Module05-02
C++ Boost：字符串与文字处理

C++ Boost - 字符串和文字处理

- 容器相关
- → 字符串和文字处理
- 正则表达式
- 智能指针
- 函数对象相关
- 序列化
- 日期与时间
- 多线程
- 网络

字符串和文字处理 - boost.lexical_cast

- 字符串和文字处理：
  - boost.lexical_cast
  - boost.format
  - boost.string_algo

字符串和文字处理 - boost.lexical_cast

- 关于 boost.lexical_cast
  - 从字符串转换成数值或将数值转换成字符串，注意：与 C++ 的 4 种类型的 cast 不同， boost.lexical_cast 不是操作符
  - 接口：

```
namespace boost {
class bad_lexical_cast;

template<typename Target, typename Source>
Target lexical_cast(const Source& arg);
}
```

  - lexical_cast 对 Target 和 Source 类型的要求：
    - Source 是可输出类型，即可以通过 operator<< 输出到输出流
    - Target 是可输入类型，即可以通过 operator>> 从输入流中输入
    - Target 必须有可用的复制构造、默认构造函数

字符串和文字处理 - boost.lexical_cast

- 示例

```
string s("3.14e12");
double d = boost::lexical_cast<double>(s);
cout << d << endl;  // 3.14e+12

try {
    int i = strtol("ff", 0, 16);
    cout << i << endl;  // 255

    // throws bad_lexical_cast
    int i2 = boost::lexical_cast<int>("ff");
    cout << i2 << endl;
} catch (boost::bad_lexical_cast& e) {
    cout << e.what() << endl;
}

string ns = boost::lexical_cast<string>(0xff);
cout << ns << endl; // 255
```

字符串和文字处理 - boost.format

- 字符串和文字处理：
  - boost.lexical_cast
  - boost.format
  - boost.string_algo

字符串和文字处理 - boost.format

- 关于 boost.format
  - 不同于 C++ 的 I/O 流使用操控符来控制输出格式， boost.format 采用了与 C 库的 printf 一族函数的风格来控制输出的格式
  - 与 printf 一族函数不同的是， boost.format 不支持不确定个数参数，但也支持任意数量的参数，同时类型更安全
  - 另外， boost.format 还支持自定义类型的格式化输出
  - format 语法：
    - `format(format-string) % arg1 % arg2 % .. % argN`

```
// 输出: 64 hello 64
cout << format ("%1% %2% %1%\n") % 64 % "hello";
format fmt("%1% %2%\n");
fmt % "str" % 128;
cout << fmt; // 输出: str 128
cout << format("%s %d\n") % "tiger" % 12; // printf-style
```

字符串和文字处理 - boost.format

- format 格式化语法
  - printf-style 语法，如： %02d
    具体的语法明细请参考 manpages： man 3 printf
  - %|*spec*|，其中 spec 遵循 printf 语法规则，如：

```
cout << format("(x,y) = (%+5d,%+5d) \n") % -23 % 35;
cout << format("(x,y) = (%|+5|,%|+5|) \n") % -23 % 35;
```

    都输出：(x,y) = (-23, +35) \n
  - %N%，类似于 printf 的 %N$d 等，如

```
// 输出: 64 hello 64
cout << format("%1% %2% %1%\n") % 64 % "hello";
cout << format("%1$d %2$s %1$d") % 64 % "hello";
```

    但不需关注输出的类型！

字符串和文字处理 - boost.format

- 使用流操纵符影响输出格式

```
using boost::format;
using boost::io::group;
cout << format("%1% %2% %1%\n") % group(hex, showbase, 40) %
50;
// prints "0x28 50 0x28\n"
```

字符串和文字处理 - boost.format

- 示例
  - (DEMO Using Boost Examples)

字符串和文字处理 - boost.format

- boost.format 异常的捕获

```
try {
    // 下面的format只需1个参数，传入一个以上的参数，将导致异常
    format(" %1% %1% ") % 101 % 102;
} catch (boost::io::too_many_args& exc) {
    cerr << exc.what() << endl;
}

try {
    // 下面的format需要3个参数，少传参数也将导致异常
    cout << format("%1%, %2%, %3%") % 12 % "hello";
    // 甚至第一、第二个参数不需要，也要传3个参数
    cerr << format(" %|3$| ") % 101;
} catch (boost::io::too_few_args& exc) {
    cerr << exc.what() << endl;
}
```

字符串和文字处理 - boost.format

- boost.format 异常的忽略

```
// fmter需要3个参数
format fmter("%1% %2% %3% %2% %1% \n");
// 忽略too_many_args异常
fmter.exceptions(boost::io::all_error_bits ^
(boost::io::too_many_args_bit));
// 现在即使传入3个以上参数，也不会出错
cout << fmter % 1 % 2 % 3 % 4 % 5 % 6;
```

字符串和文字处理 - boost.string_algo

- 字符串和文字处理：
  - boost.lexical_cast
  - boost.format
  - boost.string_algo

字符串和文字处理 - boost.string_algo

- 关于 boost.string_algo
  - boost 的字符串算法库提供如：
    - 字符大小写转换
    - 字符串分割 split
    - 修剪字符串首尾的空白字符 trim
    - 基于正则表达式的 find 和 replace
    - …
  - 四个组成部分：
    - 一族算法函数
    - Finders 和 Formatters
    - Find 迭代器
    - 字符类别判断式

字符串和文字处理 - boost.string_algo

- 算法列表

算法名称	描述	相关函数
大小写转换		
to_upper	将字符串字母转换成大写	to_upper_copy() to_upper()
to_lower	将字符串字母转换成小写	to_lower_copy() to_lower()
修剪首尾空白字符		
trim_left	去除字符串头部的空白字符	trim_left_copy_if() trim_left_if() trim_left_copy() trim_left()
trim_right	去除字符串尾部的空白字符	trim_right_copy_if() trim_right_if() trim_right_copy() trim_right()
trim	修剪首尾空白字符	trim_copy_if() trim_if() trim_copy() trim()

字符串和文字处理 - boost.string_algo

- 算法列表（续 1 ）

算法名称	描述	相关函数
判断式		
starts_with	判断一个字符串是否为另一个字符串的前缀	starts_with() istarts_with()
ends_with	判断一个字符串是否为另一个字符串的后缀	ends_with() iends_with()
contains	判断一个字符串是否出现在另一个字符串中	contains() icontains()
equals	判断 2 个字符串是否相等	equals() iequals()
lexicographical_compare	判断一个字符串在字典序上是否小于另一个字符串	lexicographical_compare() ilexicographical_compare()
all	判断一个字符串的所有字符是否都符合给定的判断式（条件）	all()

字符串和文字处理 - boost.string_algo

- 算法列表（续2）

算法名称	描述	相关函数
查找算法		
find_first	在字符串中查找第一个符合的字符串	find_first() ifind_first()
find_last	在字符串中查找最后一个符合的字符串	find_last() ifind_last()
find_nth	查找一个 string 在输入中的第 n 次（从 0 开始索引）出现	find_nth() ifind_nth()
find_head	取回一个 string 的开头	find_head()
find_tail	取回一个 string 的末尾	find_tail()
find_token	查找 string 中的第一个匹配标志	find_token()
find_regex	使用正则表达式搜索 string	find_regex()
find	通用查找算法	find()

字符串和文字处理 - boost.string_algo

- 算法列表（续 3 ）

算法名称	描述	相关函数
替换 / 删除算法		
replace/erase_first	替换 / 删除一个 string 在输入中的第一次出现	replace_first() erase_first() ...
replace/erase_last	替换 / 删除一个 string 在输入中的最后一次出现	replace_last() erase_last() ...
replace/erase_nth	替换 / 删除一个 string 在输入中的第 n 次（从 0 开始索引）出现	replace_nth() erase_nth() ...
replace/erase_all	替换 / 删除一个 string 在输入中的所有出现	replace_all() erase_all()
replace/erase_head	替换 / 删除输入的开头	replace_head() erase_head()
replace/erase_tail	替换 / 删除输入的末尾	replace_tail() erase_tail()

字符串和文字处理 - boost.string_algo

- 算法列表（续 4 ）

算法名称	描述	相关函数
替换 / 删除算法		
replace/erase_regex	替换 / 删除与给定正则表达式匹配的一个 substring	replace_regex() erase_regex() ...
replace/erase_regex_all	替换 / 删除与给定正则表达式匹配的全部 substring	replace_all_regex() erase_all_regex() ...
find_format	通用替换算法	find_format() find_format_all() ...

字符串和文字处理 - boost.string_algo

- 算法列表（续5）

算法名称	描述	相关函数
分割		
find_all	查找/选取输入中的所有匹配的 substring	find_all() ifind_all() find_all_regex()
split	将输入分给为多个字串	split() split_regex()
iter_find	在输入中迭代使用 Finder，以找到所有匹配的 substrings	iter_find()
iter_split	使用 Finder 在输入中查找匹配的 substrings，并将它们作为分隔符将输入分割为多个字串	iter_split()
合并		
join	将一个容器中的所有元素结合到一个单独的 string 中	join()
join_if	将一个容器中的所有满足条件的元素结合到一个单独的 string 中	join_if()

字符串和文字处理 - boost.string_algo

- Finders 列表

Finder	描述	生成器
first_finder	在输入中搜索 string 的第一个匹配项	first_finder()
last_finder	在输入中搜索 string 的最后一个匹配项	last_finder()
nth_finder	在输入中搜索 string 的第 n 个(从 0 开始索引)匹配项	nth_finder()
head_finder	取回输入的开头	head_finder()
tail_finder	取回输入的末尾	tail_finder()
token_finder	在输入中搜索匹配的标志	token_finder()
range_finder	不搜索，总是返回给定的 range	range_finder()
regex_finder	搜索与给定 regex 匹配的一个 substring	regex_finder()

字符串和文字处理 - boost.string_algo

- Formatters 列表

Formatter	描述	生成器
const_formatter	常量 Formatter。总是返回特定的 string	const_formatter()
identity_formatter	恒等 Formatter。返回没有改变的输入	identity_formatter()
empty_formatter	空 Formatter。总是返回一个空 string	empty_formatter()
regex_formatter	Regex Formatter。使用格式化 string 中的规范格式化 regex 匹配项	regex_formatter()

字符串和文字处理 - boost.string_algo

- Find Iterators 列表

Iterator	描述	迭代器类型
find_iterator	在输入中 迭代遍历匹配的 substrings	find_iterator
split_iterator	在输入中迭代遍历匹配的 substrings 之间的间隔	split_iterator

字符串和文字处理 - boost.string_algo

- 字符类别
  - 同 C API，如 is_space() 同 C 的 isspace()
  - 在 boost 中都作为判断式出现

字符串和文字处理 - boost.string_algo

- 示例
  - 使用 boost.string_algo 须包含头文件 : <boost/algorithm/string.hpp>
  - (DEMO Using Boost Examples)

字符串和文字处理 - More stuffs...

- 字符串和文字处理对于各种语言来讲都十分重要，当前的 C++ 标准库提供的字符串操作远远不够，为此， Boost 提供了大量字符串处理的算法，除了本单元列出来的几个组件外，还有很多有用的组件：
  - boost.regex (In TR1)，正则表达式组件，将在下个单元介绍
  - boost.spirit，非常强大和高效的 EBNF 词法、语法解析器，类似于 yacc
  - boost.tokenizer，类似于 split
  - boost.xpressive，另一个正则表达式组件
  - ...