ESTRUCTURAS DE DATOS

NOTAS SOBRE C++

STL: Algoritmos (2)

Manuel Montenegro Montes
Departamento de Sistemas Informáticos y Computación
Facultad de Informática – Universidad Complutense de Madrid

Funciones de orden superior

La función `transform`

```
transform(ini, fin, dest, fun)
```

- Definida en `<algorithm>`
- Aplica la función `fun` al conjunto de elementos contenido entre los iteradores `[ini, fin)`.
- Los resultados devueltos por fun son copiados a partir del iterador `dest`.
- Si se desea modificar la lista original, utilizar `dest = ini`.

Ejemplos

```
vector<int> v = { 3, 10, 9, 3, 15 };
transform(v.begin(), v.end(), v.begin(), [](int x) { return x * 2; });
```

```
v = [6, 20, 18, 6, 30]
```

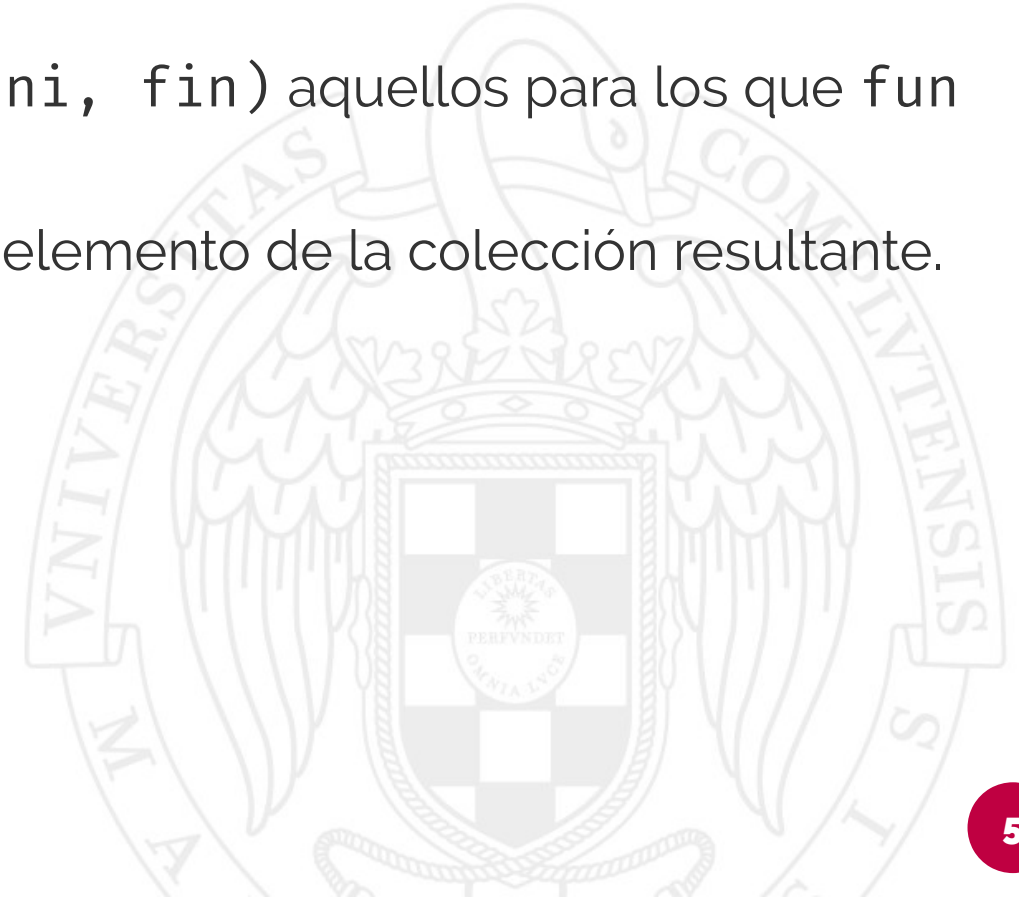
```
vector<string> nombres = {"Juan", "Rosario", "Amalia"};
vector<int> longitudes;

transform(nombres.begin(), nombres.end(),
          back_insert_iterator<vector<int>>(longitudes),
          [](const string &x) { return x.length(); });
```

```
longitudes = [4, 7, 6]
```

La función `remove_if`

```
remove_if(ini, fin, fun)
```

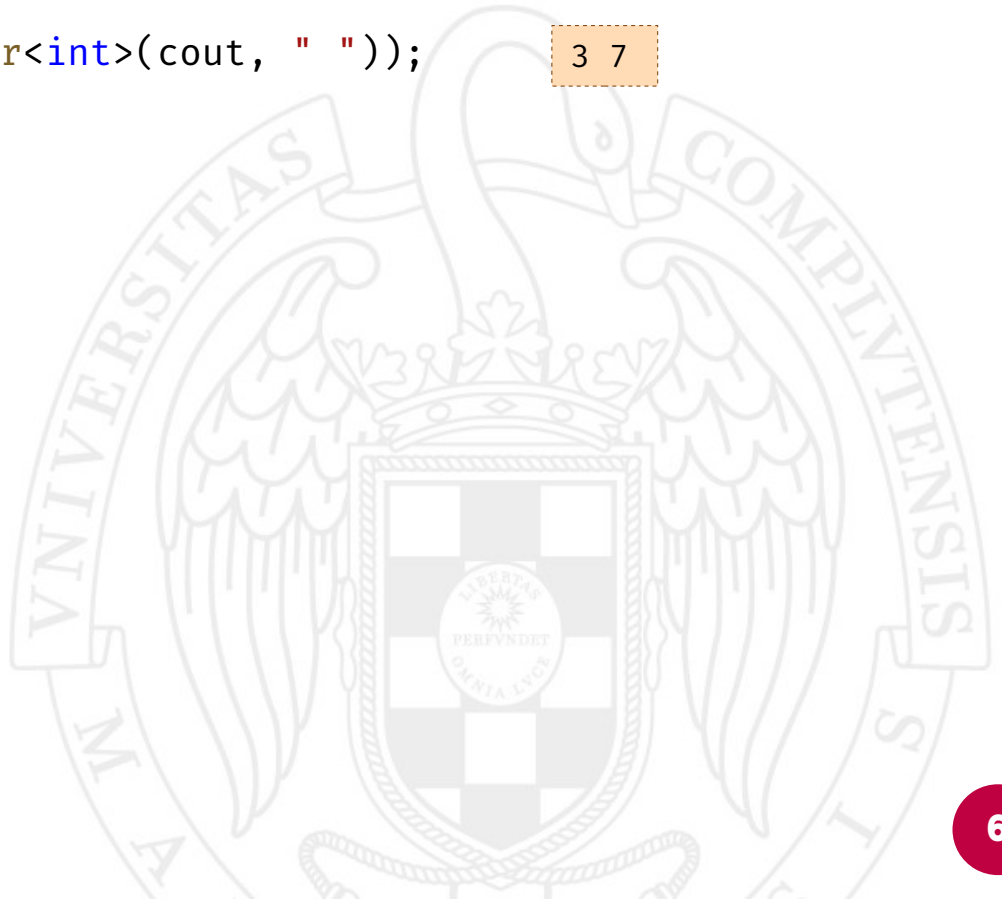
- Definida en `<algorithm>`
- Elimina del rango de elementos `[ini, fin)` aquellos para los que `fun` devuelve `true`.
- Devuelve un iterador tras el último elemento de la colección resultante.



Ejemplo

```
vector<int> v2 = { 3, 10, 8, 7, 4 };
auto it_end = remove_if(v2.begin(), v2.end(), [](int x) { return x % 2 == 0; });

copy(v2.begin(), it_end, ostream_iterator<int>(cout, " "));
```

```
3 7
```



Las funciones `find_if` y `count_if`

`find_if(ini, fin, fun)`

- Devuelve un iterador al primer elemento del rango `[ini, fin)` para el que fun devuelve `true`.

`count_if(ini, fin, fun)`

- Devuelve el número de elementos del rango `[ini, fin)` para los que fun devuelve `true`.

Ejemplo

```
vector<Fecha> fechas = {{10, 3, 2010}, {1, 6, 2019}, {28, 8, 1985}, {19, 3, 2001}};

auto it_marzo = find_if(fechas.begin(), fechas.end(),
                        [](const Fecha &f) { return f.get_mes() == 3; });

cout << *it_marzo << endl;    10/03/2010

int num_fechas_verano =
  count_if(fechas.begin(), fechas.end(),
    [](const Fecha &f) { return f.get_mes() ⩾ 6 && f.get_mes() ⩽ 8; });

cout << num_fechas_verano << endl;    2
```

Otras funciones de orden superior

- `all_of(ini, fin, fun)`
- `any_of(ini, fin, fun)`
- `none_of(ini, fin, fun)`

- `accumulate(ini, fin, valor_inicial)`
- `accumulate(ini, fin, valor_inicial, fun)`

Funciones sobre conjuntos

Funciones sobre conjuntos

- Las siguientes funciones pueden aplicarse sobre colecciones tales que, al ser iteradas, produzcan secuencias de elementos en orden ascendente. Esto incluye:
  - `set` (pero no `unordered_set`)
  - `map` (pero no `unordered_map`)
  - Listas que almacenen sus elementos en orden creciente.
  - Arrays que almacenen sus elementos en orden creciente.

Funciones sobre conjuntos

- `includes(ini1, fin1, ini2, fin2)`
- `set_union(ini1, fin1, ini2, fin2, dest)`
- `set_intersection(ini1, fin1, ini2, fin2, dest)`
- `set_difference(ini1, fin1, ini2, fin2, dest)`

Ejemplos

```
set<int> elems1 = { 6, 1, 9, 4, 3, 10 };
set<int> elems2 = { 10, 1, 4, 6 };

cout << includes(elems1.begin(), elems1.end(), elems2.begin(), elems2.end()) << endl;
```

```
true
```

```
set<string> chicos = {"Ricardo", "Jaime", "Rafa", "Enrique", "Adrián", "Jose"};
set<string> chicas = {"Clara", "Susana", "Jose", "Natalia", "Elvira"};
list<string> result;

set_union(chicos.begin(), chicos.end(),
          chicas.begin(), chicas.end(),
          back_insert_iterator<list<string>>(result));
```

```
result = [Adrián, Clara, Elvira, Enrique, Jaime, Jose, Natalia, Rafa, Ricardo, Susana]
```