プログラミング勉強会 – アルゴリズム編

第1回　C++の基本～STL

2007/05/16
@ろまんす

C++とは

- C言語を元に作られた言語
- ビャーネ・ストロヴストルップが設計
- C言語に対しての上位互換あり
- 文法はCとほぼ同じ
- 新しいバージョン（C++0x）が2010年までに

- コンパイル方法
  - 学校 ： g++ test.cpp -> ./a.out など
  - Visual Studio ： 今までと一緒

C++によるHello,World

- Cpp01.cpp
  - using namespace std;
  - cout と cin
  - C言語で使っていた関数も混ぜれる
  - 拡張子は「.cpp」
- Cpp02.cpp
  - 変数を宣言する場所に制限が無い
    - 必要なところのすぐ近くで宣言が出来る

Cとの違い

- Cpp03.cpp
  - 構造体の中で関数を宣言できる
    - 変数と一緒で「.」でアクセスできる
    - 直感的に使用できる
- Cpp04.cpp
  - アクセス制御
  - こんな事をする理由
    - 変数をいらない所まで見せなくてすむ
    - 変なところで書き換えられてバグが入る事が無くなる

クラス

- Cpp05.cpp
  - 構造体とほぼ同じなので簡単
  - 宣言を struct から class に変えればOK
  - 違いはアクセス指定のデフォルトだけ
    - 構造体：デフォルトでpublic（外から見れる）
    - クラス：デフォルトでprivate（外には見せない）

- Cpp06.cpp
  - 例

ここまでで質問、
確認したい事、
分からない事など

STL

- Standard Template Library の略
- 高機能で汎用的な標準ライブラリ
- 便利で簡単に使える
- バグが減る

#include <vector>

- 自動で大きさを管理してくれる配列

- 使い方（vec.cpp）
  - vector<型> で宣言
  - push_back(何か) で追加
  - サイズを知りたかったら size()
  - []でアクセス可能
  - clear() で初期化

#include <string>

- 文字列を扱える
  - 従来のcharの配列では大きさに制限があった
  - もっと自由に扱える＋便利な関数もついてる
- 使い方（str.cpp）
  - []でアクセス可能
  - += や + で連結可能
  - 文字を探せる
  - 入力の文字数が分からなくてもOK

#include <algorithm>

- C++ではソートや探索などのよく使われるアルゴリズムは関数として用意されています。

- 使う利点
  - 開発スピードが上がる
  - バグが無い
  - 動作が高速
  - コードが読みやすくなる

sort, stable_sort (Sort.cpp)

- 配列の要素を順番に並べる
- ゲームにおける使用例
  - 得点で並び替えてランキング表を作りたい
  - プレイヤー名をあいうえお順に並べたい
  - アイテムを名前順、値段順に並べたい　など
- 構造体などで使いたい時はoperator>を宣言しておく(SortEx.cpp)
  → 「>」を定義しておくとsetやmapで構造体が使える
  → もしくは比較関数を用意する

swap（Swap.cpp）

- 2つの値を入れ替える
- ゲームプログラミングに限らずよく使う
- swap(A, B)とする
  - AとBは比較できる同じ型であればなんでもOK

```
int tmp = a;
a = b;
b = tmp;
```

```
swap(a, b);
```

max, min (Min_max_element.cpp)

- 2つの値のうち大きいの、小さいのを返す
- これもなんにでも使える

```
if (a <= b) return a;
else return b;
```

とか

```
return ((a<=b)?a:b);
```

```
max (a, b);
```

max_element, min_element

- 配列やvector、stringなどがあったとき、その要素の中で最大（最小）の物を返す。
- ちなみにアドレスが帰ってくる。

その他（Other.cpp / OtherEx.cpp）

- find
  - 要素を探す
- count
  - 配列の中にxが何個あるか数える
- random_shuffle
  - 配列をランダムに混ぜる
- reverse, rotate
  - 配列の反転、シフト

その他

- replace
  - 値を置き換える
- fill
  - 値をつめる
- next_permutation
  - 次の順列を生成
- for_each
  - 各要素を引数として指定された関数を呼び出す

```
#include <queue>
#include <stack>
#include <set>
```

- 次回やる予定

STLについて何か質問、
確認したいこと、
もう一度聞きたい所など

C++/STLの話は終わり

C++の話だけというのもつまらないので。

最後に1つ面白いアルゴリズムを紹介

ポーカーの役判定

- 役
  - ワンペア、ツーペア
  - スリーカード、フォーカード
  - フルハウス
  - ストレート、フラッシュ
- これらを素直に判定しようとすると結構複雑になってしまう。

ソース（Poker.cpp）

```
// コアになる部分
int res = 0;
for (int i=0; i < 5; ++i) {
    for (int j=i+1; j < 5; ++j) {
        if (hand[i]==hand[j]) ++res;
    }
}
int hMax = *max_element(&hand[0], &hand[5]); // 最大要素
int hMin = *min_element(&hand[0], &hand[5]); // 最小要素
int dis =  hMax - hMin; // 最大と最小の差
// 結果
if (res==0 && dis==4) printf(" (res=%d: ストレート)¥n", res);
else printf(" (res=%d: %s)¥n", res, pat[res]);
```

何をやっているのか

- 手札を左から見ていく
- それより右のトランプを1つずつ見ていき、同じ数字を見つければ、++res
- 結果のresの値によって役が一意に分かる

役とresの対応

- res =
  - 0 → 役なし
  - 1 → ワンペア
  - 2 → ツーペア
  - 3 → スリーカード
  - 4 → フルハウス
  - 5 → ありえない
  - 6 → 4カード

ストレートは

res = 0 かつ

手札の数字の最大と
最小の差が4

であれば
ストレートと判定

フラッシュ

- フラッシュに関しては全ての記号が一緒かを調べるだけでOK
- よってストレートフラッシュも簡単。
- ロイヤルストレートフラッシュは一つ一つ見ていくしかない。

全体を通して何か質問があれば

次回予告

- 次回はもっとゲーム作りに直結するアルゴリズムを紹介

- 例えば、
  - ぷよぷよのくっつき具合判定
  - テトリスのピースの回転
  - ある位置間の最短経路　　などを予定

※このゲームのこの部分ってどうやって実装してるの？とか知りたいのがあれば教えてください。

今日伝えたかったこと

- C++はCに似ている
- STLという便利なライブラリの存在
- ポーカーの役判定は簡単

お疲れさまでした