Programación Orientada a Objetos

Unidad 8: Biblioteca STL

FICH UNL
Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas

2019
Pablo Novara

PROGRAMACIÓN GENÉRICA

```
template<typename T>
T menor(const vector<T> &v) {
    ...
}
```

Función genérica (plantilla) + argumentos


```
vector<float> v;
...
float men =
    menor<float>(v);
```

Función concreta (especializada)

```
float menor(const vector<float> &v) {
    ...
}
```

¿QUÉ ES LA STL?

- Una biblioteca con clases y funciones genéricas:
  - **Contenedores**: estructuras de datos que almacenan colecciones de otros objetos.

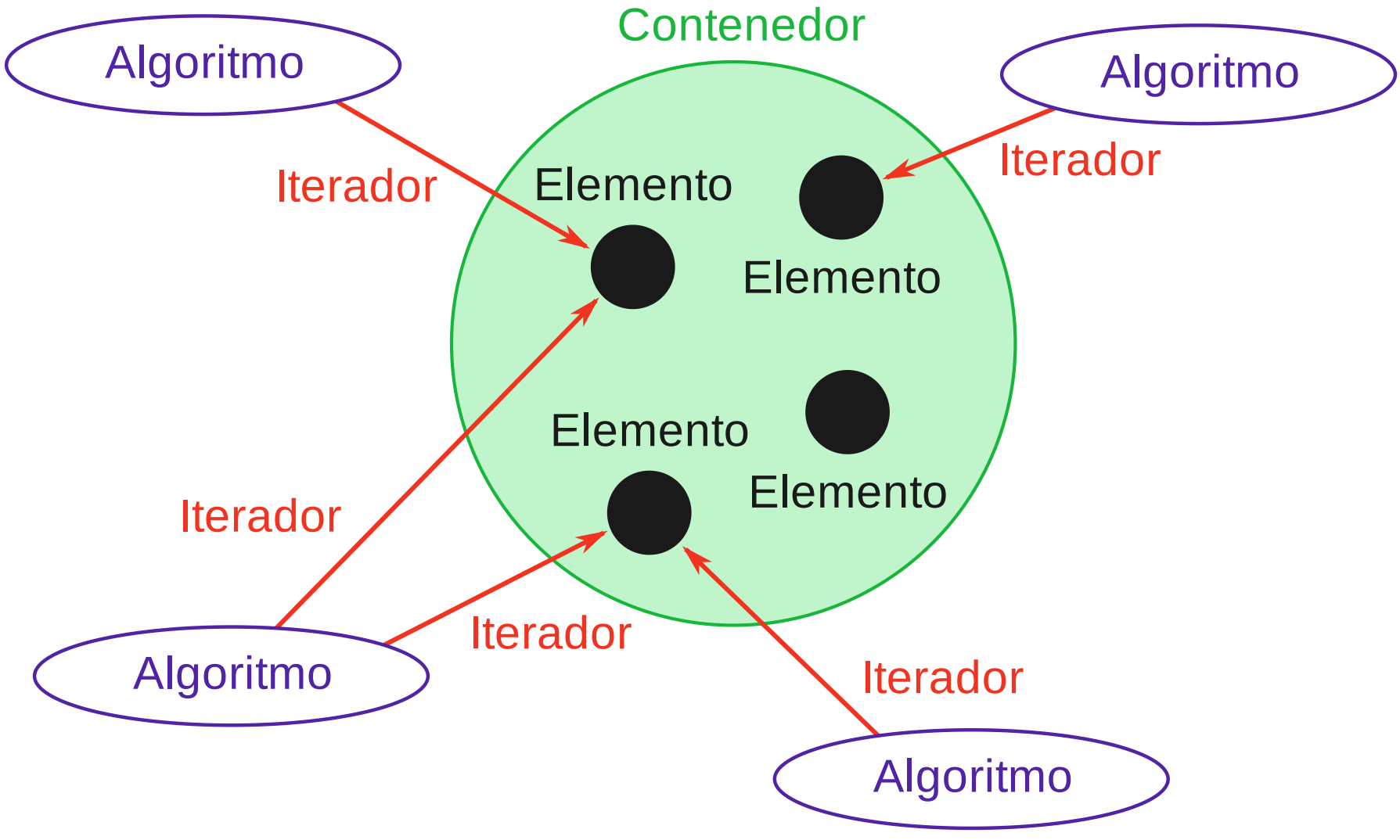
```
vector<int> v1;
```

  - **Algoritmos**: funciones que operan sobre rangos dentro de los contenedores.

```
sort(v.begin(),v.end());
```

  - **Iteradores**: objetos auxiliares para referenciar elementos y rangos dentro de un contenedor.

STL: STANDARD TEMPLATE LIBRARY



CONTENEDORES

- Secuenciales (elementos ordenados)
  - **list (lista dbl-enlazada)**
  - **vector (arreglo lineal)**
  - deque (doble cola)
- Asociativos (claves)
  - set (conjunto)
  - **map (correspondencia)**
  - multiset
  - multimap
  - bitset
- Adaptadores ("derivados")
  - stack (pila)
  - queue (cola)
  - priority_queue (cola con prioridades)
- Y más (desde C++11)
  - array (vector estático)
  - forward_list (lista simpl-enalzada)
  - unordered_* (tablas de hash)

ITERADORES

Hay distintos tipos de iteradores:

- De entrada, de salida
- **De acceso aleatorio, secuenciales**
- Direccionales, bidireccionales
- De inserción, etc

Utilizan sobrecarga de operadores para emular el comportamiento de un puntero en un arreglo

std::vector

Crear un vector (constructores):

```
vector<int> v1; // vacío
vector<int> v2(42,0); // con 42 ceros
vector<int> v3 = { 5, 7, 9 }; // con 3 elems: 5, 7 y 9
```

Obtener información del vector:

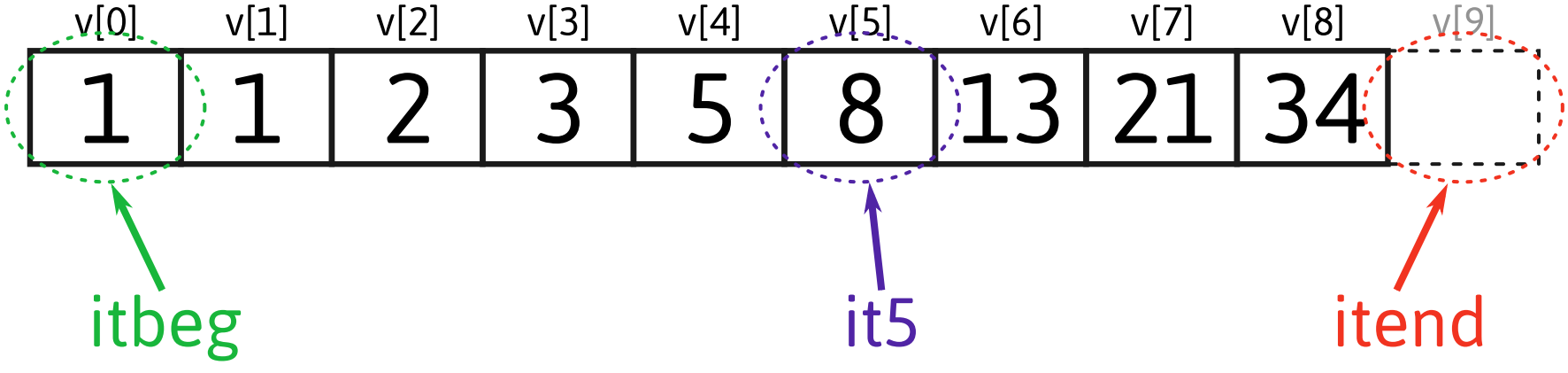
```
if (v1.empty())
    cout << "Está vacío" << endl;
else
    cout << "Tamaño: " << v1.size() << endl;
```

Redimensionar:

```
v1.resize(50,-1); // cambiar el tamaño a 50, si era menor,
                  // las nuevas posciones tendrán el valor -1
```

std::vector

```
vector<int> v = {1,1,2,3,5,8,13,21,34};
```



Obtener iteradores:

```
vector<int>::iterator itbeg = v.begin();
vector<int>::iterator itend = v.end();
vector<int>::iterator it5   = v.begin()+5;
```

std::vector

Obtener iteradores:

```
vector<int>::iterator itbeg = v.begin();
vector<int>::iterator itend = v.end();
vector<int>::iterator it5   = v.begin()+5;
```

Modificar el vector

```
v[0] = 25;                  // guardar "25" en la posición 0
*it = 18;                   // "18" en la pos. apuntada por it
v.push_back(10);            // insertar "10" al final
it = v.insert(it,20);       // insertar "20" en la posición it
v.pop_back();               // eliminar el último
it = v.erase(it);           // eliminar el de la posición it
v.erase(it1,it2);           // eliminar rango [it1;it2)
v.clear();                  // borrar todo
```

Si se agregan/eliminan elementos, los iteradores se invalidan.

CARGAR DATOS EN (O RECORRER) UN VECTOR

1. crear el contenedor vacio y agregarle datos

```
vector<int> v;
for (int i=0; i<15; ++i)
    v.push_back ( 1+rand()%100 );
```

2. crear el vector con 15 ceros y reemplazarlos

```
vector<int> v(15,0);
for (size_t i=0; i<v.size(); ++i)
    v[i] = 1+rand()%100;
```

CARGAR DATOS EN (O RECORRER) UN VECTOR

```
vector<int> v(15,0);
for (size_t i=0; i<v.size(); ++i) {
    v[i] = 1+rand()%100;
}
```

3. crear el vector con 15 ceros y reemplazarlos

```
vector<int> v(15,0);
for (vector<int>::iterator it=v.begin();
     it!=v.end(); ++it)
{
    *it = 1+rand()%100;
}
```

*En cada iteración, **`it`** apunta a un elemento de v*

CARGAR DATOS EN (O RECORRER) UN VECTOR

4. como 3, pero con **auto** para no escribir "tanto"

```
vector<int> v(15,0);

for (auto it = v.begin();
     it != v.end(); ++it)
{
    *it = 1+rand()%100;
}
```

*Si se declara una variable de tipo **auto**, el compilador deduce su tipo a partir del valor con el que se inicializa.*

El uso de auto no es exclusivo del for ni de la stl

CARGAR DATOS EN (O RECORRER) UN VECTOR

5. simil 2, 3 y 4, pero mucho mas corto y directo:

```
vector<int> v(15,0);

for (int &x : v)
    x = 1+rand()%100;
```

En cada iteración, x es una referencia a un elem. de v

`for(tipo x:v)` se lee como "para cada elemento x del contendor v"

¿QUIÉN DIJO?

> *“ Si entiendes int y vector, entonces entiendes C++, lo demás son solo detalles**

** más de 1600 páginas de detalles*

VECTOR VS LIST (COMPARACIÓN TEÓRICA)

	`std::vector`	`std::list`
Organización en memoria	Contigua (arreglo)	No contigua (lista enlaz.)
Tipo de acceso	Aleatorio	Secuencial
Inserción y eliminación	Lenta	Rápida

- `list` tiene una interfaz muy similar a la de `vector`, pero no permite acceso aleatorio:
  - no tiene sobrecarga para el operador []
  - sus iteradores solo pueden "moverse" con ++ y --

Ante la duda, usar vector

MOVIMIENTO DE ITERADORES

- Una pos. modificando el iterador:

```
++it | it++ | --it | it--
```

- Una pos, sin modifcar el iterador:

```
it1 = prev(it); it2 = next(it);
```

- Múltiples pos, modificando el iterador:

```
advance(it,N); advance(it,-N);
```

- Múltiples pos, sin modifcar el iterador:

```
it1 = it-N; it2 = it+N;          // solo para vector
it1 = prev(it,N); it2 = next(it,N);
```

EJEMPLO

2. Escriba un programa para generar una std::list de enteros con 15 elementos aleatorios entre 1 y 100.

   Luego debe mostrar los elementos y pedir al usuario que ingrese una posición para eliminarlo ese elemento de la misma.

EJEMPLO LIST

1) crear el contenedor vacio y agregarle datos

```
list<int> L;
for (int i=0; i<15; i++)
    L.push_back ( 1+rand()%100 );
```

2) recorrer sus elementos y mostrarlos:

```
for (int x : L)
    cout << x << endl;
```

3) eliminar una posicion:

```
int pos;
cin >> pos;
auto it = L.begin();  // auto := list<int>::iterator
advance(it,pos);  // equivale a for(int i=0;i<pos;i++) ++it;
L.erase(it);
```

EJEMPLOS

1. Escriba un programa que permita ingresar un conjunto de mediciones.

   Por error en el sensor, algunas mediciones no pudieron realizarse y en su lugar se registró un -1.

   Reemplace todos los -1 por el promedio de los valores adyacentes.

   Ayuda: Se sabe que no hay dos -1 consecutivos, y que tampoco están al comienzo o al final de la lista.

```
for(auto it=L.begin();it!=L.end();++it){
    if (*it==-1)
        *it = (*next(it)+*prev(it) )/2;
}
```

EJEMPLOS

2. Escriba un programa que permita ingresar una lista de valores flotantes por teclado, y luego inserte en medio de cada par de elementos consecutivos el promedio del par.

```
// El for empieza desde la 2da pos. porque ahí
// corresponde hacer la 1er inserción; y avanza
// de a 2 pos. porque it queda apuntando al nuevo
// elemento (prom) después de la inserción
for( auto it = next(L.begin());
     it!=L.end();
     advance(it,2) )
{
    int prom = (*prev(it)+*it)/2;
    it = L.insert(it,prom);
}
```

STD::MAP

- Map guarda un conjunto de asociaciones entre objetos, pares clave-valor.
- Clave y valor pueden ser de tipos diferentes.
- No puede haber claves repetidas.
- El tipo de clave debe ser ordenable (operator<).
- Insertar o buscar una clave en un map es *rápido*.

STD::MAP

Crear un mapa:

```
map<string,int> agenda; // asocia ints a strings
```

Guardar datos en el mapa:

```
agenda["Fulano"] = 15647352;
agenda["Mengano"] = 15473673;
agenda["Sultano"] = 15543455;
agenda["Chuck Norris"] = [illegible];
agenda["Juan Perez"] = 15536632;
```

✅ *parece un arreglo, pero con un tipo de índice diferente*

STD::MAP

Consultar un dato del mapa, si se que existe:

```
cout << "El nro de Chuck es: ";
cout << agenda["Chuck Norris"] << endl;
```

Consultar un dato cuando no se si existe:

```
auto it = agenda.find("Wally");
if (it==agenda.end())
    cout << "No encuentro a Wally!"
else
    cout << "Nro de Wally: " << it->second;
```

✅ *cada elemento del mapa es un struct con su clave(first) y su valor(second)*

STD::MAP

Recorrer todo el contenido de un map:

```
for(auto it=agenda.begin();
    it!=agenda.end(); ++it)
{
    cout << it->first << " está asociado a "
         << it->second << endl;
}
```

auto es map<string,int>::iterator

```
for(auto &p : m) { // cada par p
    cout << p.first << " está asociado a "
         << p.second << endl;
}
```

auto es pair<const string,int>

LA BIBLIOTECA `<algorithm>`

- Contiene algoritmos genéricos: para cualquier tipo de contenedor y de elementos
  - buscar un elemento (`find`/`find_if`)
  - buscar mayor y menor (`min_element`/`max_element`)
  - ordenar (`sort`) y desordenar(`shuffle`)
  - reemplazar (`replace`/`replace_if`)
  - contar (`count`/`count_if`), sumar (`accumulate`)
  - eliminar por valor (`remove`/`remove_if`)
  - eliminar repetidos (`unique`)
  - y muchos más...

EJEMPLO

3. Genere una cantidad arbitraria de valores enteros (n>=1) aleatorios

```
int rand_20() {
    return rand()%20;
}

int main() {

    int n;
    cout << "Cant. de datos a generar: ";
    cin >> n;

    list<int> L(n);
    generate(L.begin(),L.end(),rand_20);

    ...
```

EJEMPLO (CONT.)

3. Genere una cantidad arbitraria de valores enteros (n>=1) aleatorios y muestre:
   - la lista de datos inciales

```
for(int x:L) cout << x;
```

   - el promedio

```
float sum=accumulate(L.begin(),L.end(),0);
cout << sum/L.size() << endl;
```

   - los valores mínimo y máximo

```
auto it_max=max_element(L.begin(),L.end());
auto it_min=min_element(L.begin(),L.end());
cout << *it_max << " " << *it_min << endl;
```

EJEMPLO (CONT.)

3. Genere una cantidad arbitraria de valores enteros (n>=1) aleatorios y muestre:
   - la lista ordenada de menor a mayor

```
L.sort();
for(int x:L) cout << x << " ";
```

   - la mediana

```
auto itm = next(L.begin(),L.size()/2);
cout << *itm << endl;
```

   - la lista ordenada de mayor a menor

```
reverse(L.begin(),L.end());
for(int x:L) cout << x << " ";
```

EJEMPLO (CONT.)

3. Genere una cantidad arbitraria de valores enteros (n>=1) aleatorios y muestre:
   - la cantidad de ceros

```
cout << count(L.begin(),L.end(),0);
```

   - la cantidad de primos

```
bool es_primo(int x) { ... }
cout<<count_if(L.begin(),L.end(),es_primo);
```

   - la posición del valor 7

```
auto it_7 = find(L.begin(),L.end(),7);
if(it_7==L.end()) cout << "No está";
else cout << "Está en la pos: "
          << distance(L.begin(),it_7);
```

EJEMPLO (CONT.)

3. Genere una cantidad arbitraria de valores enteros (n>=1) aleatorios y muestre:
   - la lista de valores sin elementos repetidos

```
L.sort(); // L debe estar ordenada
auto it_rep = unique(L.begin(),L.end());
L.erase(it_rep,L.end()); // unique no puede eliminar
```

   - la lista de valores sin ceros

```
auto it_0 = remove(L.begin(),L.end(),0);
L.erase(it_0,L.end());
```

   - la lista de valores sin primos

```
auto it_pr = remove_if(L.begin(),L.end(),
                       es_primo);
L.erase(it_pr,L.end());
```

EJEMPLO

- Escriba una función que reciba el nombre de un archivo binario que contenga un conjunto de registros de un tipo genérico, y elimine del archivo todos los registros repetidos. La función debe retornar la cantidad de elementos eliminados.