

Lab 8

<https://padlet.com/dcslabcp/0503-roji14mirapdhlco>

온라인 질문 사이트

C++ Review

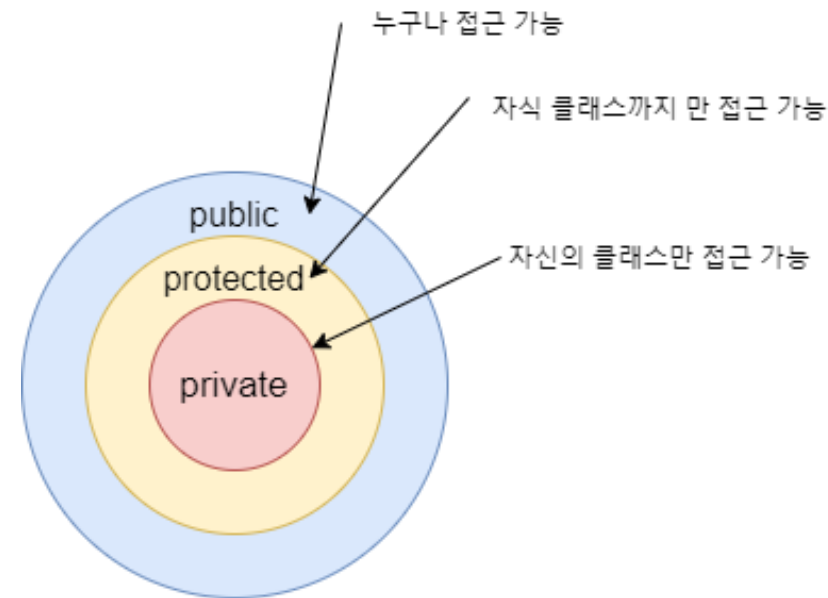
실습 시간에 배운 내용들 복습

1. C++ 개념 정리

- Class의 정의
 - 변수들과 연관된 함수들을 결합시킨 새로운 틀
 - 구조체를 선언하는 것도 새로운 틀을 생성하는 것이지만 **함수**까지 결합한다는 것이 구조체와 클래스의 차이
 - 클래스 선언 시 메모리 할당이 되는 것이 아니라 객체를 생성할 때 메모리 할당
 - 클래스 선언 Declaration of Class
 - class 클래스명
 - {
 - Member variable;
 - Member fuction;
 - };

1. C++ 개념 정리

- 접근제어 지시자
(**public**, **protected**, **private**)
- 생성자와 소멸자
객체가 생성되고 소멸될 때 호출
- 클래스의 상속
부모 클래스와 자식 클래스



1. C++ 개념 정리

- 접근제어 지시자
(public, protected, private)
- 생성자와 소멸자
객체가 생성되고 소멸될 때 호출
- 클래스의 상속
부모 클래스와 자식 클래스

```
class Fishbread{
public:
    Fishbread(){
        cout<<"creat: "<<content<<endl;
    }
    Fishbread(int argCost, string argContent){
        cost = argCost;
        content = argContent;
        cout<<"creat: "<<content<<endl;
    }
    ~Fishbread(){
        cout<<"we finish eating the <"+content+"> fishbread"<<endl;
    }
}
```

1. C++ 개념 정리

- 접근제어 지시자
(public, protected, private)
- 생성자와 소멸자
객체가 생성되고 소멸될 때 호출
- 클래스의 상속
부모 클래스와 자식 클래스

```
class Car {  
protected:  
    string name;  
    int speed = 0, totaldistance = 0, hour = 0, totalhour = 0;  
public:  
    Car(string _name){  
        name = _name;  
    }  
}
```

```
class Avante : public Car{  
private:  
    int remainingdistance = 500;  
public:  
    Avante() : Car("Avante"){  
        // ...  
    }  
}
```

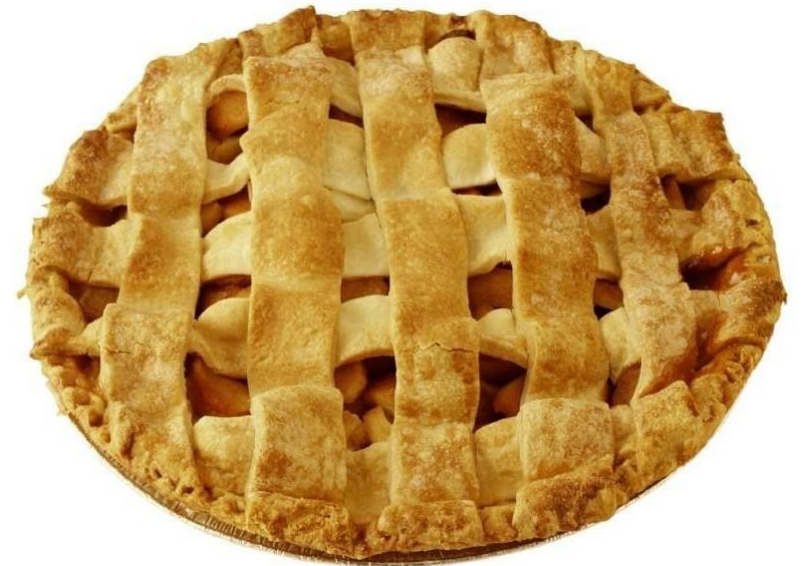
2. C++ OOP

Abstraction (추상화)

Polymorphism (다형성)

Inheritance (상속)

Encapsulation (캡슐화)



3. C++ STL

- 시퀀스 컨테이너(Sequence Container)
 - 자료를 입력하는 순서대로 저장한다.
 - 종류 : array, vector, list, deque
-
- 정렬 연관 컨테이너(Associative Container)
 - 트리 구조로 이루어져 있다.
 - 종류 : set, multiset, map, multimap

3. C++ template

- Template : 함수, 클래스, 변수가 제네릭 타입으로 동작하게 하는 것
- 함수 템플릿
컴파일 시 함수가 사용할 자료형을 결정
- 클래스 템플릿
컴파일 시 클래스가 사용할 자료형을 결정
- 자료형이 명시되어 있지 않으면 컴파일러가 추측

```
1  #include <iostream>
2  using namespace std;
3
4  template <typename T>
5
6  T AddNumbers(T num1, T num2){
7      return num1+num2;
8  }
9
10 int main(){
11     cout << AddNumbers<int>(17, 19) << endl;
12     cout << AddNumbers<double>(2.9, 3.8) << endl;
13     cout << AddNumbers(17, 19) << endl;
14     cout << AddNumbers(2.9, 3.8) << endl;
15     cout << AddNumbers<string>("ab", "cd") << endl;
16     return 0;
17 }
```

cout << AddNumbers<int>(17, 19) << endl;	36
cout << AddNumbers<double>(2.9, 3.8) << endl;	6.7
cout << AddNumbers(17, 19) << endl;	36
cout << AddNumbers(2.9, 3.8) << endl;	6.7
cout << AddNumbers<string>("ab", "cd") << endl;	abcd

4. 참고 사이트 - cppreference.com

- C++ 레퍼런스 사이트
- C++에서 제공하는 라이브러리에 대한 레퍼런스 정보 제공
- C++ 버전 별 변경사항 제공

C++ reference

C++98, C++03, C++11, C++14, C++17, C++20, C++23 | Compiler support C++11, C++14, C++17, C++20, C++23

Freestanding implementations
ASCII chart

Language

Basic concepts
Keywords
Preprocessor
Expressions
Declaration
Initialization
Functions
Statements
Classes
Overloading
Templates
Exceptions

Standard library (headers)

Named requirements

Feature test macros (C++20)

Language support library

source_location (C++20)
Type support
Program utilities
Coroutine support (C++20)
Three-way comparison (C++20)
numeric_limits - type_info
initializer_list (C++11)

Concepts library (C++20)

Diagnostics library

exception - System error
basic_stacktrace (C++23)

Memory management library

unique_ptr (C++11)
shared_ptr (C++11)

Metaprogramming library (C++11)

Type traits - ratio
integer_sequence (C++14)

General utilities library

Function objects - hash (C++11)
Utility functions
pair - tuple (C++11)
optional (C++17)
expected (C++23)
variant (C++17) - any (C++17)
String conversions (C++17)
Formatting (C++20)
bitset - Bit manipulation (C++20)

Strings library

basic_string - char_traits
basic_string_view (C++17)
Null-terminated strings:
byte - multibyte - wide

Containers library

array (C++11) - vector - deque
list - forward_list (C++11)
set - multiset
map - multimap
unordered_map (C++11)
unordered_multimap (C++11)
unordered_set (C++11)
unordered_multiset (C++11)
stack - queue - priority_queue
flat_set (C++23)
flat_multiset (C++23)
flat_map (C++23)
flat_multimap (C++23)
span (C++20) - mdspan (C++23)

Iterators library

Ranges library (C++20)

Algorithms library

Execution policies (C++17)
Constrained algorithms (C++20)

Numerics library

Common math functions
Mathematical special functions (C++17)
Mathematical constants (C++20)
Numeric algorithms
Pseudo-random number generation
Floating-point environment (C++11)
complex - valarray

Date and time library

Calendar (C++20) - Time zone (C++20)

Localizations library

locale - Character classification

Input/output library

Print functions (C++23)
Stream-based I/O - I/O manipulators
basic_istream - basic_ostream
Synchronized output (C++20)

Filesystem library (C++17)

path

Regular expressions library (C++11)

basic_regex - algorithms

Concurrency support library (C++11)

thread - jthread (C++20)
atomic - atomic_flag
atomic_ref (C++20)
memory_order - condition_variable
Mutual exclusion - Semaphores (C++20)
future - promise - async
latch (C++20) - barrier (C++20)

Technical specifications

Standard library extensions (library fundamentals TS)

Computer Programming

4. 참고 사이트 - cppreference.com

- `std::list::remove` 함수 예시
- Parameter의 value와 동일한 값의 원소들 제거
- c++17 버전까지 return 없음
- c++20 버전부터 삭제된 원소 개수 return

cppreference.com [Create account](#)

Page **Discussion**

C++ Containers library `std::list`

`std::list<T,Allocator>::remove, remove_if`

```
void remove( const T& value );           (1) (until C++20)
size_type remove( const T& value );      (1) (since C++20)

template< class UnaryPredicate >         (until C++20)
void remove_if( UnaryPredicate p );      (2)
template< class UnaryPredicate >         (since C++20)
size_type remove_if( UnaryPredicate p );
```

Removes all elements satisfying specific criteria.

- 1) Removes all elements that are equal to value.
- 2) Removes all elements for which predicate p returns `true`.

Parameters

value - value of the elements to remove

p - unary predicate which returns `true` if the element should be removed.

The expression `p(v)` must be convertible to `bool` for every argument v of type (possibly const) T, regardless of **value category**, and must not modify v. Thus, a parameter type of `T&` is not allowed, nor is `T` unless for T a move is equivalent to a copy (since C++11).

Return value

(none)	(until C++20)
The number of elements removed.	(since C++20)

Complexity

Linear in the size of the container

4. 참고 사이트 - cppreference.com

Example

- 짧은 example 제공

Run this code

```
#include <list>
#include <iostream>

int main()
{
    std::list<int> l = { 1,100,2,3,10,1,11,-1,12 };

    auto count1 = l.remove(1);
    std::cout << count1 << " elements equal to 1 were removed\n";

    auto count2 = l.remove_if([](int n){ return n > 10; });
    std::cout << count2 << " elements greater than 10 were removed\n";

    std::cout << "Finally, the list contains: ";
    for (int n : l) {
        std::cout << n << ' ';
    }
    std::cout << '\n';
}
```

Output:

```
2 elements equal to 1 were removed
3 elements greater than 10 were removed
Finally, the list contains: 2 3 10 -1
```

4. 참고 사이트 - cpp.sh

- c++ 온라인 컴파일러 사이트
- 모바일이나 태블릿에서 코드 작성해서 테스트해보거나
- 옵션으로 c++ 버전 선택하여 컴파일 및 테스트 가능

C++ shellcpp.sh
online C++ compiler
about cpp.sh

```
1 // Example program
2 #include <iostream>
3 #include <string>
4
5 int main()
6 {
7     std::string name;
8     std::cout << "What is your name? ";
9     getline (std::cin, name);
10    std::cout << "Hello, " << name << "!\n";
11 }
```

Link to this code: [\[copy\]](#) Run

options

compilation

execution

Standard

- ☐ C++98
- ☐ C++11
- ☐ C++14
- ☐ C++17
- ☒ C++20
- ☐ C++23 (experimental)

Warnings

- ☒ Many (-Wall)
- ☐ Extra (-Wextra)
- ☐ Pedantic (-Wpedantic)

Optimization level

- ☒ None (-O0)
- ☐ Moderate (-O1)

4. 참고 사이트 - cpp.sh

- c++ 20 버전에서 실행되는 코드
- c++ 17 버전에서 컴파일해보면 에러 발생

options	compilation	execution
Standard <ul style="list-style-type: none"><input type="radio"/> C++98<input type="radio"/> C++11<input type="radio"/> C++14<input checked="" type="radio"/> C++17<input type="radio"/> C++20<input type="radio"/> C++23 (experimental)	Warnings <ul style="list-style-type: none"><input checked="" type="checkbox"/> Many (-Wall)<input type="checkbox"/> Extra (-Wextra)<input type="checkbox"/> Pedantic (-Wpedantic)	Optimization level <ul style="list-style-type: none"><input checked="" type="radio"/> None (-O0)<input type="radio"/> Moderate (-O1)<input type="radio"/> Full (-O2)<input type="radio"/> Maximum (-O3)<input type="radio"/> Maximum & smallest (-Oz)

C++ shell

```
1 #include <list>
2 #include <iostream>
3
4 int main()
5 {
6     std::list<int> l = { 1,100,2,3,10,1,11,-1,12 };
7
8     auto count1 = l.remove(1);
9     std::cout << count1 << " elements equal to 1 were removed\n";
10
11     auto count2 = l.remove_if([](int n){ return n > 10; });
12     std::cout << count2 << " elements greater than 10 were removed\n";
13
14     std::cout << "Finally, the list contains: ";
15     for (int n : l) {
16         std::cout << n << ' ';
17     }
18     std::cout << '\n';
19 }
```

Link to this code: [\[copy\]](#)

options	compilation	execution
main.cpp:8:10: error: variable has incomplete type '__remove_return_type' (aka 'void') auto count1 = l.remove(1); ^		
main.cpp:11:10: error: variable has incomplete type '__remove_return_type' (aka 'void') auto count2 = l.remove_if([](int n){ return n > 10; }); ^		
2 errors generated.		