エクスリームC++11/14プログラミング

H.28/07/23

Egtra

Boost.勉強会 #20

自分

- Egtra
  - Twitter: @egtra
  - http://dev.activebasic.com/egtra/
- 仕事: 主にVisual C++ 2005/2015 (Windows)
  - 最近C++ (clang/g++, Linux)も少し

注意1: 取り扱っていないこと

- リファクタリング
  - 既存コードはまだまだこれから
- 新バージョンのコンパイラを職場に導入する方法
  - 必要性をアピールするなど

注意2: 行儀悪いコード

- Undefined behaviorな書き方が登場します
  - 積極的に推奨するものではありません
  - （良い子のみんなは真似しないでね）

目次

- 第1章 始まり
- 第2章 泥沼
- 第3章 光
- まとめ

第1章 始まり

- 201x年、あるWinアプリ開発に配属
- コンパイラ: Visual C++ 2005
  - もう2008も2010も出ていた
  - 2012も出た頃？

C++11はいいぞ

- もう体はC++11に適応し始めていた
  - ラムダ式とか使いたい。
- 将来を見据えて、少しずつC++11っぽいコードを書こうと決めた。

C++11っぽさ その1

- VC++ 2005コンパイラのがんばり
  - テンプレートの>>を空白無しで書く
  - overrideキーワードを使う
  - enumの基底型
    - enum X : int { a };
    - 上のintの指定
  - enum列挙子のスコープ付きでの参照
    - X::a

C++11っぽさ その2

- nullptr
  - C++03環境では作れる
    - https://ja.wikibooks.org/wiki/More_C%2B%2B_Idioms/nullptr
  - 全く使われていなかったが、積極的に使うようにした。

C++11っぽさ その3

- Boostを使う。
  - とくに標準ライブラリそっくりなやつ
  - 例1: _beginthreadex→boost::thread
  - いつかstd::に書き換える日を夢見て
  - 参考: C++11とBoostの対応付け
    - http://boostjp.github.io/tips/cxx11-boost-mapping.html

C++11っぽさ その3

- Boostを使う。
  - とくに標準ライブラリそっくりなやつ
  - 例2: 定数定義
    - BOOST_STATIC_CONSTEXPR int FOO = 201;

BOOST_STATIC_CONSTEXPR

こんな感じに定義されている

```
#if constexpr使える
#define BSC static constexpr
#else
#define BSC static const
#endif
```

第2章 泥沼

- 2013～2014年
  VC++ 2005を使っていた
- 世間ではC++14がリリースされる頃

C++03がとっさに書けない

- 例: まずこう書く
- ```
  auto it = std::find_if(
        v.begin(), v.end(),
        [a, b](const X& x) {…});
  ```
  ```

C++03がとっさに書けない

- その次にC++03でどう書くか考えていた
- ```
  // TODO: 将来こうする
  //auto it = std::find_if(
  // v.begin(), v.end(),
  // [a, b](const X& x) {…});
  ```
- ```
  std::vector<X>::iterator =
      std::find_if(
          v.begin(), v.end(), …);
  ```

悪魔の囁き

- C++11の機運は高まっている
- なら、その前提でC++11を取り入れても良いのでは

ヘッダーファイルcxx11.h

```
#ifdef _MSC_VER == 1400
namespace std {
        using boost::mutex;
        using boost::lock_guard;

#define noexcept throw()
```

ヘッダーファイルcxx11.h

- 名前空間stdに足を踏み入れた
- 内容のパターン
  - using boost::XXX;
  - 自分で代替実装
    - stoi, to_stringなど

第3章 光

- ついにVisual C++ 2015導入
- 少しずつVC++ 2015に転換している
  - 一部コードはVC++ 2005/2015両対応

バージョン違いの混在環境

```
class Hoge
{
#if _MSC_VER >= 1900
Hoge() = default;
Hoge(Hoge&&) = default;
.....
#endif
};
```

TODOも解消

- 昔書いたこんなコード
- // TODO: 将来こうする
  //auto it = std::find_if(
  // v.begin(), v.end(),
  // [a, b](const X& x) {…});
- std::vector<X>::iterator =
  std::find_if(
  v.begin(), v.end(), …);

TODOも解消

```
#if _MSC_VER >= 1900
auto it = std::find_if(
        v.begin(), v.end(),
        [a, b](const X& x) {…});
#else
std::vector<X>::iterator =
        std::find_if(
                v.begin(), v.end(), …);
```

今考えていること

そろそろC++17が出るんで

```
namespace std {
  using boost::optional;
  using boost::string_view;
  using boost::wstring_view;
}
```

作ろうかなあ

まとめ

- C++03環境で無理ない程度のC++11
  - コンパイラが対応しているものは使う
    - 例: overrideキーワード
  - Boostを使う
    - Thread, Regex, Chrono, ……
  - 奥の手: std名前空間に勝手に定義する
    - Boostから拝借: using boost::…
    - 自前実装: int stoi(…) {}

追加情報

- Visual Studio 2005 サポート終了のお知らせ
  Visual Studio 日本チーム Blog
  https://blogs.msdn.microsoft.com/visualstudio_jpn/2016/03/10/visual-studio-2005/
- Visual Studio 2005は2016年4月11日でサポート終了
  - Visual C++ 2005も含まれる
- お疲れ様でしたVisual Studio 2005