

C++11概要 ライブラリ編



H.24/05/26

Egtra

Boost.勉強会 #9 つくば



注意

- **網羅はしていません**
 - **規格を（N3337.pdfも可）を読む**
 - **cpprefjpを書く・読む**

Misc



スマートポインタ

- **unique_ptr**
 - 以下の上位互換
 - `std::auto_ptr`
 - `boost::scoped_ptr`,
`boost::scoped_array`
- **shared_ptr**
 - `boost::shared_ptr` とほぼ同じ
 - 注意 : `shared_array` 版はなし

スマートポインタ

- `unique_ptr<int> up(new int(1));`
- `unique_ptr<int[]> ua(
new int[]{1, 2, 3});`
- `auto sp = std::make_shared<int>(3);`

関数オブジェクト

- `std::function`
- `std::ref`, `std::bind`, `std::mem_fn`
 - ラムダ式があるので`bind`不要？

整数型 (C99)

- `<stdint.h>`
- `int8_t, uint8_t, …… (16, 32, 64)`
- `intptr_t, uintptr_t`
- その他諸々

コンテナ



新コンテナ

- **unordered_***
 - **unordered_map<>,**
 - **ハッシュマップ**
- **array<>**
 - **固定長配列 (boost::array風)**
- **forward_list<>**
 - **片方向リンクリスト**

コンテナの初期化

- 配列の初期化
- `int ra[] = {1, 2, 3};`

コンテナの初期化

- 配列の初期化、コンテナでも
- `int ra[] = {1, 2, 3};`
- `std::array<int, 3> a = {1, 2, 3};`
- `std::vector<int> v = {1, 2, 3};`

コンテナへの代入

- `a = {1, 2, 3};`
- `v = {1, 2, 3};`

With 構造体

```
struct Point {  
    double x;  
    double y;  
};
```

With 構造体

```
std::vector<Point> vp = {  
    {10, 0},  
    {0, 20},  
};
```

With 構造体

```
vp.push_back({0, -20});  
vp.insert(v.begin(), {0, 0});
```



With クラス

```
std::vector<std::fstream> vf = {  
    {"a.cpp"},  
    {"a.out", ios_base::binary},  
};
```


With クラス

```
std::vector<std::fstream> vf = {  
    std::fstream("a.cpp"),  
    std::fstream(  
        "a.out", std::ios_base::binary),  
};
```

With コピー不可の型

```
class SanJigen :  
    boost::noncopyable {  
    explicit SanJigen(  
        int x, int y, int z);  
};
```

With コピー不可の型

```
// 図形
```

```
std::vector<SanJigen> figure;
```

```
figure.emplace_back(3, 1, 4);
```

```
→ figure[0] == SanJigen(3, 1, 4)
```

map::at, unordered_map::at

```
std::map<std::string, std::string>  
    const yome = {  
    {"Nyaruko", "Mahiro"},  
    {"Kuko", "Nyaruko"},  
    {"Hasta", "Mahiro"},  
};
```

**map::at,
unordered_map::at**

auto x = yome.at("Hasta");

→ x == "Mahiro"



**map::at,
unordered_map::at**

yome.at("Mahiro");

→.....?



**map::at,
unordered_map::at**

yome.at("Mahiro");

**_人人人人_
> 突然の死 <
—^v^v^v^v—**

(註 : std::out_of_range)

**map::at,
unordered_map::at**

```
auto const yome2 = yome;
```

```
yome2.insert(  
    {"Mahiro", "Shantak-kun"});
```

```
auto y = yome2.at("Mahiro");
```



unordered_mapのキー対応

```
namespace My {  
    struct Point {  
        int x, y;  
    };  
    bool operator==(Point, Point);  
}
```

unordered_mapのキー対応

```
namespace std {  
    struct hash<My::Point> {  
        std::size_t operator()(  
            Point const& pt) const {  
            return .....;  
        }  
    }; // 特殊化  
}
```


basic_string: 要素の連続

```
auto len = GetWindowTextLength(hwnd);
```

```
std::basic_string<TCHAR> t(len + 1);
```

```
GetWindowText(hwnd, &s[0], len + 1);
```

```
s.pop_back(); // !
```



文字列・数値変換

atoi/atol/strtol類の上位互換

```
int stoi(const std::string& str,  
         std::size_t* idx = 0, int base = 10);  
int stoi(const std::wstring& str,  
         std::size_t* idx = 0, int base = 10);
```

文字列・数値変換

```
// atoiっぽく  
auto x = stoi("103");
```

```
// strtolっぽく  
std::size_t pos;  
auto y = stoi("BEEF kue", &pos, 16);
```

文字列・数値変換

- **stoi (int)**
- **stol (long)**
- **stoll (long long)**
- **stoull (unsigned long long)**
- **stof (float)**
- **stod (double)**
- **stold (long double)**

文字列・数値変換

文字列へ変換

```
auto s = std::to_string(201);
```

```
auto ws = std::to_wstring(233.1000);
```


ワイド・ナロー変換

```
std::wstring_convert<
    std::codecvt<wchar_t, char,
                std::mbstate_t>>
cvt(new std::codecvt_byname<
    wchar_t, char, std::mbstate_t>(" "));
```

ワイド・ナロー変換

```
std::wstring araragi =  
    cvt.from_bytes(' A ');  
std::wstring tsukihi =  
    cvt.from_bytes("月火");
```

ワイド・ナロー変換

```
std::string koyomi  
    = cvt.to_bytes(L'暦');  
std::string aryaryagi =  
    cvt.to_bytes(L"アララギ");
```

正規表現

```
std::regex meruado_kamo(".+@.+");  
if (std::regex_match(  
    "hoge@example.jp", meruado_kamo))  
{  
    std::cout << "メルアドかも¥n";  
}
```

メールアドレスの正規表現

```
(?:((?:((?:((?:((?:[¥x20¥x09]*(?:¥x0D¥x0A))?[¥x20¥x09]+|(?:[¥x20¥x09]+(?:¥x0D¥x0A)[¥x20¥x09]+)*))?(¥((?:((?:[¥x20¥x09]*(?:¥x0D¥x0A))?[¥x20¥x09]+|(?:[¥x20¥x09]+(?:¥x0D¥x0A)[¥x20¥x09]+)*))?(?:[¥x21-¥x27¥x2A-¥x5B¥x5D-¥x7E]|[¥x01-¥x08¥x0B¥x0C¥x0E-¥x1F¥x7F]))|(?:¥¥(?:[¥x21-¥x7E]|[¥x20¥x09])|(?:¥¥(?:¥x00|[¥x01-¥x08¥x0B¥x0C¥x0E-¥x1F¥x7F]|¥x0A|¥x0D))))|(?-1)))?(?:[¥x20¥x09]*(?:¥x0D¥x0A))?[¥x20¥x09]+|(?:[¥x20¥x09]+(?:¥x0D¥x0A)[¥x20¥x09]+)*))?(¥((?:((?:[¥x20¥x09]*(?:¥x0D¥x0A))?[¥x20¥x09]+|(?:[¥x20¥x09]+(?:¥x0D¥x0A)[¥x20¥x09]+)*))?(?:[¥x20¥x09]*(?:¥x0D¥x0A))?[¥x20¥x09]+|(?:[¥x20¥x09]+(?:¥x0D¥x0A)[¥x20¥x09]+)*))?(?:[A-Za-z0-9!#$%&'*+¥-/=?^_`{|}~)+(?:¥.[A-Za-z0-9!#$%&'*+¥-/=?^_`{|}~]+)*(?:((?:((?:[¥x20¥x09]*(?:¥x0D¥x0A))?[¥x20¥x09]+|(?:[¥x20¥x09]+(?:¥x0D¥x0A)[¥x20¥x09]+)*))?(¥((?:((?:[¥x20¥x09]*(?:¥x0D¥x0A))?[¥x20¥x09]+|(?:[¥x20¥x09]+(?:¥x0D¥x0A)[¥x20¥x09]+)*))?(?:[¥x21-¥x27¥x2A-¥x5B¥x5D-¥x7E]|[¥x01-¥x08¥x0B¥x0C¥x0E-¥x1F¥x7F]))|(?:¥¥(?:[¥x21-¥x7E]|[¥x20¥x09])|(?:¥¥(?:¥x00|[¥x01-¥x08¥x0B¥x0C¥x0E-¥x1F¥x7F]|¥x0A|¥x0D))))|(?-1)))?(?:[¥x20¥x09]*(?:¥x0D¥x0A))?[¥x20¥x09]+|(?:[¥x20¥x09]+(?:¥x0D¥x0A)[¥x20¥x09]+)*))
```


正規表現

```
std::regex last_part(  
    "(?:.*/)+([^\/*]*)");
```

```
std::string src = "/usr/bin/cc";
```

```
std::string replace = "$1";
```

```
std::string file = std::regex_replace(  
    src, last_part, replace);
```

日時入出力

```
auto time = std::time(nullptr);  
auto tm = std::localtime(&time);  
  
std::cout.imbue(std::locale(""));  
std::cout <<  
    std::put_time(tm, "%c") << std::endl;
```


日時入出力（それBoostで）

```
ptime pt = second_clock::local_time();  
  
std::locale loc(std::locale(""),  
    new time_facet<ptime, char>("%c"));  
std::cout.imbue(loc);  
std::cout << pt << std::endl;
```

並行處理關係



Atomic演算 (Windows)

```
long x;
```

```
InterlockedIncrement(&x);
```

```
InterlockedDecrement(&x);
```

```
auto old =
```

```
    InterlockedCompareExchange(  
        &x, newValue, comparand);
```

Atomic演算

```
std::atomic<int> y;
```

```
y++;
```

```
y--;
```

```
int old = newValue;
```

```
bool b = x.compare_exchange_strong(  
    &old, comparand);
```

非同期実行（スレッド）

```
int hoge(  
    std::string const& arg1, int arg2);
```

```
std::future<int> f = std::async(  
    std::launch::async, hoge, "rofi", 3);
```

.....

```
int result = f.get(); // 待機する
```



非同期実行（現Boost風）

```
void g(.....);  
std::thread th(g, .....);
```

```
th.join(); // 待機する
```

This work is licensed under a Creative Commons Attribution-ShareAlike 2.1 Japan License.

