1. Vectors

- initialize. { empty — vector <type> name ;
  { with contents assigned. — vector <type> name = { , , ... }
  { repeat elements. — vector <type> name (repeat times, content)
  { repeat 0. — vector <int> nums (times)
- Traverse. for each. — for (type elem : vector) { }
- functions
  { push_back → add behind.
  { at : vector.at(pos:) = vector[pos]
  { pop_back → delete last

2. Arrays. → in a prescribed memory

- Comparison with vector { good : more efficient
  { bad : static
- Initialize & Assign { together — Type name [len] = { , , , } ;
  { seperate. Type name [len] ;
  name[0] = ... ;
- Change value

3. 2D Arrays.

- Initialize & Assign { seperate / together
  Type name[row][col] = {
  { },
  { }
  }

- Traverse { - 2 × loop
  for(int row=0; row<max_row; row++){
  for(int col ---){}}
  - sizeof( ) → the total amount of elem inside

4 2D vector.

- Initialize
- same functions

5 Struct

- Understanding — a group of public types.
- Define Struct Struct_name { type1 var1; type2 var2; }
- Initialize & Assign { sep / tog
  Struct Struct_name name = { };
  name.elem = ;
- use it in function

6. Class

- Comparison with Struct. { class default ↑ private v.s. public → struct
  method ? → not common in struct

- data member
- member function
- scope { public
  private → default.
  protected
  how switch → use keyword "public:" before public items
- getter/setter
- Constructor → Class(,) {

7. Object
   - Def
   - Initialize.

}