**2023.10.30**

Reverse Iter

**template<typename** InIter**>**

**void** PrintRange**(**InIter beg**,** InIter end**)**

**{**

**while** **(**beg **!=** end**)**

**{**

std**::**cout **<<** **\***beg**++** **<<** ' '**;**

**}**

**}**

**int** main**()**

**{**

**using** **namespace** std**;**

vector**<int>** ivec**;**

rfill**((**ivec**,** 20**,** Irand**{**0**,** 100**};**

**auto** riter\_beg **=** ivec**.**rbegin**();**

**auto** riter\_end **=** ivec**.**rend**();**

PrintRange**(**riter\_beg**,** riter\_end**);** ***// sondan başa doğru yazar***

PrintRange**(**riter\_end**.**base**(),** riter\_beg**.**base**());** ***// baştan sona doğru yazar***

**}**

**Back Insert Iterator**

**template** **<typename** InIter**,** **typename** OutIter**>**

OutIter Copy**(**InIter beg**,** InIter end**,** OutIter destbeg**)**

**{**

**while** **(**beg **!=** end**)**

**{**

**\***destbeg**++** **=** **\***beg**++;**

**}**

**return** destbeg**;**

**}**

***/\****

***Yukarıdaki Copy algoritmasına dokunmadan ona çağrı yapıp***

***yapacağımız çağrı dest\_vec source\_vec'teki öğeleri sondan ekleyecek***

***\*/***

**template<typename** Container**>**

**class** BackInsertIterator

**{**

**public:**

BackInsertIterator**(**Container **&**c**)** **:** rc**(**c**)** **{}**

BackInsertIterator**&** **operator++()** **{return** **\*this;}**

BackInsertIterator**&** **operator++(int)** **{return** **\*this;}**

BackInsertIterator**&** **operator\*()** **{return** **\*this;}**

BackInsertIterator**&** **operator=(const** **typename** Container**::**value\_type**&** val**)**

**{**

rc**.**push\_back**(**val**);**

**return** **\*this;**

**}**

**}**

**int** main**()**

**{**

**using** **namespace** std**;**

vector**<int>** source\_vec **{**2**,** 5**,** 1**,** 3**,** 4**,** 6**,** 9**,** 7**};**

vector**<int>** dest\_vec**;** ***// empty vector***

BackInsertIterator**<**vector**<int>>** iter**(**dest\_vec**);**

BackInsertIterator iter**(**dest\_vec**);**

Copy**(**source\_vec**.**begin**(),** source\_vec**.**end**(),** BackInsertIterator**(**dest\_vec**));**

print**(**dest\_vec**);** ***// 2, 5, 1, 3, 4, 6, 9, 7***

**}**

Yukarıdaki Kodun STL ile yazılmış hali:

#include<iterator>

#include<algortihm>

**int** main**()**

**{**

**using** **namespace** std**;**

vector**<int>** source\_vec **{**2**,** 5**,** 1**,** 3**,** 4**,** 6**,** 9**,** 7**};**

vector**<int>** dest\_vec**;**

copy**(**source\_vec**.**begin**(),** source\_vec**.**end**(),** back\_inserter**(**dest\_vec**));**

print**(**dest\_vec**);**

**}**

**Back Inserter**

#include<iterator>

#include<algortihm>

**int** main**()**

**{**

**using** **namespace** std**;**

vector**<**string**>** svec**;**

rfill**(**svec**,** 10'000**,** **[]** **{return** rname**()** **+** ' ' **+** rfname**();});**

vector**<**string**>** dvec**;**

**size\_t** len **=** 13**;**

***// uzunluğu 13 olanlar kopyalanır***

copy\_if**(**svec**.**begin**(),** svec**.**end**(),** back\_inserter**(**dvec**),**

**[**len**](const** string**&** s**)** **{return** s**.**length**()** **==** len**;});**

**}**

**Front Inserter**

#include<iterator>

#include<algortihm>

**int** main**()**

**{**

**using** **namespace** std**;**

vector**<int>** source\_vec **{**2**,** 5**,** 1**,** 3**,** 4**,** 6**,** 9**,** 7**};**

list**<int>** ilist**;**

copy**(**source\_vec**.**begin**(),** source\_vec**.**end**(),** front\_inserter**(**ilist**));**

print**(**dest\_vec**);**

**}**

**STL'de iteratorleri manipüle eden algoritmalar:**

* -advance ( bir iter'i npos arttırmak için kullanılır) advance(ref)
* -distance ( iki iterator arasındaki farkı buluyor) distance(iter1, iter2)
* -next (iter, 5) 5 ilerisini alıyor
* -prev (iter, 5) 5 gerisini veriyor
* -iter\_swap (iter\_x, iter\_y) iter'in x konumuyla y konumu yer değiştirir.

**using** iter **=** std**::**vector**<int>::**iterator

**int** main**()**

**{**

iter**::**value\_type type**;** ***// iter türü yani int***

iter**::**difference\_type diff\_type**;** ***// iki iter farkından oluşan tür ptrdiff\_t***

iter**::**pointer pointer\_type**;** ***// int\****

iter**::**reference ref\_type**;** ***// int&***

iter**::**iterator\_category**;** ***// random\_access\_iterator\_tag***

**}**

**Iterator Category'sine Göre Fonksiyon Yazma**

***// iterator category'sine göre func\_impl etme***

***// tag dispatch***

**template** **<typename** Iter**>**

**void** func\_impl**(**Iter beg**,** Iter end**,** std**::**random\_access\_iterator\_tag**)** **{}**

**template** **<typename** Iter**>**

**void** func\_impl**(**Iter beg**,** Iter end**,** std**::**bidirectional\_iterator\_tag**)** **{}**

**template** **<typename** Iter**>**

**void** func\_impl**(**Iter beg**,** Iter end**,** std**::**forward\_iterator\_tag**)** **{}**

**template<typename** Iter**>**

**void** func**(**Iter beg**,** Iter end**)**

**{**

func\_impl**(**beg**,** end**,** **typename** Iter**::**iterator\_category**{});**

**}**

***// static if***

**template<typename** Iter**>**

**void** func**(**Iter beg**,** Iter end**)**

**{**

**using** cat **=** **typename** std**::**iterator\_traits**<**Iter**>::**iterator\_category**;**

**if** **constexpr** **(**std**::**is\_same\_v**<**cat**,** std**::**bidirectional\_iterator\_tag**>)**

**{**

std**::**cout **<<** " bidirectional\_iterator\_tag" **<<** "\n"**;**

**}**

**else** **if** **constexpr** **(**std**::**is\_same\_v**<**cat**,** std**::**random\_access\_iterator\_tag**>)**

**{**

std**::**cout **<<** "random\_access\_iterator\_tag" **<<** "\n"

**}**

**}**

**Advance**

***// advance***

**int** main**()**

**{**

**using** **namespace** std**;**

vector**<int>** ivec**{** 2**,** 4**,** 6**,** 7**,** 9**,** 3**,** 1**};**

list**<int>** ilist**{** 2**,** 4**,** 6**,** 7**,** 9**,** 3**,** 1**};**

**auto** vec\_iter **=** ivec**.**begin**();**

**auto** list\_iter **=** ilist**.**begin**();**

***// 3 adım ilerler***

***// ikisi farklı fonkisyonlar overloading var complite time'da kod seçimi yapılır***

***// type dispatch örneği***

advance**(**vec\_iter**,** 3**);**

advance**(**list\_iter**,** 3**);**

**}**

**template<typename** Iter**>**

**void** Advance**(**Iter**&** it**,** **int** n**,** std**::**random\_access\_iterator\_tag**)**

**{**

it **+=**n**;** ***// vector***

**}**

**template<typename** Iter**>**

**void** Advance**(**Iter**&** it**,** **int** n**,** std**::**bidirectional\_iterator\_tag**)**

**{**

**while(**n**--)**

**++**it**;** ***// list***

**}**

**template<typename** Iter**>**

**void** Advance**(**Iter**&** it**,** **int** n**)**

**{**

Advance**(**it**,** n **typename** std**::**iterator\_traits**<**Iter**>::**iterator\_category**{});**

**}**

**int** main**()**

**{**

**using** **namespace** std**;**

vector**<int>** ivec**{** 2**,** 4**,** 6**,** 7**,** 9**,** 3**,** 1**};**

**auto** iter **=** ivec**.**end**();**

advance**(**iter**,** **-**3**);** ***// \*iter 9***

**}**

**Distance**

**int** main**()**

**{**

**using** **namespace** std**;**

list mylist **{** 2**,** 5**,** 8**,** 9**,** 3**,** 1**,** 8**};**

**auto** iter1 **=** mylist**.**begin**();**

**auto** iter2 **=** mylist**.**end**();**

advance**(**iter1**,** 2**);** ***//\*iter = 8***

advance**(**iter2**,** **-**1**);** ***// \*iter 1***

**auto** n **=** distance**(**iter1**,** iter2**);** ***// n = 4***

**}**

**Next and Prev (CPP 11)**

***// next and prev --cpp11***

**int** main**()**

**{**

**using** **namespace** std**;**

vector**<**string**>** svec**{** "ali"**,** "can"**,** "ece"**,** "naz"**,** "gul"**,** "eda"**,** "tan"**};**

**auto** iter **=** next**(**svec**.**begin**(),** 3**);** ***// \*iter = naz***

iter **=** next**(**svec**.**begin**());** ***// \*iter = can***

list**<**string**>** slist**{** "ali"**,** "can"**,** "ece"**,** "naz"**,** "gul"**,** "eda"**,** "tan"**};**

std**::**cout **<<** **\***slist**.**end**()** **<<** "\n"**;** ***// tanımsız davranış***

std**::**cout **<<** **\***prev**(**slist**.**end**())** **<<** "\n"**;** ***// tan***

**}**

advance: ref parametreli iter'in kendisini artırıyor (call\_by\_reference)

next: iter değişmez ancak assign etmem gerekiyor (call\_by\_value)

**Swap**

**template** **<typename** Iter1**,** **typename** Iter2**>**

**void** IterSwap**(**Iter1 it1**,** Iter2 it2**)**

**{**

std**::**swap**(\***it1**,** **\***it2**);**

**}**

***////////***

**int** main**()**

**{**

**using** **namespace** std**;**

vector**<**string**>** svec**{** "ali"**,** "can"**,** "tan"**,** "ata"**};**

list**<**string**>** slist**{** "gul"**,** "eda"**,** "naz"**,** "ela"**};**

iter\_swap**(**next**(**svec**.**begin**()),** prev**(**slist**.**end**(),** 2**);**

**}**

**Find If**

***// find\_if***

**template<typename** InIter**,** **typename** Pred**>**

InIter FindIf**(**InIter beg**,** InIter end**,** Pred f**)**

**{**

**while(**beg **!=** end**)**

**{**

**if(**f**(\***beg**))**

**return** beg**;**

**++**beg**;**

**}**

**return** end**;**

**}**

**int** main**()**

**{**

**using** **namespace** std**;**

list**<**string**>** slist**;**

rfill**(**slist**,** 20**,** name**);**

print**(**slist**);**

**{**

**char** c **=** '0'**;**

**auto** iter **=** find\_if**(**slist**.**begin**(),** slist**.**end**(),** **[**c**](const** string **&**s**)**

**{**

**return** s**.**contains**(**c**);**

**});**

**if** **(**iter **!=** slist**.**end**())**

**{**

std**::**cout **<<** "bulundu... idx = " **<<** distance**(**slist**.**begin**(),** iter**)** **<<** '\n'**;**

**}**

**else**

**{**

std**::**cout **<<** "bulunamadı" **<<** "\n"**;**

**}**

**}**

***// ya da***

**char** c **=** '0'**;**

**if** **(auto** iter **=** find\_if**(**slist**.**begin**(),** slist**.**end**(),** **[**c**](const** string **&**s**)**

**{**

**return** s**.**contains**(**c**);**

**});** iter **!=** slist**.**end**())**

**{**

std**::**cout **<<** "bulundu... idx = " **<<** distance**(**slist**.**begin**(),** iter**)** **<<** '\n'**;**

**}**

**else**

**{**

std**::**cout **<<** "bulunamadı" **<<** "\n"**;**

**}**

**}**

**Transform**

**template<typename** InIter**,** **typename** OutIter**,** **typename** F**>**

OutIter Transform**(**InIter beg**,** InIter end**,** OutIter destbeg**,** F f**)**

**{**

**while** **(**beg **!=**end**)**

**{**

**\***destbeg**++** **=** f**(\***beg**++);**

**}**

**return** destbeg**;**

**}**

***////////***

**auto** get\_len**(const** std**::**string**&** s**)**

**{**

**return** s**.**size**();**

**}**

**int** main**()**

**{**

**using** **namespace** std**;**

vector**<**string**>** svec**;**

rfill**(**svec**,** 100**,** rname**);**

list**<size\_t>** lenlist**;**

transform**(**svec**.**begin**(),** svec**.**end**(),**

back\_inserter**(**lenlist**),** get\_len**);**

print**(**lenlist**);** ***// print svec'tekilerin size'ni yazar***

**}**

std**::**string revstr**(**std**::**string s**)**

**{**

***/\****

***std::string temp(s);***

***std::reverse(temp.begin(), temp.end());***

***return temp;***

***Eğer parametre değişikenin zaten kopyasını çıkarıyorsak (yukarıdaki gibi)***

***const string:: &s yerine std::string s kullanıp aşağdaki gibi yazabiliriz.***

***\*/***

std**::**reverse**(**s**.**begin**(),** s**.**end**());**

**return** s**;**

**}**

**int** main**()**

**{**

**using** **namespace** std**;**

vector**<**string**>** svec**;**

rfill**(**svec**,** 100**,** rname**);**

list**<size\_t>** lenlist**;**

transform**(**svec**.**begin**(),** svec**.**end**(),**

svec**.**begin**(),** revstr**);**

**}**

***// transform 2. overload'u***

**template<typename** InIter1**,** **typename** InIter2**,** **typename** OutIter**,** **typename** F**>**

OutIter Transform**(**InIter1 beg**,** InIter1 end**,** InIter2 beg2**,** OutIter destbeg**,** F f**)**

**{**

**while** **(**beg **!=** end**)**

**{**

**\***desbeg**++** **=** f**(\***beg**++,** **\*\***beg2**++);**

**}**

**return** destbeg**;**

**}**

**int** main**()**

**{**

**using** **namespace** std**;**

vector**<int>** v**{** 1**,** 4**,** 7**,** 2**,** 9**,** 9**,** 3 **};**

deque**<int>** d**{** 2**,** 5**,** 8**,** 1**,** 9**,** 3**,** 5**};**

list**<int>** ilist**;**

transform**(**v**.**begin**(),** v**.**end**(),** d**.**begin**(),** back\_inserter**(**ilist**)**

**[](int** x**,** **int** y**)** **{return** x **\*** x **+** y **\*** y**});**

**)**

**}**