# Boost.Container特有の機能

高橋 晶(Akira Takahashi)
faithandbrave@longgate.co.jp
2015/05/30 Boost.勉強会 #17 東京

# 自己紹介 1/2

- 高橋 晶(Akira Takahashi)
- システム系とかゲーム系とか、いろいろな開発をやってます。
- 最近は、プロ向けの教育の仕事とか、Emscriptenを使って C++でブラウザゲームを作っていたりしました。

# 自己紹介 2/2

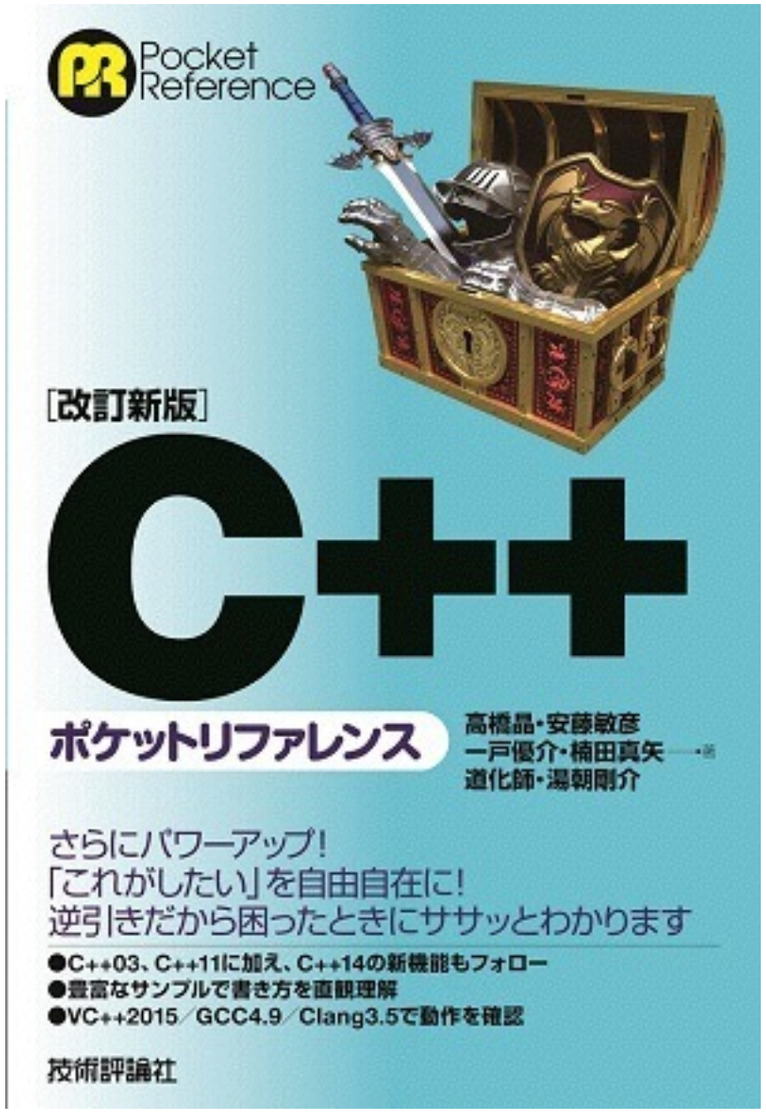
- 著書：『C++テンプレートテクニック』
  『C++ポケットリファレンス』『プログラミングの魔導書』シリーズ

- C++の日本語リファレンスサイトcpprefjpを作っています。

- Boostの日本語情報サイトboostjpを作っています。

- そのほか、たまにBoostにpull requestを投げたりしています

# C++ポケットリファレンス




- C++14に対応した第2版は、2015年6月4日(木)に発売です

# 本日のお題：Boost.Container

- この発表では、Boostのコンテナライブラリについて話します。
- Boost.Containerには、標準ライブラリとほとんど同じ機能が入っているため、あまり使われていないように思います。
- しかし実際には、標準ライブラリよりもこちらを使った方がいい、というような機能が多々入っています。
- 今回の発表では、Boost.Container特有の便利な機能を紹介していきます。

# Boost.Containerとは

- 作者Ion Gaztañaga
  - C++標準化委員会で、コンテナ関係の仕様を決めてる人

- 標準コンテナの最新仕様を、標準規格の更新を待たず、
  すぐに試せることを目的としたライブラリ。

- この発表では、C++11標準コンテナの知識を前提として、
  Boost.Containerの拡張機能やその他特徴を紹介します。

# 話すこと

- あらゆる環境向けに、ひとつの実装
- 標準以外のコンテナ
- 要素のデフォルト値を未初期化にする
- realloc機能に対応したアロケータ
- 例外送出のカスタマイズ
- 連想コンテナの拡張オプション

# あらゆる環境向けに、ひとつの実装

標準コンテナは、環境ごとにコンテナの実装が異なる。
Boost.Containerでは実装がひとつなので、以下のようなことを期待して使用できる：

- メモリ確保する場所・条件が一定
  - たとえば、デフォルトコンストラクタはメモリ確保しない (なので、例外を送出しない)
- `basic_string`のSmall String Optimization
  - 短い文字列については、スタックを使用する
- ただし、デフォルトのメモリアロケータは、環境ごとの`malloc`を使用するので、パフォーマンスは一定ではない

# 標準以外のコンテナ 1/5

Boost.Containerには、標準ライブラリにはない拡張コンテナがいくつか用意されている

- `stable_vector`
- `flat_map`／`flat_set`
- `static_vector`
- `small_vector`

# 標準以外のコンテナ 2/5

`stable_vector`

- `vector`と`list`のハイブリッド
- 伸長や消去をした際に、イテレータと参照が無効にならない
- メモリが連続していない
- 定数時間でランダムアクセスできる

# 標準以外のコンテナ 3/5

`flat_map ／ flat_set`

- ソート済みvectorとしての、順序付き連想コンテナ
- ツリー構造ではないからflat(平坦)
- イテレーションが標準連想コンテナよりも高速
- メモリ消費が標準連想コンテナよりも小さい
- 検索は対数時間

# 標準以外のコンテナ 4/5

`static_vector`

- 動的メモリ確保せず、スタックを使用するvector

- 第2テンプレート引数で、最大の要素数を指定して、それを超えないように使用する。超えたらbad alloc例外。

```
boost::container::static_vector<int, 3> v; // 3要素まで伸長できる

// 要素の追加
v.push_back(3);
v.push_back(1);
v.push_back(4); // 4回目は、やってはいけない

for (int x : v) {
  std::cout << x << std::endl;
}
```

# 標準以外のコンテナ 5/5

small_vector

- 小さい要素数に特化したvector。

- 第2テンプレート引数で、事前にメモリ確保する要素数を指定する。その要素数を超えたら再確保。

```
boost::container::small_vector<int, 3> v; // 3要素分を事前にメモリ確保

v.push_back(3);
v.push_back(1);
v.push_back(4); // ここまで、メモリの再確保なし

for (int x : v) {
  std::cout << x << std::endl;
}
```

# 要素のデフォルト値を未初期化にする

- vectorのresize()は、伸ばした分の要素を、値初期化する
- 巨大な動的配列を作る際には、値初期化のコストが無視できなくなる
- Boost.Containerのvectorは、要素を未初期化にする拡張を提供している

```
namespace cont = boost::container;

// 要素数を3個で構築
cont::vector<int> v(3, cont::default_init);

// 要素数を4に伸長する
v.resize(4, cont::default_init);
```

# realloc機能に対応したアロケータ

- vectorは、要素を伸長した際に、キャパシティを超えたら別な場所に、
  より大きな領域を確保して要素を移動する。
- boost::container::allocator<T, 2>を使用すると、
  realloc相当の機能で、いま確保しているメモリ領域を伸ばしてくれる。
- この機能は、dlmallocを改良して実装されている。

```
namespace cont = boost::container;
cont::vector<int, cont::allocator<int, 2>> v;

v.reserve(3);
v.push_back(3);
v.push_back(1);
v.push_back(4);
v.push_back(5); // メモリ領域を伸長する
```

詳細は、Boost.勉強会 #15 札幌での、池田さんの発表を参照。

# 例外送出のカスタマイズ

- Boost.Containerは、例外送出について、いくつかのカスタマイズ方法を提供している
- BOOST_NO_EXCEPTIONSをdefineする
  - 例外送出のところで、std::abort()を呼び出して異常終了する
- BOOST_CONTAINER_USER_DEFINED_THROW_CALLBACKSをdefineする
  - 例外の送出方法を自分で決める
- これによって、例外発生時に、ログやバックトレース等を埋め込める

```
#define BOOST_CONTAINER_USER_DEFINED_THROW_CALLBACKS
#include <boost/container/vector.hpp>

namespace boost { namespace container {
  void throw_out_of_range(const char* str) // カスタマイズ用のハンドラ
  {
    output_log(str); // ログ出力
    throw std::out_of_range(str);
  }
}}
```

# 連想コンテナの拡張オプション

- Boost.Containerの順序付き連想コンテナは、
  ツリー構造のオプションを指定できる。

```
using namespace boost::container;

// AVL木を使用する
using AvlTree = tree_assoc_options< tree_type<avl_tree> >::type;
using AvlSet = set<int, std::less<>, AVLTree>;

// スプレー木を使用する
// 最近アクセスした要素に、高速にアクセスできる(償却対数時間)
using SplayTree = tree_assoc_options< tree_type<splay_tree> >::type;
using SplaySet = set<int, std::less<>, SplayTree>;

// スケープゴート木を使用する
// 挿入と削除が償却対数時間になる
using ScapegoatTree = tree_assoc_options< tree_type<scapegoat_tree> >::type;
using ScapegoatSet = set<int, std::less<>, ScapeGoatTree>;
```

# そのほか

- vector<bool>の特殊化はありません。設計ミスと言われている機能は、Boost.Containerでは採用していません。
- C++1zで採用予定の、「不完全型のサポート」が、static_vectorとbasic_string以外に対して入っています。
- Boost.Interprocessと組み合わせることで、ファイルとのメモリマップができます。
  オンメモリに乗らない巨大なデータを扱えます。

# まとめ

- Boost.Containerには、いろいろな場面で役立つ機能が多々入っています。

- スタックを使用するstatic_vectorはよく使いますし、
例外送出のカスタマイズは、何かあったときに、痒いところに手が届いてくれて便利です。

- どんどん使ってください！