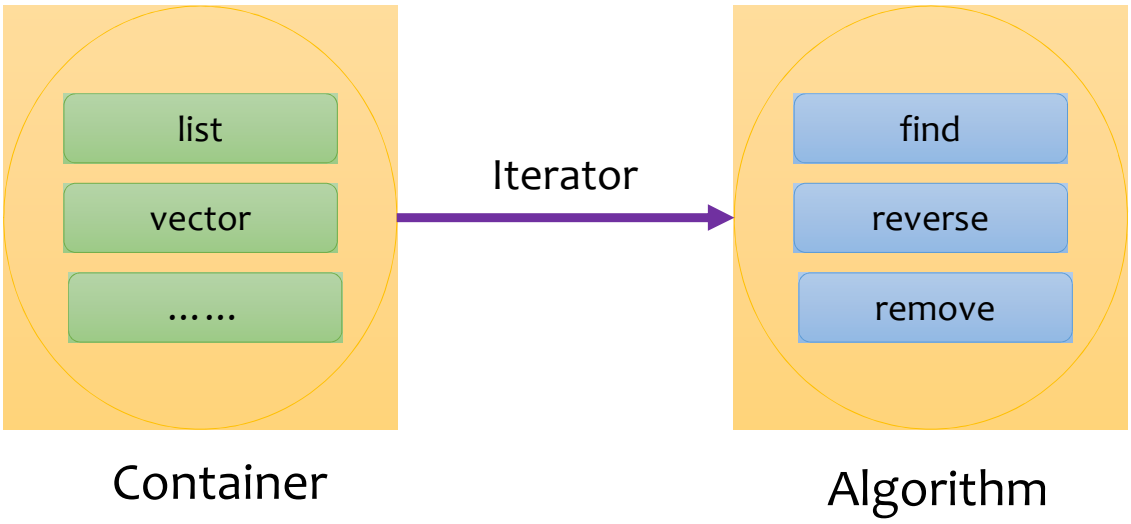
핵심 개념

- STL 구조



1. 템플릿을 사용하지 않고 find, reverse, remove 를 멤버 함수로 만든다면
   - 3개 컨테이너 * 3개 함수 * 3개 타입 = 27 개 함수가 필요

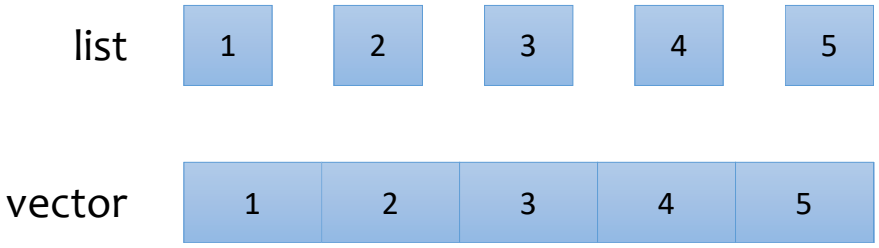
2. 템플릿을 사용해서 find, reverse, remove 를 멤버 함수로 만든다면
   - 3개 컨테이너 * 3개 함수 = 9 개 함수가 필요

3. find, reverse, remove 를 멤버 함수가 아닌 일반 함수로 만든다면
   - 모든 컨테이너에 동작하는 3개의 함수가 필요
   - 모든 컨테이너의 요소를 동일한 방식으로 접근할 수 있어야 한다. – 반복자가 필요하다.

4. STL은 자료구조와 알고리즘이 분리된 라이브러리
   - 알고리즘 함수는 자신이 어떤 자료구조에 대해 연산을 수행하는지 모른다.

- STL example

핵심 개념



1. STL 컨테이너의 모든 요소를 열거하는 방법
   - [] 연산자 사용 – vector등 일부 컨테이너만 가능. list 안됨.
   - 반복자 사용 – 대부분 컨테이너에서 사용가능.
   - range for 문 사용
   - 스트림 반복자 – SECTION 3 참고.

- STL 과 member type

핵심 개념

1. STL 에는 멤버 데이터, 멤버 함수 이외에도 "멤버 타입" 이라는 개념 제공.

2. 템플릿 의존 타입(template dependent type) 의 경우 typename을 표기 해야 한다.
   - typename T::value_type

3. 템플릿이 아닌 경우는 typename을 표기 하지 않는다.
   - list<int>::value_type

4. value_type
   - 컨테이너가 저장하는 타입
   - 반복자가 가리키는 타입

- STL 과 member type

핵심 개념

- STL 컨테이너가 제공하는 “member type”

Member type	Definition
value_type	T
allocator_type	Allocator
size_type	Unsigned integer type (usually std::size_t)
difference_type	Signed integer type (usually std::ptrdiff_t)
reference	value_type&
const_reference	const value_type&
pointer	std::allocator_traits<Allocator>::pointer
const_pointer	std::allocator_traits<Allocator>::const_pointer
iterator	컨테이너 마다 다른 정의
const_iterator	컨테이너 마다 다른 정의
reverse_iterator	std::reverse_iterator<iterator>
const_reverse_iterator	std::reverse_iterator<const_iterator>

- STL 반복자가 제공하는 “member type”

Member type	Definition
value_type	T
difference_type	Signed integer type (usually std::ptrdiff_t)
reference	value_type&
pointer	value_type*
iterator_category	컨테이너 마다 다른 정의

- C++17 과 STL

핵심 개념

1. class template type deduction
   - C++17 에서 도입된 문법
   - 클래스 템플릿도 템플릿 인자를 명시적으로 지정하지 않아도 생성자의 인자 또는 사용자가 제공한 “deduction guide” 를 통해서 컴파일러가 추론(deduction) 할 수 있다.

- Raw String Literal

핵심 정리

1. Raw String Literal
   - R"(raw_characters)"
   - \를 특수 문자(escape sequence)가 아닌 일반 문자로 인식.
   - 파일 경로, 정규 표현식 등의 표현할 때 편리하다.

2.)" 이 표현을 사용하고 싶다면?
   - 사용자 정의 delimiter 추가.
   - R"delimiter(raw_characters)delimiter"
   - R"***(raw_characters)***"

- 문자열 type & prefix

핵심 정리

1. 문자열 prefix 와 type

String literal	type	encoding
“hello”	const char[]	narrow multi-byte string literal
L”hello”	const wchar_t[]	wide string literal
u8”hello”	const char[]	UTF-8
u”hello”	const char16_t[]	UTF-16
U”hello”	const char32_t[]	UTF-32

1. CharT
   - 문자열 관련 타입(char, wchar_t, char16_t, char32_t) 를 총괄해서 표현하는 관례적인 표기법.

- char_traits

핵심 정리

1. char_traits<>
   - <string> 헤더
   - 다양한 문자 타입에 대한 문자열(문자) 연산 함수를 제공하는 클래스
   - char, wchar_t, char16_t, char32_t 타입에 대해서 부분 특수화 되어 있다.
   - 모든 멤버 함수가 static 으로 되어 있다.

Member function	description
assign	assigns a character
eqlt	compares two characters
move	moves one character sequence onto another
copy	copies a character sequence
compare	lexicographically compares two character sequences
length	returns the length of a character sequence
find	finds a character in a character sequence
to_char_type	converts int_type to equivalent char_type
to_int_type	converts char_type to equivalent int_type
eq_int_type	compares two int_type values
eof	returns an eof value
not_eof	checks whether a character is eof value

- user define character traits 를 제공할 수도 있다.

- 문자열 관련 클래스 in STL

핵심 정리

1. string 클래스

type	definition
string	basic_string<char>
wstring	basic_string<wchar_t>
u16string	basic_string<char16_t>
u32string	basic_string<char32_t>

2. stream class

type	definition
ostream	basic_ostream<char>
istream	basic_istream<char>
wostream	basic_ostream<wchar_t>
wistream	basic_istream<wchar_t>

C++ STL 에 포함된 문자열 관련 대부분 클래스는 CharT 를 인자로 가지는 템플릿 형태로 제공.