

# Zpravodaj Československého sdružení uživatelů TeXu

---

Petra Čáčková  
Sazba zdrojových kódů

*Zpravodaj Československého sdružení uživatelů TeXu*, Vol. 22 (2012), No. 2, 97–109

Persistent URL: <http://dml.cz/dmlcz/149954>

## Terms of use:

© Československé sdružení uživatelů TeXu, 2012

Institute of Mathematics of the Czech Academy of Sciences provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these *Terms of use*.



This document has been digitized, optimized for electronic delivery and stamped with digital signature within the project *DML-CZ*:  
*The Czech Digital Mathematics Library* <http://dml.cz>

Článek je věnován možnostem, které  $\text{\LaTeX}$  nabízí pro sazbu algoritmů a zdrojových kódů programů obecně. Jsou představeny vybrané balíčky, především balíček LISTINGS, jsou nastíněny výhody (a nevýhody) vybraných řešení, nechybí ani ukázky.

**Klíčová slova:**  $\text{\LaTeX}$ , algoritmy, zdrojové kódy, programování, doslovný výpis, LISTINGS.

## 1. Úvod

„V  $\text{\TeX}$ u jde všechno.“ Jednou podskupinou toho „všeho“ je i příprava dokumentace, příručky k programům, materiály na školení a sbírky příkladů. Nahlédnutí pod povrch aneb „jak to funguje“, typické pro tyto dokumenty, často vyžaduje sazbu algoritmů a zdrojových kódů programů. V tomto článku jsou představeny některé možnosti jejich sazby i s ukázkami.

## 2. Balíčky a řešení

Základní džungle balíčků je dostupná na CTANu [1], seznam řešení odpovídající klíčovému slovu „computer code, verbatim text“ má nyní 83 položky. Ty by se dále mohly členit, například takto:

- styl „psací stroj“: balíčky ALLTT, ALLTT2,
- doslovný text: CPROTECT, FANCYVRB, VERBATIM, SHORTVRB; LISTINGS... ,
- základní algoritmy, pseudokód: ALGORITHM5, ALG, ALGORITHM2E, ALGORITMICX atd.,
- podle programovacích jazyků: C2CWEB, CNOWEB, C-PASCAL, CPROG; JAVADOC atd.,
- $\text{\LaTeX}$ ový kód: CODEDOC; EXAMPLE, EXAMPLEP atd.
- zvýrazňování syntaxe (s Pygments): TEXMENTS, VERBMENTS atd.

Je tedy z čeho vybírat a jsou dostupná jak obecná řešení, tak úzce specializované balíčky. Následuje přehled vybraných řešení a balíčků s popisem, případně s ukázkami.

### 2.1. Prostředí `tabbing`

Ještě před přehledem balíčků se zastavíme u standardního prostředí `tabbing` [2, s. 98]. Díky nastavení tabulačních zárázek lze dosáhnout pěkného a přehledného

výsledku. Zde však výhody končí, kód se může jevit jako složitý a nepřehledný a přizpůsobení (například zvýrazňování klíčových slov) je třeba řešit ručně.

---

#### Prostředí tabbing

---

```
\begin{tabbing}
\textbf{procedure} Fibo(pocet: byte);\n
// \textit{procedura pro výpis celé řady až po n. člen} \n
\textbf{var} \ = clen1, clen2, pomocny: word;\n
        \> i: byte;\n
\textbf{begin} \+ \n
    clen1:=0; clen2:=1; // \textit{hodnoty prvních dvou členů} \n
    \textbf{if} \ = (pocet > 0) \textbf{then begin}\+ \n
        ... \- \n
    \textbf{end}; \- \n
\textbf{end}; \n
\textbf{end}{tabbing}
```

---

```
procedure Fibo(pocet: byte);
// procedura pro výpis celé řady až po n. člen
var clen1, clen2, pomocny: word;
    i: byte;
begin
    clen1:=0; clen2:=1; // hodnoty prvních dvou členů
    if (pocet > 0) then begin
        ...
    end;
end;
```

## 2.2. Prostředí a balíček verbatim

Prostředí **verbatim** je klasikou pro doslovný výpis neproporcionálním písmem a nepodléhající formátování. Použití je jednoduché, ale zde opět výhody končí, žádné formátování nebo zvýrazňování klíčových slov. Kromě toho je prostředí nevhodné pro tvorbu vlastních maker (nesmí být parametrem jiného příkazu).

Lepší možnosti práce s doslovným textem poskytuje balíček **VERBATIM** [3]. Název prostředí zůstává stejný a jeho možnosti v podstatě také, navíc ale umožňuje vložení textového souboru příkazem **\verbatiminput**. Dále podporuje víceřádkové komentáře v prostředí **comment**.

---

#### Balíček verbatim

---

Víceřádkový komentář, který se nevysází:  
**\begin{comment}**

Program, který má na vstupu řetězec.

Vypisuje z~něj podřetězce vždy o~jeden znak delší než předchozí.

```
\end{comment}
```

Vstup ze souboru:

```
\verbatiminput{string.pas}
```

---

Víceřádkový komentář, který se nevysází:

Vstup ze souboru:

```
var s: string; i: byte;
```

```
begin
```

```
  readln(s);
```

```
  for i:=1 to length(s) do writeln(copy(s,1,i));
```

```
end.
```

## 2.3. Balíček alltt

Další balíček pro doslovný výpis úseku dokumentu neproporcionálním písmem, složené závorky a zpětné lomítko si zachovávají svůj význam a zdroj tedy lze doplnit o příkazy [4]. Práce s tímto balíčkem je tedy o něco pružnější než u prostředí/balíčku VERBATIM, avšak použití pro výpis zdrojového kódu (který je u mnoha jazyků plný složených závorek) je poněkud nepraktické a vstup zdrojového textu přímo ze souboru je problematický. Existují však jiná vhodná řešení.

---

### Balíček alltt

---

```
\begin{alltt}
```

```
\kw{function} RFibo(n: byte): word;
```

```
\textit{// rekurzivní funkce, vrátí hodnotu n. prvku posloupnosti}
```

```
\kw{begin}
```

```
  \kw{case} n \kw{of}
```

```
    0: RFibo:=0;
```

```
    1: RFibo:=1;
```

```
    \kw{else} RFibo:=RFibo(n-1) + RFibo(n-2);
```

```
  \kw{end};
```

```
\kw{end};
```

```
\end{alltt}
```

---

```
function RFibo(n: byte): word;
```

```
// rekurzivní funkce, vrátí hodnotu n. prvku posloupnosti
```

```
begin
```

```
  case n of
```

```
    0: RFibo:=0;
```

```

1: RFibo:=1;
   else RFibo:=RFibo(n-1) + RFibo(n-2);
end;
end;

```

## 2.4. Balíčky `algorithmic` a `algorithm`

Sada Algorithms [5] obsahuje balíčky `ALGORITHMIC` a `ALGORITHM`, které poskytují stejnojmenná prostředí. `ALGORITHMIC` slouží pro sazbu (pseudo)kódu a `ALGORITHM` vytváří plovoucí prostředí pro tyto algoritmy.

Balík `ALGORITHMIC` počítá s použitím pseudokódu a definované množiny klíčových slov. Ta jsou zavedena jako makra, dají se v případě potřeby snadno předefinovat, například do národních jazyků. V prostředí lze číslovat řádky kódu (uvedením volitelného parametru, např. `\begin{algorithmic}[5]`) a také se na ně odvolávat.

Balík `ALGORITHM` nabízí další možnosti – popisky algoritmů, vytvoření jejich seznamu (příkazem `\listofalgorithms`), různé způsoby orámování. Číslování algoritmů může být odvozeno od částí dokumentu (např. uvedením volitelného parametru `[chapter]` u příkazu `\usepackage{algorithm}`).

---

### Balíčky `algorithmic` a `algorithm`

---

```

\begin{algorithm}
\caption{Fibonacci}\label{fibonacci}
\begin{algorithmic}[1]
\STATE F:
  \IF{prvek = 0} \RETURN{0}
  \ELSIF{prvek = 1} \RETURN{1}
  \ELSE \RETURN{F(prvek-1) + F(prvek-2)}
\ENDIF
\end{algorithmic}
\end{algorithm}

```

---

```

1: F:
2: if prvek = 0 then
3:   return 0
4: else if prvek = 1 then
5:   return 1
6: else
7:   return F(prvek-1) + F(prvek-2)
8: end if

```

Algorithm 1: Fibonacci

## 2.5. Balíček c-pascal

Je přizpůsobený přímo pro oba tyto programovací jazyky [6]. Jde tedy jen o dvouúčelový nástroj, ale pokud je s tím uživatel spokojen, může se jednat o velmi elegantní řešení. V kódu se automaticky zvýrazňují klíčová slova, jsou použity různé řezy písma, výpis programu je pohodlný a přehledný. K tomu může přispět i možnost načítání kódu ze zadaného souboru (vhodné např. pro sbírku příkladů).

### Balíček c-pascal

---

```
\BeginPascal
function RFibo(prvek: byte): word;
(* rekurzivni funkce, vrati hodnotu n. prvku posloupnosti *)
begin
  case n of
    0: RFibo:=0;
    1: RFibo:=1;
    else RFibo:=RFibo(n-1) + RFibo(n-2);
  end;
end;
\EndPascal

\InputPascal{string.pas}
```

---

```
function RFibo(prvek: byte): word;
(* rekurzivni funkce, vrati hodnotu n. prvku posloupnosti *)
begin
  case n of
    0: RFibo:=0;
    1: RFibo:=1;
    else RFibo:=RFibo(n-1) + RFibo(n-2);
  end;
end;

var s: string; i: byte;
begin
  readln(s);
  for i:=1 to length(s) do writeln(copy(s,1,i));
end.
```

## 2.6. Balíček lineno

1 Balíček `lineno` [7] je určen pro číslování jednotlivých řádků vstupu. Lze jej  
2 použít pro zdrojové kódy programů i odstavcový text. Odstavec je rozlámán do  
3 jednotlivých řádků, k nimž jsou připojena čísla – s možností odkazovat se na ně  
4 pomocí  $\LaTeX$ ového mechanismu křížových odkazů. Balíček nabízí různé možnosti  
5 nastavení číslování. Namísto tradičního prostředí poskytuje příkazy pro zapnutí  
6 a vypnutí číslování.

```
1 function RFibo(prvek: byte): word;  
2 // rekurzivni funkce, vrati hodnotu n. prvku posloupnosti  
3 begin  
4   case n of  
5     0: RFibo:=0;  
6     1: RFibo:=1;  
7     else RFibo:=RFibo(n-1) + RFibo(n-2);  
8   end;  
9 end;
```

### Balíček lineno

---

`\linenumbers`

Balíček `\pkg{lineno}` je určen pro číslování jednotlivých řádků vstupu.  
Lze jej použít pro zdrojové kódy programů... vypnutí číslování.

`\resetlinenumber[1]`

`\begin{verbatim}`

`function RFibo(prvek: byte): word;`

`// rekurzivni funkce, vrati hodnotu n. prvku posloupnosti`

`...`

`\end{verbatim}`

`\nolinenumbers`

---

## 2.7. Balíček fancyvrb

Balíček `FANCYVRB` [8] rozšiřuje možnosti výpisu doslovného textu. Nabízí prostředí `Verbatim`, které je univerzálně použitelné pro různé programovací jazyky i pro pseudokód, syntaxe se tedy nezvyráží. Pokud se ale v nepovinném parametru prostředí definují znaky, které uvádějí makro a obklopují skupinu (`\begin{Verbatim}[co` lze v kódu používat i příkazy a výpis tak přizpůsobit. Kromě toho jsou k dispozici rozsáhlé možnosti přizpůsobení (způsob sazby, změna typu, řezu a stupně písma, formátování, barva, rámeček, číslování řádků).

`Verbatim` se v nové verzi balíčku dá použít i v poznámce pod čarou.

Lze také definovat znak, který bude použit místo příkazu `\verb` (např.

`\DefineShortVerb{!}`) a například v odstavcových textech je pak možné používat kratší způsob zápisu (`!mujtext!`).

Vložit zdrojový kód přímo ze souboru je možné příkazem `\VerbatimInput`.

Balíček je navíc flexibilní, nabízí více prostředí typu verbatim – `BVerbatim` (materiál do boxu) a `LVerbatim` (LR mód). Je možné předefinovat existující prostředí i definovat vlastní.

Balíček `FANCYVRB` tedy poskytuje významná rozšíření pro sazbu doslovného textu s rozsáhlými možnostmi formátování a dalšího přizpůsobení. I když není určen přímo pro sazbu zdrojových kódů, i k tomuto se dá výborně využít.

---

#### Balíček `fancyvrb`, sazba kódu v $\text{\LaTeX}$ u

---

```
\begin{Verbatim}[frame=single,label=Sazba kódu v~\LaTeX u,numbers=left]
\begin{itemize}
  \item Mouka
  \item Vajíčka
  \item \textbf{Smetana}
\end{itemize}
\end{Verbatim}
```

---



---

#### Sazba kódu v $\text{\LaTeX}$ u

---

```
1 \begin{itemize}
2   \item Mouka
3   \item Vajíčka
4   \item \textbf{Smetana}
5 \end{itemize}
```

---

#### Balíček `fancyvrb`, sazba kódu v Pascalu

---

```
\DefineShortVerb{\|}
```

Ukázka pro ne $\text{\LaTeX}$ ový zdrojový kód: funkce `|RFibo|` v~Pascalu.

```
\begin{Verbatim}[frame=single,label=Sazba kódu v~Pascalu,numbers=left]
function RFibo(n: byte): word;
begin
  case n of
    0: RFibo:=0;
    1: RFibo:=1;
    else RFibo:=RFibo(n-1) + RFibo(n-2);
  end;
end;
\end{Verbatim}
```

---

Ukázka pro ne $\text{\LaTeX}$ ový zdrojový kód: funkce `RFibo` v Pascalu.



```

1 function RFibo(n: byte): word;
2 begin
3     case n of
4         0: RFibo:=0;
5         1: RFibo:=1;
6         else RFibo:=RFibo(n-1) + RFibo(n-2);
7     end;
8 end;
```

## Balíček fancyvrb, vstup ze souboru

---

```
\VerbatimInput{string.pas}
```

---

```

var s: string; i: byte;
begin
    readln(s);
    for i:=1 to length(s) do writeln(copy(s,i,i));
end.
```

Je potřeba uvést, které znaky uvádějí makro (\\) a obklopují skupinu. Pak lze používat i příkazy.

## Balíček fancyvrb, používání příkazů

---

```

\begin{Verbatim}[commandchars=\\\{\}]
function RFibo(prvek: byte): word;
\textcolor{seda}{begin}
    case prvek of (...) end;
\textcolor{seda}{end;}
\end{Verbatim}
```

---

```

function RFibo(prvek: byte): word;
begin
    case prvek of (...) end;
end;
```

## 2.8. Balíček listings

Balíček LISTINGS [9] je zřejmě nejoblíbenějším a v diskuzích nejdoporučovanějším balíčkem, který řeší sazbu zdrojových kódů. Seznam výhod je opravdu rozsáhlý:

- automatické zvýrazňování syntaxe pro desítky předdefinovaných jazyků,
- možnost dodefinovat další jazyky,
- bohaté možnosti nastavení sazby (formátování, číslování řádků, rámečky atd.),

- možnost vytvoření vlastního stylu formátování výpisů (`\lstdefinestyle`),
- vkládání úryvků kódu do odstavce i se zvýrazňováním syntaxe (příkaz `\lstinline`)
- vstup přímo ze souboru (`\lstinputlisting`)
- doplnění výpisu o titulek,
- vygenerování seznam algoritmů (na základě titulků).

Za nevýhody lze považovat výchozí nastavení formátování a práci s kódováním souboru.

Jednotlivé znaky ve výpisu kódu jsou ve výchozím nastavení zarovnány na jednotlivé sloupce, a to i přesto, že je použito proporcionální písmo. Text pak působí jako prostrkaný a je hůře čitelný. (Tento efekt se týká prostředí `lstlistings`, ne vkládaných souborů.) Popsané chování lze přepnout změnou nastavení z výchozího `columns=fixed` na `columns=flexible` v seznamu parametrů příkazu `\lstset`.

Druhou nevýhodou jsou problémy s kódováním národních znaků. Nesprávné zobrazení znaků může vyřešit parametr `extendedchars=true` u příkazu `\lstset`. Je třeba zavést `FONTENC` nebo `INPUTENC`, předpokládá se jednobajtové kódování. Má-li být vstup v UTF-8, je to řešitelné balíčkem `LISTINGSUTF8` [10], který však vstup v UTF8 převádí „zpět“ do jednobajtového kódování.

## Ukázky použití balíčku `listings`

### Ukázka nastavení formátování výpisu pro Pascal

---

```
\lstset { %
language=Pascal,
morecomment=[l]{/}, % další typ komentáře
numbers=left,      % čísla vlevo od zdrojového kódu
stepnumber=1,      % čísluje se každý řádek
numberstyle=\tiny, % velikost čísel
basicstyle=\small, % velikost písma kódu
frame=single,      % orámování kódu
columns=flexible   % odstraní zarovnání na sloupce
}
```

---

### Ukázka vlastního stylu výpisu

---

```
% Definice:
\lstdefinestyle {moje}{numbers=left, stepnumber=1, frame=lines}
...
% Použití:
\begin{lstlisting}[ title ={Ukázka balíčku listings}, style=moje]...
```

---

## Ukázka balíčku listings – Pascal

---

```
\begin{lstlisting}[ title =Rekurzivní Fibonacci,style=moje]
function RFibo(prvek: byte): word;
(* rekurzivní funkce, vrátí hodnotu n. prvu posloupnosti *)
begin
  case n of
    0: RFibo:=0;
    1: RFibo:=1;
    else RFibo:=RFibo(n-1) + RFibo(n-2);
  end;
end;
% ukončení prostředí lstlisting
```

---

## Rekurzivní Fibonacci

---

```
function RFibo(prvek: byte): word;
(* rekurzivní funkce, vrátí hodnotu n. prvu posloupnosti *)
begin
  case n of
    0: RFibo:=0;
    1: RFibo:=1;
    else RFibo:=RFibo(n-1) + RFibo(n-2);
  end;
end;
```

---

## Ukázka balíčku listings – vstup ze souboru

---

```
\lstinputlisting [ title =Výpis řetězce (ze souboru)]{string.pas}
```

---

## Výpis řetězce (ze souboru)

```
1 var s: string; i: byte;
2 begin
3   readln(s);
4   for i:=1 to length(s) do writeln(copy(s,i,1));
5 end.
```

## Ukázka balíčku listings – L<sup>A</sup>T<sub>E</sub>X

---

```
% nastavení: ...language=[LaTeX]TeX, ...
\begin{lstlisting}[ title =Tabulka v~\LaTeX u,style=moje]
\begin{tabular}{|c|l|}\hline
\bfseries BMI & \bfseries Výsledek & \bfseries Doporučení \\ \hline
méně než 18,5 & podváha & Sněž něco! \\ \hline
```

```

18,5--24,9    & v~normě & Dobré, pokračuj. \\ \hline
25,0--29,9    & nadváha & Nejez brambůrky, dej si mrkev. \\ \hline
30,0 a~vyšší & obezita & Necpi se! \\ \hline
\end{tabular}
% ukončení prostředí lstlisting

```

---

## Tabulka v L<sup>A</sup>T<sub>E</sub>Xu

```

\begin{tabular}{|c|l|l|}\hline
\bfseries BMI & \bfseries Výsledek & \bfseries Doporučení \\ \hline
méně než 18,5 & podváha & Sněz něco! \\ \hline
18,5--24,9    & v~normě & Dobré, pokračuj. \\ \hline
25,0--29,9    & nadváha & Nejez brambůrky, dej si mrkev. \\ \hline
30,0 a~vyšší & obezita & Necpi se! \\ \hline
\end{tabular}

```

---

## Ukázka balíčku listings – C

```

\begin{lstlisting}[title=Zjištění dělitelů v~C,style=moje]
#include <stdio.h>
int i,  cislo ;
int main() {
    printf("Zadej číslo pro zjištění dělitelů \n");
    scanf("%d",&cislo);
    for (i=1;i<cislo;i++) if(cislo%i==0) printf("%d ",i);
    getch();
    return(0);
}
% ukončení prostředí lstlisting

```

---

## Zjištění dělitelů v C

```

#include <stdio.h>
int i,  cislo ;
int main() {
    printf("Zadej číslo pro zjištění dělitelů \n");
    scanf("%d",&cislo);
    for(i=1;i<cislo;i++) if(cislo%i==0) printf("%d ",i);
    getch();
    return(0);
}

```

---

## 2.9. Další možnosti

Uvedený přehled balíčků zdaleka není vyčerpávající. Obsahuje vybraná čistě L<sup>A</sup>T<sub>E</sub>Xová řešení založená na stávajících připravených balíčcích. Je však možné jít ještě dál a využít například možností balíčků TEXMENTS a VERBMENTS. Oba používají zvýrazňovač syntaxe Pygments (<http://pygments.org/>). Zde už se ale nejedná jen o L<sup>A</sup>T<sub>E</sub>Xové řešení, nejprve je potřeba nainstalovat Pygments, jakož i Python.

## 3. Shrnutí

Namísto závěru následuje tabulka, která stručně shrnuje vlastnosti vybraných „univerzálních“ prostředků pro sazbu zdrojových kódů a algoritmů. Výběr konkrétního řešení však závisí především na účelu, který má být splněn.

	tabbing	verbatim	lineno	fancyvrb	listings
číslování řádků	–	–	+	+	+
autom. zvýraz.					
klíčových slov	–	–	–	–	+
formátování výpisu	+	–	–	+	+
popisek	–	–	–	+	+
úsek v odstavci	–	+	+	+	+
vložení souboru	–	–/+ <sup>1</sup>	–/+ <sup>1</sup>	+	+
plovoucí prostř.	–	–	–	–	+
seznam algoritmů	–	–	–	–	+
X <sub>E</sub> L <sup>A</sup> T <sub>E</sub> X	+	+	+	+	+ <sup>2</sup>

## Reference

- [1] *Packages by keyword* [online]. 2011 [cit. 2011-11-30]. URL [http://www.ctan.org/keyword/computer\\_code](http://www.ctan.org/keyword/computer_code).
- [2] Rybička, Jiří. *L<sup>A</sup>T<sub>E</sub>X pro začátečníky*. 3. vydání. Brno: Konvoj, 2003. 238 s. ISBN 80-7302-049-1.
- [3] Schöpf, Rainer; Rowley, Chris. *A New Implementation of L<sup>A</sup>T<sub>E</sub>X's verbatim and verbatim\* Environments* [online]. 2001-03-12 [cit. 2011-11-30]. URL <http://ftp.cvut.cz/tex-archive/macros/latex/required/tools/verbatim.pdf>.
- [4] Braams, Johannes. *The alltt environment* [online]. 1997-06-16 [cit. 2011-10-27]. URL <http://www.tug.org/texlive/devsrc/Master/texmf-dist/doc/latex/base/alltt.pdf>.

<sup>1</sup>je možné při použití balíčku VERBATIM

<sup>2</sup>potíže s kódováním národních znaků

- [5] Brito, Rogério. *The algorithms bundle* [online]. 2009-08-24 [cit. 2011-11-11]. URL <http://mirrors.ctan.org/macros/latex/contrib/algorithms/algorithms.pdf>.
- [6] Gulczynski, Michal. *PCAP — macro package for typesetting programs in Python, C and Pascal* [online]. [cit. 2011-10-20] [http://ftp.cvut.cz/tex-archive/macros/generic/c\\_pascal/README.eng](http://ftp.cvut.cz/tex-archive/macros/generic/c_pascal/README.eng).
- [7] Böttcher; Stephan, Lück, Uwe. *A L<sup>A</sup>T<sub>E</sub>X package to attach line numbers to paragraphs* [online]. 2005-11-02 [2011-11-15]. URL <ftp://ftp.cstug.cz/pub/tex/CTAN/macros/latex/contrib/lineno/lineno.pdf>.
- [8] Van Zandt, Timothy. *The ‘fancyvrb’ package: Fancy Verbatims in L<sup>A</sup>T<sub>E</sub>X* [online]. 2010-05-15 [cit. 2011-11-16]. URL <ftp://ftp.cstug.cz/pub/tex/CTAN/macros/latex/contrib/fancyvrb/fancyvrb.pdf>.
- [9] Heinz, Carsten; Moses, Brooks. *The Listings Package* [online]. 2007-02-22 [cit. 2011-11-20]. URL <http://ftp.cstug.cz/pub/tex/CTAN/macros/latex/contrib/listings/listings.pdf>.
- [10] Oberdiek, Heiko. *The listingsutf8 package* [online]. 2007-11-11 [cit. 2011-11-30]. URL <http://ftp.cstug.cz/pub/tex/CTAN/macros/latex/contrib/oberdiek/listingsutf8.pdf>.

## Summary: Source Code Typesetting

The article describes L<sup>A</sup>T<sub>E</sub>X possibilities for algorithms and source code typesetting. Chosen packages are presented, mainly the LISTINGS package. Jsou představeny vybrané balíčky, především balíček LISTINGS, advantages and disadvantages of chosen solutions are outlined, examples included.

**Key words:** L<sup>A</sup>T<sub>E</sub>X, algorithms, source code, programming, verbatim text, LISTINGS.

Petra Čáčková  
[petra.cackova@mendelu.cz](mailto:petra.cackova@mendelu.cz)