ESTRUCTURAS DE DATOS

NOTAS SOBRE C++

STL: Iteradores

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

Tipos de iteradores

Vamos a estudiar los iteradores pero en la STL

- Iteradores de entrada.
- Iteradores de salida.
- Iteradores hacia delante.
- Iteradores bidireccionales.
- Iteradores de acceso aleatorio.

Vamos a ver para qué sirve cada uno de ellos

Iteradores de entrada

Entrada

```
… = *it
```
Acceso

```
it++
```
Avance

```
it1 == it2
```
Comparación

Como vemos NO nos deja escribir sobre él.

Iteradores de salida

Entrada

`… = *it`

Acceso

`it1 == it2`

Comparación

`it++`

Avance

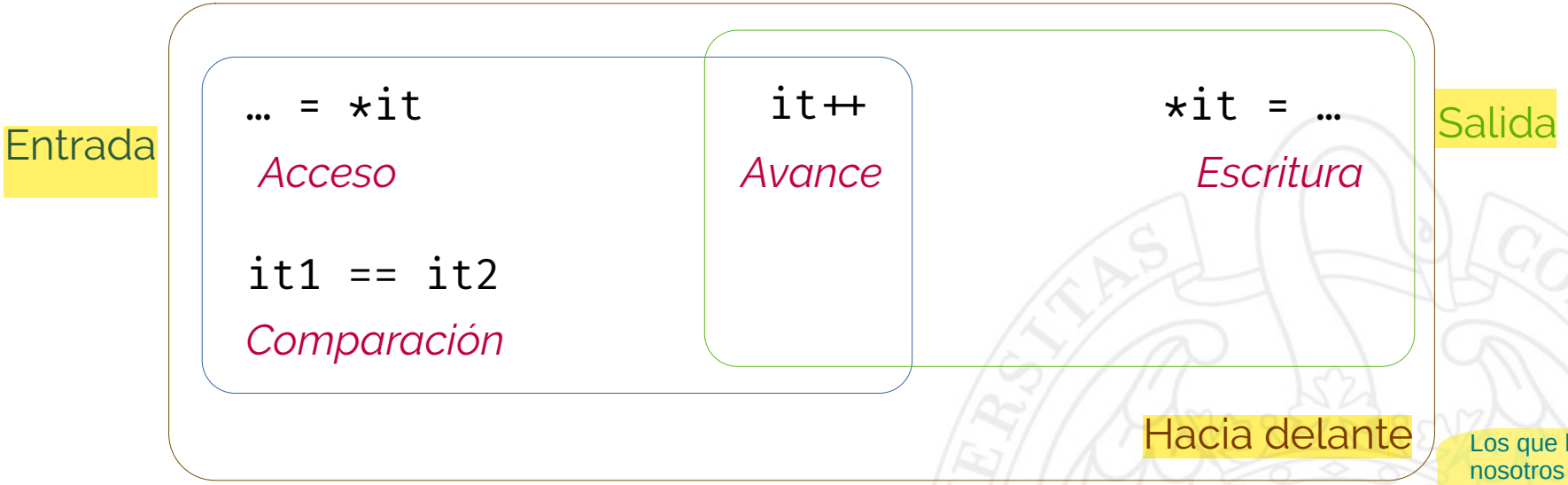
`*it = …`

Escritura

Salida

modifican el valor apuntado por ellos.

Iteradores hacia delante



Los que hemos implementado nosotros eran de este tipo.

Iteradores "Hacia delante" son iteradores de ENTRADA/SALIDA, pueden usar tanto las operaciones de entrada como las de salida.

Iteradores bidireccionales

`… = *it`

Acceso

`it++`

Avance

`*it = …`

Escritura

`it1 == it2`

Comparación

Hacia delante

`it--`

Retroceso

de uno en uno, no nos podemos saltar varios. Tampoco en el ++ creo.

Bidireccionales

iterar hacia atras

Iteradores de acceso aleatorio

`… = *it` *Acceso*

`it++` *Avance*

`*it = …` *Escritura*

`it1 == it2` *Comparación*

Hacia delante

`it--` *Retroceso*

Bidireccionales

`it = it ± n` *Avance/retroceso por saltos*

Con estos SI que nos podemos saltar más de una posición

Acceso aleatorio

Tipos de iteradores

- Cada implementación de TAD soporta un tipo de iterador determinado.

Expresión	Tipo de iterador
`vector::begin()`	Acceso aleatorio
`list::begin()`	Bidireccional
`deque::begin()`	Acceso aleatorio
`forward_list::begin()`	Hacia delante
`ostream_iterator`	Salida
`istream_iterator`	Entrada
Punteros	Acceso aleatorio

iteran solo de 1 en 1 en el caso de las listas enlazadas

listas enlazadas simples

En las siguientes diapositivas vamos a ver que es este ostream_operator.

Iterador de salida: `ostream_iterator`

- Es un iterador asociado a un flujo de salida (fichero, salida estándar, etc.)
- Cada vez que se modifica el valor apuntado por el iterador, se realiza una operación de salida.
- Cada vez que se incrementa el iterador, no se hace nada.
- Es útil para la función `copy()`

Ejemplo

```
int main() {
  std::ostream_iterator<int> it(std::cout, " ");

  *it = 10;
  it++; // Opcional. No hace nada.
  *it = 20;
  it++; // Opcional. No hace nada.

  std::cout << std::endl;

  return 0;
}
```

Flujo de salida

Separador

flujo de salida donde se escribiran

Ejemplo

Flujo de salida

Separador

```
int main() {
  std::ostream_iterator<int> it(std::cout, " ");

  *it = 10;
  it++; // Opcional. No hace nada.
  *it = 20;
  it++; // Opcional. No hace nada.

  std::cout << std::endl;

  return 0;
}
```

Ejemplo

Flujo de salida

Separador

```
int main() {
  std::ostream_iterator<int> it(std::cout, " ");

  *it = 10;
  it++; // Opcional. No hace nada.
  *it = 20;
  it++; // Opcional. No hace nada.

  std::cout << std::endl;

  return 0;
}
```