STL(Standard Template Library)

STL 은 C++ 표준 템플릿 라이브러리이며, 알고리즘을 일반화한 표현을 제공하여 데이터의 추상화와 코드를 재활용할 수 있게 한다.

구성요소

- 반복자
- 컨테이너
- 알고리즘

STL 반복자

반복자(iterator)란 STL 컨테이너에 저장된 요소를 반복적으로 순회하여, 각각의 요소에 대한 접근을 제공하는 객체이다. 즉, 컨테이너의 구조나 요소의 타입과는 상관없이 컨테이너에 저장된 데이터를 순회하는 과정을 일반화한 표현이다. 템플릿이 타입과 상관없이 알고리즘을 표현할 수 있게 해준다면, 반복자는 컨테이너와 상관없이 알고리즘을 표현할 수 있게 해준다.

반복자가 가져야 할 요구 사항과 정의되어야 할 연산자는 아래와 같다.
1. 가리키는 요소의 값에 접근할 수 있어야 한다. 따라서 참조 연산자(*)가 정의되어야 한다.
2. 반복자 사이의 대입 연산, 비교 연산이 가능해야 한다. 따라서 대입, 관계 연산자가 정의되어야 한다.
3. 가리키는 요소의 주변 요소로 이동할 수 있어야 한다. 따라서 증가 연산자(++)가 정의되어야 한다.

반복자의 종류

1. 입력 반복자(input iterator)
2. 출력 반복자(output iterator)
3. 순방향 반복자(forward iterator)
4. 양방향 반복자(bidirectional iterator)

5. 임의 접근 반복자(random access iterator)

STL 컨테이너

컨테이너(container)는 같은 타입의 여러 객체를 저장하는 일종의 집합이라 할 수 있다. 컨테이너는 클래스 템플릿으로, 컨테이너 변수를 선언할 때 컨테이너에 포함할 요소의 타입을 명시할 수 있다. 컨테이너에는 복사 생성과 대입을 할 수 있는 타입의 객체만을 저장할 수 있으며 요소의 추가 및 제거를 포함한 다양한 작업을 도와주는 여러 멤버 함수를 포함하고 있다.

컨테이너의 종류

1. 시퀀스 컨테이너(sequence container)

시퀀스 컨테이너는 데이터를 선형으로 저장하며, 특별한 제약이나 규칙이 없는 가장 일반적인 컨테이너이다.

💁‍♀️ 시퀀스 컨테이너 요구사항

1. 모든 요소가 직선 순서대로 배치되어 있어야 한다.
   첫 번째 요소와 마지막 요소를 제외한 나머지 요소들은 반드시 앞뒤로 인접한 요소를 하나씩 가지고 있어야 한다.
2. 반복자가 최소한 순방향 반복자(forward iterator) 이상이어야 한다.
   이것은 반복자가 이동할 때마다 요소들의 순서가 변하지 않음을 보장해 주는 것이다.
3. 시퀀스 컨테이너의 요소들은 명확한 순서를 가지므로, 특정 위치를 참조하는 연산이 가능해야 한다. 시
   퀀스 컨테이너에서는 삽입된 요소의 순서가 그대로 유지된다.

💁‍♀️ 시퀀스 컨테이너 종류

1. vector

벡터(vector) 컨테이너는 동적 배열의 클래스 템플릿 표현이다. 벡터 객체는 요소가 추가되거나 삭제될 때마다 자동으로 메모리를 재할당하여 크기를 동적으로 변경한다. vector 헤더 파일에 정의되어 있으며, 임의 접근을 제공하는 가장 간단한 시퀀스 컨테이너이다.

2. deque

디큐(deque) 컨테이너는 double-ended queue 를 의미하며, 양쪽에 끝이 있는 큐이다. 이 컨테이너는 컨테이너의 양 끝에서 빠르게 요소를 삽입하거나 삭제할 수 있다. 디큐 컨테이너는 deque 헤더 파일에 정의되어 있다.

3. list

리스트(list) 컨테이너는 이중 연결 리스트(doubly linked list)의 클래스 템플릿 표현이라 할 수 있다. 이 컨테이너는 컨테이너의 모든 요소에서 양방향 접근, 빠른 삽입과 삭제를 할 수 있지만, 임의 접근은 할 수는 없다. 리스트 컨테이너는 list 헤더 파일에 정의되어 있다.

함수	설명
swap()	두 요소의 위치를 서로 바꿈.
reverse()	리스트 전체의 요소의 위치를 역순으로 변경함.
sort()	리스트 전체의 요소를 정렬함.
unique()	같은 값이 인접해 있을 경우, 그 값들을 하나로 단일화함.
merge()	두 정렬된 리스트를 합병함.
splice()	두 리스트를 연결하거나, 한 쪽 리스트로 이동시킴.

4. forward_list

순방향 리스트(forward_list) 컨테이너는 단방향 연결 리스트(singly linked list)의 클래스 템플릿 표현이라 할 수 있다. C++11 부터 추가된 이 컨테이너는 모든 요소에서 순방향으로 접근할 수는 있지만, 역방향으로 접근할 수는 없다. 따라서 순방향 리스트

> 컨테이너에서는 오직 순방향 반복자(forward iterator)만을 사용한다. 리스트 객체와 비교했을 때, 순방향 리스트 객체는 더 적은 메모리를 가지고 간편하게 사용할 수 있는 장점이 있다.

2. 연관 컨테이너(associative container)

연관 컨테이너(associate container)는 키(key)와 값(value)처럼 관련있는 데이터를 하나의 쌍으로 저장하는 컨테이너이다. 키와 값을 이용한 연관 컨테이너는 요소들에 대한 빠른 접근을 제공해준다. 하지만 연관 컨테이너는 삽입되는 요소의 위치를 지정할 수는 없다.

💁‍♀️ 연관 컨테이너 종류

1. set

> 집합(set) 컨테이너는 저장하는 데이터 그 자체를 키로 사용하는 가장 단순한 연관 컨테이너이다. 이 컨테이너는 벡터와 달리 오름차순으로 정렬된 위치에 요소를 삽입하므로 검색 속도가 매우 빠르다.

2. multiset

> 집합(set)에서 키는 유일해야 하므로, 키의 중복을 허용하지 않지만 멀티집합(multiset)은 키의 중복을 허용하므로, 같은 값을 여러 번 저장할 수 있다.

3. map

> 맵(map) 컨테이너는 키와 값의 쌍으로 데이터를 관리하는 진정한 연관 컨테이너이다. 이 컨테이너는 집합 컨테이너와 마찬가지로 정렬된 위치에 요소를 삽입하므로 검색 속도가 매우 빠르다.
> 맵(map)에서 키는 유일해야 하므로, 키의 중복을 허용하지 않는다. 따라서 하나의 키에 하나의 값만이 연결될 수 있다.

4. multimap

> 멀티맵(multimap)은 값의 중복을 허용하므로, 하나의 키가 여러 개의 값과 연결될 수 있다.

3. 컨테이너 어댑터(adapter container)

컨테이너 어댑터(container adapter)는 기존 컨테이너의 인터페이스를 제한하여 만든 기능이 제한되거나 변형된 컨테이너를 의미한다. 이러한 컨테이너 어댑터는 각각의 기초가 되는 클래스의 인터페이스를 제한하여, 특정 형태의 동작만을 수행하도록 한다. 단, 반복자를 지원하지 않으므로 STL 알고리즘에서는 사용할 수 없다.

🤦‍♀️ 컨테이너 어댑터 종류

1. stack
   스택(stack) 컨테이너는 vector 클래스의 인터페이스를 제한하여, 전형적인 스택 메모리 구조의 인터페이스를 제공하며, stack 헤더 파일에 정의되어 있다. 스택 메모리 구조는 선형 메모리 공간에 데이터를 저장하면서 후입선출(LIFO)의 자료 구조이다

멤버 함수	설명
empty()	스택이 비어 있으면 true를, 비어 있지 않으면 false를 반환함.
size()	스택 요소의 총 개수를 반환함.
top()	스택의 제일 상단에 있는(제일 마지막으로 저장된) 요소에 대한 참조를 반환함.
push()	스택의 제일 상단에 요소를 삽입함.
pop()	스택의 제일 상단에 있는 요소를 삭제함.

2. queue
   큐(queue) 컨테이너는 deque 클래스의 인터페이스를 제한하여, 전형적인 큐 메모리 구조의 인터페이스를 제공하며, queue 헤더 파일에 정의되어 있다. 큐 메모리 구조는 선형 메모리 공간에

데이터를 저장하면서 선입선출(FIFO)의 자료 구조입니다.

멤버 함수	설명
empty()	큐가 비어 있으면 true를, 비어 있지 않으면 false를 반환함.
size()	큐 요소의 총 개수를 반환함.
front()	큐의 맨 앞에 있는(제일 먼저 저장된) 요소에 대한 참조를 반환함.
back()	큐의 맨 뒤에 있는(제일 나중에 저장된) 요소에 대한 참조를 반환함.
push()	큐의 맨 뒤에 요소를 삽입함.
pop()	큐의 맨 앞의 요소를 삭제함.

3. priority_queue
   우선순위 큐(priority_queue) 컨테이너는 큐와는 달리 큐의 맨 앞의 요소로 가장 먼저 저장된 요소가 아닌, 가장 큰 값을 지닌 요소가 위치하게 된다. 또한, 큐가 deque 클래스를 기반으로 하는 것과는 달리, 우선순위 큐는 vector 클래스를 기반으로 한다.

STL 알고리즘

STL 의 목적은 일반적인 알고리즘에 대한 효율적인 구현을 제공하는 것이다. 따라서 STL 은 이러한 알고리즘을 STL 알고리즘 함수나 STL 컨테이너의 멤버 함수를 사용하여 구현하고 있다. STL 컨테이너는 반드시 필요한 기능만을 포함하고 있으며, 동작하는데 필요한 모든 기능을 가지고 있지는 않다. 따라서 STL 컨테이너는 알고리즘을 제공하는 많은 전역 함수와 함께 사용해야만 제 기능을 발휘할 수 있다. 이렇게 제공되는 STL 알고리즘 함수는 반복자를 통해 임의의 컨테이너에 같은 방법으로 적용된다. 대부분의 알고리즘 함수는 algorithm 헤더 파일과 numeric 헤더 파일에 정의되어 있습니다.

알고리즘 분류

1. 읽기 알고리즘(algorithm 헤더 파일)

STL 읽기 알고리즘 함수는 컨테이너를 변경하지 않으며, 컨테이너의 지정된 범위에서 특정 데이터를 읽기만 하는 함수이다.

1. find()

   find() 함수는 두 개의 입력 반복자로 지정된 범위에서 특정 값을 가지는 첫 번째 요소를 가리키는 입력 반복자를 반환한다.

2. for_each()

   for_each() 함수는 두 개의 입력 반복자로 지정된 범위의 모든 요소를 함수 객체에 대입한 후, 대입한 함수 객체를 반환한다.

2. 변경 알고리즘(algorithm 헤더 파일)

STL 변경 알고리즘 함수는 컨테이너를 변경하지 않으며, 컨테이너의 지정된 범위에서 요소의 값만을 변경할 수 있는 함수이다.

1. copy()

   두 개의 입력 반복자로 지정된 범위의 모든 요소를 출력 반복자가 가리키는 위치에 복사한다.

2. swap()

   두 개의 참조가 가리키는 위치의 값을 서로 교환한다.

3. transform()

   두 개의 입력 반복자로 지정된 범위의 모든 요소를 함수 객체에 대입한 후, 출력 반복자가 가리키는 위치에 복사한다.

3. 정렬 알고리즘(algorithm 헤더 파일)

STL 정렬 알고리즘 함수는 컨테이너의 지정된 범위의 요소들이 정렬되도록 컨테이너를 변경하는 함수이다. 모든 정렬 알고리즘 함수는 올바른 정렬을 위해 임의 접근 반복자를 사용합니다.

1. sort()

두 개의 임의 접근 반복자로 지정된 범위의 모든 요소를 서로 비교하여, 오름차순으로 정렬한다.

2. stable_sort()

두 개의 임의 접근 반복자로 지정된 범위의 모든 요소를 서로 비교하여, 값이 서로 같은 요소들의 상대적인 순서는 유지하면서 오름차순으로 정렬한다.

3. binary_search()

ort() 함수를 사용하여 오름차순으로 정렬한 후에, 전달된 값과 같은 값이 있으면 참(true)을 반환하고 없으면 거짓(false)을 반환한다.

4. 수치 알고리즘(numeric 헤더 파일)

수치 알고리즘 함수는 다른 알고리즘 함수와는 달리 numeric 헤더 파일에 정의되어 있다.

1. accumulate()

두 개의 입력 반복자로 지정된 범위의 모든 요소의 합을 반환한다.