Reguläre Ausdrücke

C++ Übung am 30. Juni 2016

Reguläre Ausdrücke

- `Includes einbinden:`
- `#include <boost/regex.hpp> /`
  `#include <regex>`

- `Regex-Objekt erzeugen:`
- `boost::regex re("(\\w+|\\s)+[.?!]$");`

- `std::wregex re (L"(sub)(.*)");`

Reguläre Ausdrücke

- `bool regex_match(myString, RegexObj)`
- `bool regex_match(str.begin(), str.end(), RegexObj)`
- `bool regex_search(startOfSearch, endOfSearch, MatchResults, re)`
- `bool regex_replace(myString, RegexObjekt, Ersetzung)`

regex_match

```
#include <iostream>
#include <regex>

int main () {
  if (std::regex_match ("subject", std::regex("(sub)(.*)") ))
    std::wcout << "string literal matched\n";

  std::wstring s (L"subject");
  std::wregex re (L"(sub)(.*)");

  if (std::regex_match (s, re))
    std::wcout << L"wstring object matched\n";

  if ( std::regex_match ( s.begin(), s.end(), re ) )
    std::wcout << L"range matched\n";

  std::wsmatch sm;    // same as std::match_results<wstring::const_iterator> sm;
  std::regex_match (s,sm,re);
  std::wcout << L"wstring object with " << sm.size() << " matches\n";

  std::regex_match ( s.cbegin(), s.cend(), sm, re);
  std::wcout << L"range with " << sm.size() << L" matches\n";

  std::wcout << L"the matches were: ";
  for (unsigned i = 0; i < sm.size(); ++i) {
    std::wcout << L"[" << sm[i] << L"] ";
  }
  std::wcout << std::endl;
  return 0;
}
```

```
string literal matched
wstring object matched
range matched
wstring object with 3 matches
range with 3 matches
the matches were: [subject] [sub] [ject]
```

regex_search

```
#include <iostream>
#include <regex>

int main () {
  std::wstring str (L"this subject has a submarine as a subsequence");
  std::wsmatch mat;
  std::wregex re (L"\\b(sub)([^ ]*)");   // matches words beginning by "sub"

  std::wcout << L"Target sequence: " << str << std::endl;
  std::wcout << L"Regular expression: /\\b(sub)([^ ]*)/" << std::endl;
  std::wcout << L"The following matches and submatches were found:" << std::endl;

  while (std::regex_search (str, mat, re)) {
    for (auto x:mat) std::wcout << x << L" ";
    std::wcout << std::endl;
    str = mat.suffix().str();
  }

  return 0;
}
```

```
Target sequence: this subject has a submarine as a subsequence
Regular expression: /\b(sub)([^ ]*)/
The following matches and submatches were found:
subject sub ject
submarine sub marine
subsequence sub sequence
```

regex_replace

```
#include <iostream>
#include <regex>

int main () {
  std::wstring s (L"there is a subsequence in the string\n");
  std::wregex re (L"\\b(sub)([^ ]*)");   // matches words beginning by "sub"

  // print result of replace
  std::wcout << std::regex_replace(s,re,L"sub-$2");

  // write result to result wstring
  std::wstring result;
  std::regex_replace(std::back_inserter(result), s.begin(), s.end(), re, L"$2");
  std::wcout << result;

  std::wcout << std::endl;
  return 0;
}
```

```
there is a sub-sequence in the string
there is a sequence in the string
```

Klausurübung

Implementieren Sie die Klassendeklaration und drei Funktionen der Klasse Lexikon, die Elemente vom Typ wstring speichert. Die Methode rm_punct() soll am Anfang und Ende des Wortes Punktuationszeichen entfernen. Die Methode insert() soll ein Wort in das Lexikon mit dem Wert 1 eintragen, bzw. wenn das Wort schon im Lexikon existiert den Wert erhöhen. Die Methode head soll die ersten n Elemente des sortieren Lexikons ausgeben.
Überlegen Sie sich zuerst von welchem Container words sein muss und deklarieren Sie diesen.

-> Lückentext auf der Website