内容

[**3修改序列操作** 1](#_Toc506130363)

[**3.1 transform** 1](#_Toc506130364)

# **3修改序列操作**

## **3.1 transform**

1. 原型

头文件：<algorithm>

|  |  |  |
| --- | --- | --- |
| template< class InputIt, class OutputIt, class UnaryOperation >  OutputIt transform( InputIt first1, InputIt last1, OutputIt d\_first, UnaryOperation unary\_op ); | （1） | until C++20 |
| template< class InputIt, class OutputIt, class UnaryOperation >  constexpr OutputIt transform( InputIt first1, InputIt last1, OutputIt d\_first, UnaryOperation unary\_op ); | since C++20 |
| template< class ExecutionPolicy, class ForwardIt1, class ForwardIt2, class UnaryOperation >  ForwardIt2 transform( ExecutionPolicy&& policy, ForwardIt1 first1, ForwardIt1 last1, ForwardIt2 d\_first, UnaryOperation unary\_op ); | （2） | since C++17 |
| template< class InputIt1, class InputIt2, class OutputIt, class BinaryOperation >  OutputIt transform( InputIt1 first1, InputIt1 last1, InputIt2 first2, OutputIt d\_first, BinaryOperation binary\_op ); | （3） | until C++20 |
| template< class InputIt1, class InputIt2, class OutputIt, class BinaryOperation >  constexpr OutputIt transform( InputIt1 first1, InputIt1 last1, InputIt2 first2, OutputIt d\_first, BinaryOperation binary\_op ); | since C++20 |
| template< class ExecutionPolicy, class ForwardIt1, class ForwardIt2, class ForwardIt3, class BinaryOperation >  ForwardIt3 transform( ExecutionPolicy&& policy, ForwardIt1 first1, ForwardIt1 last1, ForwardIt2 first2, ForwardIt3 d\_first, BinaryOperation binary\_op ); | （4） | since C++17 |

std::transform 将给定的函数应用到某一范围的每一元素，然后将结果存储到另一个范围内，起始位置是d\_first。

1) 一元操作unary\_op被应用到由[first1, last1)指定的范围上。

3) 二元操作binary\_op被应用到两个范围里的成对元素上，这两个范围分别由[first1, last1)和[first2, last2)定义。

2,4)和(1,3)一样，只是根据策略执行。  
如果std::is\_execution\_policy\_v<std::decay\_t<ExecutionPolicy>>为true时，这类重载参与到重载决策。

|  |  |
| --- | --- |
| unary\_op和binary\_op必须没有副作用 | until C++11 |
| unary\_op和binary\_op不能使迭代器无效，包括end iterators或修改范围内的任何元素 | since C++11 |

2 参数

|  |  |
| --- | --- |
| first1，last1 | 要转化的第一个范围 |
| first2 | 要转换的第二个范围的开始 |
| d\_first | 目的范围的开始，可以等于first1或first2 |
| policy | 使用的执行策略。 |
| unary\_op | 一元操作函数，函数的签名应该等同于以下内容：  Ret fun(const Type &a);  该签名不必非有const &；类型Type必须满足，类型InputIt的对象能够被解引用且隐式地转换为类型Type。Ret必须满足，类型OutputIt的对象能够被解引用且能被类型Ret赋值。 |
| binary\_op | 二元操作函数，函数的签名应该等同于以下内容：  Ret fun(const Type1 &a, const Type2 &b);  同一元操作函数的说明。 |

类型要求：

（1）InputIt，InputIt1，InputIt2必须满足InputIterator的要求。

（2）OutputIt必须满足OutputIterator的要求。

（3）ForwardIt1, ForwardIt2, ForwardIt3必须满足 ForwardIterator的要求。

3 返回值

输出一个迭代器，迭代所有转换后的元素。

4.异常

带有名为ExecutionPolicy的模板参数的重载报告错误如下：

* 如果作为算法的一部分调用函数的执行引发异常，并且ExecutionPolicy是三个标准策略之一，则调用std :: terminate。对于任何其他ExecutionPolicy，行为都是定义的。
* 如果算法没有成功分配内存，std::bad\_alloc被抛出。

5.可能实现

第一个版本：

|  |
| --- |
| template<class InputIt, class OutputIt, class UnaryOperation>  OutputIt transform(InputIt first1, InputIt last1, OutputIt d\_first, UnaryOperation unary\_op)  {  while (first1 != last1) {  \*d\_first++ = unary\_op(\*first1++);  }  return d\_first;  } |

第二个版本：

|  |
| --- |
| template<class InputIt1, class InputIt2, class OutputIt, class BinaryOperation>  OutputIt transform(InputIt1 first1, InputIt1 last1, InputIt2 first2, OutputIt d\_first, BinaryOperation binary\_op)  {  while (first1 != last1) {  \*d\_first++ = binary\_op(\*first1++, \*first2++);  }  return d\_first;  } |

注意：

std :: transform不能保证unary\_op或binary\_op的顺序应用。要按顺序将函数应用于序列或应用修改序列元素的函数时，请使用std :: for\_each

6 例子

以下代码使用transform使用toupper函数将字符串就地转换为大写，然后将每个字符转换为其序数值：

|  |
| --- |
| #include <algorithm>  #include <cctype>  #include <iostream>  #include <string>  #include <vector>    int main()  {  std::string s("hello");  std::transform(s.begin(), s.end(), s.begin(),  [](unsigned char c) -> unsigned char { return std::toupper(c); });    std::vector<std::size\_t> ordinals;  std::transform(s.begin(), s.end(), std::back\_inserter(ordinals),  [](unsigned char c) -> std::size\_t { return c; });    std::cout << s << ':';  for (auto ord : ordinals) {  std::cout << ' ' << ord;  }  } |

输出：

HELLO: 72 69 76 76 79