Subject: What about NTL could be added to wikipedia's STL?
Posted by fudadmin on Thu, 03 Aug 2006 02:56:07 GMT
View Forum Message <> Reply to Message

What about NTL could be added to wikipedia's STL?
What points about NTL (or Core?) could be added to wikipedia's STL to criticism section?
http://en.wikipedia.org/wiki/Standard_Template_Library

Quote:
There have been a number of criticisms of the design of the STL. These can be divided into three groups.

1.Design flaws due to limitations in the C++ language
1.1. Initialization of STL containers with constants within the source code is not straightforward (like initialization of plain old structures). Objections: The hardcoded data may be kept in static raw arrays, if required (hardly any langauge allows you to hardcode data in a map/dictionary container). Also, hardcoded data are rarely used today, replaced with data read at the initialization time.

2. Perceived flaws due to the requirement to maximize speed and minimize space usage
2.1 STL containers are not intended to be used as base classes (their destructors are deliberately non-virtual). Deriving from a container is common mistake made by novices. Objections: Of course, you cannot derive it in OO style, that is using polymorphism, but you may do it to extend it with additional facilities (this is still not recommended, though). But you can still use it as a field and delegate operations from outside classes.

3. Other flaws, caused either by trade-offs in design or which have become more visible over time.
3.1 The concept of iterators as implemented by STL can de difficult to understand at first: for example, if a value pointed to by the iterator is deleted, the iterator itself gets invalidated. This is a common source of errors. Most implementations of the STL provide a debug mode which is slower but will catch these kinds of errors if used. Objections: This is not a flaw - this is a consequence of such a design. Apparently, there are no other container designs, which are the same way generic, and do not contain such a "flaw" (see containers in Java).
3.2. The design of STL allocator (used by the containers) is widely seen as flawed.
3.3. The set of algorithms is not seen as sufficiently complete (for example copy_if algorithm was left out by oversight). Objections: Probably copy_if would be more clear, but same may be achieved with remove_copy_if.
3.4. The interface of containers is often seen either as insufficient or as bloated. Objections: if it is both insufficient and bloated, then it means that it is well ballanced.
3.5. Hashing containers were left out (and will be added in a new version of the C++ Standard).