C++14の概要(draft)

高橋晶 (Akira Takahashi)
id:faith_and_brave
@cpp_akira

Boost.勉強会#11 2013/06/01(土)

Boost C++ Libraries

C++14とは

C++11のバグ修正＆マイナーアップデートバージョン。
当初は、C++98に対するC++03程度の修正を行う予定だったが、いろいろと機能追加が行われることになった。

2014年中に策定される予定。すでに新機能の受け入れは締め切られた。
C++14のあと、C++17を目指すC++1y(仮称)も予定されている。

Boost C++ Libraries

コア言語

- 2進数リテラル
- 実行時サイズの配列
- 通常の関数の戻り値型推論
- ジェネリックラムダ
- 一般化されたラムダキャプチャ
- constexpr関数の制限緩和
- 変数テンプレート
- 軽量コンセプト

Boost C++ Libraries

2進数リテラル

0bもしくは0Bプレフィックスを付けることで、
数値の2進数リテラルを記述できるようになる。

```
int x = 0b1100; // x == 12
```

Boost C++ Libraries

実行時サイズの配列

配列の要素数に、実行時の値を指定できるようにするというもの。

```
void f(int n)
{
    int ar[n]; // 要素数nの配列ar
}
```

C99とは「sizeofがとれない」等、細かい部分で非互換。
規定はされないが、スタックからメモリアロケートされる可能性がある。

通常の関数の戻り値型推論

ラムダ式と同様に、通常の関数でもreturn文から戻り値の型を推論できるようにしよう、というもの。

```
auto f(); // 関数f()の宣言。この時点では戻り値の方は「不明」。

auto f() { return 3; } // 関数f()の定義。戻り値の型はint。

int x = f(); // x == 3
```

Boost C++ Libraries

ジェネリックラムダ

ラムダ式のパラメータがジェネリックにできるようになる。

```
vector<X> v = {3, 1, 4};
sort(v.begin(), v.end(), [](const auto& a, const auto& b) {
    return a < b;
});
```

ラムダ式のパラメータをジェネリックにしたい場合は、パラメータの型をautoにする。以下のような関数オブジェクトが作られる：

```
struct F {
  template <class T, class U>
  bool operator()(const T& a, const U& b) const
    { return a < b; }
};
```

Boost C++ Libraries 一般化されたラムダキャプチャ

ラムダ式のキャプチャの際に、ひとつの変数に複数のキャプチャ方法を指定できるようになる。

```
int x = 3;

// xをコピーキャプチャした変数y、xを参照キャプチャした変数z
auto f = [y = x, &z = x] { … }
```

これは、ムーブキャプチャの代替として使用できる。

```
promise<int> p;
future<int> f = p.get_future();

thread t([p = move(p)] { p.set_value(3); });
…
```

constexpr関数の制限緩和

- if文、switch文による条件分岐の許可。
- for文、while文、do-while文によるループの許可。
- void戻り値型の許可
  - パラメータの参照で書き換える
- 初期化を伴う変数宣言の許可
  - static、thread_localは除く
- 変数書き換えの許可。

```
constexpr int abs(int x)
{
    if (x < 0) // OK : if文による条件分岐
        x = -x;  // OK : 変数書き換え
    return x;
}
```

Boost C++ Libraries

変数テンプレート

変数定義にテンプレートを使用できるようにする。
特殊化可能。

```
template <class T> // 円周率
constexpr T pi = T(3.14159265358979323385);

template <class T> // 円の面積
T circular_area(T r) {
    return pi<T> * r * r;
}
```

Boost C++ Libraries

軽量コンセプト

C++11で入らなかったコンセプトの軽量版。
テンプレートの型制約機能。

```
// <演算子を持っている、という制約の定義
template <class T>
constexpr bool LessThanComparable()
  { return has_less<T>::value; }
```

```
// 制約テンプレート
template <LessThanComparable T>
T min(T a, T b)
  { return a < b ? a : b; }
```

制約はconstexpr述語関数として記述する。
制約によるオーバーロードも可能。

Boost C++ Libraries

ライブラリ

- make_unique()
- exchange()
- コンパイル時整数シーケンス
- tupleの型指定get()
- quotedマニピュレータ
- ユーザー定義リテラルライブラリ
- Type Traitsのエイリアステンプレート版
- optional型
- 実行時サイズの配列
- 共有ミューテックス
- ファイルシステム
- ネットワークライブラリの基本的な機能

Boost C++ Libraries

make_unique()

std::unique_ptrのヘルパ関数。

```
unique_ptr<X> p = make_unique<X>(ctor_args...);
```

make_unique()の引数には、対象となる型の
コンストラクタ引数を渡す。

Boost C++ Libraries

exchange()

第1引数の値を第2引数の値で置き換えて、変更前の値を返す関数。

```
// vectorの要素をカンマ区切りで出力する
template <class T>
void print(const vector<T>& v) {
    bool first = true;
    cout << '{';
    for (const T& x : v) {
        if (!exchange(first, false)) { // 最初の1回だけカンマ出力しない
            cout << ',';
        }
        cout << x;
    }
    cout << '}';
}
```

コンパイル時整数シーケンス

主にtupleの展開のために使用する、整数のシーケンス

```
template <std::size_t...> struct index_sequence {};
```

```
// タプルを展開して関数の引数として渡す
template<typename F, typename Tuple, size_t... I>
auto apply_(F&& f, Tuple&& args, index_sequence<I...>)
{ return forward<F>(f)(get<I>(forward<Tuple>(args))...); }

template<typename F, typename Tuple,
     typename Indices = make_index_sequence<tuple_size<Tuple>::value>>
auto apply(F&& f, Tuple&& args)
{ return apply_(forward<F>(f), forward<Tuple>(args), Indices()); }
```

```
apply([](int, char, double) {}, make_tuple(3, 'a', 1.23));
```

Boost C++ Libraries

tupleの型指定get()

tupleを型の集合と見なし、N番目ではなく、指定した型の要素を取得する

```
tuple<int, char, double> t(1, 'a', 1.23);
char& c = get<char>(t); // c == 'a'
```

存在しない型を指定した場合はコンパイルエラーになる。

quotedマニピュレータ

文字列中のエスケープ文字を出力に含めるための、マニピュレータ。CSVやXMLといったフォーマットで必要になる。

```
std::cout << "She said ¥"Hi!¥"" << std::endl;
std::cout << quoted("She said ¥"Hi!¥"") << std::endl;
```

```
She said "Hi!"
"She said ¥"Hi!¥""
```

Boost.Iostreams由来。

Boost C++ Libraries ユーザー定義リテラルライブラリ

C++11で入った、リテラルに対するサフィックスを定義する機能を使用した、リテラルの型付けライブラリ

```
// 文字列
auto s = "hello"s;     // "hello"sはstring型リテラル
auto s = U"hello"s; // u32"hello"sはu32string型リテラル

// 時間
auto m = 3ms; // 3msはmilliseconds型
```

考えられている名前空間：

```
namespace std {
inline namespace literals {
inline namespace chrono_literals {
  constexpr chrono::milliseconds operator""ms(unsigned long long);
}}}
```

std::ms or std::literals::ms or std::chrono_literals::ms

Type Traitsのエイリアステンプレート版

C++11のType Traitsライブラリに対する、エイリアステンプレートのラッパーを定義する。

```
template <class T>
using remove_const_t = typename remove_const<T>::type;
```

名前に_tサフィックスが付いているのと、::typeを書く必要がなくなるのが特徴。

```
using result = remove_const_t<const int>; // result == int
```

エイリアステンプレートの制限により、特殊化はできない。

Boost C++ Libraries

optional型

有効値と無効値の統一的な表現のための、optionalクラスを定義する。boost::optional由来。

```
optional<int> a = 3;        // 有効値3を保持する
optional<int> b = nullopt; // 無効値を保持する

if (a) { // 有効値か否かを判定
    int x = a.value(); // 値を取り出す
}

optional<X> c {in_place, "res1"}; // Xのコンストラクタ引数から有効値を構築
c.emplace("res1"); // Xのコンストラクタ引数で再初期化
```

Boost C++ Libraries

実行時サイズの配列

実行時の要素数を持つ配列、dynarrayクラスを定義する。

```
void f(int n)
{
    dynarray<int> ar(n); // 要素数nのint型配列

    // 要素数の取得
    size_t size = ar.size();

    // イテレータインタフェースによる操作
    for_each(ar.begin(), ar.end(), g);
}
```

Boost C++ Libraries

共有ミューテックス

multiple-reader / single-writerなミューテックスである shared_mutexクラスを定義する。

```
shared_mutex mtx;
void reader()
{
    shared_lock<shared_mutex> lock(mtx); // mtx.lock_shared()

    // ... 共有データへの読み込みアクセス ...
} // mtx.unlock_shared()

void writer()
{
    lock_guard<shared_mutex> lock(mtx); // mtx.lock()

    // ... 共有データへの書き込みアクセス ...
} // mtx.unlock()
```

Boost C++ Libraries

ファイルシステム

ファイル属性、パス、ディレクトリのサポート。

Boost.Filesystem V3由来 + Chrono。

```
string s = read_utf8_data();
path p = u8path(s); // UTF-8のパス
create_directory(p); // ディレクトリ作成

// ファイルコピー
copy_file(path("a.txt"), path("b.txt"));

// 最終更新日時を取得(chrono::time_point)
file_time_type time = last_write_time(path("a.txt"));
```

Boost C++ Libraries ネットワークライブラリの基本的な機能

C++14段階では、ネットワークバイトオーダーの変換機能のみ。

```
// ホストがリトルエンディアン、
// ネットワークがビッグエンディアンの場合

// xはDD, CC, BB, AAのバイトオーダーで並ぶ
uint32_t x = 0xAABBCCDD;

// resultはAA, BB, CC, DDのバイトオーダーで並ぶ
uint32_t result = htonl(x); // host to network long
```

- ホストからネットワークへの変換
  - htonl()とhtons()、およびテンプレート版のhton()。
- ネットワークからホストへの変換
  - ntohl()とntohs()、およびテンプレート版のntoh()。

Boost C++ Libraries

C++14に現状入らないもの

C++1yやC++22に回されるかもしれない機能

- モジュールシステム
- トランザクショナル・メモリ
- Rangeライブラリ
- 並行ライブラリの強化
  - 並行アルゴリズム
  - 並行データ構造(キュー)
  - パイプライン
  - ラッチ
- 非同期操作の強化
  - async/await
- コルーチン
- SIMD
- 多倍長整数
- 文字列のsplit/join、検索アルゴリズム
- 整数リテラルの区切り(int a = 123_456;)
- any、variant

Boost C++ Libraries

その他の状況と議論場所

最後に、C++14やC++1yについての議論場所について話します。

- 標準C++の議論に参加するにはどうすればいいか。
- 最新のC++情報はどこで手に入るのか。

C++11までの議論グループ

C++11までは、以下の4グループに分かれて議論が行われていた:

- コア言語(CWG, Mike Miller)
- 進化(EWG, Bjarne Stroustrup)
- ライブラリ(LWG, Alisdair Meredith)
- ライブラリ進化(LWEG, Beman Dawes)

http://isocpp.org/std/the-committee

Boost C++ Libraries

Study Group

C++14/C++1y以降では、さらに細分化した専門家グループ(Study Group)で議論が行われている:

- SG1 並行・並列(Hans Boehm)
- SG2 モジュール(Doug Gregor)
- SG3 ファイルシステム(Beman Dawes)
- SG4 ネットワーク(Kyle Kloepper)
- SG5 トランザクショナルメモリ(Michael Wong)
- SG6 数値演算 (Lawrence Crowl)
- SG7 リフレクション (Chandler Carruth)
- SG8 コンセプト (Matt Austern)
- SG9 Range (Marshall Clow)
- SG10 機能テスト (Clark Nelson)
- SG11 データベース(Bill Seymour)
- SG12 未定義の振る舞い(Gabriel Dos Reis)

いくつかのStudy Groupは公開されていて、誰でも議論に参加できる

Boost C++ Libraries

標準C++関係の議論場所

- ISO C++ Standard – Discussion
  - https://groups.google.com/a/isocpp.org/forum/#!forum/std-discussion
- ISO C++ Standard - Future Proposals
  - https://groups.google.com/a/isocpp.org/forum/#!forum/std-proposals
- SG5 - Transactional Memory
  - https://groups.google.com/a/isocpp.org/forum/?fromgroups#!forum/tm
- SG8 – Concepts
  - https://groups.google.com/a/isocpp.org/forum/?fromgroups#!forum/concepts
- SG9 – Ranges
  - http://www.open-std.org/mailman/listinfo/ranges
- SG10 - Feature test macros
  - http://www.open-std.org/mailman/listinfo/features

Boost C++ Libraries

Standard C++ Foundation

- 標準C++の情報を発信し、開発者コミュニティをサポートしていくための、非営利団体(NPO)
- http://isocpp.org/
- C++に関する最新の情報は、ここで入手できる。
- イベント、開発ツール、ライブラリ、記事や書籍の情報など。

Boost C++ Libraries

ここから始めよう！

- isocpp.orgのRSSを購読しよう！
  - 最新C++のまとめ情報が手に入ります
- std-proposalsのメーリングリストに参加しよう！
  - 誰でも気軽に「こんな機能があったらいいよね」というところから新機能の展望を話し合えます
  - 日本人もけっこういます
- **英語ってむずかしい！**
  - お手伝いしますので、お気軽にご相談ください。
  - それでがんばって英語で投稿して、実際に取り入れられたこともあります

Boost C++ Libraries

まとめ

- C++14はC++11のマイナーアップデートだが、意外と便利な機能がたくさん入る
- ここで話した内容は、C++14が正式に決まるまでに機能追加・削除、変更される可能性がある
- C++14の次に、C++1y(17?)も予定されている
- 最近のC++は、誰でも議論に参加できる
- 英語むずかしいけど、がんばったらそれなりの成果は期待できます。一緒にがんばろう！

Boost C++ Libraries

この資料の情報元

- Trip Report: ISO C++ Spring 2013 Meeting
  - http://isocpp.org/blog/2013/04/trip-report-iso-c-spring-2013-meeting
- New paper: Bristol minutes—Kyle Kloepper
  - http://isocpp.org/blog/2013/05/new-paper-bristol-minutes-kyle-kloepper
- New paper: N3690, Programming Languages—C++, Committee Draft
  - http://isocpp.org/blog/2013/05/new-paper-n3690-programming-languages-c-committee-draft
  - C++14仕様のベータ版みたいなもの。
- New paper: N3692, C++ Editor's Report, May 2013—Stefanus Du Toit
  - http://isocpp.org/blog/2013/05/new-paper-n3692-c-editors-report-may-2013-stefanus-du-toit
  - 変更点リスト