solutions.	0 (false)
supple	Shows supplements.	**1 (true)**
showpoints	Shows points.	**1 (true)**
showafb	Shows requirement categories.	0 (false)
showtasktext	Shows text of tasks/subtasks.	**1 (true)**
bonuspoints	Switch Bonuspoints on.	0 (false)
nototal	Switch off summation of points.	0 (false)
aftercounterreset	Indicates reset of counters.	0 (false)

4.2.1 Umschalt-Makros

\SOLmode Schaltet den Hauptstatusindikator auf die Kontrollversion um. Nicht nur der Toggle *solmode* wird auf TRUE gesetzt, sondern auch der Toggle *showsol* und der Toggle *showsupple* werden auf FALSE gesetzt. Dem Makro SOLmode kann optional in eckigen Klammern ein Argument übergeben werden. Hat das optionale Argument einen Wert (nicht leer), so wird der Aufgabentext in der Kontrollversion unterdrückt, z.B. durch `\SOLmode[X]`.

\EXmode Schaltet den Ausdruck in die Prüfungsversion (ohne Lösungen) um. Der Toggle *showsol* wird auf FALSE und die Toggle *showsupple* und *showtasktext* werden auf TRUE gesetzt.

\showSOL aktiviert die Anzeige der Lösungen für den Ausdruck.

\notshowSOL versteckt die Lösungen in der Druckfassung.

\showSUPPLE aktiviert die Aufgabenergänzungen für den Druck.

\notschowSUPPLE nimmt die Aufgabenergänzungen aus der Druckversion.

\showAFB zeigt die Kennzeichnung der Anforderungsbereiche hinter dem Aufgabenlabel. Setzt außerdem die Länge zwischen dem Aufgabenlabel und dem Aufgabentext auf einen größeren Wert (\bigtaskbodyindent und \bigsubtaskbodyindent).

\notshowAFB verbirgt die Kennzeichnung der Anforderungsbereiche hinter dem Aufgabenlabel. Setzt außerdem die Länge zwischen dem Aufgabenlabel und dem Aufgabentext auf einen kleineren Wert (\smalltaskbodyindent und \smallsubtaskbodyindent).

4.3 Makros zur Ausführung je nach Ausgabeversion

Für die von der Ausgabeversion abhängige Ausführung gibt es folgende vordefinierte Makros:

\ifEX{} Ausführen des Arguments nur im Examensmodus (***solmode***=FALSE).

\ifSOL{} Ausführen des Arguments nur im Kontrollmodus.

\ifEXelse{}{} Ausführen des ersten Arguments nur im Examensmodus, das zweite Argument im Kontrollmodus.

Beispiel: Seitenumbruch nur in der Prüfungsversion

```
\ifEX{\pagebreak}
```

5 Organisation der Kontrolle

6 Meta – Informationen über die Aufgaben

Gewöhnlich haben die Aufgaben einer schriftlichen Leistungsmessung oder eines Übungsblattes eine Überschrift und meist auch ein Feld, in das der Prüfling seinen Namen eintragen sollte.

6.1 Der Titel der EX

Es gibt zwei Makros um einen Titel zu setzen, der in der Lösungsversion anders aussieht als in der Examensversion.

Makro: `\setEXtitle [soltitle]{extitle}`

Das erste Makro **\setEXtitle** setzt die alternativen Titel, dieses Makro kann schon in der Präambel stehen, also vor \begin{*document*}…. Voreingestellt, für den optionalen Titel für die Lösungen ist \soltitle mit dem Text »Solutions«.

Makro: `\EXtitle [Modus]`

Das zweite Makro **\EXtitle** gibt diese Titel passend zum Status von *solmode* aus. Optional kann der Modus beeinflusst werden, wie der Titel für die Lösungen erzeugt wird.

Optionen für Aufbau des Titels über der Lösungsvariante:

- r Der Titel für die Aufgaben wird durch den Titel für die Lösungen ersetzt. (Voreinstellung.)
- SE Der Titel für die Lösungen wird aus \soltitle und \extitle zusammengesetzt. (Diese Zeichenketten können mit dem Befehl \setEXtitle festgelegt werden.)
- ES Der Titel für die Lösungen ist wie bei »SE« zusammengesetzt, aber in umgekehrter Reihenfolge, also zuerst \extitle und dann \soltitle.

Beispielbox 1: EXtitle

```
<MINTED>
```

Aufgaben

Lösungen zu den Aufgaben

6.1.1 Namensfeld für den Prüfling

Bei schriftlichen Leistungsüberprüfungen ist der Name des Prüflings meist das Erste, was der Prüfling auszufüllen hat. Das lässt sich auf verschiedene Weise in LaTeXumsetzen, z.B. mit `\makebox[<Laenge>]{\hrulefill}`.

Name: _____

Statt `\hrulefill` kann auch auf Makros aus dem Paket *xhfill* von H. Voß zugegriffen werden, das EXnSOL für die Lückentexte standardmäßig eingebunden hat. Damit lässt sich an der Linie noch mancher Feinschliff anwenden, z.B. eine Absenkung mit `\makebox[<Laenge>]{\xrfill[-0.6ex]{0.2pt}}`:

Name: _____

Mit dem Paket xhfill sind auch gepunktete und farbige Linien kein Problem (`\makebox[8cm]{\xdotfill[-0.6ex]{0.5pt}[darkgray]}`):

Name: .....

6.1.2 Informationen zur Anzahl der Aufgaben

Es gibt zwei Zähler die mit Aufgaben zu tun haben:

alltasks Zähler sowohl für Aufgaben, als auch Unteraufgaben, mit zugeordneten Punkten.

bonustasks Zähler für Bonus-Aufgaben.

6.1.3 Infos zu den Anforderungsbereichen

Man kann darüber informieren, wie viele Punkte auf die verschiedenen Anforderungsbereiche (AFBI/AFBII und AFB III) entfallen, wenn dies eingesetzt wird. Außerhalb vom deutschen Bildungswesen wird man vermutlich mit dem Begriff *Anforderungsbereich* nicht viel anfangen, dieses System kann auch ganz einfach zur Kennzeichnung verschiedener Schwierigkeitsgrade benützt werden.

Dafür gibt es die Zähler afbI, afbII und afbIII. Um die jeweilige Gesamtpunktzahl der verschiedenen Anforderungsbereiche berechnen zu lassen, kommt das Makro \total zum Einsatz: `\total{afbI}` liefert also z.B. die auf den Anforderungsbereich I entfallenden Punkte. Analog liefern `\total{afbII}` und `\total{afbIII}` die Punkte für die anderen beiden Anforderungsbereiche.

7 Das Punktemanagement

7.1 Aufgabenpunkte

Makro: `\pts[<Laenge>] {<Punkte>\}`

Dieses Makro verrechnet die Aufgabenpunkte und gibt sie aus. Es sollte unmittelbar auf ein Aufgaben-Makro folgen. Siehe Beschreibung dort. Intern wird auf zwei Makros für die beiden Aufgaben zugegriffen: \setPoints {<Punkte>} und \printPoints {<Punkte>}. Standardmäßig werden die Punkte im rechten Blattrand ausgegeben. Zur Korrektur der vertikalen Position gibt es das optionale Argument in eckigen Klammern, es verlangt eine Länge, also einen Zahlenwert mit einer LaTeX-Längeneinheit. Bei der Verwendung von \subpts, siehe Beschreibung unten, kann der Punktewert zwischen den geschweiften Klammern weggelassen werden, dann wird die Aufgabenpunktzahl durch Aufsummieren der mit *subpts* gesetzten Unterpunkte automatisch berechnet.

Beispiel

```
\EX{Erste Aufgabe mit 2 Verrechnungspunkten.} \pts{2}
\EX{Zweite Aufgabe mit 3 Verrechnungspunkten.} \pts{3}

Gesamtpunkte der Aufgaben: \total{totpts} \ptssuffix
```

1. Erste Aufgabe mit 2 Verrechnungspunkten. ..... / 2 *P.*

2. Zweite Aufgabe mit 3 Verrechnungspunkten. ..... / 3 *P.*

Gesamtpunkte der Aufgaben: ?? *P.*

7.2 Bonuspunkte

Makro: `\bonuspts[<Laenge>] {<Punkte>\}`

Dieses Makro verrechnet die Bonuspunkte und gibt sie aus. Es sollte unmittelbar auf ein Aufgaben-Makro folgen.Zur Korrektur der vertikalen Position gibt es das optionale Argument in eckigen Klammern, es verlangt eine Länge, also einen Zahlenwert mit einer LaTeX-Längeneinheit. Bei der Verwendung

von \subpts, siehe Beschreibung unten, kann der Punktewert zwischen den geschweiften Klammern weggelassen werden, dann wird die Aufgabenpunktzahl durch Aufsummieren der mit *subpts* gesetzten Unterpunkte automatisch berechnet.

7.3 Teilpunkte im Aufgabentext

Makro: `\subpts{<Punkte>}`

Im Aufgabentext kann mit dem Makro `\subpts{}` innerhalb des Aufgabentextes detailliert angezeigt werden, für welche Aufgabenteile es wie viele Teilpunkte gibt. Diese Teilpunkte werden automatisch aufsummiert und als Aufgabenpunkte ausgegeben, wenn unmittelbar nach der Aufgabe mit Teilpunkten der Befehl `\pts{}` folgt, ohne einen Zahlenwert zwischen den geschweiften Klammern. Intern erfolgt die Verrechnung der Unterpunkte mit dem Makro `\setsubPoints {}`. Standardmäßig werden die Unterpunkte direkt in Klammern ausgegeben. Will man die Klammern nicht, so müssen die Makros `\subptsprefix` und `\subptssuffix` mit `\renewcommand` umdefiniert werden.

Beispiel:

```
\EX {%
  2x+2=0, x=?
  Lösungsweg \subpts{3}, Ergebnis \subpts{1}.}
  \pts{}
```

3. 2x+2=0, x=?

Lösungsweg (3 P.), Ergebnis (1 P.).

..... / 4 *P.*

7.4 Gesamtpunktzahl

Makro: `\totpts`

Damit die Gesamtpunktzahl auch im Anfangsteil eines Dokuments durch das Makro `\totpts` korrekt angezeigt wird, muss das TEX-Dokument mindestens zwei Mal compiliert werden.

Will man **Zwischensummen** der Aufgabenpunkte ausgeben, so lässt sich dies über `\arabic{totpts}` erreichen.

7.5 Gesamtbonuspunkte

Makro: `\bonustotpts`

Analog zur Gesamtpunktzahl kann mit diesem Makro die Gesamtbonuszahl ausgegeben werden.

7.6 Erwartungspunkte in den Lösungen

Makro: `\expp{<Zahl>}`

Mit den *Erwartungspunkten* wird ein Werkzeug bereitgestellt, das die Vergabe von Teilpunkten in der Musterlösung transparent macht und zur Kontrolle am Ende die Summe dieser Erwartungspunkte bildet. Eine Endsumme der Punkte wird angezeigt, sobald der Zähler *exppts* größer Null ist.

Beispiel:

Beispielbox 2: Erwartungspunkte

```
<MINTED>
```

F

r den Ansatz (1 P.), für den Lösungsweg (2 P.), für das Ergebnis (1 P.).

8 Makros für Aufgaben

Die Makroversion für Aufgaben und Unteraufgaben ist schlanker, als die Umgebungsversion, verträgt sich allerdings nicht mit Verbatim-Inhalten.

Wenn Unteraufgaben verwendet werden, bitte die Makros oder Umgebungen nicht schachteln, sondern hintereinander reihen. Damit die Bepunktung

richtig zugeordnet wird, sind die Bepunktungsmakros immer *hinter* den Aufgaben, im direkten Anschluss positionieren.

8.1 Eine Aufgabe \EX

```
\EX [<AFB>]{Aufgabentext.}
```

Optional, in eckigen Klammern, kann der Anforderungbereich (AFB) der Aufgabe angegeben werden. Als Eingabe für den AFB werden die römischen Ziffern I, II oder III akzeptiert.

Die Sternversion (`\EX*`) bzw. `\begin{EXv*)` unterdrückt die Aufgabennummernausgabe vor dem Aufgabentext. In der Sternversion wird auch nicht als »Item« einer Listenumgebung gesetzt, wie die ungesternte Version. Bei der Sternversion kann optional kein Anforderungsbereich angegeben werden.

Beispielbox 3: \EX* und \begin{*SOLv}...**

```
<MINTED>
```

a) 2+3=?

b) 12-1=?

Lösungen:

a) = 5 b) = 11

8.2 Aufgabe/Frage mit einem hervorgehobenen Titel

```
\titledEX [<AFB>]{Hervorgehoben}{Aufgabentext.}
```

Ähnlich wie \EX, nur dass hier ein Teil des Aufgabentextes hervorgehoben formatiert wird, dies ist der Inhalt zwischen den ersten geschweiften Klammern.

Die Sternversion unterdrückt die Aufgabennummernausgabe vor dem Aufgabentext. (Der interne Aufgabenzähler *NTask* wird aber um eins erhöht.)

8.3 Eine Unteraufgabe

`\subEX [<AFB>]{Aufgabentext.}`

Wie das Makro EX. Auch hier gibt es Sternversionen zur Unterdrückung der Aufgabennummernausgabe. (Der interne Unteraufgabenzähler *Nsubtask* wird aber um eins erhöht.)

Tipp bei falscher Nummerierung der Unteraufgaben: Beispielsweise bei der Verwendung vom Makro \EX innerhalb einer Minipage bleibt evtl. der Reset des Zählers für die Unteraufgaben aus und der Zähler wird von der letzten Aufgabe fortgeführt, statt wieder mit dem Anfangswert zu beginnen. In diesen Fällen kann der Befehl `\restartlist{subexenum}` Abhilfe schaffen.

8.4 Unteraufgabe mit einem hervorgehobenen Titel

`\titledsubEX [<AFB>]{Hervorgehoben}{Aufgabentext.}`

Ähnlich wie \subEX, nur dass hier ein Teil des Aufgabentextes hervorgehoben formatiert wird, dies ist der Inhalt zwischen den ersten geschweiften Klammern.

8.5 Lücken für Antworten

Länge automatisch, mit Unterstrich `\solorgap[<Laenge>]{Wort}]`

Länge automatisch, ohne Unterstrich `\solorgap*[<Laenge>]{Wort}]`

Länge fix, mit Unterstrich `\solorfixgap[<Laenge>]{Wort}]`

Länge fix, ohne Unterstrich `\solorfixgap*[<Laenge>]{Wort}]`

Länge optional `\SOL[<Laenge]{Wort(e)}`

Mit dem Makro \solorgap kann ein Wort in der EX-Version in eine Lücke verwandelt werden. Die Länge der Lücke wird vom Makro aus der Wortlänge berechnet. Das optionale Argument in eckigen Klammern ermöglicht die Anpassung der Lückenlänge durch ein Inkrement. Es sind auch negative Längen möglich, um die Lückenlänge zu verkürzen. Das Makro versucht auch zu vermeiden, dass überlange Lücken den rechten Textrand durchbrechen und im schlimmsten Fall sogar über den Seitenrand ragen. Die Sternvariante (\solorgap*) verzichtet auf den Unterstrich, ansonsten verhält es sich wie die sternlose Variante.

Will man keine unterschiedlich breiten Lücken, die mit ihrer Länge einen Hinweis auf das Lösungwort geben, so kann das Makro \solarfixgap verwendet werden. Das optionale Argument vom Typ Länge gibt an, wie breit die Lücke ist, in die geschweifte Klammer kommt als obligatorisches Argument das Lösungswort. Die Version mit Stern (\solorfixgap*) verzichtet wieder auf den Unterstrich.

Das einfachste Makro `\SOL*[<Länge]{Lösungswort}` gibt standardmäßig in der Prüfungsversion nichts aus, optional kann der Leerraum über die Länge eingestellt werden, was einfach im Makro durch `\hspace{length}` verwirklicht wird. In der Lösungsversion wird das Lösungswort ausgegeben. Dies eignet sich z.B. gut für Ausfüll-Tabellen.

Beispielbox 4: Ausfülltabelle mit \sol{}

<MINTED>

Deutsch	Englisch
Frosch	
	cat

Lösung:

Deutsch	Englisch
Frosch	frog
Katze	cat

Beispielbox 5: Alle Lücken-Makros

<MINTED>

$5 + 4 =$ __
$5 +$ __ $= 9$
$4 + 5 =$ __
$4 +$ __ $= 9$
$4 + 3 =$
$2 +$ __ $= 3$

Lösungen:
$5 + 4 =$ 9
$5 +$ 4 $= 9$
$4 + 5 =$ 9
$4 +$ = 9
$4 + 3 =$ 7
$2 +$ 2 $= 3$

8.6 Lückentext-Umgebungen

Paket *setspace*: `\begin{spacing} {Zeilenabstand} Lückentext. \end{spacing}`

Für die Erstellung von Lückentexten mit mehreren Zeilen empfiehlt es sich, zur Vergrößerung des Zeilenabstands, das Paket *setspace* von G. Tobin und R. Fairbains in der Präambel einzubinden.

```
\usepackage{setspace}
```

Beispiel:

Beispielbox 6: Lückentext mit Lücken

<MINTED>

Das Eichhörnchen hat einen langen _____ ,
dieser dient auch der _____ mit Artgenossen.

In der Kontrollversion:

Beispielbox 7: Lückentext mit Lösungen

<MINTED>

Das Eichhörnchen hat einen langen Schwanz ,
dieser dient auch der Kommunikation mit Artgenossen.

8.7 Finde-den-Fehler-Aufgaben

\textorsout[<n>]{<Text Zeigt in der Kontrollversion den *Text* durchgestrichen an, in der Farbe für Lösungen. Wird optional ein *n* in eckigen Klammern angegeben, so werden die Fehler nummeriert und diese Nummer hochgestellt hinter dem als falsch markierten Text angezeigt.

\reseterrornr Setzt den Zähler für die Fehlernummerierung auf Null zurück.

\steperrornr Erhöht den Zähler für die Fehler um 1. Dieses Makro wird automatisch von **\textorsout** aufgerufen.

Bei diesem Aufgabentyp wird ein Text mit absichtlichen Fehlern präsentiert und der Prüfling muss die Fehler finden und kennzeichnen. Es kann zudem verlangt werden, im Anschluss die Fehler zu korrigieren. In der Prüfungsversion (EX) wird ein Wort, oder ein Text, normal gesetzt angezeigt und in der Kontrollversion werden die falschen Wörter durchgestrichen in der Korrekturfarbe dargestellt. Außerdem kann optional hochgestellt eine Fehlernummer angezeigt werden, dies ermöglicht auf diese Nummer Bezug zu nehmen und den Korrekturtext zu liefern. Für die Darstellung als durchgestrichener Text wird das Paket *ulem* von Donald Arsenau verwendet.

Beispielbox 8: Finde-den-Fehler-Aufgabe

<MINTED>

Finde den Fehler:
Eisbären sind meist schwarz und haben sechs Beine.
Lösungen:
Eisbären sind meist ~~schwarz~~ [1] und haben ~~sechs~~ [2] Beine.

9 Auswahl-Aufgaben (Multiple Choice ...)

9.1 Auswahl-Option innerhalb einer Listen-Umgebung

Makro: `\mci[<x>]`

\mci steht für »Multiple-Choice-Item«. Optional kann mit einem »X« die richtige Wahl gekennzeichnet werden. Der Befehl \mci muss in einer Listen-Umgebung stehen, wie z.B. `\begin{itemize}`.

Beispiele:

Beispielbox 9

<MINTED>

Was heißt auf Spanisch Ja?

- ☒ si
- ☐ no

In der Kontrollversion (`\toggletrue{showsol}`) sieht das dann so aus:

Beispielbox 10

<MINTED>

Was heißt auf Spanisch Ja?

- ☒ si
- ☐ no

Horizontale Listen

EXnSOL bindet das Paket *enumitem* mit der Option »inline« ein. Somit kann aus einer vertikalen Liste ganz schnell auf eine horizontale Liste umgestellt werden, indem man dem Umgebungsnamen einen Stern anhängt, z.B. wird aus »description« »description*«.

Beispielbox 11

<MINTED>

Was heißt auf Spanisch Ja?
☐ si ☐ no

9.2 Auswahl-Option außerhalb von Listen

Makro: `\EXcheck[<x>]`

Dieses Makro verhält sich analog zu \mci, es zeichnet das Ankreuzkästchen aber ohne den LaTeX-Befehl \item und kann so außerhalb von Itimize-Umgebungen verwendet werden, z.B. in Tabellen:

Beispielbox 12: Auswahl-Option in Tabelle

<MINTED>

2+3 =	Wahr	Falsch
4	☐	☐
5	☐	☐

10 Grafikdatei mit Lösungs-Overlay

`\includegraphicsSOL}[grid]{\includegraphics-Befehl}{<TIKZ-Grafikbefehl(e)>}`

4. Kennzeichne in dem Röntgenbild die Elle mit einem fetten Kreuz!

11 Umgebungen für Aufgabenergänzungen und Materialien

Diese Umgebungen sollen nur in der EX-Version ausgegeben werden. Für die Kontrolle können Alternativinhalte definiert werden.

11.1 Aufgabenergänzung – Umgebung *supple*

```
\begin{SUPPLE} [Alternativinhalt für Kontrolle]
        Ergänzung zur Aufgabe.
\end{SUPPLE}
```

Bedingung für das Erscheinen oder Nichterscheinen des Inhalts der Umgebung ist das Toggle *showsupple*.

Soll die Ergänzung eingerückt werden, so kann die Umgebung »exSUPPLE« verwendet werden.

Beispiel:

```
\EX{Sein oder nicht Sein}

\begin{supplenum}
 \item Das ist hier die Frage!
\end{supplenum}
```

Ausgabe des Beispiels:

5. Sein oder nicht Sein –

Das ist hier die Frage!
Was passiert mit diesem Text?

6. Noch eine Variante.

11.2 Material – Umgebung *material*

Materialien sind Aufgabenergänzungen die üblicherweise nach dem Aufgabenteil gesetzt werden. Da sie nicht in unmittelbarer Nähe der Aufgabe sein müssen, sind Referenzverweise üblich. Die Materialien erhalten daher einen fortlaufend nummerierten Marker, z.B. M1, M2, ….. Für Referenzverweise muss innerhalb der Material-Umgebung ein \label gesetzt werden, z.B. `\label{mat:Text}`.

12 Musterlösungen für die Kontrollversion

12.1 Die Umgebung »SOLv*«

Die Umgebung *SOLv* eignet sich auch für Verbatim-Inhalte (um Quelltext exakt so abzubilden, wie eingegeben), wie folgendes Beispiel demonstriert.

Beispiel

```
 \toggletrue{showsol}

\EX{Mit welchem Macro bewirkt man in \LaTeX einen Fettdruck im Textmodus?}

\beginv}
   Mit dem Befehl \Verb|\textbf{}|.
\endv}
```

Ausgabe des Beispiels:

7. Mit welchem Macro bewirkt man in LaTeXeinen Fettdruck im Textmodus?

Mit dem Befehl `\textbf{}`. Dieser weitere Text soll nur zeigen, außerhalb von Verbatim-Inhalten, wie sich ein Zeilenumbruch auswirkt.
Hier beginnt eine weitere neue Zeile.

12.2 Das Macro \solution{}

Das Makro \solution{} erlaubt es platzsparende Musterlösungen in den Code einzufügen. Im Gegensatz zur Umgebung verträgt es sich aber nicht mit Verbatim-Inhalten, ist also weniger robust. Standardmäßig wird die Lösung eingerückt, wie der Aufgabentext.

12.3 Das Macro \SOL{}

Dieses einfache Macro ist z.B. für kurze Lösungen vermischt mit anderem Code, z.B. innerhalb von Tabellen.

Frage	Antwort
Warum ist die Banane krumm?	Wenn sie gerade wär', wäre es keine Banane mehr.

13 Antwortfelder

13.1 Grundlegendes zu den Antwortfeld-Makros

Alle Makros für Antwortfelder enden auf »field« und haben gemeinsam, dass diese Felder nur in der zu bearbeitenden Aufgabenversion (»EXmode«) erscheinen.

Alle Antwortfelder haben zwei Argumente, davon ist das erste für die horizontale Länge des Feldes (=Breite) optional, also in eckigen Klammern. Das zweite Argument in geschweiften Klammern bestimmt die vertikale Länge des Eingabebereichs (=Höhe). Wird das erste Argument weggelassen, so versucht das Makro die horizontale Länge an der Länge einer Textzeile auszurichten (\linewidth). Als Argument werden Zahlen ohne Einheiten erwartet. Mit diesen Zahlenwerten wird die Basislängeneinheit \baslu multipliziert. Voreingestellt ist für \baslu eine Länge von 5 Millimetern.

Beispiel: Ändern der Basislängeneinheit auf 0,1 Inch:

```
\setlength{\baslu}{0.1 in}
```

Kariertes Feld in 0,1-Inch-Einheiten.

Kariertes Feld in 5mm-Einheiten.

13.2 Leerraum mit \blankfield

Dies ist der Text über dem leeren Feld das mit `\blankfield[3]{2}` definiert wird:
Text vor dem Feld. Text hinter dem Feld. Text unter dem leeren Feld.

`\blankfield}[Breite]{Höhe}`

13.3 Umrahmtes Feld mit \framedfield

Beispiel:

`\framedfield[6]{2}`

Die Rahmenfarbe kann durch `\definecolor{framecolor}{rgb}{r,g,b}` geändert werden, z.B. ein grüner Rahmen durch `\definecolor{framecolor}{rgb}{0,1,0}`:

Rahmen mit gerundeten Ecken mit \rframedfield

Wird dem Makronamen ein »r« vorangestellt, also \rframedfield, so hat der Kasten gerundete Ecken.

Beispiel:

`\rframedfield[6]{3}`

13.4 Liniertes Feld mit \linedfield und \exlinedfield

Beispiel:

`\linedfield[6]{2}`

Die Linienfarbe kann mit `\definecolor{linecolor}{rgb}{r,g,b}` geändert werden, z.B. mit `\definecolor{linecolor}{rgb}{0.7,0.7,0.7}` graue Linien:

13.5 Kariertes Feld mit \gridfield

Beispiel:

```
\gridfield[6]{3}
```

Die Farbe der Karos wird mit dem Befehl `\definecolor{gridcolor}{rgb}{r,g,b}` geändert, z.B. für hellrosa `\definecolor{gridcolor}{rgb}{0.92,0.53,60.4}`

13.6 Feld mit Millimeterpapierhintergrund durch \graphfield

Beispiel:

```
\graphfield[8]{6}
```

Die Farben der Linien lassen sich wie folgt ändern:

Kästchen klein `\definecolor{smallgraphcolor}{rgb}{r,g,b}`

Kästchen mittel `\definecolor{mediumgraphcolor}{rgb}{r,g,b}`

Kästen groß `\definecolor{biggraphcolor}{rgb}{r,g,b}`

Beispiel:

Millimeterpapier mit himmelblauen Mittellinien:
`\definecolor{mediumgraphcolor}{RGB}{135,206,235}`

14 Robuste Umgebungen für Aufgaben, Unteraufgaben, Lösungen und Ergänzungen

- **EXv** (`\begin{EXv}[] …\end{EXv}`)
- **subEXv** (`\begin{subEXv}[] …\end{subEXv}`)
- **SOLv** (`\begin{SOLv}[] …\end{SOLv}`)
- **SUPv** (`\begin{SUPv}[] …\end{SUPv}`)

Die Makros für Aufgaben und Unteraufgaben vertragen sich meist nicht mit Makros und Umgebungen zur Anzeige von Programmiercode, wie z.B. das Makro \verb und die Verbatim-Umgebung. Wer Aufgaben mit solchen Code-Inhalten erstellen möchte, kann auf die robusten Umgebungen **EXv** und **subEXv** zurück greifen. Diese robusten Umgebungen sind allerdings (noch) nicht auf die Option des Ausblendens von Aufgabentext über den Toggle (showtasktext) vorbereitet.

Auch bei den Musterlösungen ist nur die Umgebung **SOLv** robust genug, für die Verwendung von Verbatim-Code. Die robusten Umgebungen haben ein kleines »v« am Ende des Namens.

8. Erste Makroaufgabe.

9.* Erste Umgebung Beispielaufgabe. Was bedeutet `\textbf{}`?

9.1 Beispielaufgabe. Was macht `$ x_{} $`?

9.2 Beispielaufgabe. Was macht `$ x_{} $`?

9.* Beispielaufgabe. Was bedeutet `\textbf{}`?

9. Eine Makroaufgabe.

9.1 Eine Makrounteraufgabe.

9.2 Eine Makrounteraufgabe.

10. Noch eine Makroaufgabe.

11.* Umgebung. Beispielaufgabe XY. Was bedeutet `\textbf{}`?

11.1 Beispielaufgabe. Was macht `$ x_{} $`?

11.2 Beispielaufgabe. Was macht `$ x_{} $`?

Die Ausgabe von `$ x_{} $` sieht so aus: x

15 Designanpassungen

Nicht nur die Geschmäcker sind verschieden, oft geben auch Institutionen ein bestimmtes Format der Aufgaben vor. Hier werden nun beispielhaft Designanpassungen aufgezeigt. Es sind sehr viel mehr Anpassungen möglich, dazu bitte die Quelldatei des Pakets studieren, je nach TEX-Kenntnisstand ist fast alles möglich.

15.1 Designanpassungen bei den Aufgabenpunkten

Vom Paket vordefiniert werden die Aufgabenpunkte an den rechten Rand gesetzt, wie zahlreiche Beispiele oben zeigen. Vorbereitet ist eine unterschiedliche Einrahmung der Punkte in Prüfungs- und Kontrollversion. In der Prüfungsversion ist eine punktierte Linie zum Eintragen der vom Prüfling erreichten Punktezahl, während in der Kontrolle die Punktezahl in runden Klammern am Rand steht.

15.1.1 Veränderung von Präfix und Suffix

Für die Veränderung der umrahmenden Zeichen der Punktezahl gibt es folgende Makros:

- `\ptsprefix`
- `\ptssuffix`
- `\ptsprefixsolmode`
- `\ptssuffixsolmode`

Angenommen, für das Eintragen der erreichten Puntkezahl soll statt der punktierten Linie ein Kästchen erscheinen und danach »von x P.«. Dann sind die beiden ersten Makros umzudefinieren:

```
\renewcommand{\ptsprefix}{\framebox(22,20){}\quad v.\thinspace}
\renewcommand{\ptssuffix}{\thinspace P.}
```

Das Ergebnis könnte dann so aussehen:

15.1.2 Veränderung der Position

Standardmäßig werden die Aufgabenpunkte in den rechten Rand gesetzt. Soll dies geändert werden, sollen z.B. die Punkte in Klammern vor den Aufgabentext gesetzt werden, so muss das Makro \pts umdefiniert werden. Die neue Definition sollte das Makro \setPoints aufrufen, damit die Punkte korrekt verrechnet werden. Das Beispiel könnte so umgesetzt werden:

```
\renewcommand{\ptsprefix}{()}
\renewcommand{\ptssuffix}{P.)\quad}
\renewcommand{\pts}[1]{\setPoints{#1}\printPoints[#1]}
```

Beispiel:

15.2 Formatierung der Aufgabenzähler

15.2.1 Änderungen der Aufgabenauszeichnung

Voreingestellt werden die Aufgaben mit arabischen Ziffern gefolgt von einem Punkt markiert. Z.B.:

11. Beispielaufgabe

Folgende Befehle können dies ändern:

\settaskprefix [< *Laenge* >] **{<Prefix>}** Damit kann dem Aufgabenzähler etwas vorangestellt werden. Das optionale Argument erlaubt einen Einzug in den üblichen Längeneinheiten, voreingestellt sind 0pt (kein Einzug).

\settasksuffix [< *Laenge* >] **{<Suffix>}** Damit kann dem Aufgabenzähler etwas angehängt werden. Das optionale Argument ist eine Länge, z.B. »1em«, die den Abstand vom Zählersuffix zum Aufgabentext angibt. Voreingestellt sind 0pt (kein zusätzlicher Abstand).

\settaskcounterstyle{<Style>} Damit kann der Aufgabenzähler mit den in LaTeX üblichen Stilarten formatiert werden:

arabic Der Standard. Zähler mit arabischen Zahlen: 1, 2, 3, ...

roman Zähler mit kleinen römischen Zahlen: i, ii, iii, ...

Roman Zähler mit kleinen römischen Zahlen: I, II, III, ...

alph Zähler mit kleinen römischen Zahlen: a, b, c, …

Alph Zähler mit kleinen römischen Zahlen: A, B, C, …

\settaskbodyindent {<length>} Das Makro nimmt als Argument eine Länge, welche die Einrückung (engl. *Indentation*) des Aufgabentextes festlegt.

Beispiel:

```
\settaskprefix{Aufgabe~}
\settaskcounterstyle{Roman}
\settasksuffix{:}
\EX{Beispiel mit römischem Zähler}
```

Ausgabe Beispiel:

Aufgabe XII: Beispiel mit römischem Zähler.

Das ist die nichtsagende Lösung!

Änderungen des Unteraufgabendesigns Für die Anpassung der Unteraufgabenauszeichnung existieren analoge Befehle wie zur Aufgabenauszeichnung (siehe oben):

\setsubtaskprefix [< *Laenge* >] **{<Prefix>}** Damit kann dem Unteraufgabenzähler etwas vorangestellt werden. Das optionale Argument erlaubt einen Einzug in den üblichen Längeneinheiten, voreingestellt sind 0pt (kein Einzug).

\setsubtasksuffix [< *Laenge* >] **{<Suffix>}** Damit kann dem Unteraufgabenzähler etwas angehängt werden.

\setsubtaskcounterstyle{<Style>} Damit kann der Unteraufgabenzähler mit den in LaTeX üblichen Stilarten formatiert werden:

arabic Der Standard. Zähler mit arabischen Zahlen: 1, 2, 3, …

roman Zähler mit kleinen römischen Zahlen: i, ii, iii, …

Roman Zähler mit kleinen römischen Zahlen: I, II, III, …

alph Zähler mit kleinen römischen Zahlen: a, b, c, …

Alph Zähler mit kleinen römischen Zahlen: A, B, C, …

\taskcounter Dieses Makro fügt die aktuelle Aufgabennummer in der aktuellen Formatierung ein. Dies erlaubt Aufgaben so zu nummerieren, wie man es von Gliederungen kennt, z.B.: 1.1, 1.2, 1.3, ….

\setsubtaskbodyindent {<length>} Das Makro nimmt als Argument eine Länge, welche die Einrückung (engl. *Indentation*) des Unteraufgabentextes fest legt.

Beispiel:

```
\settaskprefix{A}
\settaskcounterstyle{arabic}
\settasksuffix{}
\setsubtaskprefix{}
\setsubtaskcounterstyle{alph}
\setsubtasksuffix{)}
\EX{Beispiel mit Unteraufgaben.}
\subEX{Erste Sub.}
\subEX{Zweite Sub.}
```

Ausgabe Beispiel:

A13: Aufgabe mit Unteraufgaben.

A14: Aufgabe mit Unteraufgaben.

A15: Aufgabe mit Unteraufgaben.

A16: Aufgabe mit Unteraufgaben.

A17: Aufgabe mit Unteraufgaben.

A18: Aufgabe mit Unteraufgaben.

A19: Aufgabe mit Unteraufgaben.

A20: Aufgabe mit Unteraufgaben.

a) Erste Sub.

b) Zweite Sub.

Die Musterlösung ist rot.

15.3 Aufgabentitel neu gestalten

Angenommen, Sie möchten bei den Aufgaben mit Titeln, siehe **??**, den Titel in großen Buchstaben und in einer serifenlosen Schrift. Dann muss das Makro `\labeltitledtask` umdefiniert werden. Es sieht im Original wie folgt aus:

```
\newcommand{\formatquestiontitle}[1]{%
        \textbf{#1}\par  }
```

Die neue Definition könnte dann so aussehen:

```
\renewcommand{\formatquestiontitle}[1]{%
{\Large \textsf{#1}}\par  }
\titledEX{Titel}{Aufgabentext}
```

A21: Titel

Aufgabentext

15.4 Lösungen formatieren

Voreingestellt werden Lösungen durch die Farbe »solcolor« gekennzeichnet. Diese ist mit der Farbe »rot« vordefiniert. Weiterhin ist keine Überschrift über dem Musterlösungstext voreingestellt, das kann aber optional geändert werden.

Befehle zur Umformatierung der Lösungen:

\definecolorsolcolor{rgb}{1,0,0} Empfohlen wird »rgb« als Farbpalette im ersten Argument, das zweite Argument nimmt die drei Farbwerte zwischen 0 und 255 durch Kommata getrennt auf. Der Befehl wird durch das Paket *oPlotSymbl* von B. M. Döhring zur Verfügung gestellt und nutzt seinerseits *xcolor*.

\setsolutiontitle{<Solutiontitle>} Falls erwünscht, kann über der Musterlösung ein Titel angezeigt werden. Voreingestellt wird dieser in der normalen Schriftart in der Farbe »solcolor« dargestellt. Wird eine andere Formatierung gewünscht, so kann der Befehl zur Formatierung neu definiert werden: \renewcommand {\formatsolutiontitle}[1]{<NeueDefinition>}. Übergibt man dem Befehl \setsolutiontitle ein leeres Argument, so wird auch der Zeilenumbruch nach dem Titel nicht aktiviert, so dass Platz gespart wird.

16 Tipps zu EXnSOL innerhalb der Koma-scrartl-Klasse

Wie schon anfangs geschrieben, wurden bei diesem Paket bewusst die Makros und Umgebungen nicht in eine Dokumentenklasse gepackt. So bleibt es dem Anwender überlassen, welchen Dokumentenklasse-Überbau er für seine Aufgabenzusammenstellung wählt. Da ich diese Dokumentation auch für mich selbst zur Unterstützung meines immer schlechter werdenden Gedächtnisses schreibe, möchte ich hier noch einige Kniffe beschreiben, die sich auf Makros und Umgebungen beziehen, die nicht von mir sind, die aber für die Erstellung von Aufgabenblättern sehr nützlich sind.

Sowohl Arbeitsblätter, oder Tests, sind im Regelfall auf wenige Seiten Umfang beschränkt und werden meist in loser Form, also nicht gebunden, höchstens zusammen geklammert, verteilt. Dafür eignen sich die Klassen »article« und die von Markus Kohm an kontinentaleuropäische Vorlieben angepasste und erweiterte Klasse »scrartcl« gut. Ich bevorzuge meist die letztere Dokumentenklasse, wegen ihren vielen eingebauten Features, die nicht nur Anpassungen an deutsche Gepflogenheiten erleichtern, sondern auch Belange, die beim Erstellen von Arbeitsblättern und Tests von großem Nutzen sind. Andererseits scheint mir die ältere Klasse *article* solider und schlanker zu sein und durch Ergänzung mit passenden Paketen kann das Ergebnis ähnlich sein.

16.1 Gleiten oder nicht gleiten, das ist hier die Frage!

Eigentlich braucht man bei losen Kopien, mit wenigen Seiten Umfang, wie sie für Tests und Arbeitsblätter typisch sind, überhaupt keine Gleitumgebungen. Oft werden sie deshalb vom Nutzer durch den Layoutparameter »h« (hier) am Gleiten gehindert. Oder gar durch den Parameter »H« (*genau* hier), der das Paket *float* erfordert. Warum also ein Gleitobjekt benützen, wenn das Objekt nicht gleiten soll? Meist, weil die automatische Objektnummerierung, die einheitliche Betitelung und das typenbezogenes Referenzsystem erwünscht sind. Aber das geht auch ohne Gleitobjekte. (Ein anderer Grund ist Bequemlichkeit, so bringen viele LaTeX-Editoren Assistenten mit, die bequem eine Gleitumgebung für Grafiken setzen.)

\captionof {Objekttyp}{Langer Titel}

Bei der Klasse *scrartcl* bekommt man den Befehl \captionof {Objekttyp}{Langer Titel} gleich mitgeliefert. Optional kann diesem Makro noch ein Kurztitel für einen Verzeichniseintrag mitgegeben werden, aber das wird man bei Tests oder Arbeitsblättern selten benötigen. Bei der Standardklasse *article* lässt sich diese Funktionalität durch das Paket *capt-of* von R. Fairbains, oder das umfangreichere Paket *caption* von Axel Sommerfeldt nachrüsten. (Unter *scrartcl* ist die Verwendung dieser Pakete wegen Überschneidungen der Funktionalität weniger empfehlenswert.)

Layoutbeispiele mit \captionof

Klassisches zentriertes Layout mit Untertitel

```
\begin{minipage}{\linewidth}
 \centering
 \includegraphics[width=0.3\linewidth]{Bilder/Bildbeispiel}%
 \captionof{figure}{Landschaftsbild. \label{fig:bild1}}%
\end{minipage}
```

Abbildung 1: Landschaftsbild.

Die Minipage verhindert hier (Abb. **??**), dass die Bildunterschrift durch einen Seitenumbruch vom Bild getrennt wird. Die »Label-Funktion« scheint am zuverlässigsten zu funktionieren, wenn sie in den langen Titel von *captionof* integriert wird.

Fazit: Auch ohne die gleitende Umgebung *figure* muss nicht auf die automatische Nummerierung, die typische Formatierung und das Referenzsystem verzichtet werden.

Tabelle zentriert mit Übertitel

Beispielcode:

```
\begin{minipage}{\linewidth}
 \centering
 \captionof{table}
 {Eine Beispieltabelle \label{tab:Tabelle1}}
```

```
 \begin{tabular}{|c|c|}
  \hline
  Spalte 1 & Spalte 2 \\
  \hline
  A & B \\
  \hline
 \end{tabular}
 \label{tab:Tabelle1}
So sieht also nun Tabelle \ref{tab:Tabelle1} aus!
\end{minipage}
\bigskip
Tabelle \ref{tab:Tabelle1} ist hier ein nicht gleitendes Objekt.
```

Tabelle 1: Eine Beispieltabelle

Spalte 1	Spalte 2
A	B

Tabelle **??** ist hier ein nicht gleitendes Objekt. Der \bigskip nach der Minipage könnte umgangen werden, indem man die Minipage durch das optionale Argument für eine fixe Höhe entsprechend streckt.

Tabelle zentriert mit Untertitel

Beispielcode:

```
\begin{minipage}[c][5ex]{\linewidth}
 \centering
 \begin{tabular}{|c|c|}
  \hline
  Spalte 1 & Spalte 2 \\
  \hline
  A & B \\
  \hline
 \end{tabular}
 \captionof{table}
 {Eine Beispieltabelle \label{tab:Tabelle2}}
 \label{tab:Tabelle2}
\end{minipage}
\medskip
So sieht also nun Tabelle \ref{tab:Tabelle2} aus!
```

Ausgabe (Code-Beispiel):

Spalte 1	Spalte 2
A	B

Tabelle 2: Eine Beispieltabelle

So sieht also nun Tabelle **??** aus!

Der Umgebung *minipage* werden hier zusätzlich eine fixe Höhe `[4ex]` als optionales Argument übergeben, ohne dieses wäre der Abstand zum Text darunter und darüber für meinen Geschmack zu klein.

Tabelle linksbündig mit Untertitel

Da *caption* meist zentriert vordefiniert ist, muss das Caption-Setup zunächst geändert werden. Bei *scrartcl* erfolgt dies durch das Makro **\setcapwidth** `[Ausrichtung]{Breite}`:

```
\setcapwidth[l]{\linewidth}
\begin{tabular}{|c|c|}
 \hline
 Spalte 1 & Spalte 2 \\
 \hline
 A & B \\
 \hline
\end{tabular}
\captionof{table}{Eine Beispieltabelle}
\label{tab:Tabelle3}
```

Spalte 1	Spalte 2
A	B

Tabelle 3: Eine Beispieltabelle

So sieht also nun die linksbündige Tabelle **??** aus!

Bildtitel neben dem Bild mit minipages

Gleitende Objekte vertragen sich nicht mit *minipages*, nichtgleitende Objekte können dagegen problemlos in *minipages* eingebettet werden und man kann über diesen Kniff auch die Bildbeschreibung samt Bildtitel neben das Bild setzen und so oft wertvollen Vertikalplatz sparen.

Abbildung 2: Landschaftsbild

So sieht das Bild **??** nun mit dem Text daneben aus. Um die vertikale Ausrichtung zu verändern, kann die raisebox verwendet werden, wie das nächste Beispiel zeigt.

Bildtitel neben dem Bild mit minipages und raisebox

Die vertikale Ausrichtung zwischen Bild und Bildtitel lässt sich mit Hilfe des Makros \raisebox einstellen. Damit dies einen einigermaßen vorhersehbaren Effekt hat, wird hier die *minipage* mit dem optionalen Parameter `[t]` (für *top*).

Abbildung 3: Landschaftsbild

Bei Abb.**??** wurde durch das Argument -\height erreicht, dass der Bildtitel bündig zur Bildoberkante anschließt.

Dies ist der Code dazu:

```
\begin{minipage}[t]{3.5cm}
 \raisebox{-\height}{\includegraphics[width=3cm]{Bilder/Bildbeispiel}}
 \label{fig:bildlinks2}
\end{minipage}
\begin{minipage}[t]{6cm}
 \captionof{figure}{Landschaftsbild}
\end{minipage}
```

Abbildung 4: Landschaftsbild

So sieht nun Abb. **??** mit der raisebox und dem Wert -0.6\height aus. Änderung gegenüber Abb. **??**:

```
\raisebox{-0.6\height}{\includegraphics[width=3cm]{Bilder/Bildbeispiel}}
```

Abschließend noch bündig mit der Unterkante:

Abbildung 5: Landschaftsbild

So sieht nun Abb. **??** aus, diesmal also auf Höhe der Bildunterkante. Der Code für die raisebox diesmal:

```
\raisebox{-0.3\height}{\includegraphics[width=3cm]{Bilder/Bildbeispiel}}
```

Bildtitel formatieren

Bei der Dokumentenklasse *scrartcl* kann der Befehl

\setkomafont {captionlabel}{<Format>}

verwendet werden, um dem Bildtitel eine andere Formatierung zu geben, z.B. erreicht man mit `\setkomafont{captionlabel}{\bfseries}` einen Fettdruck des Labels.

Beispiel:

Abbildung 6: Noch ein Bild!

Bildlabel ändern

Wem das Label »Abbildung« zu viel Platz wegnimmt, kann dieses auch abkürzen lassen.

Dies erreicht man z.B. entweder durch

```
\renewcommand{\figurename}{Abb.}
```

oder, auf die deutsche Sprache begrenzt durch

```
\renewcaptionname{ngerman}{\figurename}{Abb.}
```

Abb. 7: Bildtitel

Weitergehende Änderungswünsche erlaubt die Neudefinition von \figureformat in der Klasse *scrartcl*, zusätzlich wird hier das Trennzeichen hinter der Abbildungsnummer verändert, mit

```
\renewcommand*{\captionformat}{~}
```

Die Neudefinition `\renewcommand*{\figureformat}{[Abb.]}` bewirkt folgende Ausgabe:

[Abb. 8] Bildtitel

16.2 Tabellenlabel formatieren

Die Befehle lauten hier:

- `\renewcaptionname{ngerman}{\tablename}{Tab.}`
- `\renewcommand*{\tableformat}{}`

A	B
1	2

Tab. 4: Beispieltabelle

17 Tipps zum Erstellen von Vorlagen

17.1 Klassenarbeit mit Deckblatt

17.2 Lösungen hinter den Aufgaben

Das kennt man von Arbeitsblättern verschiedener Verlage: nach den Aufgaben folgen die Lösungen. Bei dieser Vorlage soll alles in einer PDF enthalten sein, das erleichtert oft die Organisation der Dateien.

Um dieses Ziel zu erreichen wird die Umgebung *filecontents* verwendet, welche in dieser Form seit 2019 in den LaTeX-Distributionen enthalten ist. Die Auslagerungsdatei wird dann über \input {Auslagerungsdatei} zwei Mal eingebunden, einmal im Aufgabenmodus (ohne Lösungen) und dann im Lösungsmodus. Im Beipiel unten, wird der Befehl \jobname verwendet, um den aktuellen Dateinamen für den Namen der Auslagerungsdatei zu verwenden, im Beispiel wird ».swapfile« an den aktuellen Dateinamen angehängt. Zwischen den beiden Einbindungen müssen die Zähler zurück gestellt werden, dafür gibt es den Befehl \EXreset. Wurden nicht alle Zähler zurück gesetzt, da der Anwender eigene definiert hat, oder weil Befehle oder Umgebungen genutzt werden, deren Zähler von \EXreset nicht zurück gesetzt wurden, dann sind die betreffenden Zähler zu ermitteln und über \setcounter {Zähler}{0} auf den Anfangswert zu setzen. Bitte notfalls die betreffenden Dokumentationen studieren, wie die Zähler zurück gestellt werden können.

Beispiel für eine Vorlage mit Lösungen nach dem Aufgabenteil

```
\documentclass[parskip=full]{scrartcl}
\usepackage{EXnSOL}
\setEXtitle[Lösungen:]{Aufgaben:}
\begin{document}
 % Beginn der Auslagerungsdatei:
 \begin{filecontents*}[force]{\jobname.swapfile}
  \part* {\EXtitle}
  \EX { Aufgabe 1 }
  \solution{Die Lösung zu Aufgabe 1}
  \EX { Aufgabe 2 }
  \solution{Die Lösung zu Aufgabe 2}
  % ... weitere Aufgaben mit ihren Lösungen
 \end{filecontents*}
```

```
% Einbinden der Auslagerungsdatei im EXmodus
 \input{\jobname.swapfile}

% Zurücksetzen der Zähler:
 \EXreset
% In den SOLmode schalten und neue Seite beginnen:
 \SOLmode
 \newpage
% Zweites Einbinden der Auslagerungsdatei:
 \input{\jobname.swapfile}
\end{document}
```

Mögliche Probleme und ihre Lösung

Das wiederholte Einfügen eines ausgelagerten Aufgabenteils mit Lösungen kann nicht nur bei Zählern zu unerwünschten Ergebnissen führen, auch bei Referenzen kann es Probleme geben. So sollte ein *Label* nicht mehrfach vergeben werden. Um dies zu vermeiden, kann man **\label** durch eine Bedingung nur einmal ausführen. Dafür wurde das *Toggle* ***firsttime*** und der Befehl **\ifFirstTime {<Code>}** in das Paket implementiert.

17.3 Aufgaben-Punkte-Tabellen

18 Anhang

18.1 Zähler

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% Counter %%%%%%%%%%%%%%%%%%%%%%
\newcounter{konto} % 0 = Punkte ohne Kategorie, 1 = AFB I, 2 = AFB II, 3 = AFB
\setcounter{konto}{0}
\newcounter{material}
\newcounter{subtasknumber}
\setcounter{subtasknumber}{0}
\newcounter{tasknumber}
\setcounter{tasknumber}{0}
```

afbI,afbII,afbIII Punkte der einzelnen Anforderungsbereiche.

alltasks Anzahl aller bepunkteter Aufgaben (oder Unteraufgaben).

bonuspts Bonuspunkte.

errornumber Zähler für Fehlernummer beim Aufgabentyp Finde-den-Fehler.

expages Seitenzahl nur vom Aufgabenteil.

expp Erwartungspunkte in Lösungen.

pts Punkte für Aufgaben.

subpts Unterpunkte einer Aufgabe.

totpts Gesamtpunktzahl.

bonustotpts Gesamtbonuspunktezahl.

18.2 Textbausteine

19 Sonstige Makros

19.1 \umu{} – zum Setzen von Einheiten physikalischer Größen

`\umu{}` ist ein Makro, um Einheiten mit aufrechter Schrift nach einem mit `\thinspace` gesetzten Abstand auszugeben. Ähnlich wie das Makro `\qty{number}{unit}` aus dem Paket *siunitx*. Es wird im Makro abgefragt, ob der Mathemodus aktiv ist und ob das Paket unicode-math geladen wurde.

Beispiel: Der Widerstand beträgt 12 Ω bei 22 °C.

20 Ideen für die Zukunft

* Statistikfunktionen * Befehl für nofloatpic (untertitelt, nebenbetitelt)

```
\newcommand{\nofloatpic}[4][s]{%
\ifstrequal{#1}{st}{% if st=subtitled
 #2\par
 \captionof{figure}{#3\label{fig:#4}}
}{%
\ifstrequal{#1}{rt}{#2 \captionof{figure}{#3\label{fig:#4}}} % if rt = right t
 {\captionof{figure}{#3\label{fig:#4}} #2}}
}
```