1. Qualified name, eg: v.x(), std::vector, p->m_y, search in v, std, and p (Qualifying construct)

2. Unqualified name, eg: get<1>(), Search in successively more enclosing scope (until one found then stop)

☆ In 1 and 2, if the searched class has base, include it (transivly)

☆ Template class phase one look up for non-dependent name, not include dependent base class

☆ It is called ordinary name look up

3. Argument Dependent Lookup (ADL) for unqualified name, in addition to 2, also goes through ADL

☆ Not happening if ordinary lookup finds:

1) a declaration of class member

2) (NOT using declaration) a declaration of function in block scope

3) any declaration that is not a function of function template (eg, a function obj or any other obj)

☆难点: Associated set of namespaces and classes

☆ ptr or array is treated as pointed-to type or element type

1) Fundamental types such as int, float: Empty

2) Enum: enclosing class and ns.

3) Function type: TRANSITIVELY add param type and return type

4) Ptr to member function: apply rule 3, TRANSITIVELY add its class

5) ptr to member data: TRANSITIVELY add this member class, TRANSITIVELY add its class

6) class, union
- a) associated classes are: itself, direct and indirect, enclosing nested class
- b) associated namespace: enclosing ns of classes listed in a)
- c) template class specialization: a, b apply
  - template<> class<W>{}: (w is not template template param) TRANSITIVELY add W
  - template<> class<W<Y>>: (w is template template param) w's enclosing nested class & w's enclosing namespace

7) address of expression for a set of overloaded functions

Apply rule 3 to all overloads, if set of overloads is template, TRANSITIVEly add template (or template template arguments)

☆ ADL and unqualified ordinary Look up results are merged:

☆ using-directives in associated namespaces are ignored.

☆ namespace-scoped friend func (include templates) that are declared in an associated class are visible through ADL, even if they are not visible through Ordinary name look up.

☆ all names except for the function (or func templates) are ignored (no collision with variables)

4. Class Name Injection: class inject its name

class also inject ~~tem~~ its name if it's template.
- injected can have template arguments
- can be without <>
  - with expect type: current class specialization.
  - with expect template: treated as template