# - Modules

This class has been about toy programs.

Can we work on some big projects?

- How to organize code?

  Header files / packages / namespaces / classes / Libraries / Inheritance / Functions / Abstraction

  - Syntactic organization: C/C++ header files

    "Where does the code live on disk?" — .java files containing a single class

  - Namespaces: To avoid name clashes

    C++ namespaces
    Java packages

  - Abstraction: Hide details of implementation

    C++ / Java / Python: classes

  - Code reuse: Inheritance / Parametric polymorphism

    - Widgets can be drawn.
    - Buttons have "state": "clicked" or "not"
    - Scroll bars have position

- Modules: Ocaml's way to structure large pieces of code.

Two "sublanguages" within Ocaml

- Vernacular: values, expression, types, functions
- Module: structures signatures functors

C: C "proper" + "preprocessor"

#include
#define
#ifdef ...

Lisp: Lisp "proper" + macros

C++: classes / structs / pointers / expressions / ... : Vernacular

Template mechanism

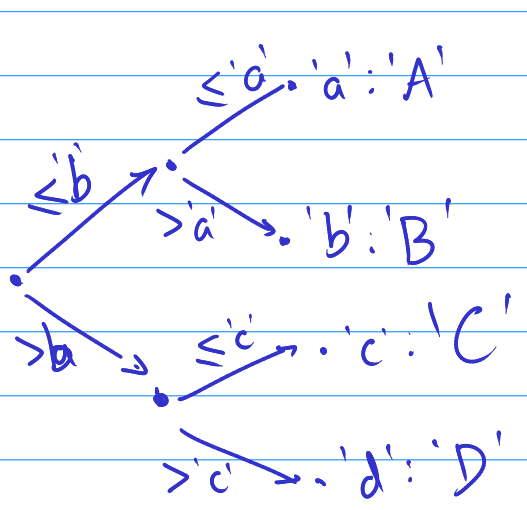
"Template metaprogramming"

Boost C++

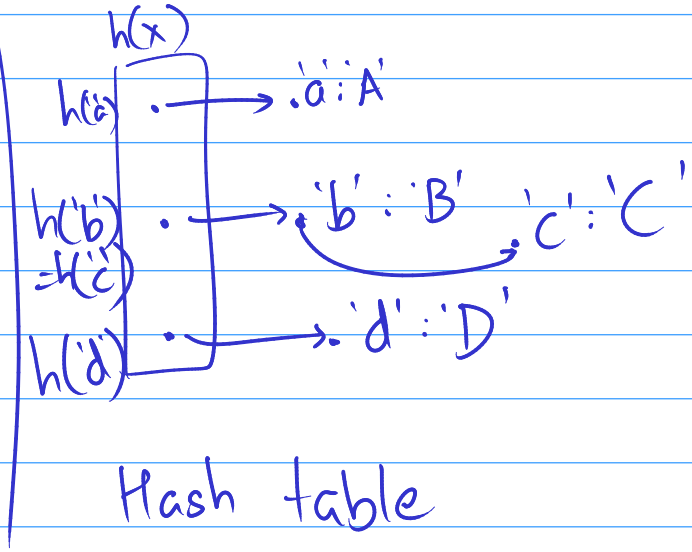
- How does one represent a dictionary?

Associative data structure, Map

{'a': 'A', 'b': 'B', 'c': 'C', 'd': 'D'}




Balanced BST
B-tree



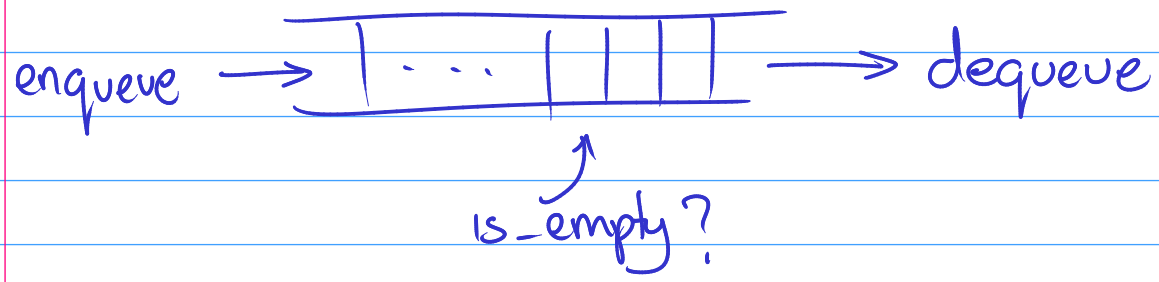

Hash table

Associative list

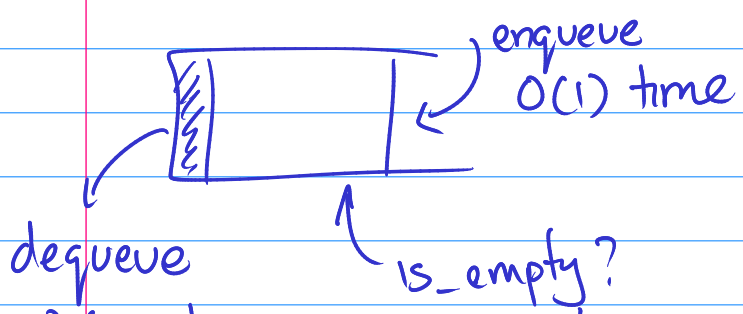
[('a';A'); ('b','B'); ('c','C'); ('d','D')]

(key, value)

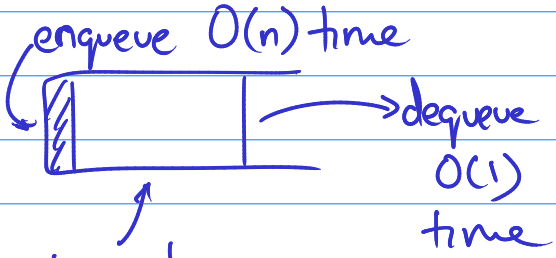
- How to implement queues?




## Implementation 1




## Implementation 2




enqueue x q = q@[x]

## Implementation 3