Faith and Brave - C++で遊ぼう

<[C++] deprecated属性がほしい | [C++] Boostのロゴ>

プロフィール

faith_and_brave

アキラ. C++プログラマです.「プログラムは美しく」がモットー

日記の検索

詳細 一覧

カレンダー

<<	2010/02					>>
日	月	火	水	木	金	土
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28						

C++0x基礎文法最速マスター - Faith and Brave - C++で遊ぼう 118users

C++0xの言語拡張まとめ(※随時更新) - Faith and Brave - C++で遊ぼ... 99users

『C++テンプレートテクニック』もうすぐ発売です。 - Faith a... 40users

Boost.勉強会のまとめとか感想とか今後についてとか - Faith a... 38users

constメンバ関数は重要 - Faith and Brave - C++で遊ぼう 37users

Visual C++ の今後 - Faith and Brave - C++で遊ぼう 32users

Faith and Brave - C++で遊ぼう 28users

2010-02-01

■[C++] C++0x基礎文法最速マスター ★15★

C++0xになると、C++03でごちゃごちゃした部分がだいぶすっきり書けるようになるので、C++0xでの入門はこんな感じになるよー、という気持ちで書きました。

1. Hello World

C++0xでの入出力には、IOStreamというものを使用します。

<<演算子でどんどんつないでいきます。

以下のプログラムの読み方は

「標準出力(cout)に"Hello World"という文字列と、改行(endl)を出力する」

です。

```
#include <iostream>

int main()
{
    std::cout << "Hello World" << std::endl;
    return 0;
}
```

```
Hello World
```

coutとendlを使用するには、

```
#include <iostream>
```

のように、<iostream>という標準ライブラリのファイルを読み込む必要があります。

2. コメント

行コメント

```
std::cout << "Hello"; // これはコメント
```

ブロックコメント

```
/*
これは
  コメント
    です
*/
std::cout << "Hello";
```

3. 変数

C++0xでは「const 型 変数名 = 値」のようにして変数を宣言します。

constを付けない場合、変数の値を変更でき、初期値も省略できますが、

バグの温床となりえるので、基本的にconstを付けることをオススメします。

```
#include <string>
#include <vector>

int main()
{
    const int              a = 3;          // 整数型(int)の変数(a)を宣言し、3という値を割り当てる
    const std::string      s = "abc";      // こちらは文字列型(std::string)
    const std::vector<int> v = {1, 2, 3}; // リスト(std::vector)。ここでは整数型(int)を格納できる

    return 0;
}
```

4. 関数

雑記
C++
VC++
GCC
Comeau
C#
LISP
D言語
Scala
Haskell
Oz
SlickEdit
Objective-C++
OpenGL
まとめ

関数は処理に名前を付けるために使用します。

基本的な文法は2種類あり、戻り値の型を前置するものと後置するものがあります。

これはスタイルで使い分けてください。

```
// 戻り値の型を前置する構文
// 「戻り値の型 関数名(パラメータの型 パラメータ名)」
int square(int x)
{
    return x * x;
}
```

```
// 戻り値の型を後置する構文
// 「auto 関数名(パラメータの型 パラメータ名) -> 戻り値の型」
auto square(int x) -> int
{
    return x * x;
}
```

呼び出し側:

```
const int s = square(3); // s == 9
```

戻り値で何も返さない場合は、voidという特別な型が必要になります。

入出力などで使用します。

```
void disp(const std::string& s)
{
    std::cout << s << std::endl;
}

disp("Hello World");
```

5. 型

C++0xには以下のような型が用意されています(一部です)。

```
基本データ型
int    : 整数型(0, 1, 2, 3, -1, etc...)
char   : 文字型('a', 'b', 'c', '1', etc...)
double : 浮動小数点数型(0.1, 3.14, etc...)
bool   : 論理型(true, false)

複合データ型
```

Life like a clown - はてな的プログラミング言語人気ランキング

2010-02-01

きまぐれメモ - 各種言語による基礎文法最速マスターまとめ

2010-02-01

なんとなく日記 - JavaScript基礎文法最速マスター

リンク集

C++ Final Draft International Standard

C++ Standards Committee Papers

C++ Reference

東方算程譚

USK's Dialy

melpon日記 - C++すらまともに扱えないへたれのページ

melt日記

Cry's Diary

いろきゅう.jp

本の虫

p_stade

ときどきの雑記帖 i戦士篇

わんくま同盟

囚人のジレンマな日々

神様なんて信じない僕らのために

TrickDiary

じょ～だんの日記

かそくそうち

口の中にご飯が入ってるのにすごくクシャミがしたくなった時の焦り日記

HILOG on Hatena

予定は未定Blog版

```
std::array<T, N>      : 配列(<array>をインクルード)
std::vector<T>        : リスト(<vector>をインクルード)
std::string           : 文字列(<string>をインクルード)
std::map<Key, Value>  : 辞書(<map>をインクルード)
std::set<T>           : 集合(<set>をインクルード)
```

6. 基本演算

四則演算

```
const int num = 3 + 2; // num == 5
const int num = 3 - 2; // num == 1
const int num = 3 * 2; // num == 6
const int num = 5 / 2; // num == 2
const int num = 5 % 2; // num == 1
```

論理演算

```
const int a = 1;
const int b = 2;

const bool r = a == b;
const bool r = a != b;
const bool r = a < b;
const bool r = a > b;
const bool r = a <= b;
const bool r = a >= b;

const bool p = true;
const bool q = false;

const bool r = p && q;
const bool r = p || q;
```

7. 参照

参照は変数に別名を付けます。

```
int  a = 1;
int& b = a; // bはaを参照する

b = 2; // bの参照先であるaを2に書き換える
```

関数の戻り値が複数ある場合などに使用します。

```
void get_ip(int& aa, int& bb, int& cc, int& dd)
```

```
{
    aa = 127;
    bb = 0;
    cc = 0;
    dd = 1;
}

int a = 0;
int b = 0;
int c = 0;
int d = 0;
get_ip(a, b, c, d);
  // a == 127
  // b == 0
  // c == 0
  // d == 1
```

8. 制御構文

if/else文、もしくは条件演算子を使用することで条件分岐できます。

```
// if/else文を使用した場合
int abs(int x)
{
    if (x >= 0) {
        return x;
    }
    else {
        return -x;
    }
}
```

```
// 条件演算子を使用した場合
int abs(int x)
{
    return x >= 0 ? x : -x;
}
```

switch文は、値と処理の対応表として使用できます。

```
int next_scene(int scene)
{
    switch (scene)
    {
        case title: return menu;
        case menu:  return game;
        case game:  return gameover;
    }
```

```
    throw std::invalid_argument("不正なシーン");
}
```

while文は、条件を満たす間ループするための構文です。

```
int retry = 0;
while (!connect()) {
    std::cout << "まだ接続できない" << std::endl;
    ++retry;
}
```

for文は、while文をより書きやすくした構文です。

```
for (int retry = 0; !connect(); ++retry) {
    std::cout << "まだ接続できない" << std::endl;
}
```

さらに、データ構造を1要素ずつ処理するための範囲for文というものもあります。

```
const std::vector<int> v = {1, 2, 3};
for (const int x : v) {
    std::cout << x << std::endl;
}
```

9. テンプレート

std::vector<int>などですでにひっそり出てきている<int>です。

あらゆる型を格納するデータ構造や

あらゆる型で振舞うアルゴリズムを作るときに

型のプレースホルダーとしてテンプレートというものを使用します。

```
template <class T>
void disp(T x)
{
    std::cout << x << std::endl;
}
```

int型の値/変数を渡した場合、もしくは明示的にintを指定した場合

```
disp(3);
disp<int>(3);
```

disp関数のTはintに置き換えられます。

```
void disp(int x)
{
    std::cout << x << std::endl;
}
```

double型の場合も同様です。

```
disp(3.14);
disp<double>(3.14);
```

```
void disp(double x)
{
    std::cout << x << std::endl;
}
```

10. リスト操作

```
#include <vector>

const std::vector<int> v = {1, 2, 3};

const int front  = v.front(); // front  == 1 : 先頭要素を取得
const int back   = v.back();  // back   == 3 : 最後尾要素を取得
const int second = v[1];      // second == 2 : 添字を指定したランダムアクセス

// 全ての要素を出力
for (const int x : v) {
    std::cout << x << std::endl;
}


// 破壊的な操作 : constにできない
std::vector<int> v = {1, 2, 3};

v.push_back(4);       // v == {1, 2, 3, 4} 末尾に要素を追加する
v.pop_back();         // v == {1, 2, 3} 末尾の要素を削除
v.assign({4, 5, 6}); // v == {4, 5, 6} 要素の再割り当て
```

11. 連想配列の操作

```
#include <map>
#include <string>
```

```
const std::map<std::string, int> m = {
    {"Akira",  24}, // {キー, 値}
    {"Millia", 16},
    {"Johnny", 38}
};

const int age = m.at("Akira"); // age == 24


// 破壊的な操作
std::map<std::string, int> m = {
    {"Akira",  24},
    {"Millia", 16},
    {"Johnny", 38}
};

m["Sol"] = 150; // 要素の追加。キーがすでにあったら値を書き換え
```

12. アルゴリズム

データ構造を操作するアルゴリズムが多数用意されています。

多くのアルゴリズムは破壊的です。

```
#include <iostream>
#include <vector>
#include <algorithm>

void disp(int x)
{
    std::cout << x << std::endl;
}

int main()
{
    const std::vector<int> v = {1, 2, 3};

    // for_each : 指定された範囲の要素全てに関数を適用する
    std::for_each(v.begin(), v.end(), &disp);

    // find : 範囲内の指定された値を検索する
    decltype(v)::const_iterator it = std::find(v.begin(),
v.end(), 2);
    if (it != v.end()) {
        std::cout << "見つかった：" << *it << std::endl;
    }
    else {
        std::cout << "見つからなかった！" << std::endl;
    }
```

```
    // 破壊的な操作
    std::vector<int> vv = {3, 1, 4};

    // reverse : 範囲を反転する
    std::reverse(vv.begin(), vv.end());

    // sort : 並び替え
    std::sort(vv.begin(), vv.end());

    return 0;
}
```

13. ラムダ式

アルゴリズムでは、各要素を処理するための関数を作ることが多くなりますが
関数をわざわざ作るのがめんどくさい場合に、ラムダ式というものを使って
その場に関数を作れます。

```
#include <iostream>
#include <vector>
#include <algorithm>

int main()
{
    const std::vector<int> v = {1, 2, 3};
    std::for_each(v.begin(), v.end(), [](int x) { std::cout << x
<< std::endl; });

    return 0;
}
```

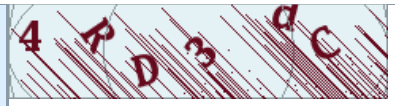
アルゴリズムのあたりはOvenのようなRangeライブラリが入れば
もっとすっきり書けそうですが、C++0xでもだいぶ書きやすくなりましたね。

Permalink | コメント(1) | トラックバック(7) | 13:03 tB 118 users

コメントを書く

hito_hpp 2010/02/01 22:29
デフォルトをconstにしろってのは、確かにいいスタイルかも。

なまえ メール(非公開) URL



画像認証

画像内の文字列を入力して下さい

投稿

トラックバック - http://d.hatena.ne.jp/faith_and_brave/20100201/1264997004

燈明日記 - 基礎文法最速マスターぞくぞくキターーー！
(rubikitch loves (Emacs Ruby CUI)) - Emacs Lisp基礎文法最速マス...
どうでもいい情報置き場 - Whitespace基礎文法最速マスター
なんとなく日記 - JavaScript基礎文法最速マスター
きまぐれメモ - 各種言語による基礎文法最速マスターまとめ
Life like a clown - はてな的プログラミング言語人気ランキング
なんとなく日記 - 基礎文法最速マスターシリーズのまとめ

リンク元

196 http://d.hatena.ne.jp/
181 http://b.hatena.ne.jp/hotentry
139 http://b.hatena.ne.jp/hotentry/it
74 http://reader.livedoor.com/reader/
56 http://twitter.com/
47 http://www.google.com/reader/view/
40 http://www.google.co.jp/reader/view/
38 http://news.atode.cc/
37 http://d.hatena.ne.jp/rubikitch/20100201/elispsyntax
37 http://pipes.yahoo.com/pipes/pipe.info?_id=faa858a20082ef6d25ad27557e37e011

おとなり日記

2010-02-01 while(c++); 6/30 20%
2010-02-01 IKB: 雑記帖 7/45 15%
2010-02-01 Life like a clown 5/49 10%
2010-02-01 Selene日記 5/61 8%
2010-02-02 プログラミング言語を作る日記 7/90 7%
2010-02-01 はてなかよっ！ 9/136 6%
2010-02-01 notes plastiques 6/94 6%
2010-01-31 何かしらの言語による記述を解析する日記 6/106 5%

<[C++] deprecated属性がほしい | [C++] Boostのロゴ>