ESTRUCTURAS DE DATOS Y DE LA INFORMACIÓN

Biblioteca estándar de plantillas (STL: Standard Templates Library)

Prof. Dr. Nicolás Luis Fernández García

Departamento de Informática y Análisis Numérico
Escuela Politécnica Superior
Universidad de Córdoba

Curso académico 2009 - 2010

Contenido del tema

Contenido de la sección

Biblioteca estándar de plantillas (STL)

Contenedores: Definición

Definición (Contenedor)

Un contenedor es un objeto que contiene otros objetos.

Ejemplos

Las listas, vectores y los conjuntos (sets) son ejemplos de contenedores.

Acciones

Se pueden añadir, consultar y eliminar objetos de un contenedor.

Biblioteca estándar de plantillas (STL)

Contenedores: Definición

Definición (Contenedor)

Un contenedor es un objeto que contiene otros objetos.

Ejemplos

Las listas, vectores y los conjuntos (sets) son ejemplos de contenedores.

Acciones

Se pueden añadir, consultar y eliminar objetos de un contenedor.

Biblioteca estándar de plantillas (STL)

Contenedores: Definición

Definición (Contenedor)

Un contenedor es un objeto que contiene otros objetos.

Ejemplos

Las listas, vectores y los conjuntos (sets) son ejemplos de contenedores.

Acciones

Se pueden añadir, consultar y eliminar objetos de un contenedor.

Biblioteca estándar de plantillas (STL)

Contenedores: Tipos de contenedores

Tipos de contenedores

- **Contenedores de secuencia**:
  - Almacenan los elementos en posiciones contiguas de memoria
- **Contenedores asociativos**:
  - Permite el acceso a los elementos mediante claves.
- **Adaptadores de contenedores**:
  - Se definen a partir de los **contenedores de primera clase**, utilizando sólo algunas de sus características.
- **"Casi" contenedores**:
  - No poseen todas las capacidades de los contenedores de primera clase.

Biblioteca estándar de plantillas (STL)

Contenedores: Tipos de contenedores

Definición (Contenedores de primera clase)

Se denominan **contenedores de primera clase** *a los contenedores de* **secuencia** *y a los contenedores* **asociativos**.

Biblioteca estándar de plantillas (STL)

Contenedores: Tipos de contenedores

Tipos de contenedores

- **Contenedores de secuencia**:
  - **vector**: array unimidensional
  - **list**: lista doblemente enlazada
  - **deque**: cola con doble extremo
- **Contenedores asociativos**:
  - **set**: conjunto
  - **multiset**: multiconjunto o bolsa (con duplicados).
  - **map**: array asociativo
  - **multimap**: array asociativo (con duplicados).

Biblioteca estándar de plantillas (STL)

Contenedores: Tipos de contenedores

Tipos de contenedores

- **Adaptadores de contenedores**:
  - **stack**: pila
  - **queue**: cola
  - **priority_queue**: cola con prioridad
- **"Casi" contenedores**:
  - **Array del tipo del lenguaje C**
  - **string**: cadena de caracteres
  - **bibset**: conjunto de mapas de bits
  - **valarray**: permite operaciones vectoriales matemáticas de alta velocidad.

Biblioteca estándar de plantillas (STL)

Contenedores: Ficheros de cabecera

Ficheros de cabecera

- <**vector**>: vector.
- <**list**>: list.
- <**deque**>: deque.
- <**set**>: set y multiset.
- <**map**>: map y multimap.
- <**stack**>: stack.
- <**queue**>: queue y priority_queue.
- <**string**>: string
- <**bitset**>: bibset

Biblioteca estándar de plantillas (STL)

Contenedores: Funciones miembro

Funciones miembro de los contenedores (1/2)

- **Constructor predeterminado y constructor de copia**
- **Destructor**
- **c.empty()**: devuelve **true** si no hay elementos; en caso contrario, **false**

donde **c** representa un contenedor.

Biblioteca estándar de plantillas (STL)

Contenedores: Funciones miembro

Funciones miembro de los contenedores (2/2)

- **c.max_size()**: devuelve el número máximo de elementos del contenedor.
- **c.size()**: devuelve el número de elementos actuales del contenedor
- **c.resize()**: modifica el tamaño de un contenedor de secuencia. Sólo válido para "vector, list, y deque"
- **c1.swap(c2)**: intercambia los elementos de los contenedores c1 y c2.
- **operator=**: asigna un contenedor a otro.

donde **c, c1 y c2** representan contenedores.

Biblioteca estándar de plantillas (STL)

Contenedores: Operadores relacionales

Operadores relacionales entre contenedores

- **operator**<: operador "menor que" entre contenedores.
- **operator**<=: operador "menor o igual que" entre contenedores.
- **operator**>: operador "mayor que" entre contenedores.
- **operator**>=: operador "mayor o igual que" entre contenedores.
- **operator**==: operador "igual que" entre contenedores.
- **operator**! =: operador "distinto" entre contenedores.

Nota

El contenedor **priority_queue** *no dispone de los operadores relacionales*

Biblioteca estándar de plantillas (STL)

Contenedores: Operadores relacionales

Operadores relacionales entre contenedores

- **operator**<: operador "menor que" entre contenedores.
- **operator**<=: operador "menor o igual que" entre contenedores.
- **operator**>: operador "mayor que" entre contenedores.
- **operator**>=: operador "mayor o igual que" entre contenedores.
- **operator**==: operador "igual que" entre contenedores.
- **operator**! =: operador "distinto" entre contenedores.

Nota

El contenedor **priority_queue** *no dispone de los operadores relacionales*

Biblioteca estándar de plantillas (STL)

Contenedores: Operadores relacionales

Nota

Más adelante se describirán los operadores relacionales que pueden ser utilizados por los **iteradores** *que apuntan a los elementos de los contenedores.*

Biblioteca estándar de plantillas (STL)

Contenedores: Iteradores

Iteradores de los contenedores de primera clase

- **c.begin()**: devuelve un **iterator** o **const_iterator** que hace referencia al **primer** elemento.
- **c.end()**: devuelve un **iterator** o **const_iterator** que hace referencia a la **siguiente** posición después del final del contenedor.
- **c.rbegin()**: devuelve un **reverse_iterator** o **const_reverse_iterator** que hace referencia al **último** elemento.
- **c.rend()**: devuelve un **reverse_iterator** o **const_reverse_iterator** que hace referencia a la posición **anterior** al primer elemento.

donde **c** representa un contenedor.

Biblioteca estándar de plantillas (STL)

Contenedores: Operaciones de borrado

Operaciones de borrado en un contenedor de primera clase

- **c.erase**:
  - c.erase(x): elimina el elemento con valor x.
  - c.erase(p): elimina el elemento apuntado por el iterador p.
  - c.erase(primero,último): elimina los elementos del intervalo [primero,último)
- **c.clear()**: elimina todos los elementos del contenedor.

donde **c** representa un contenedor.

Biblioteca estándar de plantillas (STL)

Contenedores: Operaciones en los extremos (1/2)

Operaciones al final de vectores, listas, deques, pilas y colas

- **c.push_back()**: añade al final.
- **c.pop_back()**: elimina el último elemento.

donde **c** representa un contenedor.

Biblioteca estándar de plantillas (STL)

Contenedores: Operaciones en los extremos (2/2)

Operaciones al principio de listas y deques

- **c.push_front()**: añade un primer elemento nuevo
- **c.pop_front()**: elimina el primer elemento

donde **c** representa un contenedor.

Contenido de la sección

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: Tipos

Tipos de contenedores de secuencia

- **vector**: array unidimensional.
- **list**: lista doblemente enlazada.
- **deque**: array unidimensional de doble extremo.

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: Acceso a los elementos

Modos de acceso a elementos

- **c.front()**: primer elemento.
- **c.back()**: útimo elemento.
- Indexación de elementos (sólo válido para "vector" y "deque")
  - **c.[i]**: indexación del elemento i-ésimo; acceso sin verificación
  - **c.at(i)**: indexación elemento i-ésimo; acceso con verificación

donde **c** representa un contenedor.

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: Funciones miembro

Funciones miembro (1/2)

- **assign**:
  - c.assign(primero,último);
    asigna a c los elementos de otro contenedor indicados por los iteradores [primero,último).

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: Funciones miembro

Funciones miembro (2/2)

- **insert**:
  - c.insert(c.begin(), 0);
    Inserta el valor 0 en la posición indicada del contenedor.
  - c.insert(c.begin(), 5, 0);
    Inserta 5 veces el valor 0 a partir de la posición indicada del contenedor.
  - c1.insert(c1.begin(),c2.begin(),c2.end());
    Inserta en c1, a partir de la posición c1.begin(), los elementos de c2 indicados en el rango de valores [c2.begin(),c2.end()).

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: Vector

Definición (vector)

Un **vector** *es un array unidimensional con posiciones contiguas de memoria.*

Ejemplos

- *vector<int> v;*
- *vector<double> temperatura(10);*
- *vector<Donante> donantes(persona.begin(),personas.end());*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: Vector

Definición (vector)

Un **vector** *es un array unidimensional con posiciones contiguas de memoria.*

Ejemplos

- *vector<int> v;*
- *vector<double> temperatura(10);*
- *vector<Donante> donantes(persona.begin(),personas.end());*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: Vector

Caracerísticas del contenedor **vector**

- Se recomienda su uso si los datos deben ordenarse y ser fácilmente accesibles.
- Se puede recorrer con iteradores de acceso aleatorio o directo usando los operadores de subíndice **[]** o **at**
- Si la memoria asignada a un vector se agota entonces:
  1. Se le asigna un área contigua más extensa
  2. Se copian los elementos originales.
  3. Se suprime la asignación a la memoria anterior.

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: Vector

Funciones miembro de **vector**

El contenedor **vector** posee

- Las **funciones miembro comunes a todos contenedores**.

  En particular

  - **v.push_back()**: añade al final.
  - **v.pop_back()**: elimina el último elemento.

- Las funciones miembro comunes a los contenedores de **secuencia**.

  En particular las de acceso aleatorio **v.[i]** y **v.at(i)**.

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: Vector

Funciones miembro específicas de **vector**

- **c.capacity()**: espacio asignado.
- **c.reserve()**: reserva espacio para expansiones futuras.

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: List

Definición (list)

Un contenedor **list** *es una lista doblemente enlazada.*

Ejemplos

- *list<int> lista_enteros;*
- *list<double> notas(30);*
- *list<Mensaje> lista_mensajes(avisos.begin(),avisos.end());*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: List

Definición (list)

Un contenedor **list** *es una lista doblemente enlazada.*

Ejemplos

- *list<int> lista_enteros;*
- *list<double> notas(30);*
- *list<Mensaje> lista_mensajes(avisos.begin(),avisos.end());*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: List

Características del contenedor **list**

- Permite la inserción y eliminación en cuaquier posición del contenedor.
- Se implementa como una lista doblemente enlazada.
- Se puede recorrer con iteradores bidireccionales.

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: List

Funciones miembro de **list**

El contenedor **list** posee

- Las **funciones miembro comunes a todos contenedores**.

  En particular

  - **lista.push_back()**: añade al final.
  - **lista.pop_back()**: elimina el último elemento.
  - **lista.push_front()**: añade un primer elemento nuevo
  - **lista.pop_front()**: elimina el primer elemento

- Las funciones miembro comunes a los contenedores de **secuencia**.

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: List

Funciones miembro específicas de **list** (1/2)

- **merge**:
  - lista1.merge(lista2);
    Elmina los elementos de lista2 y los añade al final de lista1.
- **remove**:
  - lista1.remove(4);
    Elmina los elementos de lista1 iguales a 4
- **remove_if**:
  - lista1.remove_if(predicado);
    Elmina los elementos de lista1 que hagan verdadero el *predicado*

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: List

Funciones miembro específicas de **list** (2/2)

- **sort**:
  - lista1.sort();
    Ordena los elementos de lista1.
- **splice**:
  - lista1.splice(lista1.end(),lista2);
    Elmina los elementos de lista2 y los añade al final de lista1.
- **unique**:
  - lista1.unique();
    Elmina los elementos contiguos y duplicados de lista1.

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: Deque

Definición (deque)

Un contenedor **deque** *es un array unidimensional de doble entrada.*

Ejemplos

- *deque<int> deque_enteros;*
- *deque<double> datos(100);*
- *deque<Deportista> clasificacion_ciclistas(deportistas.begin(),deportistas.end());*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: Deque

Definición (deque)

Un contenedor **deque** *es un array unidimensional de doble entrada.*

Ejemplos

- *deque<int> deque_enteros;*
- *deque<double> datos(100);*
- *deque<Deportista> clasificacion_ciclistas(deportistas.begin(),deportistas.end());*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: Deque

Características del contenedor **deque**

- Permite la inserción y eliminación al principio y al final del contenedor.
- Se implementa como un array unidimensional con posiciones no necesarimente contiguas..
- Se puede recorrer con iteradores de acceso aleatorio.
- El término **deque** se pronuncia **"diik"**.
- El contenedor **deque** es la implementación subyacente para el adaptador de cola **queue**.

Biblioteca estándar de plantillas (STL)

Contenedores de secuencia: Deque

Funciones miembro de **deque**

El contenedor **deque** posee

- Las **funciones miembro comunes a todos contenedores**.

  En particular

  - **d.push_back()**: añade al final.
  - **d.pop_back()**: elimina el último elemento.
  - **d.push_front()**: añade un primer elemento nuevo
  - **d.pop_front()**: elimina el primer elemento

- Las funciones miembro comunes a los contenedores de **secuencia**.

  En particular las de acceso aleatorio **d.[i]** y **d.at(i)**.

donde **d** es un contenedor **deque**.

Contenido de la sección

Biblioteca estándar de plantillas (STL)

Contenedores asociativos: Introducción

Contenedores asociativos

- **set**: conjunto (sin duplicados).
- **multiset**: multiconjunto o bolsa (permite duplicados).
- **map**:
  - Array asociativo compuesto por pares de elementos
  - No permite duplicados: asocia un único valor a cada clave única.
- **multimap**:
  - Array asociativo compuesto por pares de elementos
  - Permite duplicados: puede asociar varios valores a una clave.

Biblioteca estándar de plantillas (STL)

Contenedores asociativos: Función miembro

Función miembro específica

- **c.insert(x)**: añade x al contenedor asociativo **c**.

Biblioteca estándar de plantillas (STL)

Contenedores asociativos: Set

Definición (set)

Un contenedor **set** *representa un conjunto con los elementos ordenados.*

Ejemplos

- *set<int> conjunto_enteros;*
- *set<Persona> conjunto_amigos(personas.begin(),personas.end());*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Contenedores asociativos: Set

Definición (set)

Un contenedor **set** *representa un conjunto con los elementos ordenados.*

Ejemplos

- *set<int> conjunto_enteros;*
- *set<Persona> conjunto_amigos(personas.begin(),personas.end());*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Contenedores asociativos: Set

Características del contenedor **set**

- Se utiliza para almacenar y recuperar claves únicas
- No posee elementos duplicados.
- Se puede recorrer con iteradores de bidireccionales, pero no con los iteradores de acceso aleatorio.
- Generalmente se implementa como un árbol binario de búsqueda.

Biblioteca estándar de plantillas (STL)

Contenedores asociativos: Multiset

Definición (multiset)

Un contenedor **multiset** *representa una bolsa con los elementos ordenados.*

Ejemplos

- *multiset<int> bolsa_enteros;*
- *multiset<Fecha> bolsa_reunion(fechas.begin(),fechas.end());*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Contenedores asociativos: Multiset

Definición (multiset)

Un contenedor **multiset** *representa una bolsa con los elementos ordenados.*

Ejemplos

- *multiset<int> bolsa_enteros;*
- *multiset<Fecha> bolsa_reunion(fechas.begin(),fechas.end());*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Contenedores asociativos: Multiset

Características del contenedor **multiset**

- Se utiliza para almacenar y recuperar claves que no son únicas
- Permite elementos duplicados.
- Se puede recorrer con iteradores de bidireccionales, pero no con los iteradores de acceso aleatorio.
- Generalmente se implementa como un árbol binario de búsqueda.

Biblioteca estándar de plantillas (STL)

Contenedores asociativos: Map

Definición (map)

Un contenedor **map** *representa un conjunto de pares de elementos ordenados.*

Ejemplos

- *map<int,string> conjunto_dni_personas;*
- *map<Par> conjunto_pares(pares.begin(),pares.end());*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Contenedores asociativos: Map

Definición (map)

Un contenedor **map** *representa un conjunto de pares de elementos ordenados.*

Ejemplos

- *map<int,string> conjunto_dni_personas;*
- *map<Par> conjunto_pares(pares.begin(),pares.end());*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Contenedores asociativos: Map

Características del contenedor **map**

- El primer elemento del par representa la clave y el segundo, el valor.
- Se utiliza para almacenar y recuperar claves que son únicas
- No permite elementos duplicados.
- Se puede recorrer con iteradores de bidireccionales, pero no con los iteradores de acceso aleatorio.
- Generalmente se implementa como un árbol binario de búsqueda.

Biblioteca estándar de plantillas (STL)

Contenedores asociativos: Multimap

Definición (multimap)

Un contenedor **multimap** *representa una bolsa de pares de elementos ordenados.*

Ejemplos

- *multimap<int,string> bolsa_dni_personas;*
- *multimap<Par> bolsa_pares(pares.begin(),pares.end());*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Contenedores asociativos: Multimap

Definición (multimap)

Un contenedor **multimap** *representa una bolsa de pares de elementos ordenados.*

Ejemplos

- *multimap<int,string> bolsa_dni_personas;*
- *multimap<Par> bolsa_pares(pares.begin(),pares.end());*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Contenedores asociativos: Multimap

Características del contenedor **multimap**

- El primer elemento del par representa la clave y el segundo, el valor.
- Se utiliza para almacenar y recuperar claves que no son únicas
- Permite elementos duplicados.
- Se puede recorrer con iteradores de bidireccionales, pero no con los iteradores de acceso aleatorio.
- Generalmente se implementa como un árbol binario de búsqueda.

Contenido de la sección

Biblioteca estándar de plantillas (STL)

Adaptadores de contenedores: Stack

Definición (stack)

Un contenedor **stack** *representa una* **pila** *elementos.*

Ejemplo

- *stack<int> pila_enteros_1; // basada en deque*
- *stack<int, vector<int> > pila_enteros_2; // basada en vector*
- *stack<int, list<int> > pila_enteros_3; // basada en list*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Adaptadores de contenedores: Stack

Definición (stack)

Un contenedor **stack** *representa una* **pila** *elementos.*

Ejemplo

- *stack<int> pila_enteros_1; // basada en deque*
- *stack<int, vector<int> > pila_enteros_2; // basada en vector*
- *stack<int, list<int> > pila_enteros_3; // basada en list*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Adaptadores de contenedores: Stack

Características del contenedor **stack**

- Representa una **pila** de elementos: *el último en entrar es el primero en salir*.
- Se puede implementar con cualquier contenedor de secuencia: vector, list, deque.
- La implementación subyacente predeterminada de la pila es con deque.
- No posee iteradores.

Biblioteca estándar de plantillas (STL)

Adaptadores de contenedores: Stack

Funciones miembro de **stack**

El contenedor **stack** posee

- Las **funciones miembro comunes a todos contenedores**.

  En particular

  - **pila.empty()**: añade al final.
  - **pila.size()**: elimina el último elemento.
  - Etc.

Biblioteca estándar de plantillas (STL)

Adaptadores de contenedores: Stack

Funciones miembros específicas de **stack**

- **pila.push()**: inserta un elemento en la cima de la pila.
- **pila.pop()**: extrae el elemento situado en la cima de la pila.
- **pila.top()**: consulta el elemento situado en la cima de la pila.

Biblioteca estándar de plantillas (STL)

Adaptadores de contenedores: Queue

Definición (queue)

Un contenedor **queue** *representa una* **cola** *elementos.*

Ejemplo

- *queue<int> cola_enteros_1; // basada en deque*
- *queue<int, list<int> > cola_enteros_2; // basada en list*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Adaptadores de contenedores: Queue

Definición (queue)

Un contenedor **queue** *representa una* **cola** *elementos.*

Ejemplo

- *queue<int> cola_enteros_1; // basada en deque*
- *queue<int, list<int> > cola_enteros_2; // basada en list*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Adaptadores de contenedores: Queue

Características del contenedor **queue**

- Representa una **cola** de elementos: *el primero en entrar es el primero en salir*.
- Se puede implementar con los contenedores de secuencia list o deque.
- La implementación subyacente predeterminada de la cola es con deque.
- No posee iteradores.

Biblioteca estándar de plantillas (STL)

Adaptadores de contenedores: Queue

Funciones miembro de **queue**

El contenedor **queue** posee

- Las **funciones miembro comunes a todos contenedores**.

  En particular

  - **cola.empty()**: añade al final.
  - **cola.size()**: elimina el último elemento.
  - Etc.

Biblioteca estándar de plantillas (STL)

Adaptadores de contenedores: Queue

Funciones miembros específicas de **queue**

- **cola.push()**: inserta un elemento al final de la cola.
- **cola.pop()**: extrae el primer elemento de la cola.
- **cola.front()**: consulta el primer elemento de la cola
- **cola.back()**: consulta el último elemento de la cola

Biblioteca estándar de plantillas (STL)

Adaptadores de contenedores: Priority_queue

Definición (priority_queue)

Un contenedor **priority_queue** *representa una* **cola** *elementos con prioridades.*

Ejemplo

- *priority_queue<int> prioridades_enteros_1; // basada en vector*
- *priority_queue<int, deque<int> > prioridades_enteros_2; // basada en deque*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Adaptadores de contenedores: Priority_queue

Definición (priority_queue)

Un contenedor **priority_queue** *representa una* **cola** *elementos con prioridades.*

Ejemplo

- *priority_queue<int> prioridades_enteros_1; // basada en vector*
- *priority_queue<int, deque<int> > prioridades_enteros_2; // basada en deque*
- *Etc.*

Biblioteca estándar de plantillas (STL)

Adaptadores de contenedores: Priority_queue

Características del contenedor **priority_queue**

- Representa una **cola** de elementos: *se hacen inserciones ordenadas y siempre se extrae por su parte inicial*.
- Se puede implementar con los contenedores de secuencia vector o deque.
- La implementación subyacente predeterminada de la cola con prioridades es con vector.
- No posee iteradores.

Biblioteca estándar de plantillas (STL)

Adaptadores de contenedores: Priority_queue

Funciones miembro de **priority_queue**

El contenedor **priority_queue** posee

- Las **funciones miembro comunes a todos contenedores**.

  En particular

  - **cola.empty()**: añade al final.
  - **cola.size()**: elimina el último elemento.
  - Etc.

Biblioteca estándar de plantillas (STL)

Adaptadores de contenedores: Priority_queue

Funciones miembros específicas de **priority_queue**

- **cola_prioridades.push()**: inserta un elemento en la cola teniendo en cuenta su prioridad.
- **cola_prioridades.pop()**: extrae el primer elemento de la cola.
- **cola_prioridades.front()**: consulta el primer elemento de la cola
- **cola_prioridades.back()**: consulta el último elemento de la cola

Contenido de la sección

Biblioteca estándar de plantillas (STL)

Iteradores

Tipos de iteradores

- **iterator**: iterador que apunta al tipo de elemento almacenado en un contenedor.
- **const_iterator**: iterador constante que apunta al tipo de elemento almacenado en un contenedor; sólo se puede usar para la **leer** elementos.
- **reverse_iterator**: iterador inverso que apunta al tipo de elemento almacenado en un contenedor; permite iterar en sentido inverso.
- **const_reverse_iterator**: iterador inverso y constante que apunta al tipo de elemento almacenado en un contenedor; permite iterar en sentido inverso; sólo se puede usar para la **leer** elementos.

Biblioteca estándar de plantillas (STL)

Iteradores

Tipos de iteradores

Tipo de iterador	Dirección de ++	Capacidad
iterator	avance	lectura y escritura
const_iterator	avance	lectura
reverse_iterator	retroceso	lectura y escritura
const_reverse_iterator	retroceso	lectura

Biblioteca estándar de plantillas (STL)

Iteradores

Categoría de los iteradores

Categoría	Capacidad	Movimiento	Pasadas
Entrada	Lectura	Adelante	Una
Salida	Escritura	Adelante	Una
Avance	Lectura y escritura	Adelante	Una
Bidireccional	Lectura y escritura	Adelante y atrás	Varias
Aleatorio	Lectura y escritura	Directo, adelante y atrás	Varias

Biblioteca estándar de plantillas (STL)

Iteradores: Operaciones de los iteradores

Operaciones de todos los iteradores de los contenedores

Tipo de operación	Descripción
++p	incremento previo del iterador
p++	incremento posterior del iterador

Biblioteca estándar de plantillas (STL)

Iteradores: Operaciones de los iteradores

Operaciones de los iteradores de entrada

Tipo de operación	Descripción
***p**	acceso al contenido del iterador (para usar su valor, rvalue)
p1 = p2	asginar el iterador p2 al iterador p1
p1 == p2	igualdad de iteradores
p1 != p2	desigualdad de iteradores

Biblioteca estándar de plantillas (STL)

Iteradores: Operaciones de los iteradores

Operaciones de los iteradores de salida

Tipo de operación	Descripción
***p**	acceso al contenido del iterador (para asignarle un valor, lvalue)
p1 = p2	asginar el iterador p2 al iterador p1

Biblioteca estándar de plantillas (STL)

Iteradores: Operaciones de los iteradores

Operaciones de los iteradores de avance

- Los iteradores de avance poseen todos las operaciones de los iteradores de entrada y de salida

Operaciones adicionales de los iteradores bidireccionales

Tipo de operación	**Descripción**
$--\mathbf{p}$	decremento previo del iterador
$\mathbf{p}--$	decremento posterior del iterador

Biblioteca estándar de plantillas (STL)

Iteradores: Operaciones de los iteradores

Operaciones adicionales de los iteradores de acceso aleatorio (1/2)

Tipo de operación	Descripción
p += i	incrementar el iterador p en i posiciones
p -= i	decrementar el iterador p en i posiciones
p + i	crea un iterador situado en p incrementado en i posiciones
p - i	crea un iterador situado en p decrementado en i posiciones
p[i]	contenido del elemento referenciado por p incrementado en i posiciones

Biblioteca estándar de plantillas (STL)

Iteradores: Operaciones de los iteradores

Operaciones adicionales de los iteradores de acceso aleatorio (2/2)

Tipo de operación	Descripción
p1 < p2	devuelve **true** si p1 se encuentra antes que p2
p1 <= p2	devuelve **true** si p1 se encuentra antes o en la misma posición que p2
p1 > p2	devuelve **true** si p1 se encuentra después que p2
p1 >= p2	devuelve **true** si p1 se encuentra después o en la misma posición que p2
p1 == p2	devuelve **true** si p1 se encuentra en la misma posición que p2
p1 != p2	devuelve **true** si p1 se encuentra en una posición diferente que p2

Biblioteca estándar de plantillas (STL)

Iteradores: Iteradores de los contenedores

Tipo de iterador soportado por los contenedores

Contenedor	Tipo de iterador
vector	acceso aleatorio
deque	acceso aleatorio
list	bidireccional
set	bidireccional
multiset	bidireccional
map	bidireccional
multimap	bidireccional
stack	no soporta iterador
queue	no soporta iterador
priority_queue	no soporta iterador

Contenido de la sección

Biblioteca estándar de plantillas (STL)

Algoritmos de contenedores de primera clase: Argumentos de los algoritmos

Argumentos de los algoritmos: iteradores bidireccionales o de acceso aleatorio

- Algunos de los argumentos de los algoritmos son pares de iteradores que limitan los elementos del contenedor:
  - Iterador que apunta al primer elemento que se quiere procesar del contenedor.
  - Iterador que apunta a la posición situada después del último elemento que se quiere procesar del contenedor.

Biblioteca estándar de plantillas (STL)

Algoritmos de contenedores de primera clase: Argumentos de los algoritmos

Argumentos de los algoritmos: iteradores bidireccionales o de acceso aleatorio

Ejemplo

algoritmo(c.begin(),c.end(), ...);

donde **c** *es un contenedor y*

- **c.begin()**: *iterador que apunta al primer elemento del contenedor.*
- **c.end()**: *iterador que apunta a la posición situada después del último elemento del contenedor.*

Biblioteca estándar de plantillas (STL)

Algoritmos de contenedores de primera clase

Algoritmos sobre contenedores de <algorithm>

- **Algoritmos invariantes**: realizan una consulta sobre el contenedor.
- **Algoritmos modificadores**: modifican el contenido del contenedor.

Biblioteca estándar de plantillas (STL)

Algoritmos de contenedores de primera clase: Algoritmos invariantes

Tipos de algoritmos invariantes

1. **Iterador**: for_each.
2. **Búsqueda en un contenedor no ordenado**: find, find_if, ...
3. **Búsqueda en un contenedor ordenado**: binary_search, lower_range, ...
4. **Contar y comparar**: count, equal, ...
5. **Búsqueda de subsecuencias ordenadas**: search.
6. **Inclusión**: includes.
7. **Máximo y mínimo**: max_element, min_element.
8. **Numéricos**: accumulate, inner_product.

Biblioteca estándar de plantillas (STL)

Algoritmos de contenedores de primera clase: Algoritmos invariantes (1/8)

Algoritmo iterador

- **for_each**: simula la ejecucion de un bucle que aplica una **función** a los elementos del contenedor.

Nota

Este algoritmo será invariante o modificador según el tipo de acción que realice la **función** *sobre los elementos del contenedor.*

Biblioteca estándar de plantillas (STL)

Algoritmos de contenedores de primera clase: Algoritmos invariantes (2/8)

Búsqueda en un contenedor no ordenado

- **find**: devuelve un iterador al primer elemento que coincida con un valor.
- **find_if**: devuelve un iterador al primer elemento que haga verdadero a un predicado.
- **find_first_of**: devuelve un iterador al primer elemento del primer contenedor que coincide con un elemento del segundo contenedor (o que hacen verdadero un predicado).

Biblioteca estándar de plantillas (STL)

Algoritmos de contenedores de primera clase: Algoritmos invariantes (3/8)

Búsqueda en un contenedor ordenado

- **binary_search**: busca un elemento en un contenedor ordenado que sea igual a un valor o que haga verdadero un predicado.
- **lower_bound**: busca el primer elemento en un contenedor ordenado que sea igual a un valor o que haga verdadero un predicado.
- **upper_bound**: busca el último elemento en un contenedor ordenad0 que sea igual a un valor o que haga verdadero un predicado.

Biblioteca estándar de plantillas (STL)

Algoritmos de contenedores de primera clase: Algoritmos invariantes (4/8)

Contar y comparar

- **count**: cuenta las ocurrencias de un valor en un contenedor.
- **count_if**: cuenta el número de elementos que hacen verdadero a un predicado.
- **equal**: comprueba si son iguales dos contenedores (o si los elementos hacen verdadero a un predicado).
- **mismatch**: busca el primer el primer par de elementos que son diferentes (o hacen falso a un predicado) en dos contenedores y devuelve iteradores a dichos elementos.
- **lexicographical_compare**: compara lexicográficamente dos contenedores.

Biblioteca estándar de plantillas (STL)

Algoritmos de contenedores de primera clase: Algoritmos invariantes (5/8)

Búsqueda de subsecuencias

- **search**: busca la segunda secuencia como subsecuencia de la primera secuencia; devuelve un iterador al primer elemento coincidente de la primera secuencia.

Biblioteca estándar de plantillas (STL)

Algoritmos de contenedores de primera clase: Algoritmos invariantes (6/8)

Inclusión

- **includes**: determina si todos los elementos del segundo contenedor pertenecen al primer contenedor.

Biblioteca estándar de plantillas (STL)

Algoritmos de contenedores de primera clase: Algoritmos invariantes (7/8)

Máximo y mínimo

- **max_element**: busca el elemento máximo.
- **min_element**: busca el elemento mínimo.

Biblioteca estándar de plantillas (STL)

Algoritmos de contenedores de primera clase: Algoritmos invariantes (8/8)

Algoritmos numéricos de <numeric>

- **accumulate**: acumula resultados de una operación sobre un contenedor.
- **inner_product**: acumula resultados de una operación sobre dos contenedores
  la operación por defecto es la multiplicación, obteniéndose el producto escalar.

Biblioteca estándar de plantillas (STL)

Algoritmos de contenedores de primera clase: Algoritmo modificador

Algoritmo modificador

1. **Intercambiar**: swap.

Biblioteca estándar de plantillas (STL)

Algoritmos de contenedores de primera clase: Algoritmo modificador

Intercambiar

- **swap**: intercambia los elementos de dos contenedores.

Contenido de la sección

Biblioteca estándar de plantillas (STL)

Algoritmos adicionales sobre contenedores de secuencia

Algoritmos sobre contenedores de secuencia de <algorithm>

- **Algoritmos invariantes**: relizan una consulta sobre el contenedor de secuencia.
- **Algoritmos modificadores**: modifican el contenido del contenedor de secuencia.

Biblioteca estándar de plantillas (STL)

Algoritmos adicionales sobre contenedores de secuencia: Algoritmos invariantes

Tipos de algoritmos invariantes adicionales

1. **Búsqueda en una secuencia no ordenada**: adjacent_find.
2. **Búsqueda en una secuencia ordenada**: equal_range.
3. **Búsqueda de subsecuencias ordenadas**: search_n, find_end

Biblioteca estándar de plantillas (STL)

Algoritmos adicionales sobre contenedores de secuencia: Algoritmos invariantes (1/3)

Búsqueda en una secuencia no ordenada

- **adjacent_find**: busca dos elementos adyacentes que coincidan (o que hagan verdadero un predicado).

Biblioteca estándar de plantillas (STL)

Algoritmos adicionales sobre contenedores de secuencia: Algoritmos invariantes (2/3)

Búsqueda en una secuencia ordenada

- **equal_range**: busca el primer y el último elemento en una secuencia ordenada que sean iguales a un valor o que haga verdadero un predicado.

Biblioteca estándar de plantillas (STL)

Algoritmos adicionales sobre contenedores de secuencia: Algoritmos invariantes (3/3)

Búsqueda de subsecuencias

- **search_n**: encuentra una secuencia de, al menos, n coincidencias de su parámetro valor en la secuencia; devuelve un iterador al primer elemento de la secuencia de n coincidencias.
- **find_end**: busca la segunda secuencia como subsecuencia de la primera secuencia; devuelve un iterador que apunta a la última coincidencia en la primera secuencia (es decir, "search al revés").

Biblioteca estándar de plantillas (STL)

Algoritmos adicionales sobre contenedores de secuencia: Algoritmos modificadores

Tipos de algoritmos modificadores adicionales

1. **Copiar y asignar**: copy, fill, generate, ...
2. **Intercambiar**: swap_ranges, iter_swap, partition, ...
3. **Permutar**: random_shuffle, next_permutation, prev_permutation.
4. **Borrar y copiar**: remove_copy, unique_copy, ...
5. **Sustituir**: replace,...
6. **Invertir y rotar**: reverse, rotate, ...
7. **Ordenar**: sort, stable_sort, ...
8. **Fusionar**: merge, inplace_merge
9. **Sumas y diferencias parciales**: suma_partial, adjacent_difference.

Biblioteca estándar de plantillas (STL)

Algoritmos adicionales sobre contenedores de secuencia: Algoritmos modificadores (1/9)

Copiar y asignar

- **copy**: copia una secuencia en otra.
- **copy_backward**: copia una secuencia comenzando por la parte final de la otra.
- **fill**: asigna a los elementos un valor indicado.
- **fill_n**: asigna a los "n" primeros elementos un valor indicado.
- **generate**: asigna a los elementos valores proporcionados por una función.
- **generate_n**: asigna a los "n" primeros elementos valores proporcionados por una función.

Biblioteca estándar de plantillas (STL)

Algoritmos adicionales sobre contenedores de secuencia: Algoritmos modificadores (2/9)

Intercambiar

- **swap_ranges**: intercambia los elementos especificados en los rangos de entrada.
- **iter_swap**: intercambia los elementos apuntados por los parámetros iteradores.
- **partition**: coloca al principio los elementos que satisfacen un predicado (no se respeta el orden relativo de los elementos con igual valor).
- **stable_partition**: coloca al principio los elementos que satisfacen un predicado (respeta el orden relativo de los elementos con igual valor).

Biblioteca estándar de plantillas (STL)

Algoritmos adicionales sobre contenedores de secuencia: Algoritmos modificadores (3/9)

Permutar

- **random_shuffle**: permuta aleatoriamente los elementos.
- **next_permutation**: si existe la siguiente permutación lexicográfica, la genera; en caso contrario, devuelve **false**.
- **prev_permutation**: si existe la anterior permutación lexicográfica, la genera; en caso contrario, devuelve **false**.

Biblioteca estándar de plantillas (STL)

Algoritmos adicionales sobre contenedores de secuencia: Algoritmos modificadores (4/9)

Borrar y copiar

- **remove_copy**: realiza una copia pero omite los elementos con un valor indicado.
- **remove_copy_if**: realiza una copia pero omite los elementos que hagan verdadero un predicado.
- **unique_copy**: realiza una copia sin duplicados.

Biblioteca estándar de plantillas (STL)

Algoritmos adicionales sobre contenedores de secuencia: Algoritmos modificadores (5/9)

Sustituir

- **replace**: sustituye los elementos con un valor indicado por un nuevo valor.
- **replace_if**: sustituye los elementos que hagan verdadero un predicado por un nuevo valor.
- **replace_copy**: realiza una copia sustituyendo los elementos con un valor indicado por un nuevo valor.
- **replace_copy_if**: realiza una copia sustituyendo los elementos que hagan verdadero un predicado por un nuevo valor.
- **transform**: copia en otra secuencia una transformación de su entrada basada en una operación suministrada por el usuario.

Biblioteca estándar de plantillas (STL)

Algoritmos adicionales sobre contenedores de secuencia: Algoritmos modificadores (6/9)

Invertir y rotar

- **reverse**: invierte el orden de los elementos
- **reverse_copy**: produce una copia con los elementos en orden inverso
- **rotate**: rota los elementos de forma que el elemento elegido pasa a ser el primer elemento.

  El elemento de la posición *primero + i*
  se traslada a la posición
  primero + (i + (último - elegido)) % (último - primero)
- **rotate_copy**: hace una copia con los elementos rotados, de forma que que el elemento elegido pasa a ser el primer elemento.

Biblioteca estándar de plantillas (STL)

Algoritmos adicionales sobre contenedores de secuencia: Algoritmos modificadores (7/9)

Ordenar un **vector** o **deque**

- **sort**: ordena los elementos de la secuencia.
- **stable_sort**: ordena los elementos de la secuencia respetando el orden relativo.
- **partial_sort**: sólo garantiza la ordenación de los primeros elementos de la secuencia.
- **partial_sort_copy**: sólo garantiza la ordenación de los primeros elementos de la secuencia y el resultado lo copia en otra secuencia. El tamaño de la segunda secuencia determina cuántos elementos se van a ordenar.
- **nth_element**: se garantiza que el n-ésimo elemento está en el lugar que le correspondería si la secuencia estuviera ordenada.

Biblioteca estándar de plantillas (STL)

Algoritmos adicionales sobre contenedores de secuencia: Algoritmos modificadores (8/9)

Fusionar o mezclar secuencias ordenadas

- **merge**: fusiona dos secuencias ordenadas y el resultado lo copia en otra secuencia.
- **inplace_merge**: fusiona dos partes ordenadas de un contenedor.

Biblioteca estándar de plantillas (STL)

Algoritmos adicionales sobre contenedores de secuencia: Algoritmos modificadores (9/9)

Algoritmos numéricos adicionales de <numeric>

- **partial_sum**: dada la secuencia a, b, c, d, ..., se genera la secuencia a, a + b, a + b + c, a + b + c + d,...
- **adjacent_difference**: dada la secuencia a, b, c, d, ..., se genera la secuencia a, b - a, c - b, d - c,...

Contenido de la sección

Biblioteca estándar de plantillas (STL)

Algoritmos adicionales sobre contenedores asociativos: Algoritmos de conjuntos

Unión e intersección

- **set_union**: realiza la unión de dos contenedores ordenados.
- **set_intersection**: realiza la intersección de dos contenedores ordenados.
- **set_difference**: diferencia del primer contenedor menos el segundo, ambos ordenados.
- **set_symmetric_difference**: diferencia simétrica de dos contenedores ordenados.

ESTRUCTURAS DE DATOS Y DE LA INFORMACIÓN

Biblioteca estándar de plantillas (STL: Standard Templates Library)

Prof. Dr. Nicolás Luis Fernández García

Departamento de Informática y Análisis Numérico
Escuela Politécnica Superior
Universidad de Córdoba

Curso académico 2009 - 2010