正则表达式（regular expression）程序库regex，允许使用通配符（wildcar）和pattern（模式）来查找和替换string中的字符。

文法：用来定义正则表达式的语法。

例：程序stl\_test85

*regex* reg1("<.\*>.\*</.\*>");

*string* source = "<tag>value</tag>";

bool found = *regex\_match*(source, reg1);

out(found); // found

*regex* reg2("<(.\*)>.\*</\\1>");

found = *regex\_match*(source, reg2);

out(found); // found

*regex* reg3("<\\(.\*\\)>.\*</\\1>", *regex\_constants*::*grep*);

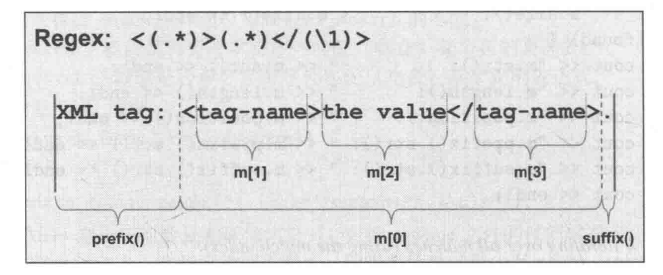
found = *regex\_match*(source, reg3);

out(found); // found

found = *regex\_match*(source, *regex*("<(.\*)>.\*</\\1>"));

out(found); // found

获取正则表达式的匹配细节：



例：程序stl\_test86

*string* data = "XML tag: <tag-name>the value </tag-name>.";

*cout* << "data: " << data << *endl*;

*cout* << *endl*;

*smatch* m; // for returned details of the match

// regex\_search：部分匹配

bool found = *regex\_search*(data, m,

*regex*("<(.\*)>(.\*)</(\\1)>"));

// print match details

*cout* << "m.empty(): " << *boolalpha* << m.*empty*() << *endl*;

*cout* << "m.size(): " << m.*size*() << *endl*;

if (found)

{

// 匹配合格的所有字符

*cout* << "m.str(): " << m.*str*() << *endl*;

// 匹配合格的所有字符长度

*cout* << "m.length(): " << m.*length*() << *endl*;

// 匹配合格的整体字符的位置

*cout* << "m.position(): " << m.*position*() << *endl*;

// 第1个匹配合格的字符前的所有字符

*cout* << "m.prefix().str(): " << m.*prefix*().*str*() << *endl*;

// 最后一个匹配合格的字符后的所有字符

*cout* << "m.suffix().str(): " << m.*suffix*().*str*() << *endl*;

*cout* << *endl*;

// iterating over all matches(using the match index)

for (int i = 0; i < m.*size*(); ++i)

{

*cout* << "m[" << i << "].str(): " << m[i].*str*() << *endl*;

*cout* << "m.str(" << i << "): " << m.*str*(i) << *endl*;

*cout* << "m.position(" << i << "): " << m.*position*(i)

<< *endl*;

}

*cout* << *endl*;

// iterating over all matches(using iterators)

*cout* << "matches: " << *endl*;

for (auto pos = m.*begin*(); pos != m.*end*(); ++pos)

{

*cout* << " " << \*pos << " ";

*cout* << "(length: " << pos->*length*() << ")" << *endl*;

}

}

输出为：

data: XML tag: <tag-name>the value </tag-name>.

m.empty(): false

m.size(): 4

m.str(): <tag-name>the value </tag-name>

m.length(): 31

m.position(): 9

m.prefix().str(): XML tag:

m.suffix().str(): .

m[0].str(): <tag-name>the value </tag-name>

m.str(0): <tag-name>the value </tag-name>

m.position(0): 9

m[1].str(): tag-name

m.str(1): tag-name

m.position(1): 10

m[2].str(): the value

m.str(2): the value

m.position(2): 19

m[3].str(): tag-name

m.str(3): tag-name

m.position(3): 31

matches:

<tag-name>the value </tag-name> (length: 31)

tag-name (length: 8)

the value (length: 10)

tag-name (length: 8)

正则表达式迭代器：

例：程序stl\_test87

*string* data = "<person>\n"

"<first>Nico</first>\n"

"<last>Josuttis</last>\n"

"</person>\n";

*regex* reg("<(.\*)>(.\*)</(\\1)>");

*sregex\_iterator* pos(data.*cbegin*(), data.*cend*(), reg);

*sregex\_iterator* end; // 默认的结尾

for (; pos != end; ++pos)

{

*cout* << "match: " << pos->*str*() << *endl*;

*cout* << " tag: " << pos->*str*(1) << *endl*;

*cout* << " value: " << pos->*str*(2) << *endl*;

}

输出为：

match: <first>Nico</first>

tag: first

value: Nico

match: <last>Josuttis</last>

tag: last

value: Josuttis