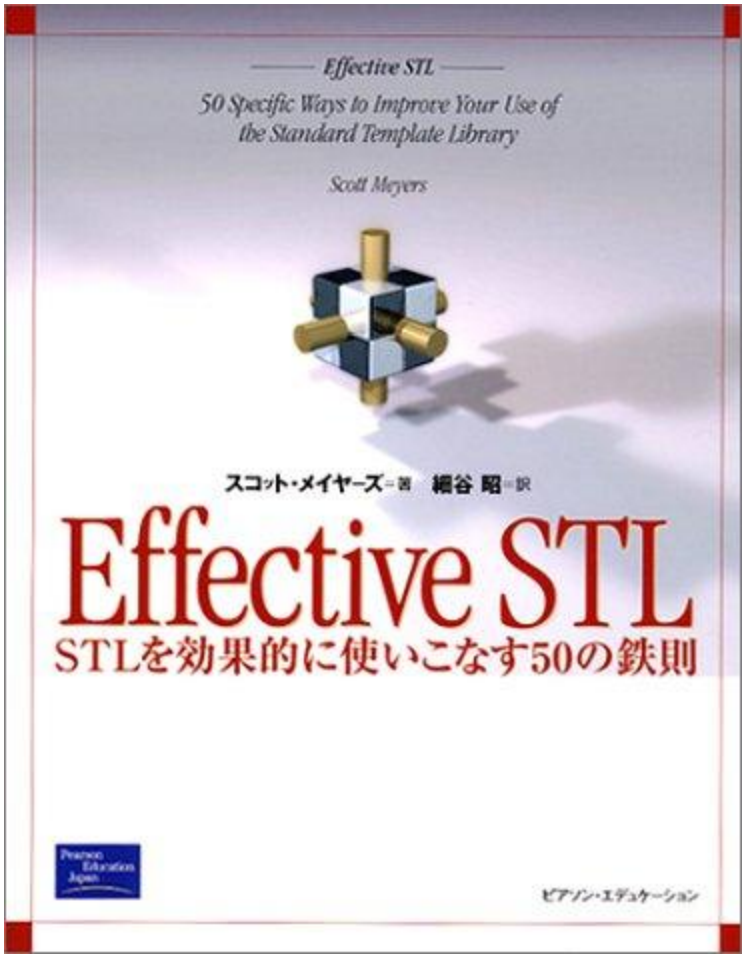
Effective STL 11


Effective STL
50 Specific Ways to Improve Your Use of the Standard Template Library
Scott Meyers
スコット・メイヤーズ=著 細谷 昭=訳
Effective STL
STLを効果的に使いこなす50の鉄則
Pearson Education Japan
ピアソン・エデュケーション

当然読んでますよね？

Effective STL の

C++11で不要になったテクニック

を紹介します

第4項 size()で0を調べる代わりにemptyを呼び出そう

第4項 size()で0を調べる代わりにemptyを呼び出そう

C++03:

- list::size()は線形時間の実装になる可能性があるよ
- empty()なら必ず定数時間になるよ

第4項 size()で0を調べる代わりにemptyを呼び出そう

C++11:

- list::size()は線形時間の実装になる可能性があるよ
  - size()は定数時間が必須になったよ！
- empty()なら必ず定数時間になるよ
  - size()でも定数時間になるよ！

第10項 アロケータの規則と制限に注意しよう

第10項 アロケータの規則と制限に注意しよう

C++03:

- カスタムアロケータが状態持てるなんて幻想ですよ

第10項 アロケータの規則と制限に注意しよう

C++11:

- カスタムアロケータが状態持てるなんて幻想ですよ
  - まずはそのふざけた幻想をぶち殺す
  - scoped_allocator_adaptorとかでぐぐりましょう

第13項 動的に割り当てられる配列よりvectorとstringを優先して使おう

第15項 stringの実装の違いに注意しよう

第13項 動的に割り当てられる配列よりvectorとstringを優先して使おう

第15項 stringの実装の違いに注意しよう

C++03:

- stringの実装は参照カウント使ってるものがあるよ！

第13項 動的に割り当てられる配列よりvectorとstringを優先して使おう

第15項 stringの実装の違いに注意しよう

C++11:

- stringの実装は参照カウント使ってるものがあるよ！
  - 参照カウント禁止されました

第16項 vectorとstringのデータをレガシーAPIに渡す方法を学ぼう

C++03:

- doSomething(&v[0], v.size())は危険だよ！
- ちゃんとemptyかどうかチェックしよう！

- stringをレガシーAPIに渡すのはc_str()しか無いよ！
- なので読み取りしかできないよ！

C++11:

- doSomething(&v[0], v.size())は危険だよ！
- ちゃんとemptyかどうかチェックしよう！
  - v.data()を使おう！

- stringをレガシーAPIに渡すのはc_str()しか無いよ！
- なので読み取りしかできないよ！
  - doSomething(&s[0], s.size())しても大丈夫になったよ！

第17項 余分な容量を取り除くには「swap技法」を使おう

第17項 余分な容量を取り除くには「swap技法」を使おう

C++03:

- vec.erase()しても容量は減らないよ！
- vector<Hoge>(vec).swap(vec)で余分な容量が消えるよ！

第17項 余分な容量を取り除くには「swap技法」を使おう

C++11:

- vec.erase()しても容量は減らないよ！
- vector<Hoge>(vec).swap(vec)で余分な容量が消えるよ！
  - それvec.shrink_to_fit()でできるよ！

第25項 標準以外のハッシュコンテナに慣れよう

第25項 標準以外のハッシュコンテナに慣れよう

C++03:

- STLにハッシュコンテナは無いよ！
- 探せ！ハッシュの全てをそこに置いてきた！

第25項 標準以外のハッシュコンテナに慣れよう

C++11:

- STLにハッシュコンテナは無いよ！
- 探せ！ハッシュの全てをそこに置いてきた！
  - unordered_xxxという名前で標準入り

第26項 const_iterator、reverse_iterator、const_reverse_iteratorよりiteratorを優先的に使おう

C++03:

- `iterator insert(iterator pos, const T& x);`
- const_iteratorを渡せないんです＞＜
- コンパイラのバグもあったりするんです＞＜
- いろいろ面倒だからもうiteratorでいいよね！

C++11:

- `iterator insert(const_iterator pos, const T& x);`
- const_iteratorを渡せないんです＞＜
  - const_iterator渡せるようになったよ！
- コンパイラのバグもあったりするんです＞＜
  - 無いよ！
- いろいろ面倒だからもうiteratorでいいよね！
  - const教徒が見てるよ！

第33項 ポインタのコンテナには注意してremove風アルゴリズムを使おう

C++03:

- Boostのスマートポインタを使おう！

C++11:

- 標準のスマートポインタを使おう！

第35項 mismatchまたはlexicographicsl_compareを使って、大文字小文字を区別しない単純な文字列比較を実現しよう

C++03:

- 2位: `lexicographical_compare` (23文字)
- 1位: `set_symmetric_difference` (24文字)

C++11:

- m位: lexicographical_compare (23文字)
- n位: set symmetric difference (24文字)
- 1位: atomic compare_exchange_strong_explicit (39文字)

第36項 copy_ifの正しい実装について理解しよう

第36項 copy_ifの正しい実装について理解しよう

C++03:

- 勢い余って`copy_if`を標準から消しちゃったてへ

第36項 copy_ifの正しい実装について理解しよう

C++11:

- 勢い余って`copy_if`を標準から消しちゃったてへ
  - 入りました

第37項 範囲に関する要約情報を取得するには、accumulateまたはfor_eachを使おう

C++03:

- accumulateの副作用禁止とかバカバカしいよね！

C++11:

- accumulateの副作用禁止とかバカバカしいよね！
  - 副作用OKになりました

第40項 ファンクタクラスを変換可能にしよう

第40項 ファンクタクラスを変換可能にしよう

C++03:

- 関数オブジェクトにはunary_function, binary_functionを使おう！

第40項 ファンクタクラスを変換可能にしよう

C++11:

- 関数オブジェクトにはunary_function, binary_functionを使おう！
  - それdeprecatedだよ！

第41項 ptr_fun、mem_fun、および mem_fun_refの使用理由を理解しよう

C++03:

- ptr_funとかmem_funとかmem_fun_refを使って呼び出し処理をジェネリックにしよう！

C++11:

- ptr_funとかmem_funとかmem_fun_refを使って呼び出し処理をジェネリックにしよう！
  - それdeprecatedだよ！
  - bindとかラムダ式使えばいいと思うよ！

おまけ

付録B MicrosoftのSTLプラットフォームについて

付録B MicrosoftのSTLプラットフォームについて

C++03:

- “以下に示すのは、Microsoft Visual C++ (MSVC) Version **4〜6** を使っている開発者向けの情報である”

C++03:

- “以下に示すのは、Microsoft Visual C++ (MSVC) Version **4〜6** を使っている開発者向けの情報である”
  - そんなのは実在**しません**。
  - このページに辿り着いたら安心して本を閉じよう！

おわり