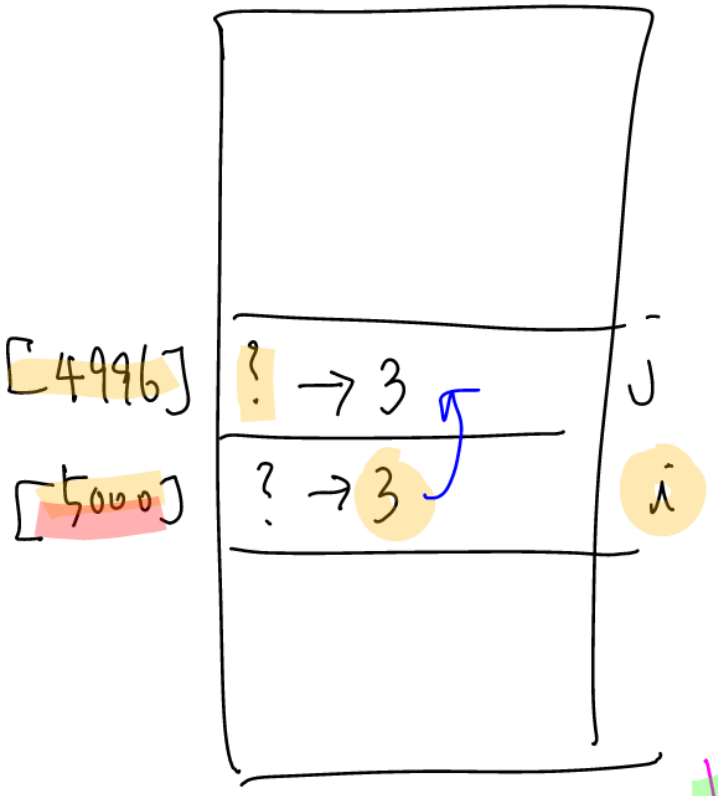
[2011.4.15, Sat] 「C++ OCW」

@ Perfect Forwarding in C++.

l-value
r-value
reference.
copy constructor
move constructor
template specialization
std::move
perfect forwarding



i = 3 ; // [5000] ← 3

machine code

j = i ; // [4996] ← *[5000] .

[5000] = 3

[4996] ← *[5000]

l-value
i
left

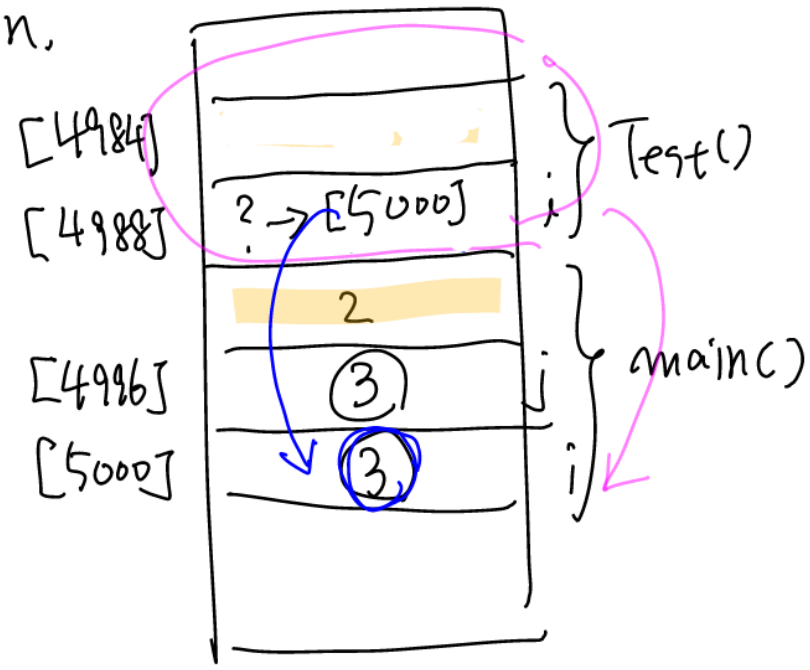
r-value.
right

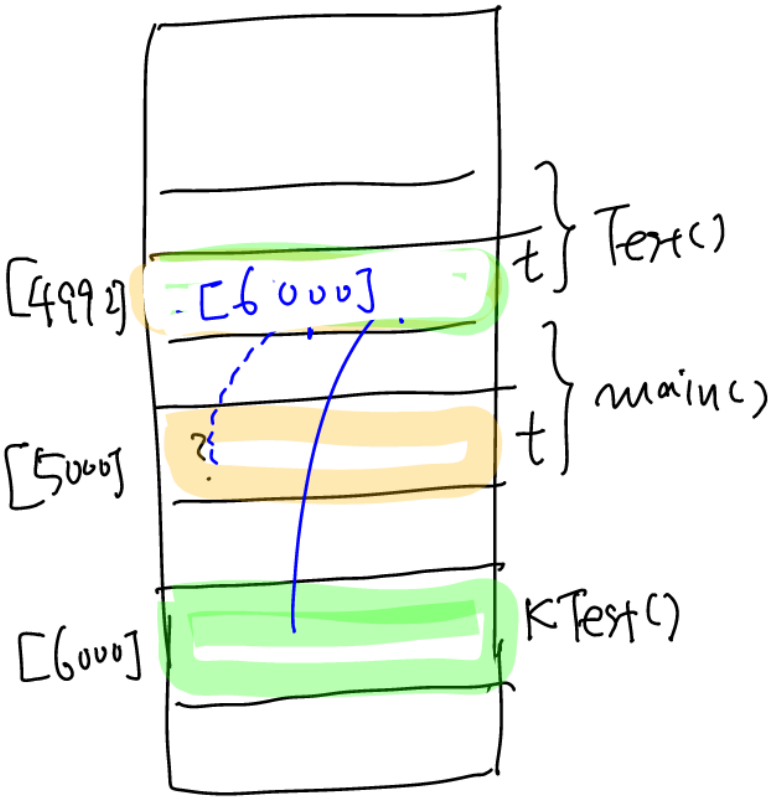
i=3;

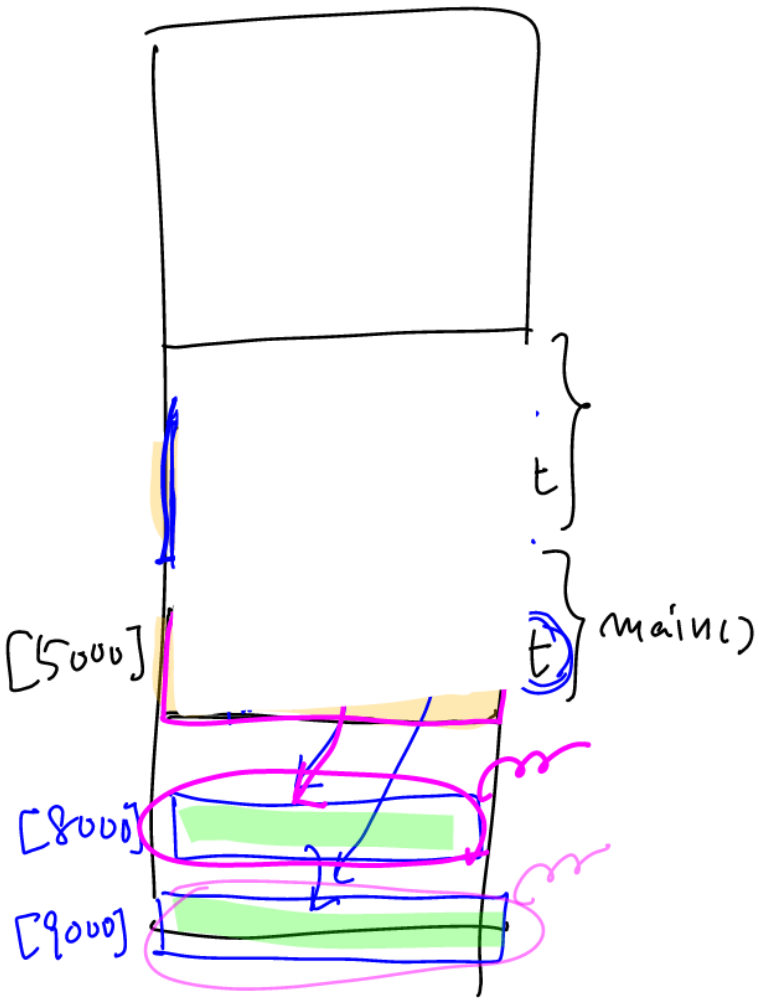
i+1; expression.

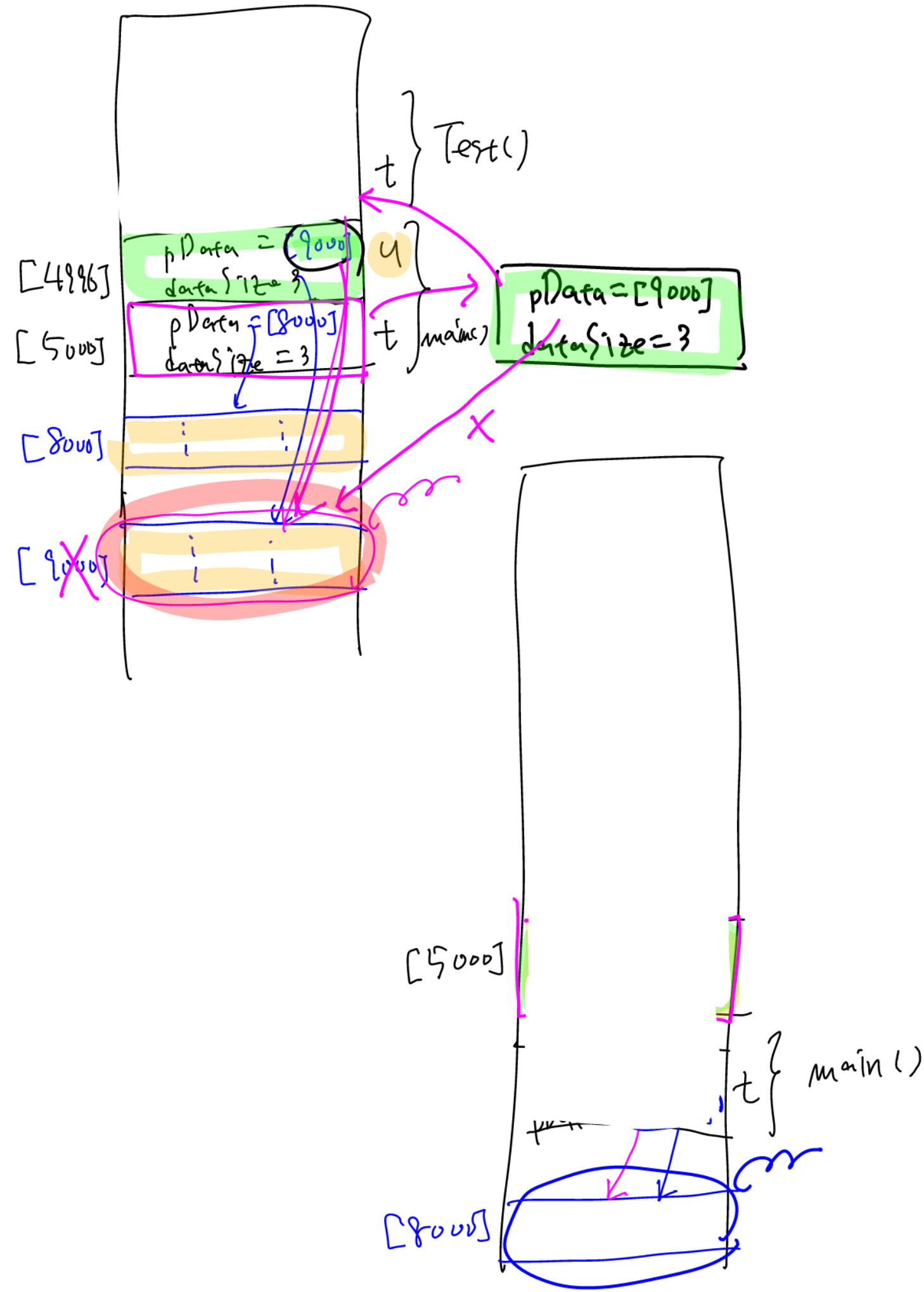
l-value(X)

r-value.








t
Test()
pData = [9000]
dataSize = 3
[4996]
4
[5000]
pData = [8000]
dataSize = 3
t
main()
pData=[9000]
dataSize=3
[8000]
X
[9000]
[5000]
t
main()
[8000]

std::move() will be explained later~;

(2017.4.15)

[2017.4.20, Thu]

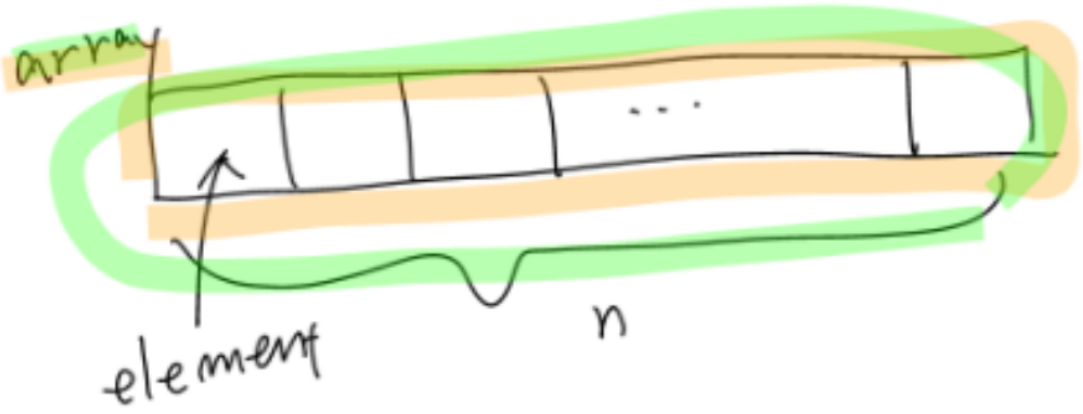
std::forward<>();

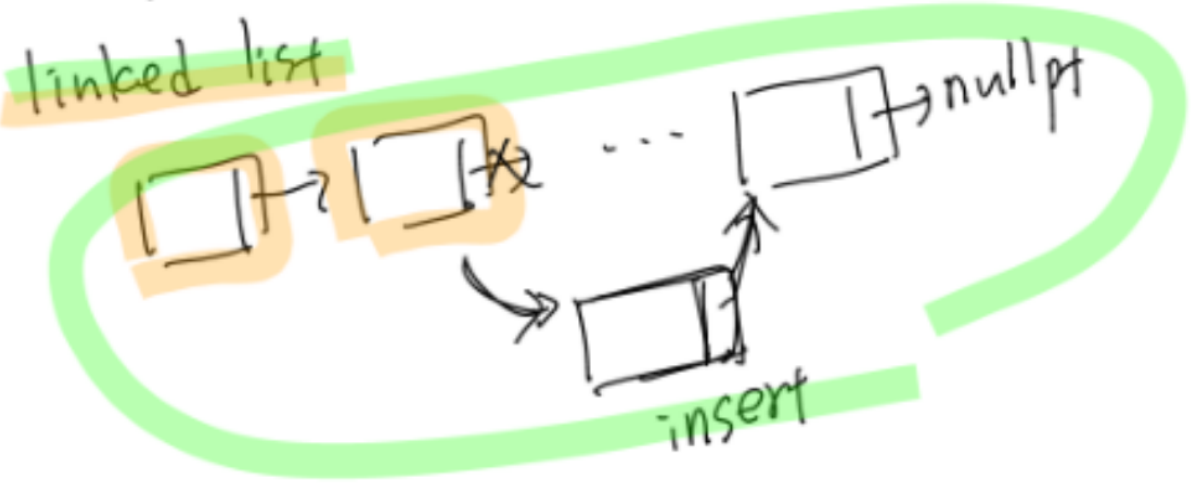
(2017.4.20)

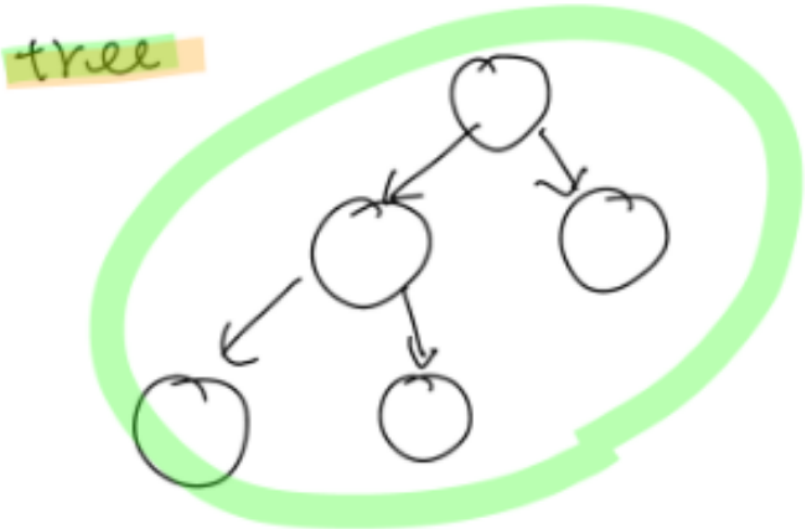
* C++ Ocw (2017.4.25)

Dongseo University Jintaek Seo.

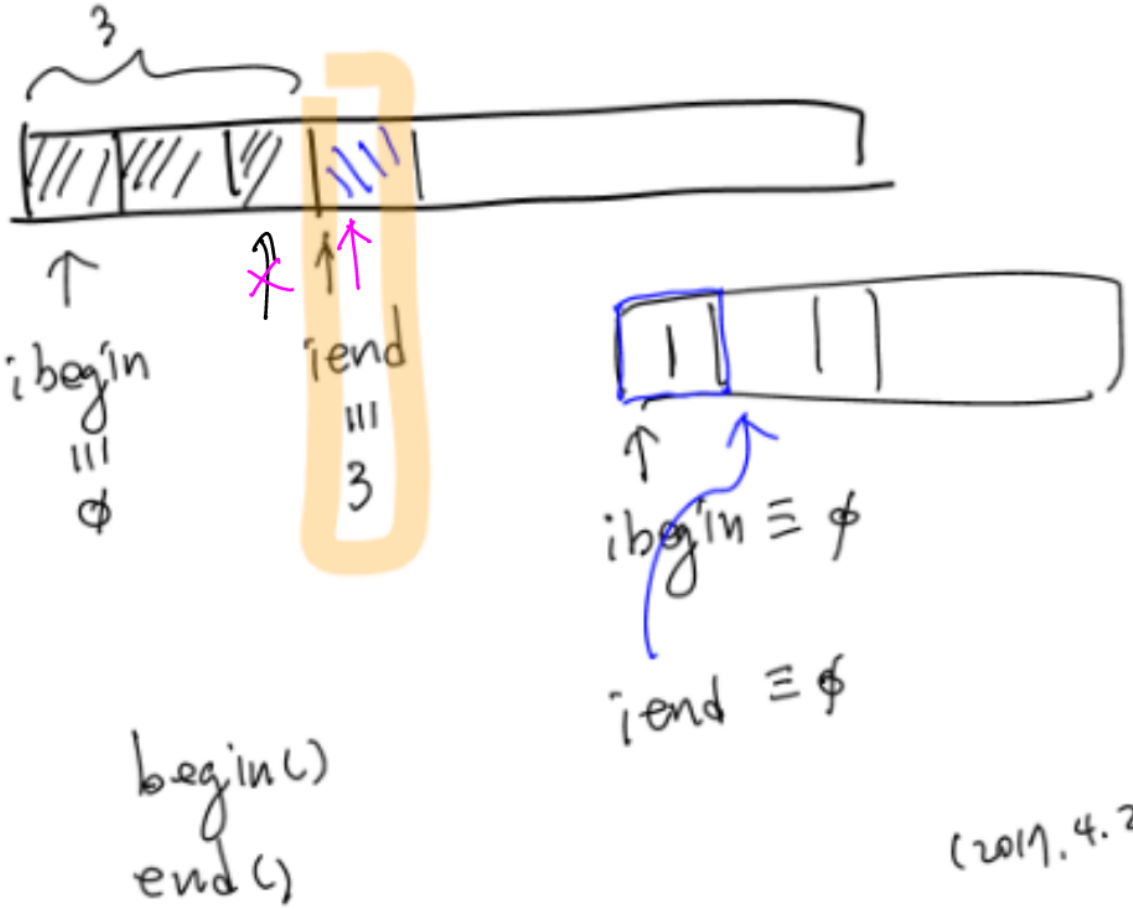
* Container, Herator.







Standard template library.



begin()
end()

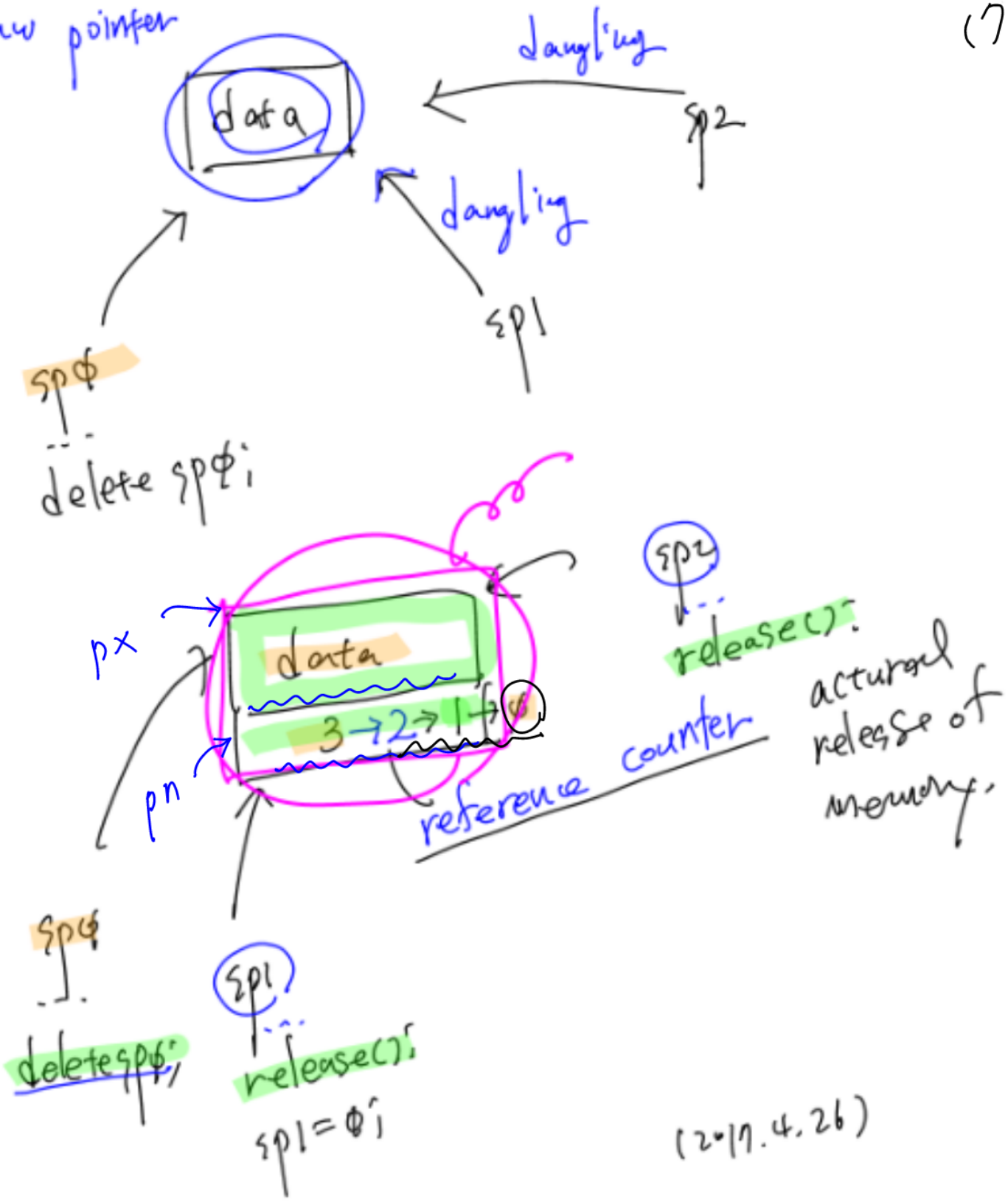
(2017.4.25)

---

② Smart pointor. [2017.4.26]

shared_ptr example.
pre-requisite
simple implementation of shared_ptr
weak_ptr
shared_from_this

① share_ptr_traits
② if (expression)
statement;



(2017.4.26)

[2017.5.1, Mon]

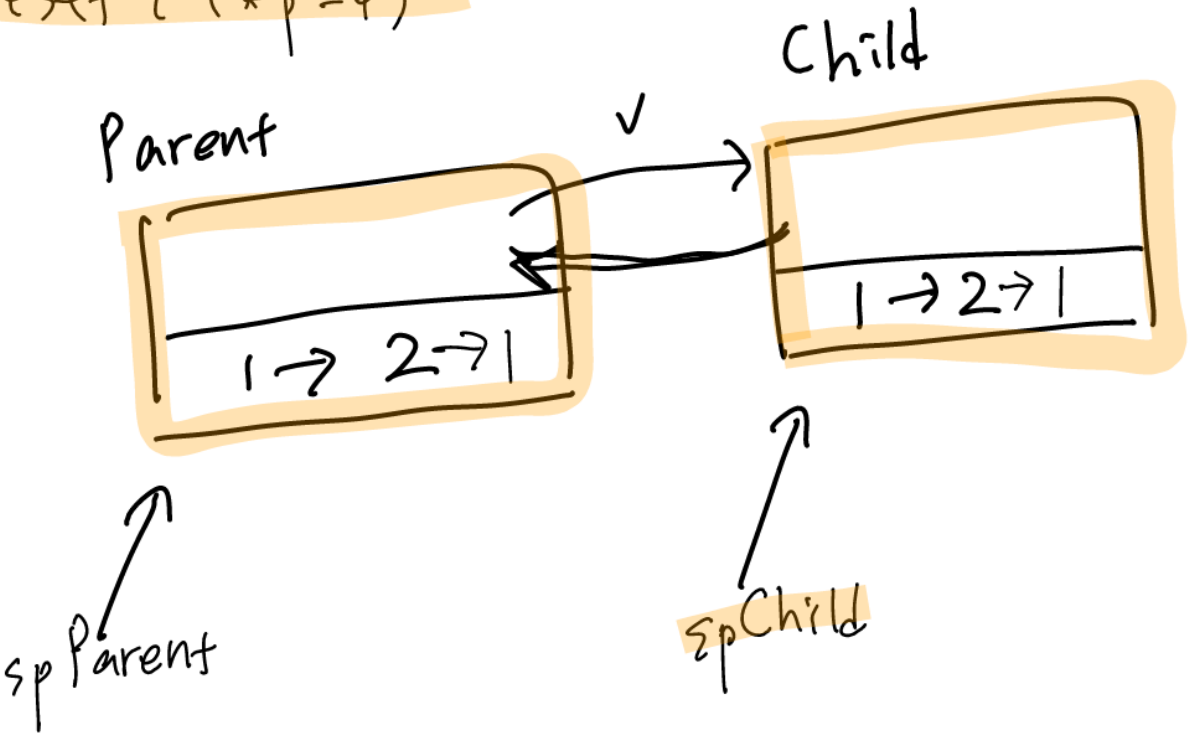
Non-throwing swap idiom.

Copy and swap idiom.

```
class shared_ptr
{ public:
        this_type;
        value_type;
        pointer;
        reference;

};
```

- ✓ constructor (T *p = 0)
- ✓ copy construction
- ✓ destruction
- ✓ release
- ✓ swap
- ✓ operator =
- ✓ reset (T *p = 0)



Cyclic Reference.

@ enable_shared_from_this

unique_ptr<> will not be explained.

(2017.5.1)

@ Lambda [2017.5.3]

function pointer

function object

bind2nd, bind1st

std::bind

lambda

[std::bind internal]

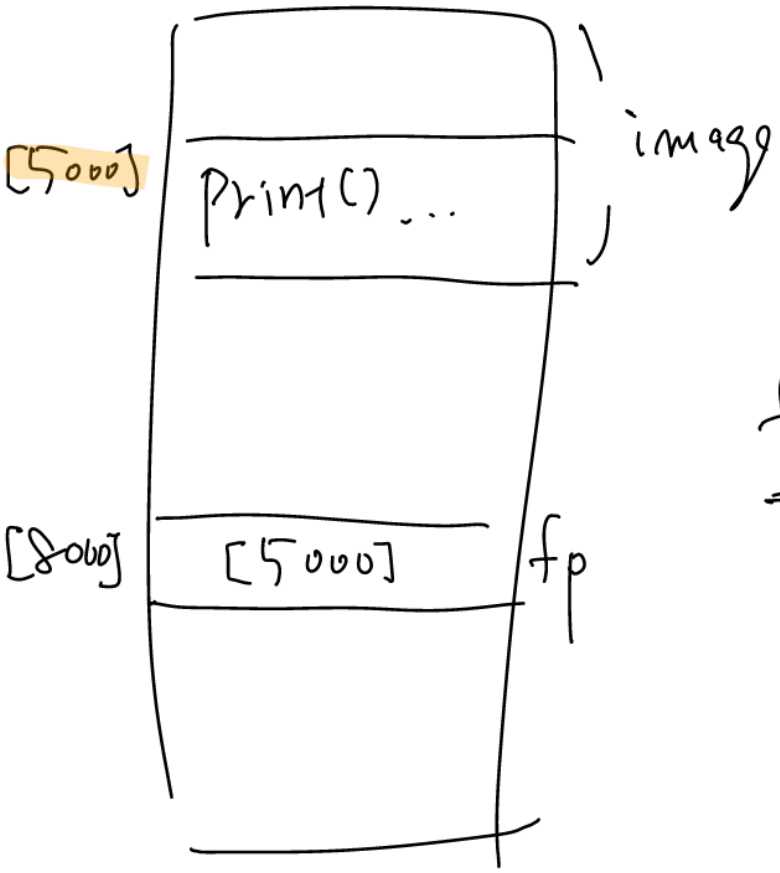
@ std::bind [2017.5.4]

member function pointer

binder

bind

placeholders



fp = Print [5000]

Print();

function call operator

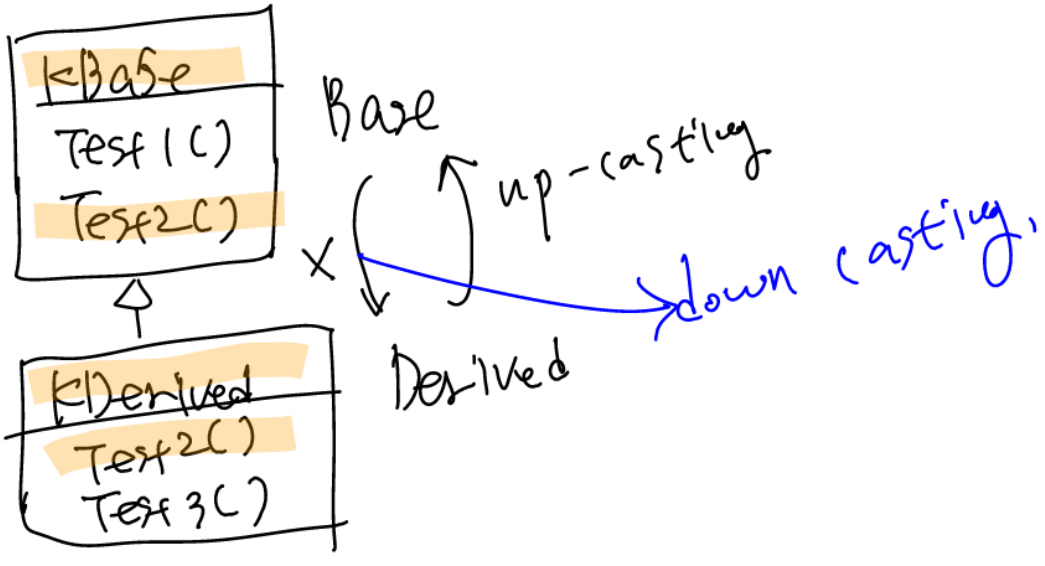
fp();

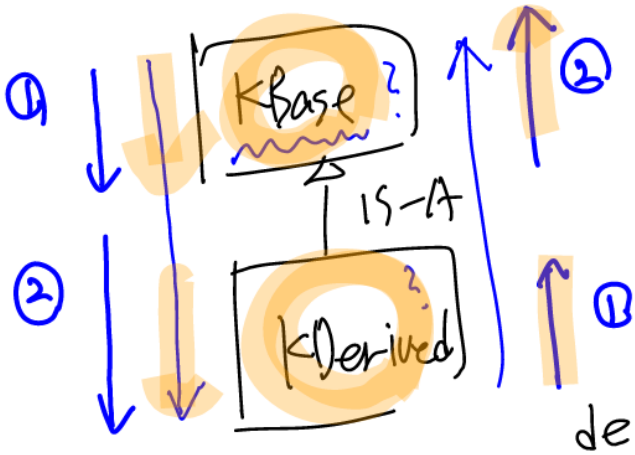
@ virtual functions. [2017.5.10]

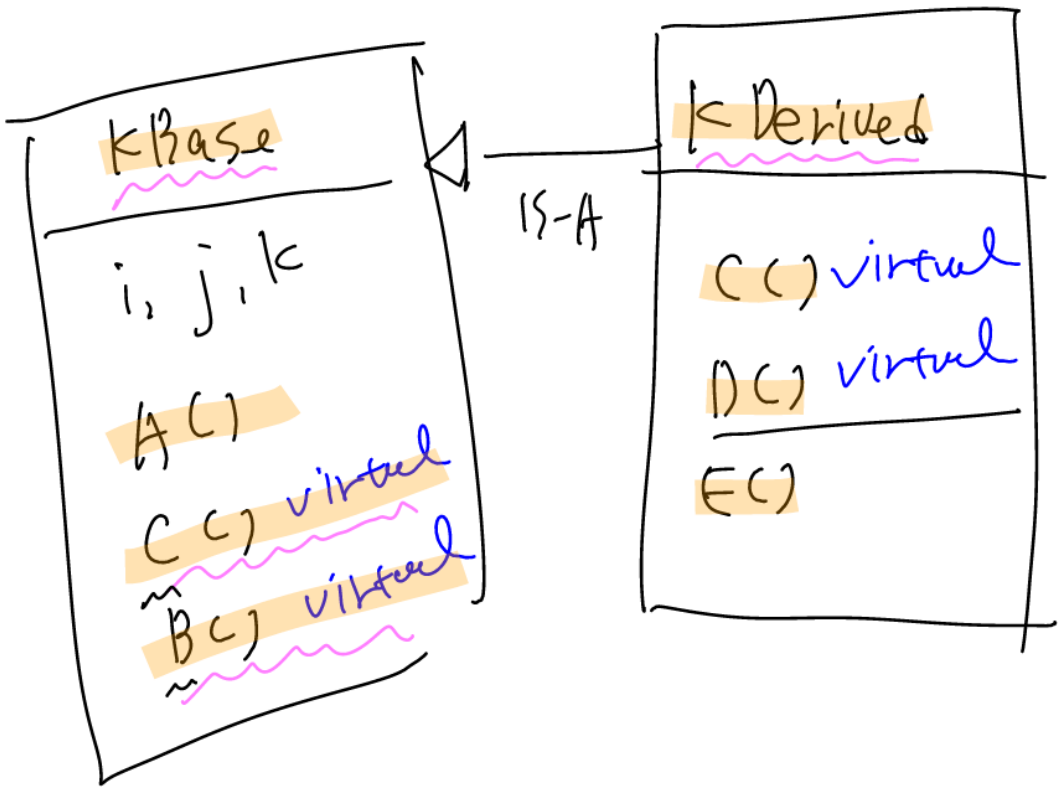
virtual function

virtual destructor, pure virtual function.

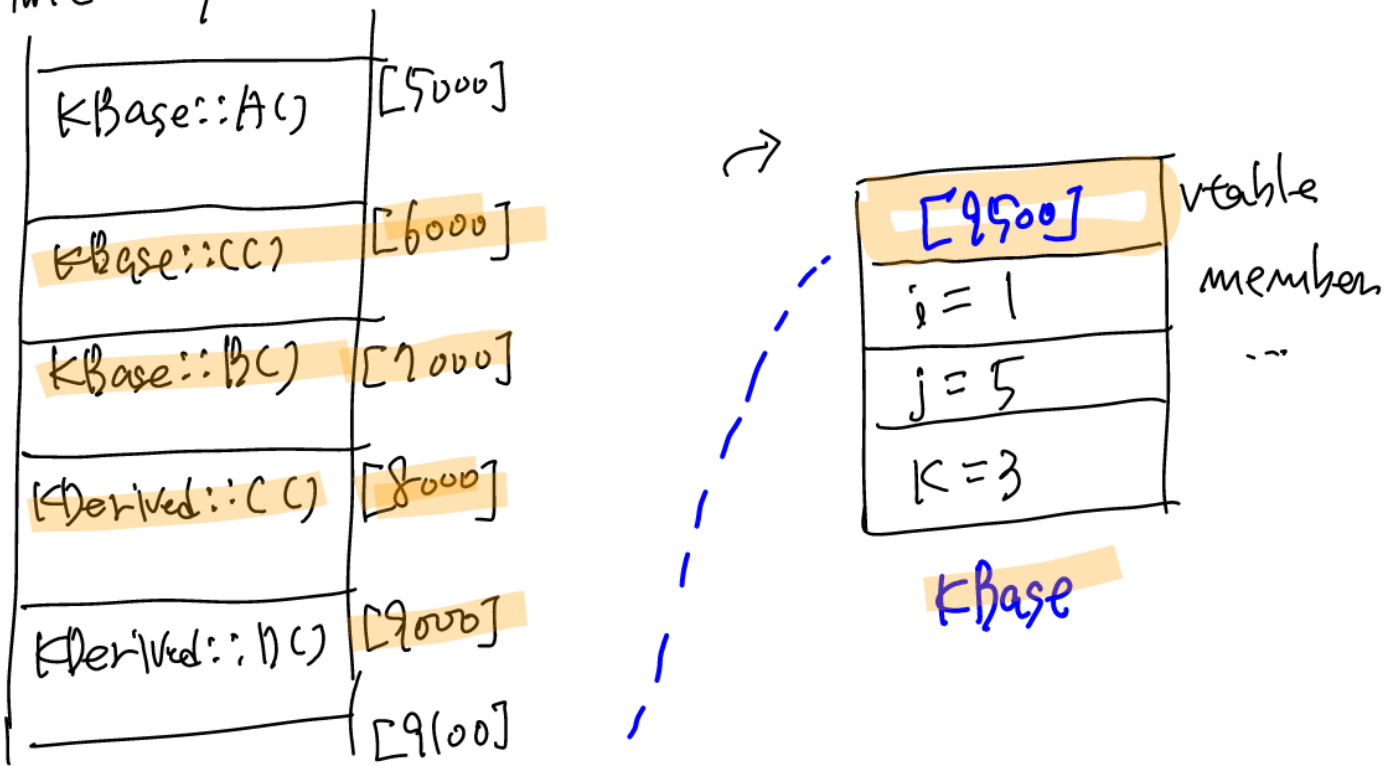
; internal operation (vtable)

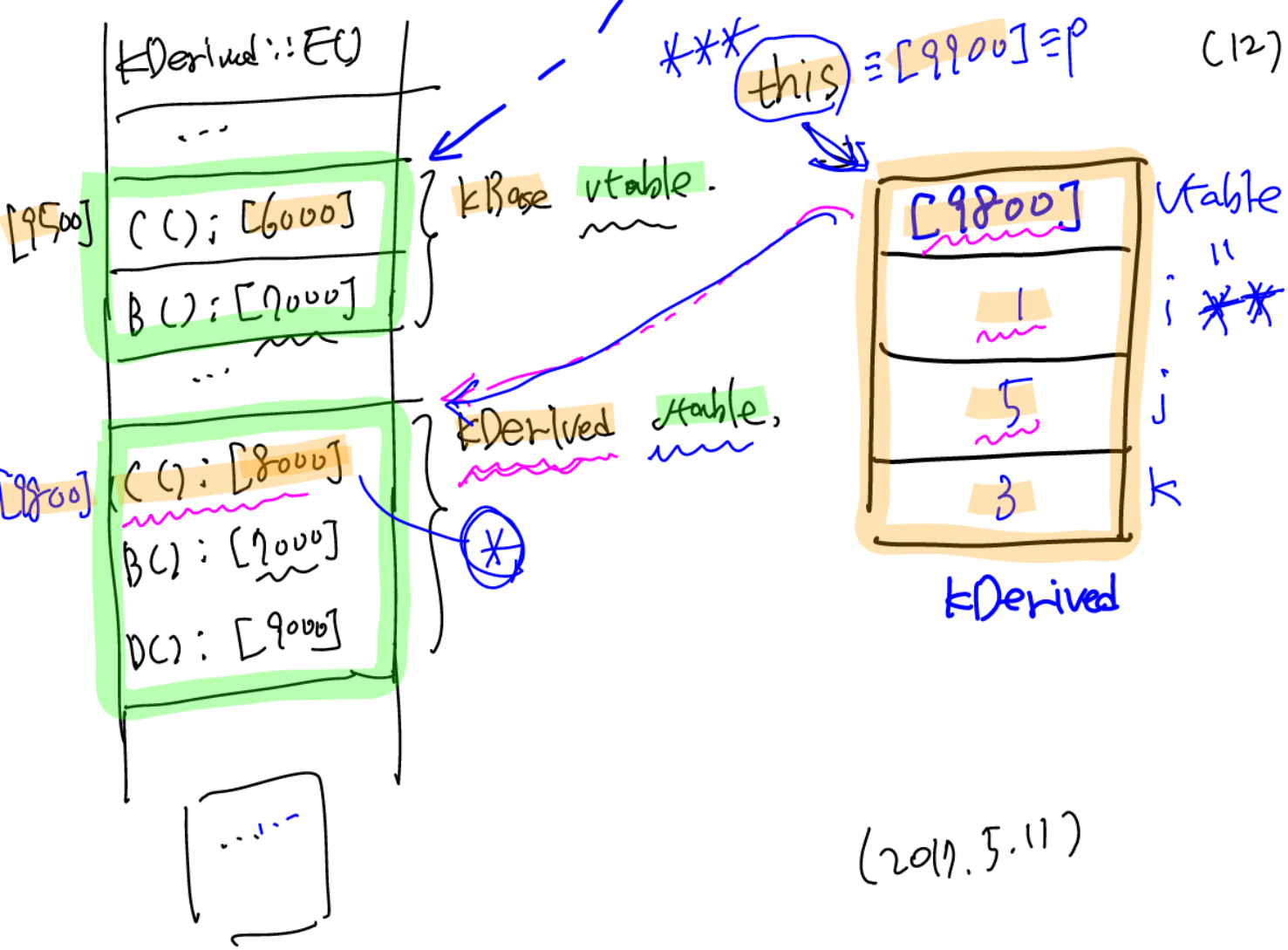



①
KBase
②
IS-A
②
KDerived
①
destructor.
Constructor


KBase
i, j, k
A()
C() virtual
B() virtual
IS-A
KDerived
C() virtual
D() virtual
E()

memory.


KBase::A()
[5000]
KBase::C()
[6000]
KBase::B()
[7000]
KDerived::C()
[8000]
KDerived::D()
[9000]
[9100]
[9500]
vtable
i = 1
member
j = 5
...
k = 3
KBase



(12)

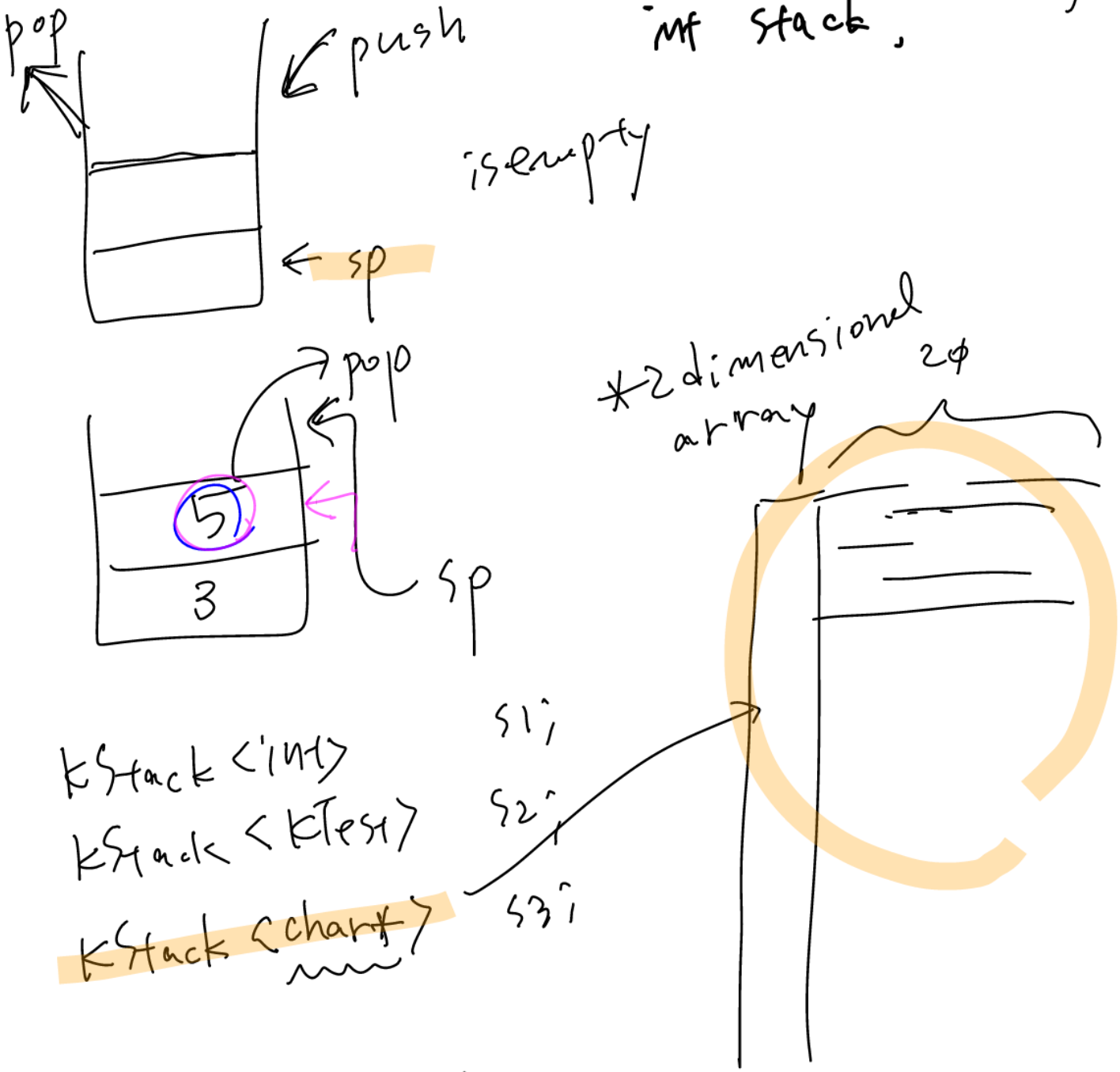
(2017.5.11)

(2017.5.12)

@ Template

- overloading & name mangling
- function template
- class template
- template specialization
- template meta programming
- variadic template.

* KStack class



@ template specialization

(2017.5.16)

full template specialization

partial template specialization

@ template meta programming.

Compile time class generation

static polymorphism.

* recursion.

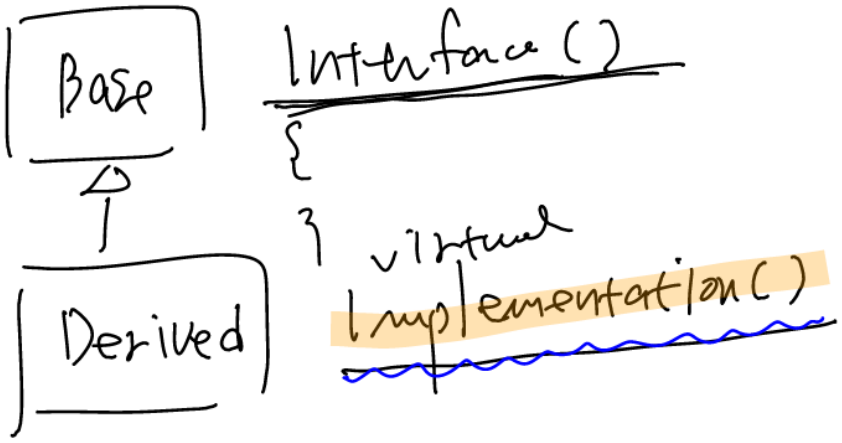
$n! = n * n-1 * \cdots * 1 = Fact(n)$

$0! = 1$

$$Fact(n) = \begin{cases} 1 & (n == 0) \\ n * fact(n-1) & (n \geq 1) \end{cases}$$

$4! = 4 \times 3 \times 2 \times 1 = 24$

* template method pattern



(2019.5.17)

@ Variadic Templates

template parameter pack

function parameter pack

typename... Args

Args... args

BOOST_PP

variable argument ...

va_list

va_start

Func<int, const char*, float>

Arg1 Args

* Args * ...

Args ...

const char * , float ...

Func(args...)

("hello", 3.1f);

Func

const char*, float>