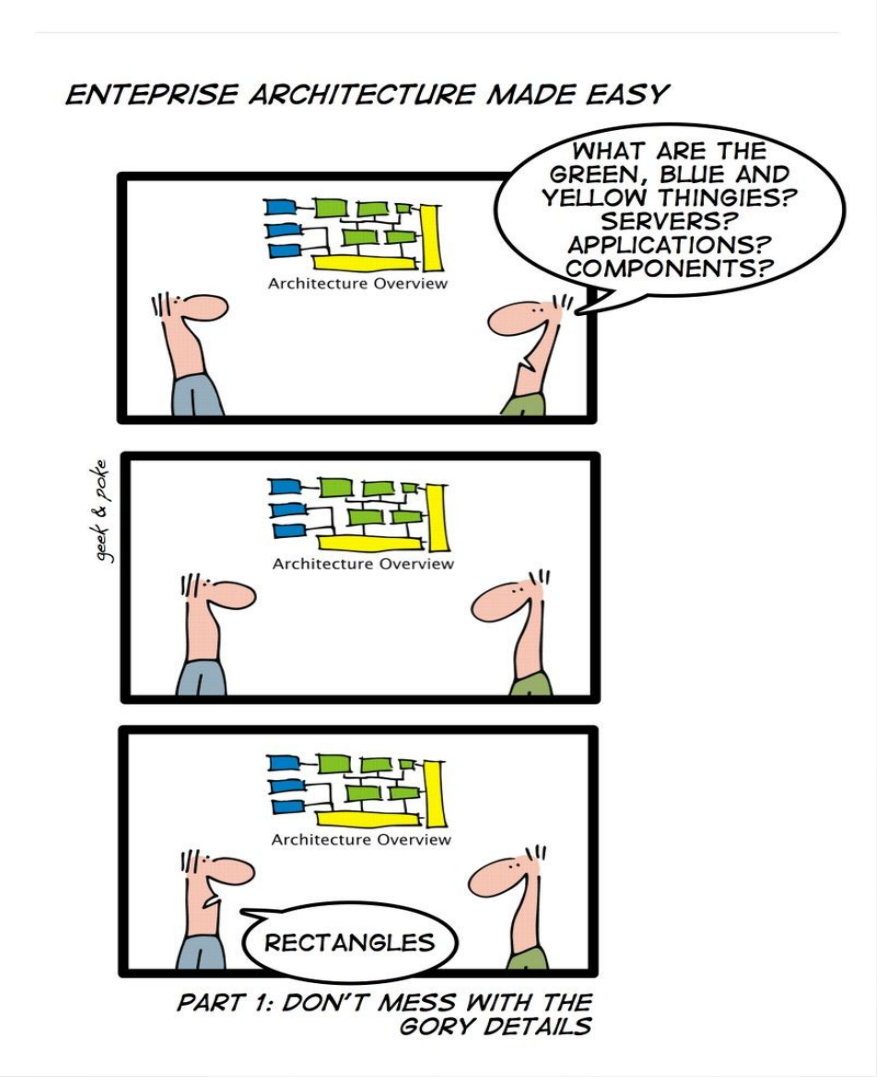
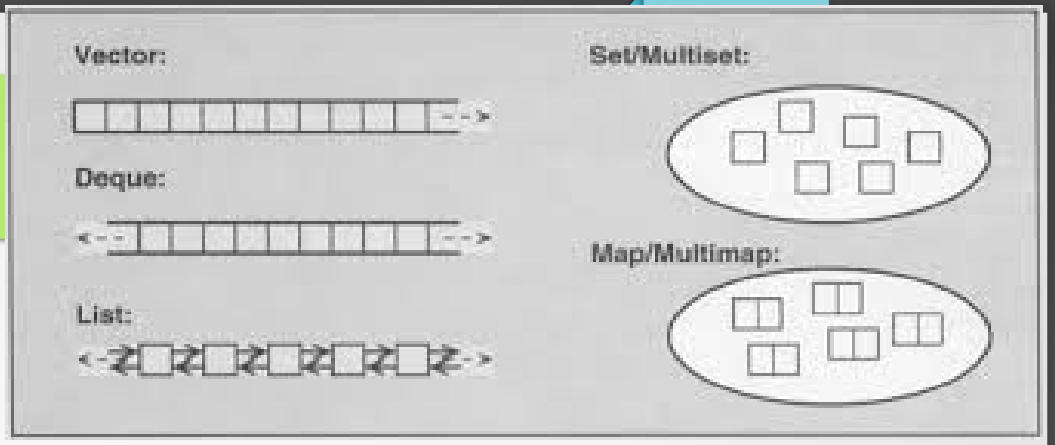
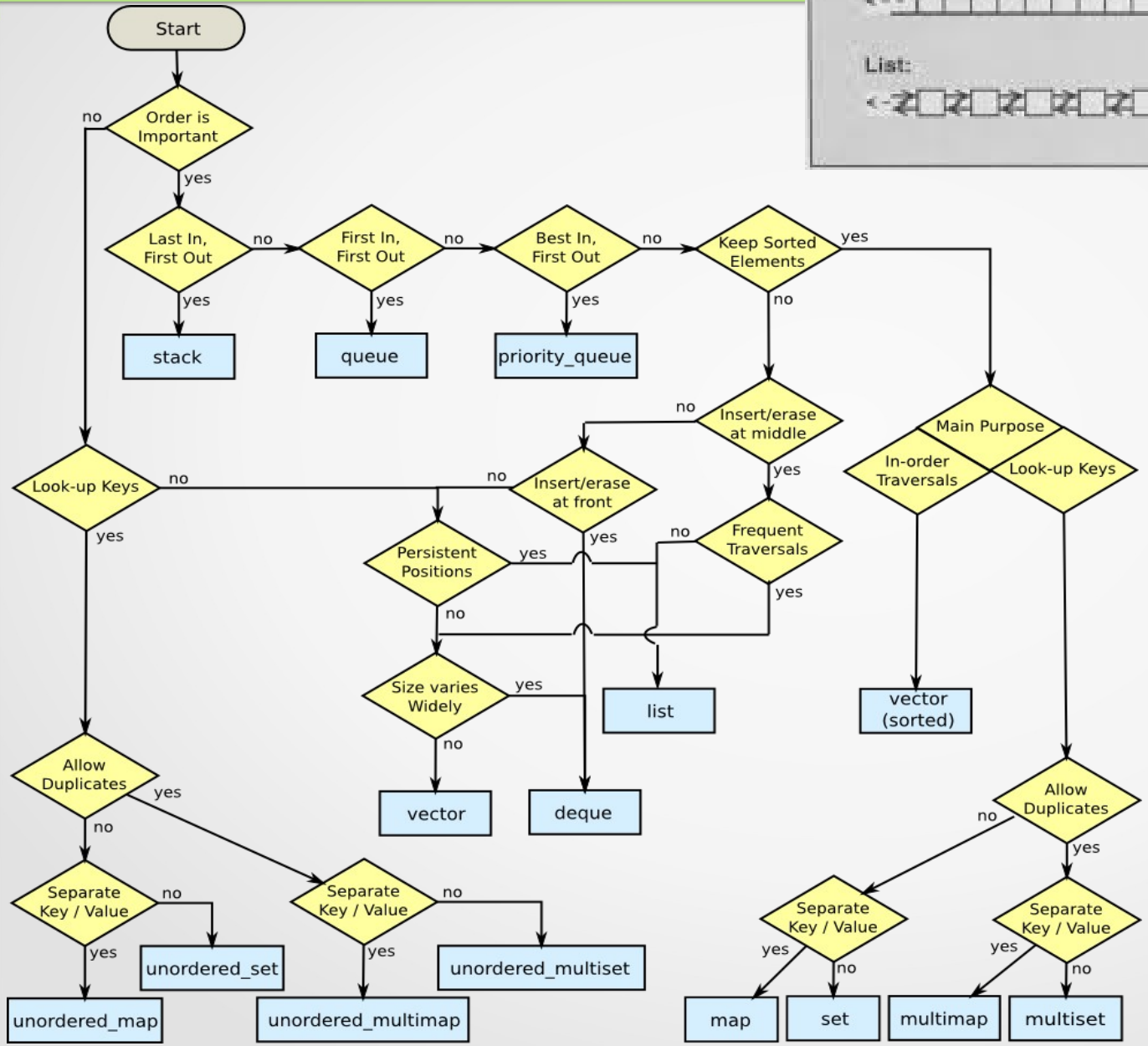
Tuple & Variadic Templates



Krishna Kumar

STL Containers





Tuple

- Tuples are objects that pack elements of -possibly- different types together in a single object, just like pair objects do for pairs of elements, but generalized for any number of elements.
- One way to imagine using a tuple is to hold a row of data in a database. The row might contain the attributes of a person, such as the person's name, account number, address, and so on. All the elements might have different types, but they all belong together as one row.
- Conceptually, they are similar to plain old data structures (C-like structs) **but instead of having named data members, its elements are accessed by their order in the tuple.**
- The selection of particular elements within a tuple is done at the template-instantiation level, and thus, it must be specified at compile-time, with helper functions such as get and tie.

Manipulating a tuple

Tuple Function	Explanation
make_tuple	Pack values in a tuple
forward_as_tuple	Pack Rvalue reference in the tuple
std::get<i>(mytuple)	Get element “i” in the tuple - mytuple
std::tie	Unpack values from a tuple
tuple_size<decltype(mytyple)>::value	Size of tuple
tuple_element<i, decltype(mytuple)>::type	Get element type of element “i” in mytuple
tuple_cat (mytuple, std::tuple<int,char>(mypair))	Concatenate tuples

References

- Bjarne Stroustrup's “C++ Programming Language 4ed”
- Scott Meyer's “Effective Modern C++”
- Herb Sutter's Exceptional C++ and More Exceptional C++
- C++ Templates: the Complete Guide
-
- http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2080.pdf