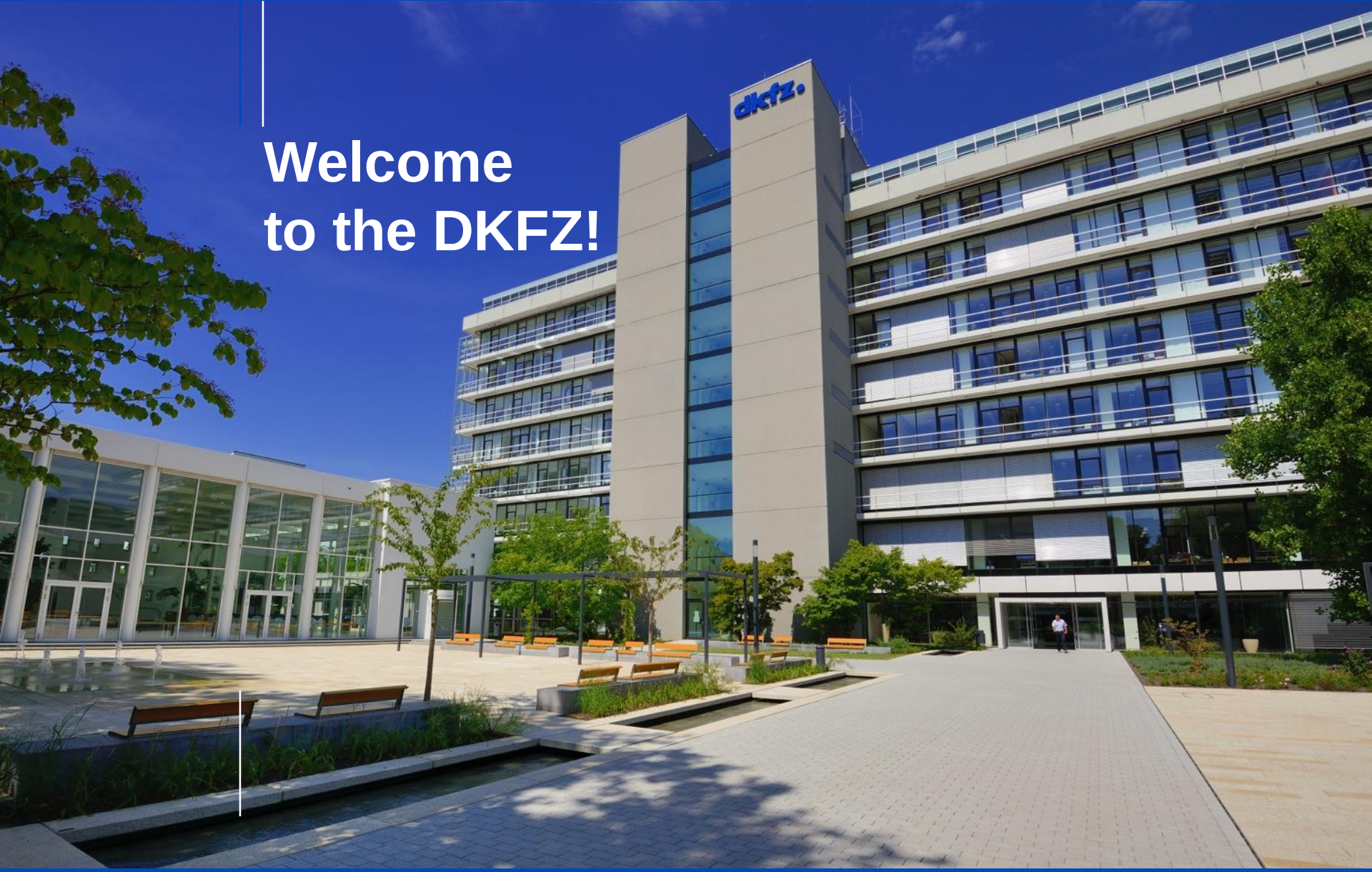
Regular Expressions

Bugsquashing Seminar
Jens Petersen
Medical Image Computing (E132)



Welcome to the DKFZ!

dkfz.
GERMAN CANCER RESEARCH CENTER
IN THE HELMHOLTZ ASSOCIATION

50 Years – Research for
A Life Without Cancer

Regular Expressions (RegExp, Regex)

Used to define string patterns for pattern matching

1. Find occurrences of pattern in string (and replace, split, …)
2. Check whether string matches pattern

Sidenote:
Regular Expressions form regular languages.
RegExp/Regex more expressive, but not regular.

C++ Implementations

1. `std::regex`
   C++11, GNU ≥ 4.9

2. `boost::regex`
   Additional dependency

3. `QRegExp/QRegularExpression`
   QRegularExpression since Qt 5.0

Usually compiled at runtime

How to define patterns

Define patterns using strings:

"^(Hi|Hello|Dear)\s\w+,"

Literals:

"Hello" → *"Hello"*

Quantifiers:

"()?" → zero or one times

"()"* → any number of times

"()+" → one or more times

"(){n,m}" → between n and m times

"ab"* → *"a"*, *"ab"*, *"abb"*, *"abbb"*, …

Ranges and grouping

Specify range or set with brackets:

"[abc]" → *"a"*, *"b"*, *"c"*

"[a-zA-Z0-9]" → one alphanumeric character

"[-a-c]" → *"-"*, *"a"*, *"b"*, *"c"*

"[^abc]" → any character except *"a"*, *"b"* or *"c"*

Define subpatterns using parentheses:

"(ab)"* → *""*, *"ab"*, *"abab"*, …

"(Hi|Hello|Dear)" → *"Hi"*, *"Hello"*, *"Dear"*

Metacharacters (selection)

"^" → beginning of line (or string)
"$" → end of line (or string)
"\A", "\Z" → beginning, end of string

"\d" / "\D" → digit / non-digit character
"\w" / "\W" → word (alphanumeric) / non-word character

"\s" / "\S" → whitespace / non-whitespace character
"\t", "\n" → tab, newline character
".", "\." → any character, dot

Example: DICOM UIDs

Format

Any positive number of non-negative integers, separated by dots.

Maximum length of 64 characters including dots

(not always enforced)

Non-zero integers must not contain leading zeros

1.2.3.4.5

123.0.45

123.04.5

12.a.4.5,6

Example: DICOM UIDs

```
#include <QtGlobal>
#if QT_VERSION >= 0x050000
  #include <QRegularExpression>
#else
  #include <QRegExp>
#endif


bool QmitkDicomBulkRetrieveWidget::IsValidDicomUID(const QString& uid)
{
#if QT_VERSION >= 0x050000
  QRegularExpression uidPattern("\\A([1-9][0-9]*|0)(\\.([1-9][0-9]*|0))*\\z");
  return uidPattern.match(uid).hasMatch();
#else
  QRegExp uidPattern("([1-9][0-9]*|0)(\\.([1-9][0-9]*|0))*");
  return uidPattern.exactMatch(uid);
#endif
}
```

More advanced functionality

Independent subexpressions

Backreferences

Recursive patterns

Conditional matching

Look-ahead and look-behind assertions

…

Thank you for your attention!

Further information on www.dkfz.de

dkfz. GERMAN CANCER RESEARCH CENTER IN THE HELMHOLTZ ASSOCIATION

50 Years – Research for A Life Without Cancer