UNSA

UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

ESCUELA PROFESIONAL DE CIENCIAS DE LA COMPUTACIÓN

CIENCIA DE LA COMPUTACIÓN II

GRUPO LABORATORIO A

SEMESTRE 2021-A

Laboratorio 6:

Estructura de datos básica,

lista enlazada – patrón iterator

comparación list vs LinkedList

Alejandro Villa Herrera

21 julio del 2021

1. Standard Templates Library (STL)

La Standard Template Library (STL) es una librería de software para el lenguaje de programación C++. Esta provee de cuatro componentes denominados algoritmos, contenedores, iteradores y funciones.

Una de las dificultades del lenguaje C es la implementación de contenedores genéricos, de fácil uso y eficaces [1].

STL no es la primera de tales librerías, así la mayor parte de los compiladores de C++ disponen (o disponían) de librerías similares y, también, están disponibles varias librerías comerciales. Uno de los problemas de estas librerías es que son mutuamente incompatibles, lo que obliga a los programadores a aprender nuevas librerías y a migrar de un proyecto a otro y de uno a otro compilador. Sin embargo, STL ha sido adoptado por el comité ANSI de estandarización del C++, lo que significa que está (o estará) soportado como una extensión más del lenguaje por todos los compiladores [2].

1.1. Contenedores

Los contenedores son estructuras de datos que almacenan otras estructuras de datos, en un contenedor se pueden realizar diferentes acciones tales como: añadir elementos, buscar elementos, eliminar elementos, ordenar todos los elementos o un rango de elementos en el contenedor, etc.

Los contenedores de la STL se dividen en tres grupos: contenedores de secuencia, contenedores asociativos y contenedores adaptativos.

a) Contenedores de secuencia: almacenan los elementos en posiciones contiguas de memoria, dentro de esta categoría están.
- **Vector.**
- **List.**
- **Deque.**

b) Contenedores asociativos: cada elemento del contenedor tiene asociado una clave, se puede acceder a estos mediante su clave, dentro de esta categoría se encuentran.
- **Set.**
- **Multiset.**
- **Map.**
- **Multimap.**

c) Contenedores adaptativos: se definen a partir de los contenedores de primera clase.
- **Stack.**
- **Queue.**
- **Priority_queue.**

1.2. Iteradores

Método o forma que se usa para poder navegar sobre los elementos de un contenedor específico, dependiendo del tipo de iterador podremos recorrer y acceder a los elementos del contenedor de diferentes formas, podemos recorrer ya sea de inicio a fin (izquierda a derecha), fin a inicio (derecha a izquierda), acceder de forma aleatoria a los elementos, etc.

Un iterador es una especie de puntero que es utilizado por un algoritmo para recorrer los elementos almacenados en un contenedor. Dado que los distintos algoritmos necesitan recorrer los contenedores de diversas maneras para realizar diversas operaciones, y los contenedores deben ser accedidos de formas distintas, existen diferentes tipos de iteradores [3].

a. Categorías de los iteradores.
   - Acceso aleatorio: permite lectura y escritura, movimiento directo, adelante y atrás.
   - Bidireccional: permite lectura y escritura, movimiento adelante y atrás.
   - Avance: permite lectura y escritura, movimiento adelante.
   - Salida: permite escritura, movimiento adelante.
   - Entrada: permite lectura, movimiento adelante.

b. Tipos de iteradores.
   - Iterator: permite avance, lectura y escritura.
   - Const_Iterator: permite avance, solo lectura.
   - Reverse_Iterator: permite retroceso, lectura y escritura.
   - Const_reverse_Iterator: permite retroceso, solo lectura.

c. Tipo de iterador soportado por 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

1.3. Algoritmos

STL proporciona una amplia variedad de algoritmos comunes entre los que se incluyen los de ordenación, búsqueda y algoritmos numéricos. Los algoritmos se pueden utilizar con estructuras de datos de la librería, vectores, o estructuras de datos definidas por el usuario y provistas de iteradores [2].

2. List.

Un contenedor list es una lista doblemente enlazada, se puede recorrer con iteradores bidireccionales (en ambos sentidos).

Entre los diferentes tipos de operaciones que proporcionan las listas destacan:

- Inserciones y borrados al principio y al final.
- Inserciones y borrados en posiciones intermedias.

2.1. Propiedades.

- Los elementos son almacenados en una estricta secuencia linear.
- El contenedor usa un objeto asignador para manejar dinámicamente sus necesidades de almacenamiento.

- Cada elemento guarda información sobre cómo ubicar el elemento anterior y el siguiente, lo que permite operaciones de inserción y borrado de tiempo constante antes o después de un elemento específico (incluso de rangos completos), pero sin acceso aleatorio directo.
- Se puede recorrer sus elementos con iteradores bidireccionales.

2.2. Parámetros de plantilla

- T: tipo de elemento.
- Alloc: Tipo de objeto asignador utilizado para definir el modelo de asignación de almacenamiento.

2.3. Funciones miembro.

a) Constructores.
   - Default constructor: construye un contenedor vacío.
   - Fill constructor: construye un contenedor con n elementos (primer parámetro), cada elemento es una copia del valor dado (segundo parámetro, o 0 si no se proporciona).
   - Range constructor: construye un contenedor con los elementos dado por el rango [first, last).
   - Copy constructor: construye un contenedor copiando cada uno de los elementos de otro contenedor.
   - Move constructor: construye un contenedor que adquiere los elementos de otro contenedor.
   - Initializer list constructor: construye un contenedor con cada elemento dado por la lista de inicialización.

b) Destructor.
   - Destruye todos los elementos del contenedor.

c) Funciones miembro comunes con otros contenedores:
   - Lista.push_back(): añade al final.
   - Lista.push_front(): añade al comienzo.
   - Lista.pop_back(): elimina al final.
   - Lista.pop_front(): elimina al comienzo.

d) Funciones miembro comunes a los contenedores de secuencia:
   - Merge(): lista1.merge(lista2): elimina los elementos de la lista 2 y los añade al final de la lista 1.
   - Remove(): lista1.remove(4): elimina los elementos de la lista iguales a 4.
   - Remove_if(): lista1.remove_if(predicado): elimina los elementos de la lista que hagan verdadero el predicado.
   - Sort(): lista1.sort(): ordena los elementos de la lista.
   - Splice(): lista1.splice(lista.end(),lista2): elimina los elementos de la lista2 y los añade en la posición indicada por el primer parámetro.
   - Unique(): lista1.unique() Elimina los elementos contiguos y duplicados de lista1.

e) Funciones miembro para trabajar con iteradores:

- Iterator:
    - Lista.begin( ): devuelve un iterator al primer elemento.
    - Lista.end(): devuelve un iterator al primer elemento después de ultimo.
- Reverse_iterator:
    - Lista.rbegin(): devuelve un reverse_iterator al final.
    - Lista.rend(): devuelve un reverse_iterator al elemento antes de primero.
- Const_iterator:
    - Lista.cbegin( ): devuelve un const_iterator al primer elemento.
    - Lista.cend(): devuelve un const_iterator al primer elemento después de ultimo.
- Const_reverse_iterator:
    - Lista.crbegin( ): devuelve un const_reverse_iterator al final.
    - Lista.crend(): devuelve un const_reverse_iterator al elemento antes de primero.

3. LinkedList.

Implementada como una lista simplemente enlazada, se puede recorrer con iteradores bidireccionales (en ambos sentidos).

Operaciones disponibles:

- Inserciones y borrados al principio y al final.
- Inserciones y borrados en posiciones intermedias.
- Se puede recorrer sus elementos con iteradores bidireccionales.

3.1. Propiedades.

- Los elementos almacenados en una estricta secuencia linear.
- Brinda información de como acceder a elementos anteriores o siguientes a los iteradores.

3.2. Parámetros de plantilla

- T: tipo de elemento.

3.3. Funciones miembro.

a. Constructores.
    - Default constructor: construye un contenedor vacío.
    - Fill constructor: construye un contenedor con n elementos (primer parametro), cada elemento es una copia del valor dado (segundo parámetro).
    - Copy constructor: construye un contenedor copiando cada uno de los elementos de otro contenedor.
    - Move constructor: construye un contenedor que adquiere los elementos de otro contenedor.

b. Destructor.
    - Destruye todos los elementos del contenedor.

c. Funciones miembro:
    - Lista.push_back(): añade al final.
    - Lista.push_front(): añade al comienzo.

- Lista.pop_back(): elimina al final.
- Lista.pop_front(): elimina al comienzo.
- Lista.insert(value , posición): inserta un elemento en la posición indicada.
- Lista.erase(posición): elimina según posicion
- Lista.clear(): deja vacio el contenedor
- Lista.front(): devuelve el primer elemento.
- Lista.back(): devuelve el ultimo elemento.
- Find(nodo , bool): devuelve un puntero al elemento que se encuentra antes o después, esto se define según el segundo parámetro:
    - bool = true, devuelve el siguiente.
    - bool = false, devuelve el anterior.

d. Funciones miembro para trabajar con iteradores:

- Iterator:
    - Lista.begin(): devuelve un iterator al primer elemento. Si no hay elementos en la lista, devuelve un iterator a _end.
    - Lista.end(): devuelve un iterator al primer elemento después de ultimo. El elemento _end se ubica despues del ultimo elemento siempre y cuando exista al menos un elemento en el nodo, si no existe ningún elemento, head apunta a nada y _end no tiene valor pero si existe en memoria.

4. Comparación List vs LinkedList.

LIST.	LINKEDLIST.
Constructors. • Empty container constructor (default constructor). • Fill constructor. • Range constructor. • Copy constructor. • Move constructor. • Initializer list constructor.	**Constructors.** • Empty container constructor (default constructor). • Fill constructor. • Copy constructor. • Move constructor.
Destructor	**Destructor**
Operator=. • Copy assignment. • Move assignment. • Initializer list assignment.	**Operator=.** • Copy assignment. • Move assignment.
Iterator (bidirectional). • Iterator. • Const_iterator. • Reverse_iterator.	**Iterator (bidirectional).** • Iterator.

• Const_reverse_iterator.	
Capacity. • Empty. • Size. • Max_size.	**Capacity.** • Empty. • Size.
Element access. • Front. • Back.	**Element access.** • Front. • Back.
Modifiers. • Assign. • Emplace_front. • Push_front. • Pop_front. • Emplace_back. • Push_back. • Pop_back. • Emplace. • Insert. • Erase. • Swap. • Resize. • Clear.	**Modifiers.** • Push_front. • Pop_front. • Push_back. • Pop_back. • Insert. • Erase. • Clear.
Operations. • Splice. • Remove. • Remove_if. • Unique. • Merge. • Sort. • Reverse.	**Operations.**
Overload. • Operator==. • Operator!=. • Operator<. • Operator<=. • Operator>. • Operator>=.	**Overload.** • Operator==. • Operator!=.

Bibliografía

[1] NGuerrero, «Programa en línea,» 17 setiembre 2019. [En línea]. Available: https://www.programaenlinea.net/introduccion-la-stl-c-standard-template-library/.

[2] «Departamento de Ciencias de la Computacion e Inteligencia Aritificial,» Universidad de Granada, [En línea]. Available: https://ccia.ugr.es/~jfv/ed1/c++/cdrom4/paginaWeb/stl.htm. [Último acceso: 26 julio 2021].

[3] «Curso C++,» Zator Systems, [En línea]. Available: https://www.zator.com/Cpp/E5_1_2.htm.

[4] N. L. Fernandez Garcia, «STL: Standard Templates Library,» Escuela Politécnica Superior de Córdova, Unicersidad de Córdova, Córdova.

[5] «Cplusplus <Iterator>,» Cplusplus, [En línea]. Available: https://www.cplusplus.com/reference/iterator/. [Último acceso: 20 julio 2020].

[6] «Cplusplus <list>,» Cplusplus, [En línea]. Available: https://www.cplusplus.com/reference/list/list/. [Último acceso: 20 julio 2021].