maXbox Regex
THE DELPHI SYSTEM

Regular Expressions

EKON 16

REGULAR EXPRESSION

- A pattern of special characters used to match strings in a search
- Typically made up from special characters called metacharacters

- Regular expressions are used thoughout Linux:
  - Utilities: ed, vi, grep, egrep, sed, and awk
  - You can validate e-mail addresses, extract phone numbers or ZIP-codes from web-pages or documents, search for complex patterns in log files and all You can imagine! Rules (templates) can be changed without Your program recompilation! (add IBAN as ex.)

REGULAR EXPRESSION IN DELPHI

- The regular expression engine in Delphi XE is PCRE (Perl Compatible Regular Expression). It's a fast and compliant (with generally accepted regex syntax) engine which has been around for many years. Users of earlier versions of delphi can use it with TPerlRegEx, a delphi class wrapper around it.

  www.softwareschule.ch/maxbox.htm

Regular expressions are used in Editors:

- Search for empty lines: '^$'
- Utilities: ed, vi, grep, egrep, sed, TRex and awk
- The main type in RegularExpressions.pas is TRegEx.
- TRegEx is a record with a bunch of methods and static class methods for matching with regular expressions.

REGEX CORE UNIT IN DELPHI

- I hope it helps! I've always used the RegularExpressionsCore unit rather than the higher level stuff because the core unit is compatible with the unit that Jan Goyvaerts has provided for free for years. That was my introduction to regular expressions. So I forgot about the other unit. I guess there's either a bug or it just doesn't work the way one might expect.

  - Record: G:= TRegex.Match(...)
  - Object: regEx:= TPerlRegEx.Create;
    → Ex. 309_regex_powertester2.txt
  - RegExStudio: TRegExpr.Create

METACHARACTERS

RE Metacharacter	Matches…
.	**Any one character, except new line**
[a-z]	**Any one of the enclosed characters (e.g. a-z)**
*	**Zero or more of preceding character**
? or \?	**Zero or one of the preceding characters**
+ or \+	**One or more of the preceding characters**

- any non-metacharacter matches itself

EKON 16

THE PURPOSE

- There are 3 main operators that use regular expressions:

  1. matching (which returns TRUE if a match is found and FALSE if no match is found.

  2. substitution, which substitutes one pattern of characters for another within a string

  3. split, which separates a string into a series of substrings

- Regular expressions are composed of characters, character classes, metacharacters, quantifiers, and assertions.

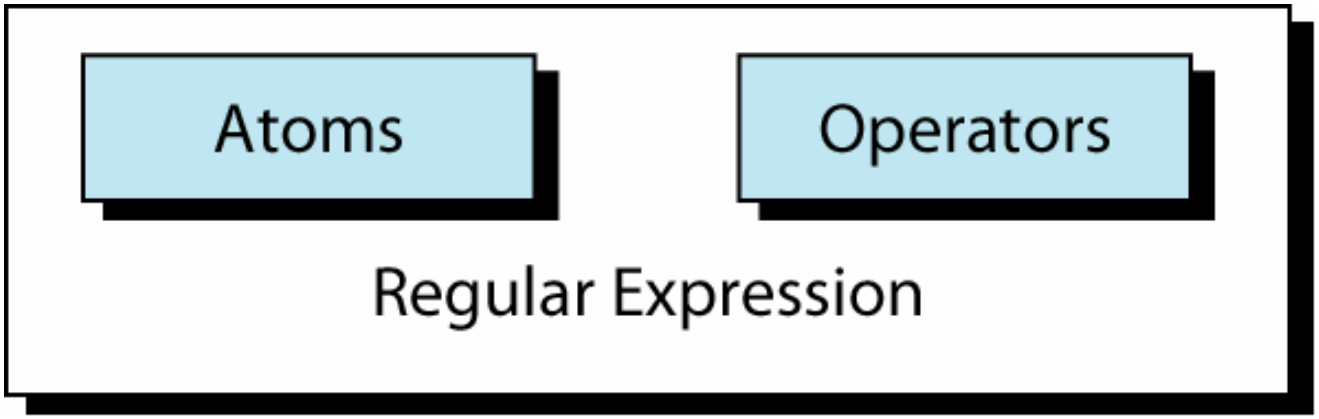
EKON 16

MORE METACHARACTERS

RE Metacharacter	Matches…
^	**beginning of line**
$	**end of line**
char	**Escape the meaning of *char* following it**
[^]	**One character <u>not</u> in the set**
\<	**Beginning of word anchor**
\>	**End of word anchor**
( ) or \(\)	**Tags matched characters to be used later (max = 9)**
| or |	**Or grouping**
x\{m\}	**Repetition of character x, m times (x,m = integer)**
x\{m,\}	**Repetition of character x, at least m times**
x\{m,n\}	**Repetition of character x between m and m times**

EKON 16

Regular Expression

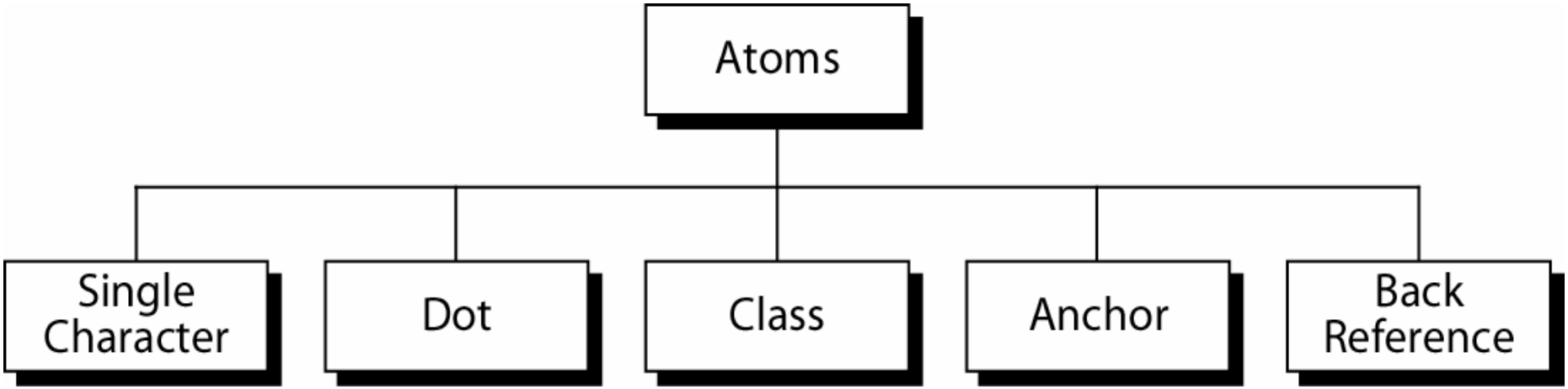


An atom specifies <u>what</u> text is to be matched and <u>where</u> it is to be found.

An operator combines regular expression atoms.

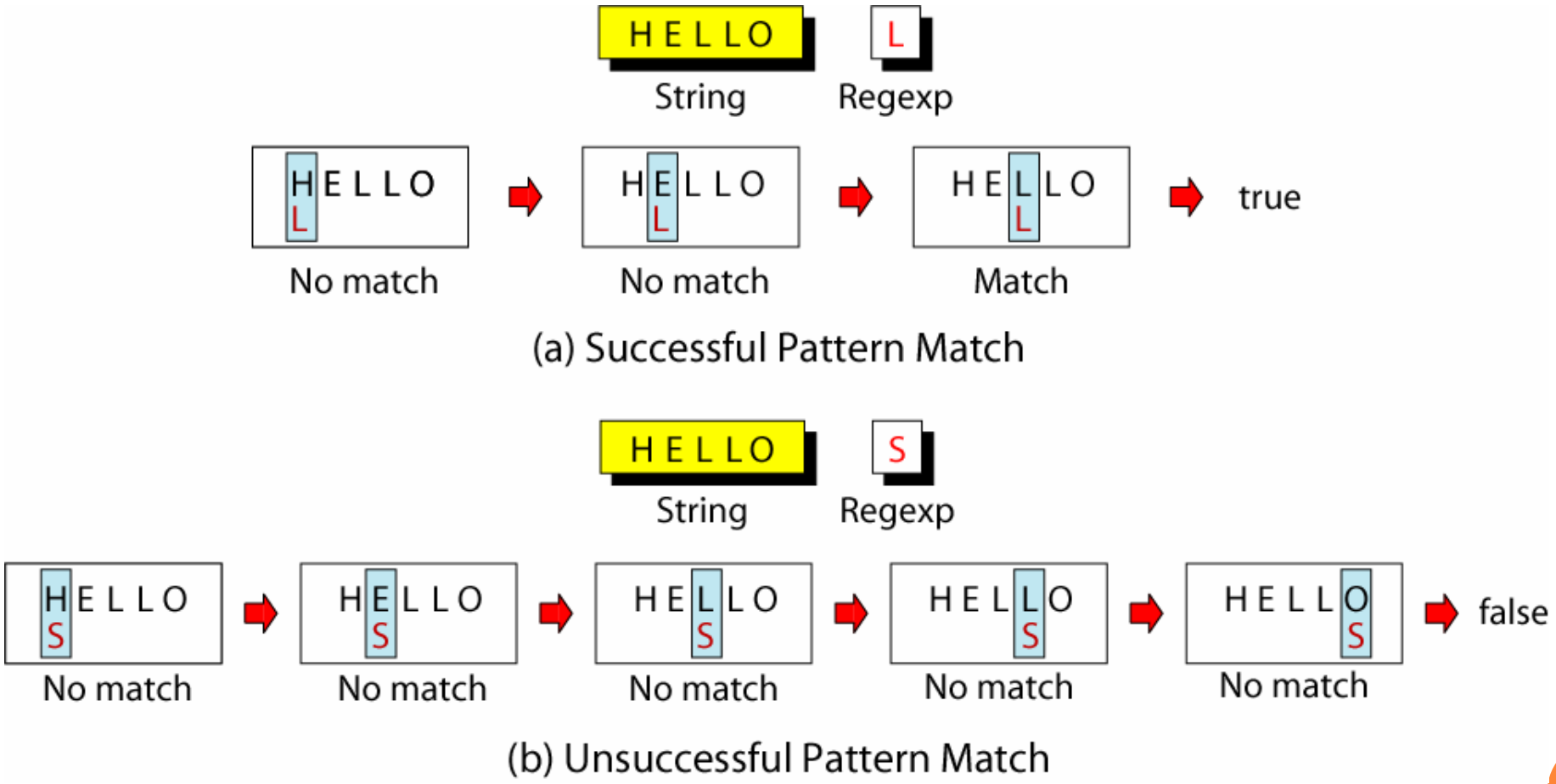
Atoms

An atom specifies <u>what</u> text is to be matched and <u>where</u> it is to be found.



Single-Character Atom

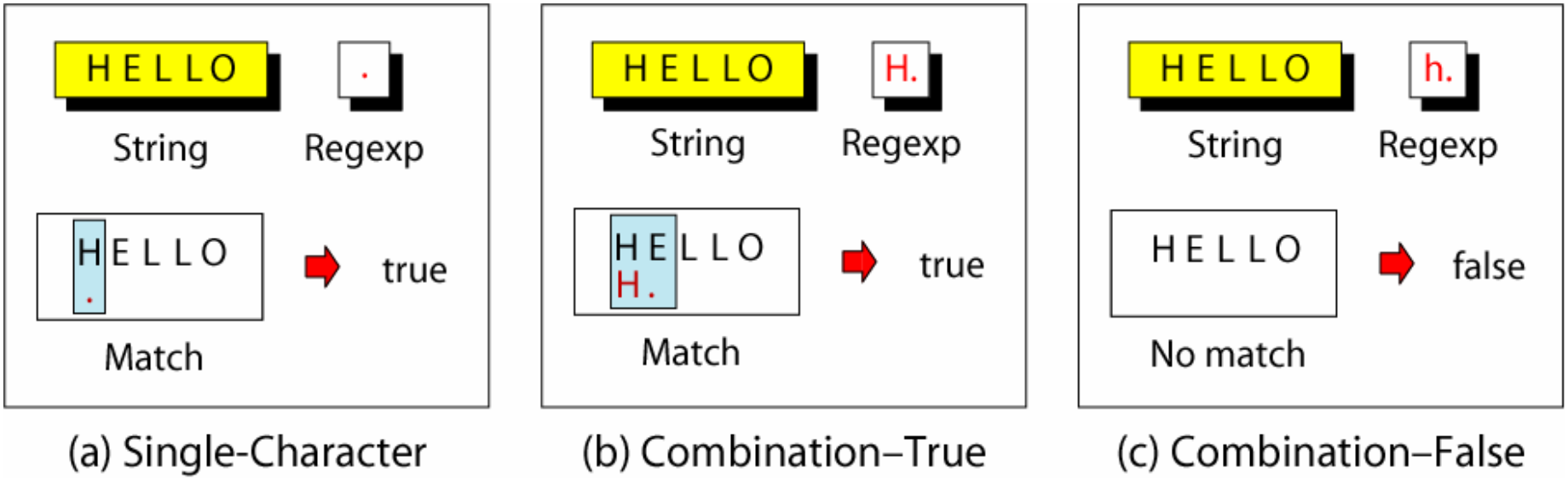
A single character matches itself



EKON 16

Dot Atom

matches any single character except for a new line character (\n)

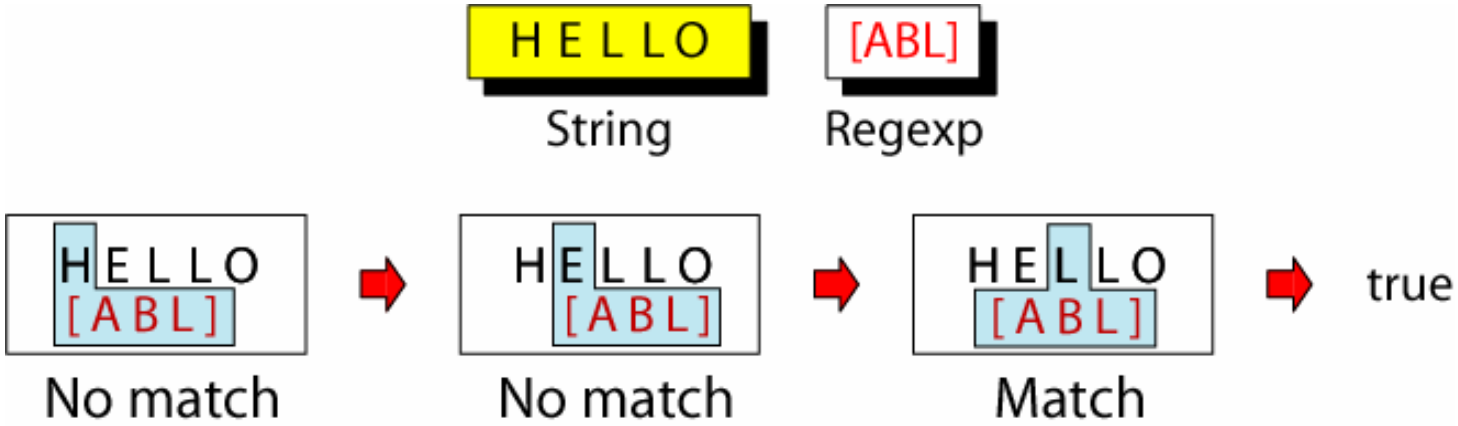


(a) Single-Character (b) Combination–True (c) Combination–False

Class Atom

matches only single character that can be any of the characters defined in a set:

<u>Example:</u> [ABC] matches either A, B, or C.



Notes:

1) A range of characters is indicated by a dash, e.g. [A-Q]
2) Can specify characters to be excluded from the set, e.g. [^0-9] matches any character other than a number.

Example: Classes

RegExpr	Means
[A-H]	[ABCDEFGH]
[A-Z]	Any uppercase alphabetic
[0-9]	Any digit
[[a]	[or a
[0-9\-]	digit or hyphen

RegExpr	Means
[^AB]	Any character except A or B
[A-Za-z]	Any alphabetic
[^0-9]	Any character except a digit
[]a]	] or a
[^\^]	Anything except^

EKON 16

REPLACEMENT

```
procedure DelphiPerlRegex;
begin
  with TPerlRegex.create do try
    Options:= Options + [preUnGreedy];
    Subject:= 'I like to sing out at Foo bar';
    RegEx:= '([A-Za-z]+) bar';
    Replacement:= '\1 is the name of the bar I like';
    if Match then ShowMessageBig(ComputeReplacement);
  finally
    Free;
  end;
end;
```

Anchors

Anchors tell where the next character in the pattern must be located in the text data.

Anchor	Means	Example
^	Beginning of line	One line of text.\n
$	End of line	One line of text.\n
\<	Beginning of word	One line of text.\n
\>	End of word	One line of text.\n

EKON 16

BACK REFERENCES: \N

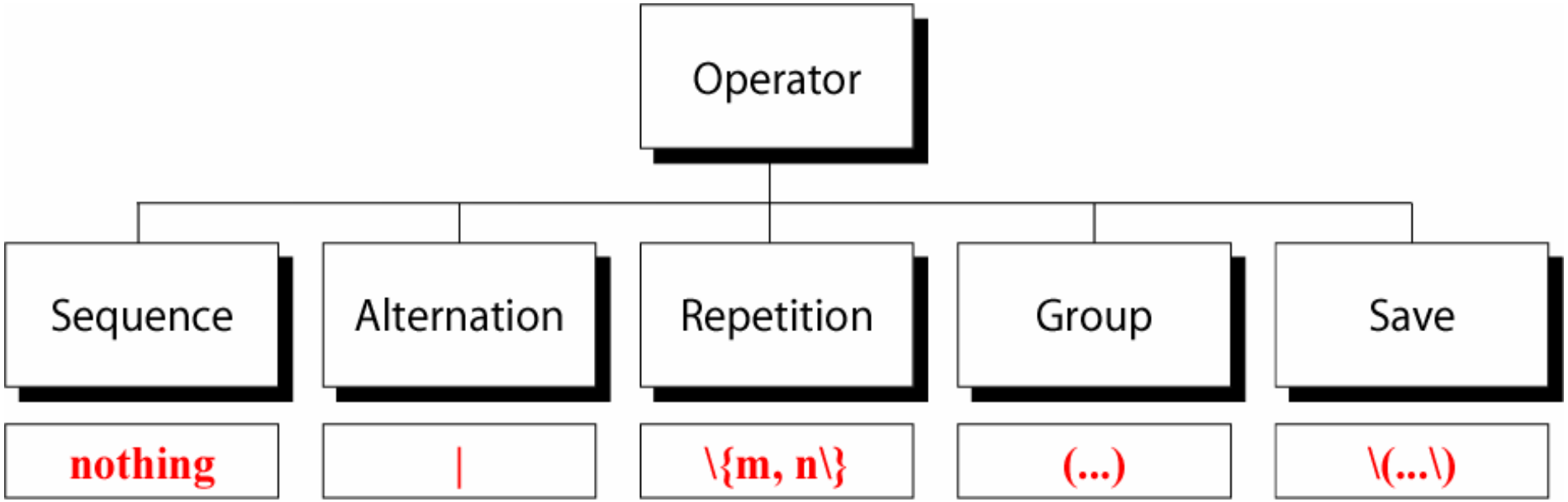
- used to retrieve saved text in one of nine buffers
- can refer to the text in a saved buffer by using a back reference:

  ex.: \1 \2 \3 ...\9

- more details on this later
- rex:= '.*(es).*\1.*'; //subpattern → Ex.
-

EKON 16

Operators



EKON 16

Sequence Operator

In a sequence operator, if a series of atoms are shown in a regular expression, there is no operator between them.

`dog`	matches the pattern "dog"
`a..b`	matches "a" , any two characters, and "b"
`[2-4][0-9]`	matches a number between 20 and 49
`[0-9][0-9]`	matches any two digits
`^$`	matches a blank line
`^.$`	matches a one-character line
`[0-9]-[0-9]`	matches two digits separated by a "-"

EKON 16

Alternation Operator: | or \|

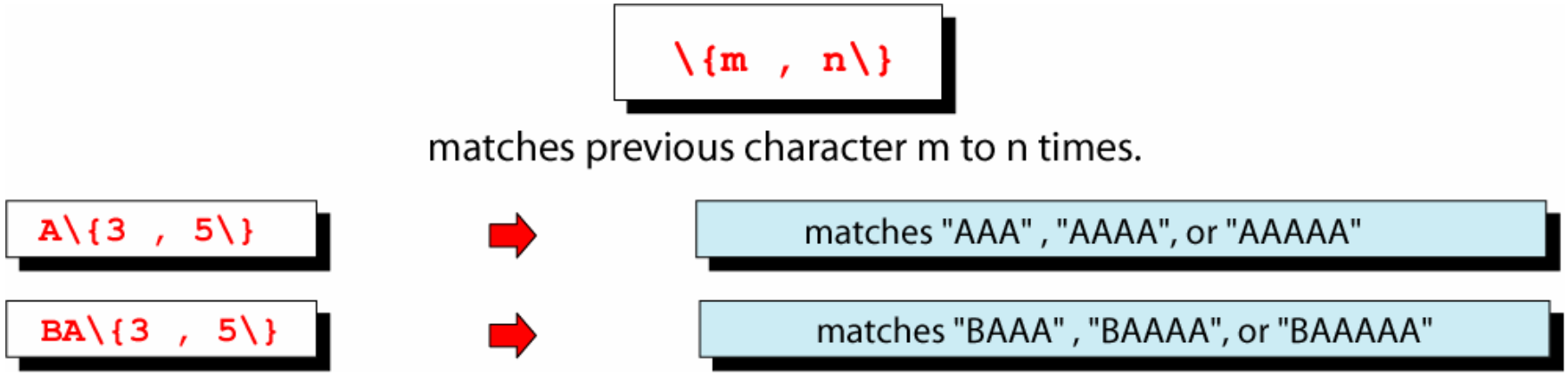
operator (| or \|) is used to define one
or more alternatives

`UNIX|unix`	→	matches "UNIX" or "unix"
`Ms|Miss|Mrs`	→	matches "Ms" or "Miss" or "Mrs"

Note: depends on version of

Repetition Operator: \{…\}

The repetition operator specifies that the atom or expression immediately before the repetition may be repeated.



Basic Repetition Forms

Formats

`\{m\}`	➡	matches previous atom exactly m times
`\{m, \}`	➡	matches previous atom m times or more
`\{, n\}`	➡	matches previous atom n times or less

Examples

`CA\{5\}`	➡	`CAAAAA`
`CA\{3,\}`	➡	`CAAA, CAAAA, CAAAAA, …`
`CA\{,2\}`	➡	`C, CA, CAA`

EKON 16

Short Form Repetition Operators:

Formats

`*`	special case: matches previous atom zero or more times
`+`	special case: matches previous atom one or more times
`?`	special case: matches previous atom 0 or one time only

Examples

`BA*`	B, BA, BAA, BAAA, BAAAA, . . .
`B.*`	B, BA . . . BZ, BAA . . . BZZ, BAAA . . . BZZZ, . . .
`.*`	zero or more characters
`.+`	one or more characters
`[0-9]?`	zero or one digit

EKON 16

Group Operator

In the group operator, when a group of characters is enclosed in parentheses, the next operator applies to the whole group, not only the previous characters.

Regexp		Matches
`A(BC)\{3\}`	➡	ABCBCBC
`(F(BC)\{2\}G)\{2\}`	➡	FBCBCGFBCBCG

Note: depends on version of “unit”
use \(and \) instead

DELPHI DETAIL AND EXAMPLES

- For new code written in Delphi XE, you should definitely use the RegularExpressions unit that is part of Delphi rather than one of the many 3rd party units that may be available. But if you're dealing with UTF-8 data, use the RegularExpressionsCore unit to avoid needless UTF-8 to UTF-16 to UTF-8 conversions.

COMMONLY USED “D” OPTIONS:

```
-c   CL.AddTypeS('TPerlRegExOption', ‘
     ( preCaseLess,
      preMultiLine,
      preSingleLine,
      preExtended,
      preAnchored,
      preUnGreedy,
      preNoAutoCapture )');
```

EXAMPLE: WITH \<north \>

```
% cat grep-datafile
northwest        NW      Charles Main            300000.00
western          WE      Sharon Gray             53000.89
southwest        SW      Lewis Dalsass           290000.73
southern         SO      Suan Chin               54500.10
southeast        SE      Patricia Hemenway       400000.00
eastern          EA      TB Savage               440500.45
northeast        NE      AM Main Jr.             57800.10
north            NO      Ann Stephens            455000.50
central          CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print the line if it contains the word "north".

```
% grep '\<north\>' grep-datafile
north            NO      Ann Stephens            455000.50
```

C309_regex_powertester2.txt

EXAMPLE: EGREP WITH +

```
% cat grep-datafile
northwest        NW      Charles Main            300000.00
western          WE      Sharon Gray             53000.89
southwest        SW      Lewis Dalsass           290000.73
southern         SO      Suan Chin               54500.10
southeast        SE      Patricia Hemenway       400000.00
eastern          EA      TB Savage               440500.45
northeast        NE      AM Main Jr.             57800.10
north            NO      Ann Stephens            455000.50
central          CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print all lines containing one or more 3's.

```
% egrep '3+' grep-datafile
northwest        NW      Charles Main            300000.00
western          WE      Sharon Gray             53000.89
southwest        SW      Lewis Dalsass           290000.73
```

Note: line works with \.*

C309_regex_powertester2.txt

EXAMPLE: EGREP WITH RE: ?

```
% cat grep-datafile
northwest        NW      Charles Main            300000.00
western          WE      Sharon Gray             53000.89
southwest        SW      Lewis Dalsass           290000.73
southern         SO      Suan Chin               54500.10
southeast        SE      Patricia Hemenway       400000.00
eastern          EA      TB Savage               440500.45
northeast        NE      AM Main Jr.             57800.10
north            NO      Ann Stephens            455000.50
central          CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

C309_regex_powertester2.txt

Print all lines containing a 2, followed by zero or one period, followed by a number.

```
% egrep '2\.?[0-9]' grep-datafile
southwest        SW      Lewis Dalsass           290000.73
```


EKON 16

Note: Delphi PCRE grep Test

EXAMPLE: FGREP

```
% cat grep-datafile
northwest        NW      Charles Main            300000.00
western          WE      Sharon Gray             53000.89
southwest        SW      Lewis Dalsass           290000.73
southern         SO      Suan Chin               54500.10
southeast        SE      Patricia Hemenway       400000.00
eastern          EA      TB Savage               440500.45
northeast        NE      AM Main Jr.             57800.10
north            NO      Ann Stephens            455000.50
central          CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Find all lines in the file containing the literal string "[A-Z]****[0-9]..$5.00". All characters are treated as themselves. There are no special characters.

```
% fgrep '[A-Z]****[0-9]..$5.00' grep-datafile
Extra [A-Z]****[0-9]..$5.00
```

C309_regex_powertester2.txt

EXAMPLE: Mail Finder

```
procedure delphiRegexMailfinder;
begin
  // Initialize a test string to include some email addresses. This
would normally be your eMail.
  TestString:= '<one@server.domain.xy>, another@otherserver.xyz';
  PR:= TPerlRegEx.Create;
  try
    PR.RegEx:= '\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b';
    PR.Options:= PR.Options + [preCaseLess];
    PR.Compile;
    PR.Subject:= TestString; // <-- tell PR where to look for matches
    if PR.Match then begin
       WriteLn(PR.MatchedText); // Extract first address
       while PR.MatchAgain do
       WriteLn(PR.MatchedText); // Extract subsequent addresses
    end;
  finally
    PR.Free;
  end;
  //Readln;
end;
```

EXAMPLE: Songfinder

```
with TRegExpr.Create do try
  gstr:= 'Deep Purple';
  modifierS:= false; //non greedy
  Expression:= '#EXTINF:\d{3},'+gstr+' - ([^\n].*)';
     if Exec(fstr) then
      Repeat
         writeln(Format ('Songs of ' +gstr+': %s', [Match[1]]));
        (*if AnsiCompareText(Match[1], 'Woman') > 0 then begin
             closeMP3;
     PlayMP3('..\EKON_13_14_15\EKON16\06_Woman_From_Tokyo.mp3');
         end;*)
      Until Not ExecNext;
  finally Free;
end;
```

EXAMPLE: GREP WITH \CHAR

```
% cat grep-datafile
northwest        NW      Charles Main            300000.00
western          WE      Sharon Gray             53000.89
southwest        SW      Lewis Dalsass           290000.73
southern         SO      Suan Chin               54500.10
southeast        SE      Patricia Hemenway       400000.00
eastern          EA      TB Savage               440500.45
northeast        NE      AM Main Jr.             57800.10
north            NO      Ann Stephens            455000.50
central          CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print all lines containing the number 5, followed by a literal period and any single character.

```
% grep '5\..' grep-datafile
Extra [A-Z]****[0-9]..$5.00
```

EKON 16

EXAMPLE: HTTP RegEx[ ]

```
% cat get russian rouble rate - datafile

procedure getHttpREGEX(Sender: TObject);
var http1: TIDHTTP;
    htret: string;
begin
  http1:= TIDHTTP.Create(self);
  htret:= HTTP1.Get('http://win.www.citycat.ru/finance/finmarket/_CBR/');
  //writeln(htret);
  with TRegExpr.Create do try
     Expression:= russTemplate;
     if Exec(htret) then begin
       //if output success
       writeln(Format ('Russian rouble rate at %s.%s.%s: %s',
                 [Match [2], Match [1], Match [3], Match [4]]));
      end;
      //writeln(dump)
    finally Free;
   end;
   //text2html
   //writeln('deco: '+#13+#10+DecorateURLs(htret,[durlAddr, durlPath]))
end;
```

EXAMPLE: GREP WITH X\{M\}

```
% cat grep-datafile
northwest        NW      Charles Main            300000.00
western          WE      Sharon Gray             53000.89
southwest        SW      Lewis Dalsass           290000.73
southern         SO      Suan Chin               54500.10
southeast        SE      Patricia Hemenway       400000.00
eastern          EA      TB Savage               440500.45
northeast        NE      AM Main Jr.             57800.10
north            NO      Ann Stephens            455000.50
central          CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print all lines where there are at least six consecutive numbers followed by a period.

```
% grep '[0-9]\{6\}\.' grep-datafile
northwest        NW      Charles Main            300000.00
southwest        SW      Lewis Dalsass           290000.73
southeast        SE      Patricia Hemenway       400000.00
eastern          EA      TB Savage               440500.45
north            NO      Ann Stephens            455000.50
central          CT      KRush                   575500.70
```

EXAMPLE: Extract Phones\<city code 812

```
% cat grep-delphi-maXbox_datafile

procedure ExtractPhones(const AText: string; APhones: TStrings);
  begin
  with TRegExpr.Create do try
     Expression := '(\+\d*)?(\((\d+)\)*)?(\d+(-\d*)*)';
     if Exec (AText) then
       REPEAT
         if Match[3] = '812'
          then APhones.Add(Match [4]);
       UNTIL not ExecNext;
     finally Free;
   end;
end;

writeln('Formula Gauss : '+
     floatToStr(maXcalc('1/SQRT(2*PI*3^2)*EXP((-
                                        0.0014^2)/(2*3^2))')));
```

SUMMARY

- regular expressions
- Delphi Implementation
- for PCRE family of commands
- Examples to train your brain