

13. Шаблони. Необходими ф-ции в шаблонен клас /  
шаблонна ф-ција. Темплейтка спецификации.  
Пример за шаблонни класове от стандард. Гидри.  
Уникатни указатели. Употреба и идеја на `shaped-ptr`,  
`weak-ptr` и `unique-ptr`.

Шаблон  
- параметричен  
- ~~атрибуутен~~ полиморфизъм (статичен)

- клас/<sup>функција</sup> с един предназначение ~~средство~~  
тuna

Тип:

```
template <typename T>
class Vector { ... };
```

- ключова дума `template`

Необходими ф-ции в шаблонен клас / ф-ции

Имете следната доктрина

Тип: `template <typename T>`

```
const const T& max(const T& l, const T& r) {
    return r > l ? r : l;
```

3

Тук всички класове за конто се извиква  
макс () трябва да има предофициран  
оператор  $\geq$ , ако иначе компилът.

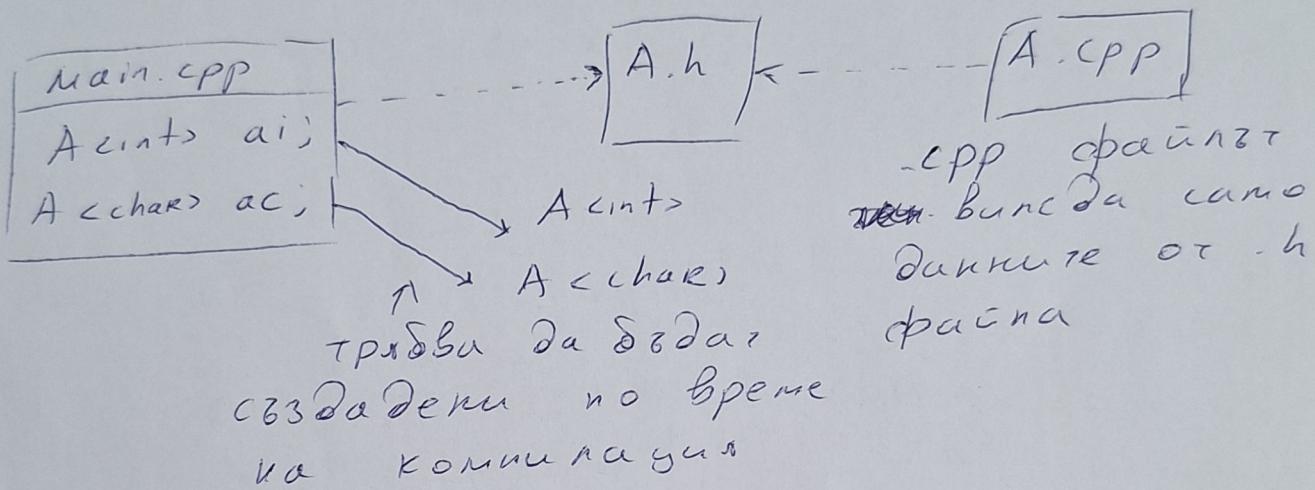
⚠ Ceed C++ има методология за описване  
на задължителните оператори

Компилатор ~~нужда~~ заменя тегови - а  
с конкретни идентификатори

нужда

Нп:  
max<int>(1,2);  
генерира се код за max и  
re заменя +

## Δ Трудност с разделяната комуникация



### 2 Решения:

#### I) Разделяне на .h и .cpp файл

Като на последния ред на .cpp се казва, кой са тяховете.

+  
разделена комуникация (не трябва + да се комуникира много между)

-  
ограничен брой предварителни тяхове

#### II) Темплементен клас

+  
не ограничен брой различни тяхове

.hpp

-  
ниски разделини комуникации

Темпнелга сиңуфикация  
-клас / ф-ции с конкретно пред назначение  
справо тұна

Рп: std::vector<<sup>bool</sup>~~char~~> vec;

Не используется масив a Bitset за  
да cream намет.

Рп: template <class T>  
sort( T<sup>n</sup> arr, size\_t )  
=> using Quick Sort

template << темпнелга сиңуфикация  
sort( char<sup>\*</sup> arr, size\_t )  
=> using Counting Sort

Умни ықазданы

Создаден ол no стандарт за бүздеңе на  
ақырақтыра и изболжанено на нөхшед-  
жето на new и delete.

I auto\_ptr

// legacy  
использование конструктор за init  
и при исполнении не  
делено кипи за деинициализатор

Рп:

auto\_ptr<A> ptr(new A);

## II Unique-ptr

Нарече наим е обикновен  
онреденен pointer.

Известно е make-unique  $\Rightarrow$  (.) за  
да се избере new-то.

Задължено е да CC и DTC.

Фп: class A

A(int, bool)

$\sim A$

~~Да се избере~~

main

{ unique - ptr<A> up =

make - unique <ptr<A>(2, true);

A()  $\beta \in \sim A()$

~~Unique-ptr~~

Edge case

{

$A^* \alpha = \text{new } A();$

unique - ptr<A> up1 = make - unique(a);

unique - ptr<A> up2 = make - unique<A>(a);

$\beta \leq$  не избрани защото CC  
онредна да избере правилни данни.

## III shared\_ptr

Членувачите на членовете на класа се искажат  
многократно и възникват проблеми.

Те трябва да бъдат защищени от 1 и  
чрез копиране да се създават дупки.

Но следният проблем да е изпълнен

Ип: shared\_ptr < A> sp = make\_shared<A>(2, true);

s-p <A> sp2 = sp;

s-p <A> sp3 = sp;

sp2.release();

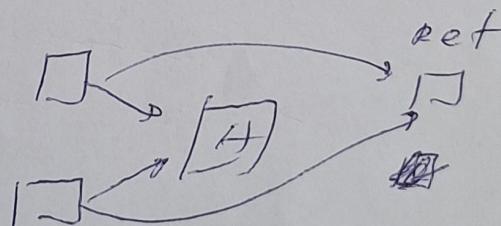
sp - release();

sp3 → f(); // неправилно

Решаването

int\* ref;

Но следните изпълнения  
тази членовете.



## IV. weak\_ptr

Установка се засяда с shared\_ptr.

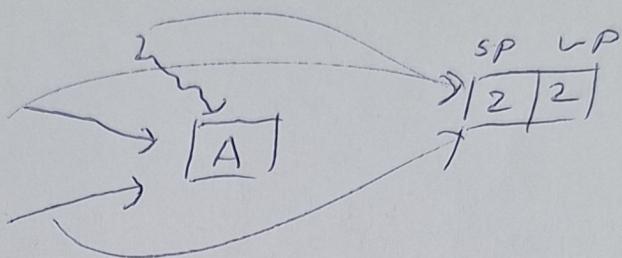
този указател не отговаря за деструктора  
и dealокирането на данни  
а също наблюдава конкуренция в  
наметка.

Безмодело е съсността на weak\_ptr  
Да е вебамдка.

Реализация:

Counter - counter  
масив съдържащ 2 инт група на кояде  
6 номера

mo - wp  
→ - sp



Този shared\_ptr е отговорен за  
изпълнение на A и Counter.

weak\_ptr отговоря за изпълнение  
да съдържа Counter.

Моне да има 3 weak\_ptr към  
не съществуващи обекти и те няма как  
да знаят, ако няма Counter.

Ние Ø wp => valid Counter

Леде промоупане

- lock() - от weak\_ptr не съдържа  
SharedPtr, ако - data е бавнодан  
обект.