Тема 13

Шаблони. Необходими функции в шаблонен клас/шаблонна функция. Темплейтни специализации. Примери за шаблонни класове / функции от стандартната библиотека. Умни указатели. Употреба и идея на shared-ptr, weak_ptr, unique-ptr

Пример за стек (с шаблонен капацитет) и опашка (с функция resize)

Шаблони - функции / класове / структури с общо предназначение спрямо типа (параметричен полиморфизъм)

```
template < typename T >
           /class
```

- шаблоните се заместват по време на компилация

- проблем с разделната компилация - .h не винаги .cpp и не може да бъдат генерирани файлове за конкретния тип, указан от друг фай

⇒ решение - указваме в края на .cpp типовете, с които ще работим

или

- пишем целия шаблон в .hpp файл

- темплейтна специализация - ф-я/клас, който се дърпи различно за конкретен тип

(vector<bool> - битсет)

- Необходими ф-ии на тип
  - < - най-важен оператор за всяка колекция
  - ако някъде се вика оп =, то Т трябва да има разписан оп =

- Примери от стандартната библиотека
  - → std::vector
  - → std::stack
  - → std::queue
  - → std::sort
  - → std::find

и # умни указатели

Умни указатели - обвиващ клас на указател, който се грижи за менажира-нето на паметта

I. unique_ptr - 1 обект, 1 указател
- изтрити коп к-р и оп =
- make_unique<A>(3,7,9,'A') - спестява употребата на new

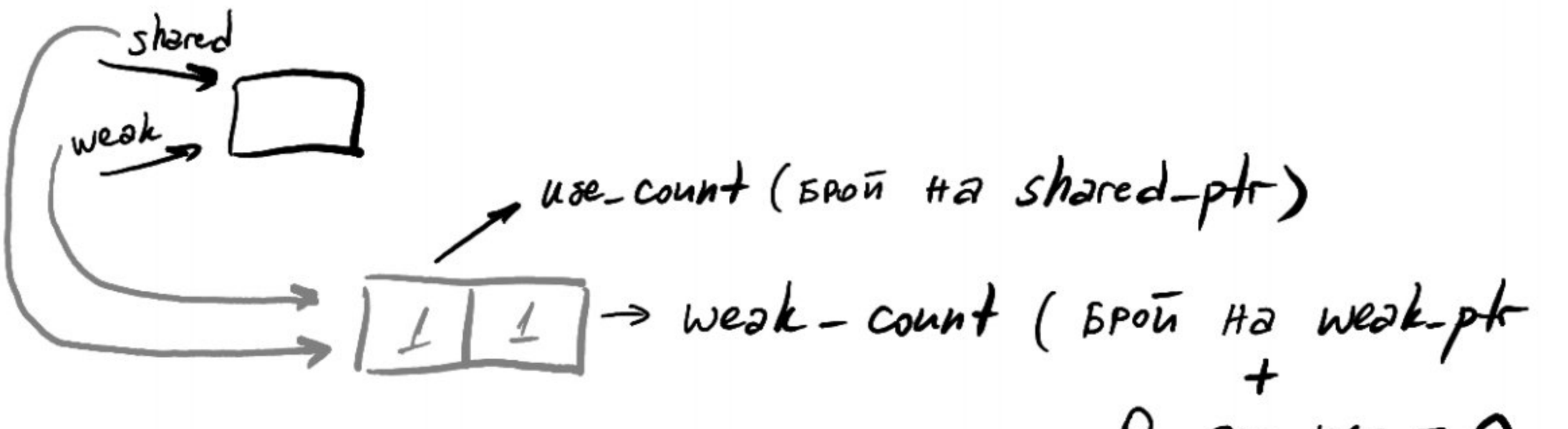
(UniquePtr(new A))

II. shared_ptr - 1 обект, много указатели
- При 1вия указател се заделя обекта

- При изтриването на последния указател към обекта се изтрива и самият обект
- има указател към брояч, който следи броя на shared-ptr към един обект

III. weak_ptr - not-owning ref.
- Насочване указател към обект, който е мениджран от shared_ptr, не влияе на трието и създаването

- възможно е промотиране от weak в shared
- ако обектът бъде междувременно изтрит, има достъп до брояча на shared_ptr



→ обектът се трие, когато use_count = 0
→ броячът се трие, когато weak_count = 0