Struct: a group of named vars (in diff type)

Eg

```
Struct  Stu{
        String  name
        int     age
        - ~
    }
Stu  a;
a. name =      ;  a.age =  ;
```

Abbreviated notion

Stu a = { str, int } → original order

easier

Pairs with 2 fields → first & second

→ use for return success + result.

```
if ( pair. first  == true)
    - ~ -
else ——
```

Tuple: with lots of fields → don't use

auto -> use for shortening code.

Structure Binding

```
auto p = std::make_pair("s", 5);
auto [a,b] = p;
```

Vector.

Initialize:

```
std::vector<type> v;
std::vector<type> v(n,k);
```

add_elem:

```
v.push_back
```

access:

```
auto k = v[i]
```

modify:

```
v[i] = x
```

isEmpty():

v.empty()

size():

v.size()

clear():

v.clear()

Uniform Initialization.

if we have a struct and we don't want to assign vars 1 after another

we can use "= {}" after var name to assign var together

E.g. std::pair <bool, int> somepair {false, 6}

Stu s{str, str, int}

even int x{3}

A even quicker way for same nums need to be assigned to 1 struct.

```
std::vector<int> v(3,5); // {5,5,5}
```

↓ constructor

, ~ ~ ~~~ {3,5} // {3,5}

Reference

int & a.l.

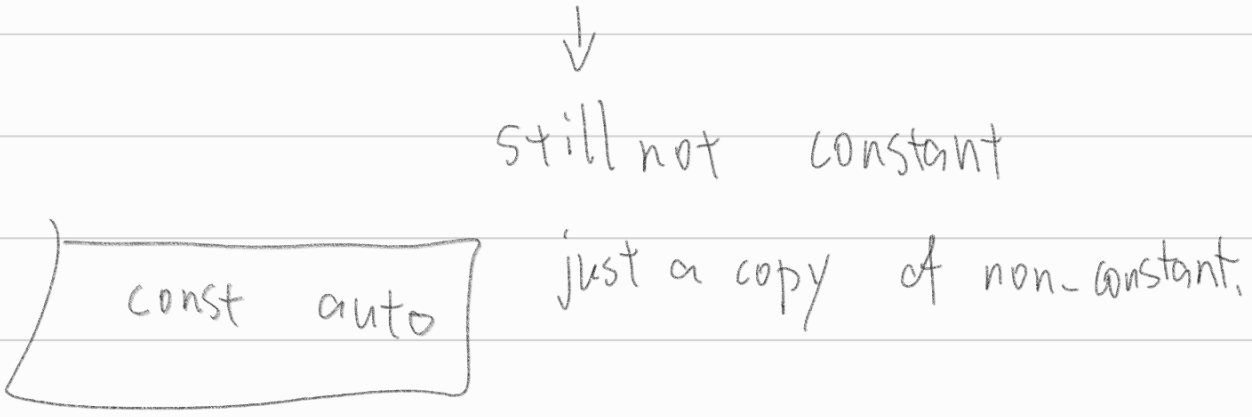
Const & Const Reference.

const ⇒ can not modified

not_constant_var = const_var ✗

Cons ~~~~ = non~~~~ ✗

|


still not constant
const auto
just a copy of non-constant.