|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **작성자** | **2014132002 구태균** | **팀명** | - |
| **주차** | **9, 10, 11**  **주차** | **기간** | **2020.03.09 ~ 2020.03.15**  **2020.03.16 ~ 2020.03.22**  **2020.03.23 ~ 2020.03.29** | **지도교수** | **정 내 훈** (서명) |
| **이번주 한일** | * shared\_ptr<> weak\_ptr<> 구현 | | | | |

**template<typename Tp>**

**shared\_ptr<Tp>**

1. **생성자**

* shared\_ptr()
* shared\_ptr(nullptr\_t)
* shared\_ptr(Tp\* other)
* shared\_ptr(const shared\_ptr& other)
* shared\_ptr(const weak\_ptr<Tp>& other)

1. **= operator**

* shared\_ptr& operator=(const shared\_ptr& other)
* shared\_ptr& operator=(const weak\_ptr<Tp>& other)

1. **소멸자**

* ~shared\_ptr()

1. **atomic reference counting**

* class ctr\_block
* void multi\_test()

1. **get**

* Tp\* get()

1. **\* operator**

* Tp& operator\*()

1. **-> operator**

* Tp\* operator->()

1. **== operator & !=operator**

* template<typename \_Tp1, typename \_Tp2>

bool operator==(const shared\_ptr<\_Tp1>& \_\_a, const shared\_ptr<\_Tp2>& \_\_b)

* template<typename \_Tp>

bool operator==(const shared\_ptr<\_Tp>& \_\_a, nullptr\_t)

* template<typename \_Tp>

bool operator==(nullptr\_t, const shared\_ptr<\_Tp>& \_\_a)

* template<typename \_Tp1, typename \_Tp2>

bool operator!=(const shared\_ptr<\_Tp1>& \_\_a, const shared\_ptr<\_Tp2>& \_\_b)

* template<typename \_Tp>

bool operator!=(const shared\_ptr<\_Tp>& \_\_a, nullptr\_t)

* template<typename \_Tp>

bool operator!=(nullptr\_t, const shared\_ptr<\_Tp>& \_\_a)

1. **bool operator**

* bool operator bool()

1. **use\_count**

* int use\_count()

1. **reset**

* void reset()

1. **make\_shared**

* template<typename Tp, typename... Args>

shared\_ptr<Tp> make\_shared(Args&&... \_Args)

1. **atomic\_load (atomic 미완성)**

* template<typename Tp>

Tp atomic\_load(const shared\_ptr<Tp>\* sptr)

* template<typename Tp>

Tp atomic\_load(const weak\_ptr<Tp>\* wptr)

1. **atomic\_store (atomic 미완성)**

* template<typename Tp>

Tp atomic\_store(const shared\_ptr<Tp>\* sptr, Tp ptr)

* template<typename Tp>

Tp atomic\_store(const weak\_ptr<Tp>\* wptr, Tp ptr)

1. **enable\_shared\_from\_this (미완성)**

* template<typename Tp>

class enable\_shared\_from\_this

**template<typename Tp>**

**weak\_ptr<tp>**

1. **생성자**

* weak\_ptr()
* weak\_ptr(nullptr\_t)
* weak\_ptr(const shared\_ptr<Tp>& other)
* weak\_ptr(const weak\_ptr<Tp>& other)

1. **= operator**

* weak\_ptr& operator=(const weak\_ptr& other)
* weak\_ptr& operator=(const shared\_ptr<Tp>& other)

1. **소멸자**

* ~weak\_ptr()

1. **atomic reference counting**

* class ctr\_block
* void multi\_test()

1. **lock**

* shared\_ptr<Tp> lock() const

|  |  |  |  |
| --- | --- | --- | --- |
| **문제점 정리** | Shared\_ptr<int>에 shared\_ptr<float>을 넣으면??   * Std::shared\_ptr에서는 불가   공부한 내용과 다르게 동작하는 코드   * 원래 코드에서 strong\_count = 1, weak\_count = 1로 시작, * 하지만 이해하지 못했음 * (결론) 이해한 내용으로 작업   Weak\_ptr lock()동작 사용x  Atomic\_load, atomic\_stroe   * atomic 해결x   Enable\_shared\_from\_this   * 동작 해결x | **해결 방안** |  |
| **다음 주차** | **11 주차** | **다음 기간** | **2020.03.23~2020.03.29** |
| **다음주 할 일** |  | | |
| **지도교수**  **Comment** |  | | |