Standard Template Library (STL)

Dr. Luis Gerardo de la Fraga

Cinvestav, Departamento de Computación
Av. IPN 2508, 07360 Ciudad de México
E-mail: fraga@cs.cinvestav.mx

7 de noviembre, 2017

La Biblioteca de Plantillas Estándar (the Standard Template Library)

Provee clases y métodos en C++ para realizar:

- Contenedores (containers) para almacenar información
- Iteradores para accesar a la información almacenada, y
- Algoritmos para manipular el contenido de los contenedores

Contenedores de la BPS

Existen dos tipos:

- **Contenedores secuenciales:**
  Vectores, colas y listas
- **Contenedores asociativos:**
  Conjuntos, mapas, multiconjuntos y multimapas

Contedores secuenciales (1/2)

- Están caracterizados por un tiempo de inserción rápido
- Son relativamente lentos en la operación 'encontrar'

Contenedores secuenciales (2/2)

- `std::vector` Opera igual que un arreglo dinámico, pero puede crecer al final
- `std::deque` Implementa una cola: los elementos pueden insertar o borrarse en ambos extremos.
- `std::list` Implemente una cola doblemente ligada.
- `std::forward_list` Implemente una cola simple.

Contenedores asociativos (1/2)

- Actúan como diccionarios
- Tienen tiempos de inserción lentos, pero
- presentan ventajas significativas en la búsqueda

Contenedores asociativos (2/2)

- `std::set` inserta en un árbol, resultando en una complejidad logarítmica
- `std::unordered_set` La complejidad es constante
- `std::map` Almacena pares llave–valor ordenado por sus llaves únicas en un contenedor con complejidad logarítmica (un árbol)
- `std::unordered_map` Almacena también pares llave–valor, ordenado por sus llaves únicas en un contenedor con complejidad constante
- `std::multiset`
- `std::unordered_multiset`
  Pueden almacenar varios elementos con el mismo valor
- `std::multimap`
- `std::unordered_multiset`
  La llave puede esta repetida, entonces está no es única.

La *complejidad* es un indicador del rendimiento del contenedor con relación a su número de elementos.

- *Complejidad constante* (como en el caso de `std::unordered_map`, no está relacionada con el número de elementos. Se tarda lo mismo en un contenedor con mil o un millón de elementos.
- *Complejidad logarítmica* (como en el `srd::map`) indica que el rendimiento es proporcional al logaritmo del número de elementos en el contenedor. Se tardaría tres veces más con un millón de elementos que con mil.
- *Complejidad lineal* indica que el rendimiento es proporcional al número de elementos. El contendor será mil veces más lento en procesar un millón de elementos que en procesar mil.

Contenedores adaptados

- `std::stack`
- `std::queue`
- `std::priority_queue`

Iteradores de la BPS

- El ejemplo más sencillo es un apuntador
- Los iteradores son generalizaciones de un apuntador
- Con estos iteradores se puede manipular y realizar operaciones sobre un contenedor.
- Existen **iteradores de entrada** (de solo lectura) e **iteradores de salida** (para escribir en los elementos)
- **Iterador de avance** se usa generalmente en una lista simple. Permite tanto la entrada como la salida.
- **Iterador bidireccional** se usa normalmente en una lista doblemente ligada.
- **Iterador de acceso aleatorio** se usa en arreglos.

Algoritmos de la BPS

- `std::find`
- `std::find_if`
- `std::reverse`
- `std::remove_if`
- `std::transform`
- Se necesita incluir el espacio de nombres `std` y el archivo de encabezado `<algorithm>`.

Interacción entre algoritmos y contenedores usando iteradores

Un ejemplo que usa:

- El contenedor secuencial `std::vector` para almacenar números enteros.
- El algoritmo `std::find`
- Los iteradores forman el puente entre ambos

Una página WEB para referencia a la BPE:

- `https://www.sgi.com/tech/stl/`