A LaTeX Package for Typesetting Crossword Puzzles*

Gerd Neugebauer
Mainzer Straße 8
56321 Rhens (Germany)
Net: gerd@informatik.uni-koblenz.de

Documentation date: 1996/11/25

Abstract

cpwuzzle.dtx provides a package to typeset crossword puzzles. There is an graphical user interface written in Tcl/Tk which does most of the work in creating a file which is conform with this style. Thus it is clear that the macros provided are not tailored towards a human user.

The package can be used to produce several types of puzzles like the classical crossword puzzle, a number puzzle, and fill-in puzzles.

Contents

*This file documents cwpuzzle.dtx version 1.4 as of 1996/11/25.

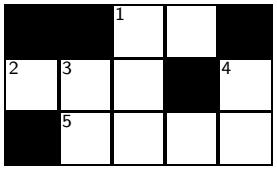
1 About Crossword Puzzles

Crossword puzzles are can be a an amusing but also a challenging hobby. Unfortunately at the time of this writing I am not aware of any good package to typeset crossword puzzles with LaTeX. Thus I decided to make one which at least fits my needs.

There are several types of crossword puzzles aroung. This package can only be used to typeset several of them. The basic assumption in this package is that puzzles are rectangular arragnements of boxes. Some of these boxes are black and others are prepared to take single letters. Each word in the grid is enclosed in black boxes or the outside.

Optionally there may be rectangular regions left blank inside the puzzle. They can be used to place ads or other informative texts inside the puzzle.

1.1 Classical Crossword Puzzles

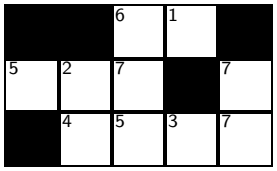


Across 1 unit of measure
2 ∗ 5 sectioning unit

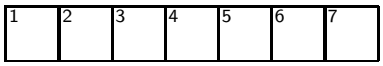
Down 1 η 3 unit of measure
4 nonproportional font

The "classical" type of a crossword puzzle words are marked with numbers and each word is acomapanied with a clue which should help (or confuse) the reader. Those clues are listed after the frame of the puzzle.

1.2 Number Crossword Puzzles

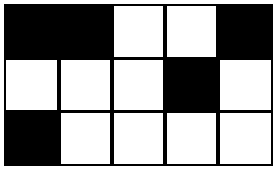


The following letters are used: AEPRSTX



The "number puzzle" variant contains only numbers instead of letters. Different numbers denote different letters. There are no clues. The reader is assumed to find a complete list of letters by filling appropriate words into the grid. Sometimes a word is already entered into the grid to ease the start.

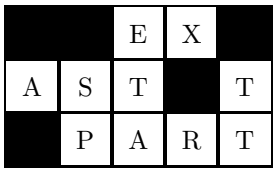
1.3 Fill-In Crossword Puzzles



Words of length 2: EX SP TT
Words of length 3: AST ETA
Words of length 4: PART

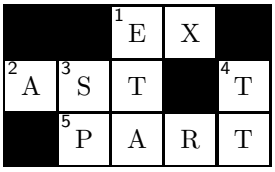
The "fill-in puzzle" variant consists of a frame containing only black and white boxes. Additionally a list of words is given which have to be put into the frame until none is left and the frame is completed.

1.4 Solutions

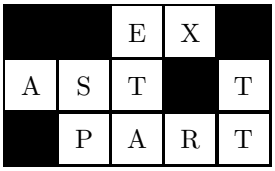


Often it is not only desirable to typeset the unsolved crossword puzzle but also the solution. This means that all the letters have to be filled in. This should be possible with the same source as the questions to avoid typos or redundancies leading to additional work.

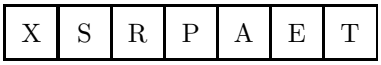
Several variants of solutions come to mind. Primarily the solution should show the letters and suppress any clues. One major distinction is also wether or not the numbers of the words should be shown in the solution as well.



Finally there are the lists of letters in numbered puzzles. In the solution they will show the letters in them as well.



The following letters are used: AEPRSTX



2 Input of Crossword Puzzles

The basic idea behind this package is that a crossword puzzle is specified in a seperate file. The actual appearance of the puzzle is controlled by several options. Thus it should be possible to produce the unsolved and the solved puzzle from the same source. Before we describe the various options we will have a look at the basic environments and macros used to specified a crossword puzzle.

`Puzzle` This package provides the environment `Puzzle` which typesets the frame of a crossword puzzle. This environment takes two arguments. These arguments are the number of columns and the number of the rows of the puzzle. This means that essentially only rectangular puzzles can be typeset.

The example from section 1.1 has been entered as follows:

```
\begin{Puzzle}{5}{3}%
  |*   |*   |[1]E|X  |*   |.
  |[2]A|[3]S|T   |*  |[4]T|.
  |*   |[5]P|A   |R  |T   |.
\end{Puzzle}
```

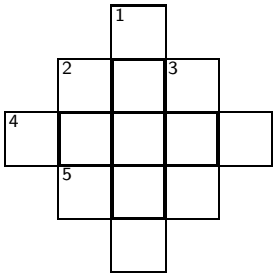
In this example we can see that inside the `Puzzle` environment there is one special character. This is the bar `|`. This bar is an activge character in TEX. Thus you can think of it like a macro.

The `|` macro takes two arguments. The first argument is optional, i.e. enclosed in brackets if present. This optional argument denotes the number for numbered boxes.

The second argument is either empty `{}` or it consists of a single character. This argument describes the action to be performed.

- If this argument is a letter then it is simply shown in the solution and suppressed in the unsolved crossword puzzle.
- If this argument is an asterisk `*` then a black box is produced.
- If this argument is a dot `.` then this marks the end of the current row. The next box is typeset at the beginning of the following row.
- If this argument is empty `{}` then a white box is typeset. This box does not contain a letter, nor does it have a frame. This macro can be used to leave room for larger boxed with ads. Alternatively this can be used to disable certain boxes to make a non-rectangular crossword puzzle.

```
\begin{Puzzle}{5}{5}
  |{}  |{}  |[1]S|.
  |{}  |[2]M|I   |[3]D|.
  |[4]T|I   |M   |E   |S   |.
  |{}  |[5]N|E   |G   |.
  |{}  |{}  |Q   |.
\end{Puzzle}
```



Across: 2 | 4 × 5 ¬

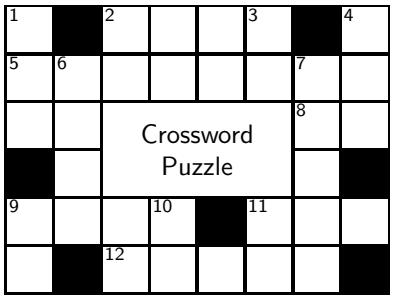
Down: 1 ≃ 2 loglike function 3 loglike function

Note that whitespace is ignored after the arguments but not between the bar and the arguments.

`\Frame` The macro `\Frame` can be used to typeset ads or other text into larger boxes inside the frame of the crossword puzzle. For this purpose five arguments are required. The first two arguments are used to specify the leower left corner of the frame. The lower left corner has the coordinates 0,0 and the numbers increase upwards and to the right.

The third argument is the width of the frame and the fourth argument is the height of the frame measured in number of boxes. Finally, the fifth argument contains the text to be typeset. Per default it is typeset in a minipage of the appropriate width centered horizontally and vertically.

```
\begin{Puzzle}{8}{6}
  \Frame{2}{2}{4}{2}{\sf Crossword\\Puzzle}
  |[1]E|*   |[2]N |U    |L   |[3]L |*   |[4]V|.
  |[5]T|[6]R|I    |A    |N   |G    |[7]L|E   |.
  |A   |U   |{}   |{}   |{}  |{}   |[8]C|C   |.
  |*   |L   |{}   |{}   |{}  |{}   |E   |*   |.
  |[9]B|E   |T    |[10]A|*   |[11]L|I   |M   |.
  |F   |*   |[12]L|A    |B   |E    |L   |*   |.
\end{Puzzle}
```



Across: 2 empty 5 △ 8 carbon copy (letter.sty) 9 β 11 limes 12 mark it

Down: 1 η 2 $\ni$ 3 logarithm 4 $\vec{}$ 6 black rectangle 7 $\lceil$ 9 bold face 10 Å 11 $\leq$

`PuzzleClues` The clues in the classical crossword puzzle are typeset with the use of the environment `PuzzleClues`. This environment takes one argument which is typeset before the clues. The environment takes roughly the half of the textwidth and make a minipage with this width. Thus two invocations of this environment are typeset side by side.

Alternatively if the solution is typeset then the environment `PuzzleClues` has no effect.

```
\begin{PuzzleClues}{\textbf{Across}}%
  \Clue{1}{EX}{unit of measure}%
  \Clue{2}{AST}{\(\ast\)}%
  \Clue{5}{PART}{sectioning unit}%
\end{PuzzleClues}%
\begin{PuzzleClues}{\textbf{Down}}%
  \Clue{1}{ETA}{\(\eta\)}%
  \Clue{3}{SP}{unit of measure}%
  \Clue{4}{TT}{nonproportional font}%
\end{PuzzleClues}%
```

`Clue` The environment `PuzzleClues` defines one local macro. This macro is named `\Clue` and takes three arguments. The first argument is the number of the word. This should correspond to the number in the puzzle frame. The second argument is the word itself. Currently this not used at all. Finally the third argument is the clue for the word.

If the unsolved puzzle is typeset then the first and the third argument are used. Otherwise all arguments are silently absorbed.

`\PuzzleLetters` The macro `\PuzzleLetters` can be used to typeset the list of used letters in numbered crossword puzzles. It has one argument which are the used letters (preferably in alphabethical order.

`\PuzzleNumbers` The macro `\PuzzleNumbers` can be used to generate a numbered list of boxes

for the numbered crossword puzzles. The user is supposed to collect the found letters here.

`PuzzleWords` The environment `PuzzleWords` can be sued to typeset the list of words for a fill-in puzzle. It takes one argument. This is the length of the words listed. For each length there should be an invocation of this environment. The words in this environment are supposed to be ordered alphabetically.

`\Word` The macro `\Word` is defined inside the environment `PuzzleWords`. It takes one argument which is the word itself.

```
\begin{PuzzleWords}{2}
  \Word{EX}%
  \Word{SP}%
  \Word{TT}%
\end{PuzzleWords}%
\begin{PuzzleWords}{3}
  \Word{AST}%
  \Word{ETA}%
\end{PuzzleWords}%
\begin{PuzzleWords}{4}
  \Word{PART}%
\end{PuzzleWords}%
```

3 Parameters and Options

`\PuzzleUnitlength` The length `\PuzzleUnitlength` deternimes the width and height of each single box in the frame of a crossword puzzle. The default value is `20pt`.

`\PuzzleBlackBox` The macro `\PuzzleBlackBox` contains the commands to produce the black boxes. It has to produce at most of width and height of `\PuzzleUnitlength`. Per default it just produces a black rectangle of this size.

The following list shows some variants which can be achieved by redefining the macro `\PuzzleBlackBox`.

```
\renewcommand{\PuzzleBlackBox}{\rule{.75\PuzzleUnitlength}%
                                {.75\PuzzleUnitlength}}
```

```
\renewcommand{\PuzzleBlackBox}{\framebox(.75,.75){%
                                \framebox(.5,.5){}}}
```

Additional effects can be achieved by using shades of gray (with the `graphics` package).

`\PuzzleFont` The macro `\PuzzleFont` contains the font changing command issued before the frame of the crosword puzzle.

`\PuzzleNumberFont` The macro `\PuzzleNumberFont` contains the font changing command issued before a number in the frame of the crosword puzzle is typeset.

`\PuzzleClueFont` The macro `\PuzzleClueFont` contains the font changing command issued before the clues are typeset.

`\PuzzleWordsText` The macro `\PuzzleWordsText` contains the text which is typeset at the beginning of the environment `PuzzleWords`. It has one argument which contains the length of the words listed.

`\PuzzleLettersText` The macro `\PuzzleLettersText` contains the text which is typeset at the beginning of the macro `PuzzleLetters`.

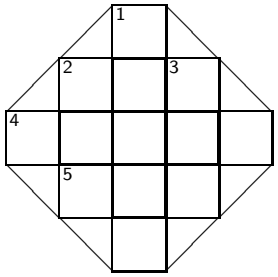
`\PuzzleSolution` The macro `\PuzzleSolution` arranges everything that the following puzzles are typeset in the "solution" mode, i.e. the letters are shown and the clues are suppressed.

This macros has one optional argument which has to be `true` or `false`. This argument determines whether or not the numbers should also be shown in the solution. The default is `false` which means that the numbers are suppressed in the solution.

`\PuzzleUnsolved` The macro `\PuzzleUnsolved` arranges everything that the following puzzles are typeset in the "unsolved" mode, i.e. the letters are suppressed and the clues are shown.

`\PuzzleHook` The macro `\PuzzleHook` is called at the end of the `Puzzle` environment. It can be used to place additional graphical elements in the puzzle frame.

The following example shows a crossword puzzle which we have seen before and the definition for the `\PuzzleHook`.




```
\newcommand\PuzzleHook{
  \put(0,2){\line(1,-1){2}}
  \put(0,3){\line(1,1){2}}
  \put(5,2){\line(-1,-1){2}}
  \put(5,3){\line(-1,1){2}}
}
```

4 The Related Program

There is a related program written in Tcl/Tk. This program can be used to manually construct crosswrod puzzles and save them in a format suitable for this package. Alternativly it can also store the crossword puzzle prepared for `crosswrd.sty`.

Other features include the creation of a proper frame and filling with words.

Right now I have not prepared a distribution of this program yet since this program requires dictionaries which I can not distribute legally.

The examples in this documentation have been computed with the help of the `cwp` program.

5 Further Plans

Maybe I will add a mode for further variants of crossword puzzles sometimes.

6 The Implementation

The crossword puzzle is basically implemented with the LaTeX picture environment. This gives us enough flexibility and provides an high enough abraction such that we do not have to fiddle around with to many low level details.

The natural unit in a crossword puzzle is a box which is empty or black. Thus the `unitlength` is set to the width (and height) of such a box.

6.1 Basic Definitions and Parameters

First we identify this package.

```
\ProvidesPackage{cwpuzzle}[\filedate gene]
```

The dimen register `\PuzzleUnitlength` stores the height and width of a box of the puzzle. The default is `20pt` which is also shown in this documentation.

```
\newdimen\PuzzleUnitlength
\PuzzleUnitlength=20pt
```

`\PuzzleClueFont` The macro `\PuzzleClueFont` contains font changings commands issued before the clues are typeset.

```
\newcommand\PuzzleClueFont{\footnotesize}
```

`\PuzzleFont` The macro `\PuzzleFont` contains font changings commands issued before the puzzle is typeset.

```
\newcommand\PuzzleFont{\rm\normalsize}
```

`\PuzzleNumberFont` The macro `\PuzzleNumberFont` contains font changings commands issued before the numbers in a puzzle are typeset.

```
\newcommand\PuzzleNumberFont{\sf\scriptsize}
```

`\PuzzleHook` Puzzles are typeset with the LaTeX picture environment. At the end of this environment the macro `\PuzzleHook` is called. The package prodised an empty default. Users may want to use this place to typeset additional elements on top of the puzzle.

The puzzle uses a `\unitlength` of `\PuzzleUnitlength`. Thus it is rather easy to adress the boxes in the puzzle.

```
\let\PuzzleHook=\relax
```

6.2 The Frame of the Crossword Puzzle

To describe the coordinates where the next box should be typeset we need two counters for the coordinates. These counters are now allocated (even though we could use temporary counters from LaTeX).

```
\newcount\Puzzle@X
\newcount\Puzzle@Y

\begingroup
\catcode`\|=13
\gdef\Puzzle@@solution{
  \let|=\Puzzle@Box@@solution
  \let\Frame=\Puzzle@Frame@@solution
}
```

```
\gdef\Puzzle@@normal{
  \let|=\Puzzle@Box@@normal
  \let\Frame=\Puzzle@Frame@@normal
}
\endgroup
```

`Puzzle` The environment `Puzzle` typesets the frame of a crossword puzzle. It is implemented utilizing a `picture` environment. The unilength is set to the `\PuzzleUnitlength`. Thus the navigation is faily easy. The basic unit is width and height of a single box.

The macros which are local to the environment are activated. Thus we avoid collisions with other packages where the same macro names might be used.

Finally the counter which contain the x and the y coordinate have to be initialized.

The last action in the `picture` environment is the expansion of the macro `\PuzzleHook`. This can be used to include additional material in the `picture` environment. Primarily I have use this to include the ads. But now there is the macro `\Frame` for this purpose.

```
\newenvironment{Puzzle}[2]{\par\noindent\mbox{}\hfill
  \catcode`\|=13
  \@nameuse{Puzzle@@\Puzzle@TYPE}%
  \unitlength=\PuzzleUnitlength
  \Puzzle@Y=#2
  \begin{picture}(#1,#2)
    \Puzzle@Box@@normal.
}{%
    \PuzzleHook
  \end{picture}\hfill\null\par\noindent
}
```

`\Puzzle@Frame@@normal` The macro `Puzzle@Frame` is used to place additional rectangular regions into the puzzle frame. This frame can contain arbitrary text which is typeset in a centered environment.

This macro takes five arguments. The first two arguments are the coordinates of the upper left corner of the frame. The coordinates are logical coordinates starting from the lower left corner of the puzzle. The next two arguments are the width and the height of the frame given in the number of boxes covered. Finally the fifth argument contains the text which should apear in this frame.

```
\newcommand\Puzzle@Frame@@normal[5]{\put(#1,#2){\framebox(#3,#4){%
      \begin{minipage}{#3\unitlength}\begin{center} #5
      \end{center}\end{minipage}}}}
```

`\Puzzle@Frame@@solution` For the solution the framed ads are simply ignored.

```
\newcommand\Puzzle@Frame@@solution[5]{}
```

`\PuzzleBlackBox` The macro `\PuzzleBlackBox` is called to typeset the black boxes. It should produce a box of at most width and height of `\PuzzleUnitlength`.

```
\newcommand\PuzzleBlackBox{\rule{\PuzzleUnitlength}{\PuzzleUnitlength}}
```

`\Puzzle@Box@@normal` The macro `\Puzzle@Box@@normal` performs all tasks when a box should be typeset in “normal” mode. The arguments are evaluated and the appropriate tpye of box typeset or other actions performed.

```
\newcommand\Puzzle@Box@@normal[2][]{%
  \def\Puzzle@tmp{#2}%
  \if\Puzzle@tmp.
    \Puzzle@X=0\relax \advance\Puzzle@Y-1
  \else
    \ifx\@empty\Puzzle@tmp
    \else
    \if\Puzzle@tmp*
    \put(\Puzzle@X,\Puzzle@Y){\framebox(1,1){\PuzzleBlackBox}}
    \else
    \put(\Puzzle@X,\Puzzle@Y){\framebox(1,1){}}
    \fi
    \fi
    \def\Puzzle@tmp{#1}%
    \ifx\@empty\Puzzle@tmp\else
      \put(\Puzzle@X,\Puzzle@Y){%
        \makebox(1,.95)[tl]{\PuzzleNumberFont\,#1}}%
    \fi
    \advance\Puzzle@X 1
  \fi
}
```

`\Puzzle@Box@@solution` The macro `\Puzzle@Box@@solution` performs all tasks when a box should be typeset in "solution" mode. The arguments are evaluated and the appropriate tpye of box typeset or other actions performed.

```
\newcommand\Puzzle@Box@@solution[2][]{%
  \def\Puzzle@tmp{#2}%
  \if\Puzzle@tmp.
    \Puzzle@X=0\relax \advance\Puzzle@Y-1
  \else
    \ifx\Puzzle@tmp\@empty
    \else\if\Puzzle@tmp*
    \put(\Puzzle@X,\Puzzle@Y){\framebox(1,1){\PuzzleBlackBox}}
    \else
    \put(\Puzzle@X,\Puzzle@Y){\framebox(1,1){\uppercase{#2}}}%
    \fi
    \fi
    \def\Puzzle@tmp{#1}%
    \ifx\Puzzle@tmp\@empty\else
      \ifPuzzle@SolutionNumbered
      \put(\Puzzle@X,\Puzzle@Y){%
        \makebox(1,.95)[tl]{\PuzzleNumberFont\,#1}}%
    \fi
    \fi
    \advance\Puzzle@X 1
  \fi
}
```

6.3 Clues

`\Puzzle@Clue@@normal` The first and the third argument are shown as clue. This macro is used for unsolved puzzles.

```
\newcommand\Puzzle@Clue@@normal[3]{\textsf{#1} #3 }
```

\Puzzle@Clue@@solution In solutions clues are simply suppressed. Thus all three arguments are discarted.

```
\newcommand\Puzzle@Clue@@solution[3]{}
```

Puzzle@Clues@@normal The environment Puzzle@Clues@@normal is mapped to PuzzleClues in "normal" mode. It typesets its contents in a minipage of appropriate half textwidth.

```
\newenvironment{Puzzle@Clues@@normal}[1]{%
  \null\hfill
  \let\Clue\Puzzle@Clue@@normal
  \begin{minipage}[t]{.45\textwidth}%
    \PuzzleClueFont{#1}%
}{\end{minipage}\hfill\null }
```

Puzzle@Clues@@solution The environment Puzzle@Clues@@solution is mapped to PuzzleClues in "solution" mode. It just suppressed any output.

```
\newenvironment{Puzzle@Clues@@solution}[1]{%
  \let\Clue\Puzzle@Clue@@solution
}{}
```

\PuzzleWordsText The macro \PuzzleWordsText is the text typeset at the beginning of the environment PuzzleWords. It takes one argument which is the length of the words listed.

```
\newcommand\PuzzleWordsText[1]{Words of length #1: }
```

Puzzle@Words@@normal The environment Puzzle@Words@@normal will be mapped to the environment PuzzleWords in "normal" mode. It just arranges that words are typeset after the \PuzzleWordsText has shown the length of the words. Finally a new paragraph is started.

```
\newenvironment{Puzzle@Words@@normal}[1]{%
  \PuzzleWordsText{#1}%
  \let\Word\relax
}{\par}
```

Puzzle@Words@@solution The environment Puzzle@Words@@solution will be mapped to the environment PuzzleWords in "solution" mode. It arranges things that the contents is silently ignored.

```
\newenvironment{Puzzle@Words@@solution}[1]{%
  \newcommand\Word[1]{}%
}{}
```

6.4 Numbers

\PuzzleNumbers The macro \PuzzleNumbers will produce a list of boxes with numbers for letters. It is intended for numbered crossword puzzles.

```
\newcommand\PuzzleNumbers[1]{\begingroup
  \@nameuse{Puzzle@@\Puzzle@TYPE}%
  \Puzzle@Y=0
  \Puzzle@X=1
  \unitlength=\PuzzleUnitlength
  \Puzzle@Numbers#1.\endgroup}
```

`\Puzzle@Numbers` The macro `\Puzzle@Numbers` loops through the arguments until it finds a dot. For each argument it produces a box, either with the numbers or with the letters or both, depending on the current settings.

The loop is implemented via recursion. The box is tyepset by the `|` macro which takes care of the current settings. For this purpose this characterhas to be made active temporarily.

```
\begingroup
\catcode`\|=13
\gdef\Puzzle@Numbers#1{%
  \if#1.
    \let\next\relax
  \else
    \begin{picture}(1,1)
      \xdef\X{\the\Puzzle@X}%
      \Puzzle@X=0
      |[\X]{#1}%
    \end{picture}%
    \let\next\Puzzle@Numbers
    \advance\Puzzle@X 1
  \fi
  \next
}
\endgroup
```

`\PuzzleLettersText` The macro `\PuzzleLettersText` contains the text typeset at the beginning of the `\PuzzleLetters` environment.

```
\newcommand\PuzzleLettersText{The following letters are used: }
```

`\PuzzleLetters` The macro `\PuzzleLetters` is intended to show the letters used in a numbered crossword puzzle. The argument is the (sorted) list of characters used.

```
\newcommand\PuzzleLetters[1]{\PuzzleLettersText #1\par}
```

`\Puzzle@TYPE` The macro `\Puzzle@TYPE` contains the type of the puzzle. It is used find the appropriate initialization macro.

```
\newcommand\Puzzle@TYPE{normal}
```

`\PuzzleSolution` The macro `\PuzzleSolution` arranges everything that the following puzzles are typeset in the "solution" mode, i.e. the letters are shown and the clues are suppressed.

This macros has one optional argument which has to be `true` or `false`. This argument determines whether or not the numbers should also be shown in the solution. The default is `false` which means that the numbers are suppressed in the solution.

```
\newcommand\PuzzleSolution[1][false]{%
  \@nameuse{Puzzle@SolutionNumbered#1}%
  \let\PuzzleClues\Puzzle@Clues@@solution
  \let\endPuzzleClues\endPuzzle@Clues@@solution
  \let\PuzzleWords\Puzzle@Words@@solution
  \let\endPuzzleWords\endPuzzle@Words@@solution
  \xdef\Puzzle@TYPE{solution}}
```

\PuzzleUnsolved The macro \PuzzleUnsolved arranges everything that the following puzzles are typeset in the "unsolved" mode, i.e. the letters are suppressed and the clues are shown.

```
\newcommand\PuzzleUnsolved{%
  \let\PuzzleClues\Puzzle@Clues@@normal
  \let\endPuzzleClues\endPuzzle@Clues@@normal
  \let\PuzzleWords\Puzzle@Words@@normal
  \let\endPuzzleWords\endPuzzle@Words@@normal
  \xdef\Puzzle@TYPE{normal}}
```

The boolean Puzzle@SolutionNumbered deternimes whether or not the solution should contain numbers. Initially it is set to "false".

```
\newif\ifPuzzle@SolutionNumbered
\Puzzle@SolutionNumberedfalse
```

Finally we arrange that the default behaviour is to typeset an unsolved crossword puzzle.

```
\PuzzleUnsolved
```

That's all.