第２６回世界コンピュータ将棋選手権

きふわらべ
アピール文書

２０１６年０５月１２日　高橋智史

大会を終えて☆

DARE

誰なんだぜ☆？

開発者

高橋　智史

第９回UEC杯コンピュータ囲碁大会（３月１９日）　３０位☆　（３２チーム中）
C++言語の取り扱いに自信をつけて
第２６回コンピュータ将棋選手権（５月３日）　に望んだんだぜ☆

北白河ちゆり
／TOHO PROJECT
FANMADE　※

「 きふわらべは作ったものの、
コンピュータ将棋開発者です、と言うには　コンピュータ将棋の知識不足だよな☆
もっと　他人のライブラリを読んで　将棋プログラムを　覚えようぜ☆？」

そして　大会の１か月前。

「 C++の template がわからないぜ……☆
１個所いじれば　コンパイル・エラーになるぜ☆
ほんとに　C++言語　は、使われている言語なのかだぜ☆？」

コンピュータ将棋エンジン
きふわらべ

「 お父んを含めて、４１チームがC++言語を使っているようだぜ☆？
４１人も　１個所いじれば　コンパイル・エラーになっているとは　驚きだぜ☆」

「 ちょっと、１次予選　きふわらべ　ちゃんが　優勝候補の技巧と
１手差　の大善戦じゃないの！
どんな　いたずら　を加えたのよ！」

岡崎夢美
／TOHO PROJECT
FANMADE　※

※北白河ちゆり、岡崎夢美は　東方夢時空　の登場キャラクター／（Ｃ）上海アリス幻樂団　様の著作物です。

エイプリー＿大樹の枝　ライブラリー

APERY CHILDREN

エイプリー・チルドレンなんだぜ☆

5月3日　1次予選　1回戦　▲きふわらべ　ＶＳ　△技巧

http://live4.computer-shogi.org/wcsc26/kifu/WCSC26_L1_KFW_GIK.html

「まあ、ギコちゃん、１手差とか気にせず
ずっと　勝ってる　と思ってたけどな☆」

「１００点ぐらい、きふわらべがいい……
と、技巧に　言わしめた瞬間があったんだぜ☆！」

「先手のアドバンテージの１００点よ！」

エイプリー＿大樹の枝　と　きふわらべ　の違いは☆

JIKAN
よく時間をつかって　考えて　指そうぜ☆

「早指しにならないよう、
１手に　１７秒～２５秒ほど　考えるように　時間制御　を改造したのが
対局で確認できる　**変化**　だぜ☆」

「なんで　そんな改造をしたの？」

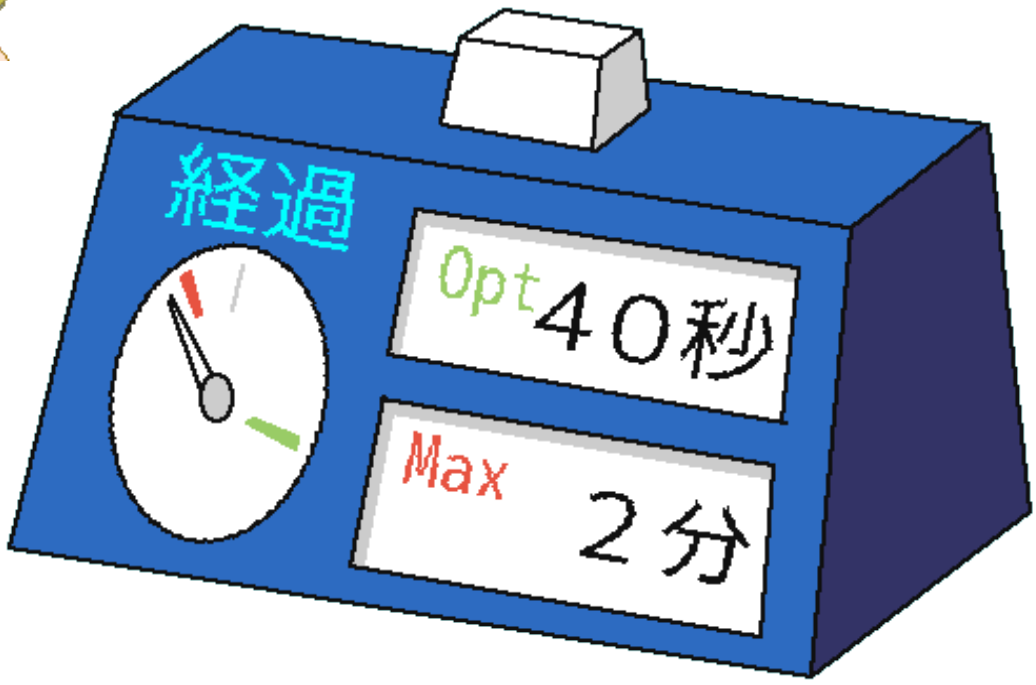
「**時間を　３分も残したまま**
負けてしまう、　というのを　練習中に見ていて、　嫌　だったんだぜ☆

時間を使っていれば　まだ粘れたんじゃないか、と思っていたんだぜ☆」

時間管理をいじろうぜ☆！

MITERU
２つの時間を見てるだけ☆

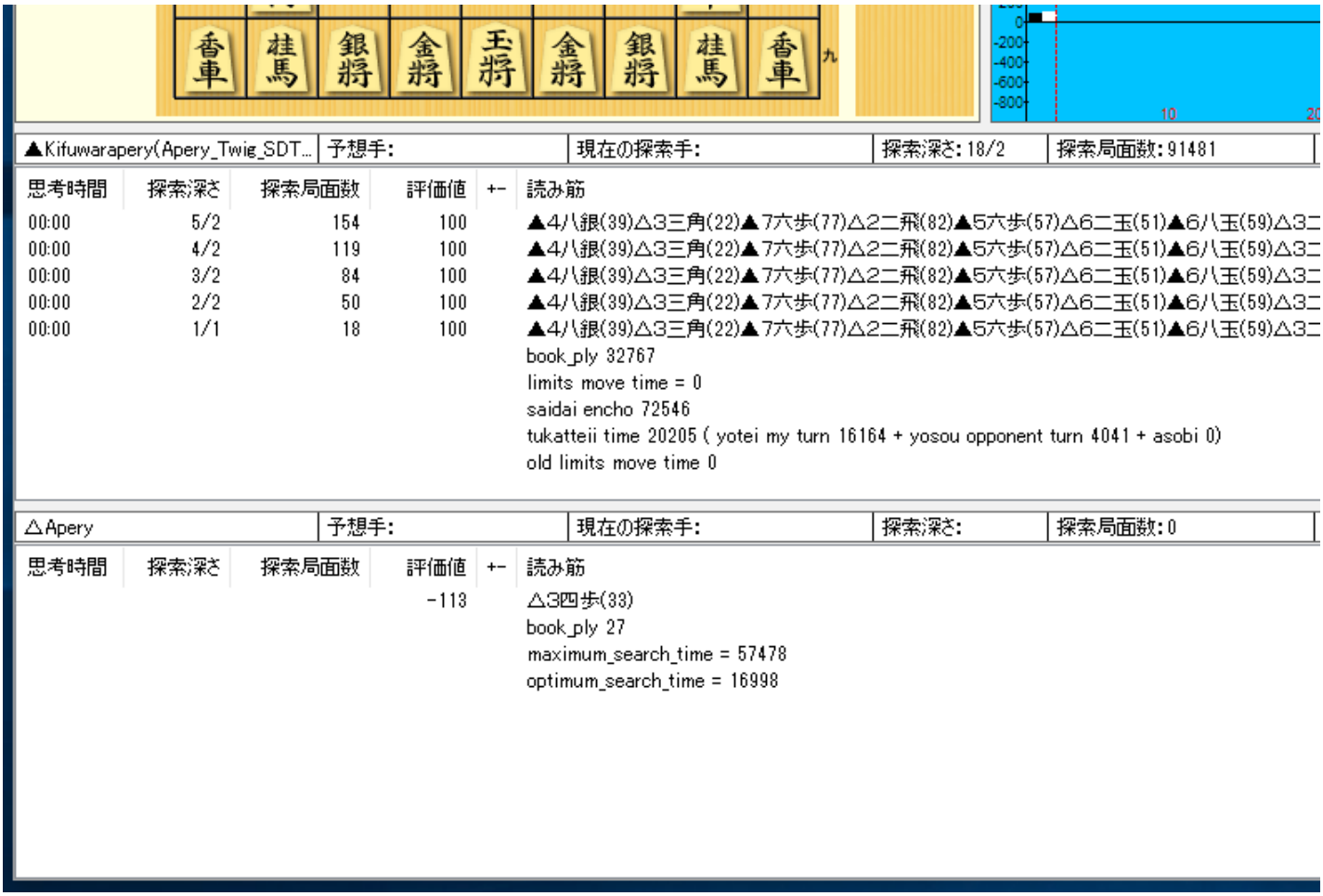
「エイプリー＿大樹の枝　は、こんな　**時間**　を見ているのな☆」



「 ここは４０秒　使っていいのか……、任せろだぜ☆」

「 将棋所の　表示を　見てみようぜ☆」

上：　きふわらべ（ＷＣＳＣ２６版）　　下：　Ａｐｅｒｙ＿ＳＳＥ４２．ｅｘｅ



「△Ａｐｅｒｙ　の方には、

ｍａｘｉｍｕｍ＿ｓｅａｒｃｈ＿ｔｉｍｅ　＝　５７４７８
ｏｐｔｉｍｕｍ＿ｓｅａｒｃｈ＿ｔｉｍｅ　＝　１６９９８

と書いてあるわね」

「 下から読む☆
単位は　ミリ秒　なので、3桁　削ったのが　秒数　だぜ☆

この１手には　１６秒　使おうぜ、
長考してもいいけど、最大で　５７秒　な☆

という意味だぜ☆」

「 わたしの方には

```
ｔｕｋａｔｔｅｉｉ　ｔｉｍｅ　２０２０５
（
　　ｙｏｔｅｉ　ｍｙ　ｔｕｒｎ　１６１６４
　　＋
　　ｙｏｓｏｕ　ｏｐｐｏｎｅｎｔ　ｔｕｒｎ　４０４１
　　＋
　　ａｓｏｂｉ　０
）
ｓａｉｄａｉ　ｅｎｃｈｏ　７２５４６
```

と書いてあるぜ☆」

「 この手に使っていい時間は　２０　秒☆
内訳は、
　　自分の手番で　１６秒　使う予定で、
　　相手は　　４秒　考えるだろう。
　　おまけは　　０秒　付けておくぜ☆

長考したいときは、最大で　７２秒　まで使ってもいいぜ☆
という意味だぜ☆」

「 コンピューターは、
経過時間、
この手に使う予定の時間、
この手に使っていい最大の時間
しか見てないわけ？」

「 探索部　がな☆
そういうことになるぜ☆」

ライブラリを　いじった知見が　ウリ☆

YATTEIRU
ソースコードが　何やっているのか　読めてきた☆

「 今大会では、 改造して 元のライブラリより 弱くしてしまったんだが、
将棋プログラムを覚えよう
という 今年の マイ課題 については 少しのナットク が得られたぜ☆

そこで、
今年は ここまで
エイプリー＿大樹の枝 を
独習した、ということを ＰＲ していくぜ☆」

「 はい」

「 その前に、他の ライブラリ も読んだので、軽く 触れておくぜ☆」

新ＣＳＡライブラリ☆

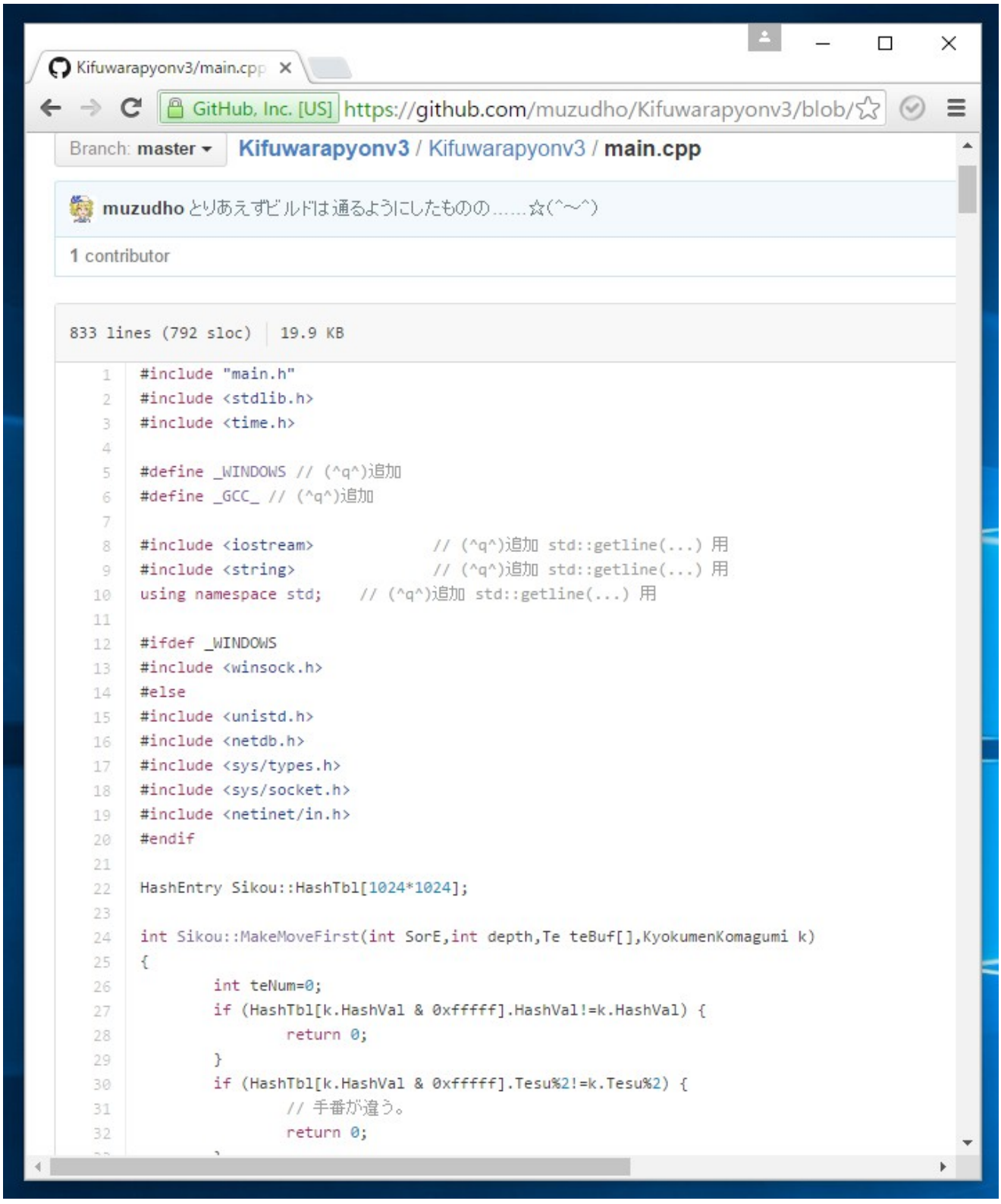
LIBRARY
どのライブラリにしようかなんだぜ☆？

「 新ＣＳＡライブラリの

やねうらおうｎａｎｏ、
なのはｍｉｎｉ、
れさぴょんｖ３、
エイプリー＿大樹の枝

に、目を通しておいたんだぜ☆」

きふわらぴょんｖ３　（れさぴょんｖ３ベース）
未使用☆（＾ｑ＾）



「れさぴょんｖ３は　ＣＳＡプロトコル通信なんだぜ☆
将棋所で　エンジン登録できなかったので　今回　見送ったんだぜ☆」

きふわらぼう　（やねうらおうｎａｎｏベース）
未使用☆（＾ｑ＾）

Kifuwarabou/n900_400_M

GitHub, Inc. [US] https://github.com/muzudho/Kifuwarabou/blob/ma

Branch: master ▾ **Kifuwarabou** / Kifuwarabou / source / n900_main____ / **n900_400_Main**

muzudho UsiLoop

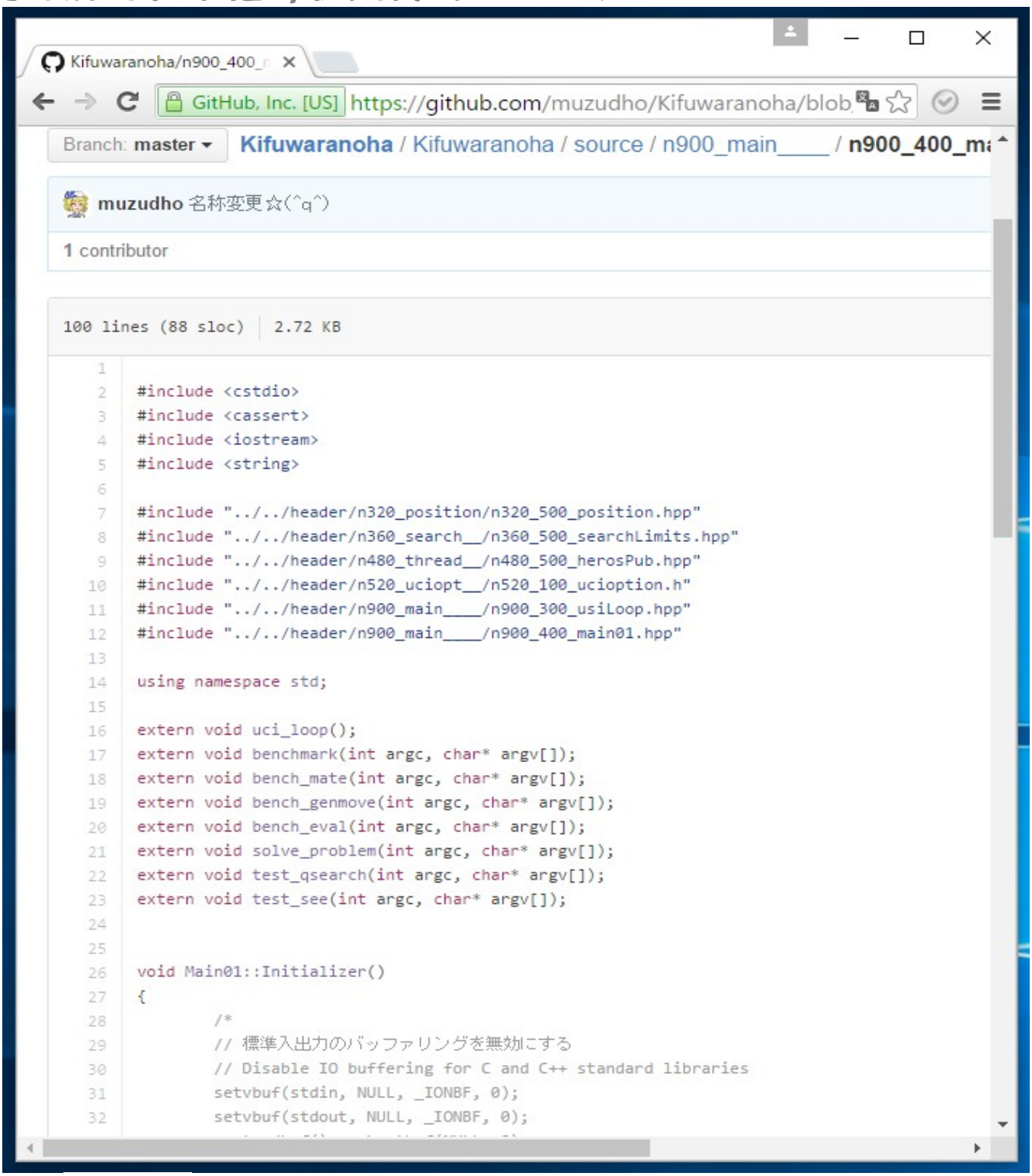
1 contributor

31 lines (26 sloc) | 751 Bytes

```
1  #include "..\..\header\n100_shogi___\n100_095_usiOption.h"
2  #include "../../header/n200_position/n200_100_position.h"
3  #include "../../header/n280_mate____/n280_100_thread.h"
4  #include "../../header/n900_main____/n900_300_usiLoop.hpp"
5  #include "..\..\header\n900_main____\n900_400_Main01.hpp"
6
7  void Main01::Initialize()
8  {
9          // --- 全体的な初期化
10         USI::init(Options);
11         Bitboards::init();
12         Position::init();
13         Search::init();
14         Threads.init();
15         Eval::init(); // 簡単な初期化のみで評価関数の読み込みはisreadyに応じて行なう。
16
17 }
18
19 void Main01::Body(int argc, char * argv[])
20 {
21         // USIコマンドの応答部
22         UsiLoop usiLoop;
23         usiLoop.mainloop01();
24 }
25
26 void Main01::Finalize()
27 {
28         // 生成して、待機させていたスレッドの停止
29         Threads.exit();
30 }
```

「やねうらおうｎａｎｏ　と　わたし　が　１０分で対局したんだが、
わたしが　勝ったので　萎えたんで　止めたんだぜ☆」

きふわらのは　　（なのはｍｉｎｉベース）
きふわらのぷりー　（　〃　Ａｐｅｒｙ評価値使用版）
読み筋の表示だけ使用☆（＾▽＾）



「なのはｍｉｎｉ　のソースコードは、無料のチェス・ソフトで強くて有名な
Ｓｔｏｃｋｆｉｓｈ（ストックフィッシュ）　ベースなんだが、
改造前の　Ｓｔｏｃｋｆｉｓｈ　のコードも併記して　残してくれているんだぜ☆」

search.cpp （なのはmini）

E:¥Muzudho¥Shogi¥新ライブラリ¥07_なのはmini (0.2.2.1)¥NanohaMini0221¥src¥search.cpp

ファイル(F) 編集(E) 検索(S) 整形(P) 表示(V) ツール(T) フォルダ(L) HTMLタグ(M) ウインドウ(W) ヘルプ(H)

```
581
582    // Could be MOVE_NONE when searching on a stalemate position
583#if defined(NANOHA)
584    if (bestMove == MOVE_NONE) {
585        cout << "bestmove resign";
586    } else {
587        cout << "bestmove " << move_to_uci(bestMove);
588    }
589#else
590    cout << "bestmove " << bestMove;
591#endif
592
593    // UCI protol is not clear on allowing sending an empty ponder move, instead
594    // it is clear that ponder move is optional. So skip it if empty.
595#if defined(NANOHA)
596    if (ponderMove != MOVE_NONE && Options["Ponder"].value<bool>())
597        cout << " ponder " << move_to_uci(ponderMove);
598
599    searchMoves[0] = bestMove;
600#else
601    if (ponderMove != MOVE_NONE)
602        cout << " ponder " << ponderMove;
603#endif
604
```

search.cpp 1 : 1 標準 C/C++ SJIS LF 挿入 A

「＃ｉｆ　ｄｅｆｉｎｅｄ（ＮＡＮＯＨＡ）　と書かれているところが、
なのはｍｉｎｉ　で改造された　将棋のプログラム　だと思うんだぜ☆

反対側の　＃ｅｌｓｅ　は多分、
Ｓｔｏｃｋｆｉｓｈの　チェス・プログラム　だと思うんだぜ☆」

「評価値ファイルは　Ａｐｅｒｙ　のものに差し替えられるというのも
面白いところだぜ☆」

「将棋所での読み筋の表示のために　なのはｍｉｎｉ　のソースコードを
きふわらぷりー　に　持ってきたよな☆　なぜだぜ☆？」

「もっと　なのはｍｉｎｉ　と　エイプリー＿大樹の枝　の
いいとこどり　しようと思っていたんだが、
まだまだ　ソースコードを読み解くちからが　足りなかったので　そこだけだぜ☆」

きふわらぷりー　（エイプリー_大樹の枝ベース）
今大会のベースとして使用だぜ☆！＼（ ^ ▽ ^ ）／

Kifuwarapery/n900_400_m

GitHub, Inc. [US] https://github.com/muzudho/Kifuwarapery/blob/

Branch: master ▾ **Kifuwarapery** / Kifuwarapery / source / n900_main____ / **n900_400_mai**

muzudho 指し手生成部整理途中☆(^q^)

1 contributor

123 lines (96 sloc) | 4.67 KB

```
1   #include <iostream>
2   #include "../../header/n080_common__/n080_100_common.hpp"
3   #include "../../header/n110_square__/n110_400_squareRelation.hpp"
4   #include "../../header/n160_board___/n160_100_bitboard.hpp"
5   #include "../../header/n160_board___/n160_110_silverAttackBb.hpp"
6   #include "../../header/n160_board___/n160_120_bishopAttackBb.hpp"
7   #include "../../header/n160_board___/n160_130_lanceAttackBb.hpp"
8   #include "../../header/n160_board___/n160_140_goldAttackBb.hpp"
9   #include "../../header/n160_board___/n160_150_rookAttackBb.hpp"
10  #include "../../header/n160_board___/n160_160_kingAttackBb.hpp"
11  #include "../../header/n160_board___/n160_180_knightAttackBb.hpp"
12  #include "../../header/n160_board___/n160_190_pawnAttackBb.hpp"
13  #include "../../header/n161_sqDistan/n161_500_squareDistance.hpp"
14  #include "../../header/n220_position/n220_750_charToPieceUSI.hpp"
15  #include "../../header/n351_bonaDir_/n351_600_utilSquareDistance.hpp"
16  #include "../../header/n480_tt______/n480_300_tt.hpp"
17  #include "../../header/n520_evaluate/n520_500_kkKkpKppStorage1.hpp"
18  #include "../../header/n600_book____/n600_500_book.hpp"
19  #include "../../header/n760_thread__/n760_400_herosPub.hpp"
20  #include "../../header/n885_searcher/n885_040_rucksack.hpp"
21  #include "..\..\header\n900_main____/n900_300_usiLoop.hpp"
22  #include "..\..\header\n900_main____\n900_400_main01.hpp"
23
24
25
26
27  Main01::Main01()
28  {
29          this->m_rucksack = std::unique_ptr<Rucksack>(new Rucksack);
30  }
31
32  Main01::~Main01()
```

「なのはｍｉｎｉ　や、
Ａｐｅｒｙの評価値ファイルに差し替えた　なのはｍｉｎｉ　より
　強かったので　使うことにしたんだぜ☆」

「きふわらべ（きふわらぷりー）の　ＷＣＳＣ２６出場版のソースと　実行ファイルは
開発現行版とは分けた場所で　配布しているぜ☆」

ソースコード
https://github.com/muzudho/Kifuwarabe_WCSC26

実行ファイル
https://github.com/muzudho/Kifuwarabe_WCSC26_bin

「やねうら王ｎａｎｏ、　なのはｍｉｎｉ、　Ａｐｅｒｙ＿大樹の枝　の
３つ　を読んで思ったんだが、
どれも　変数の名前や、　全体的なつくりが　似ているんだぜ☆
恐らく、　Ｓｔｏｃｋｆｉｓｈ　の知識をそのまま使えるように
Ｓｔｏｃｋｆｉｓｈ　の書き方を　残しているんだと　思うんだぜ☆」

「Ａｐｅｒｙ＿大樹の枝　の学習部分は、
Ｂｏｎａｎｚａ６．０　を読んでいると、
なんとなく　アタリを付けて　似ている雰囲気をつかめるぜ☆」

「勉強したな☆」

「勉強したぜ☆」

きふわらべは　ほとんど　エイプリー＿大樹の枝☆

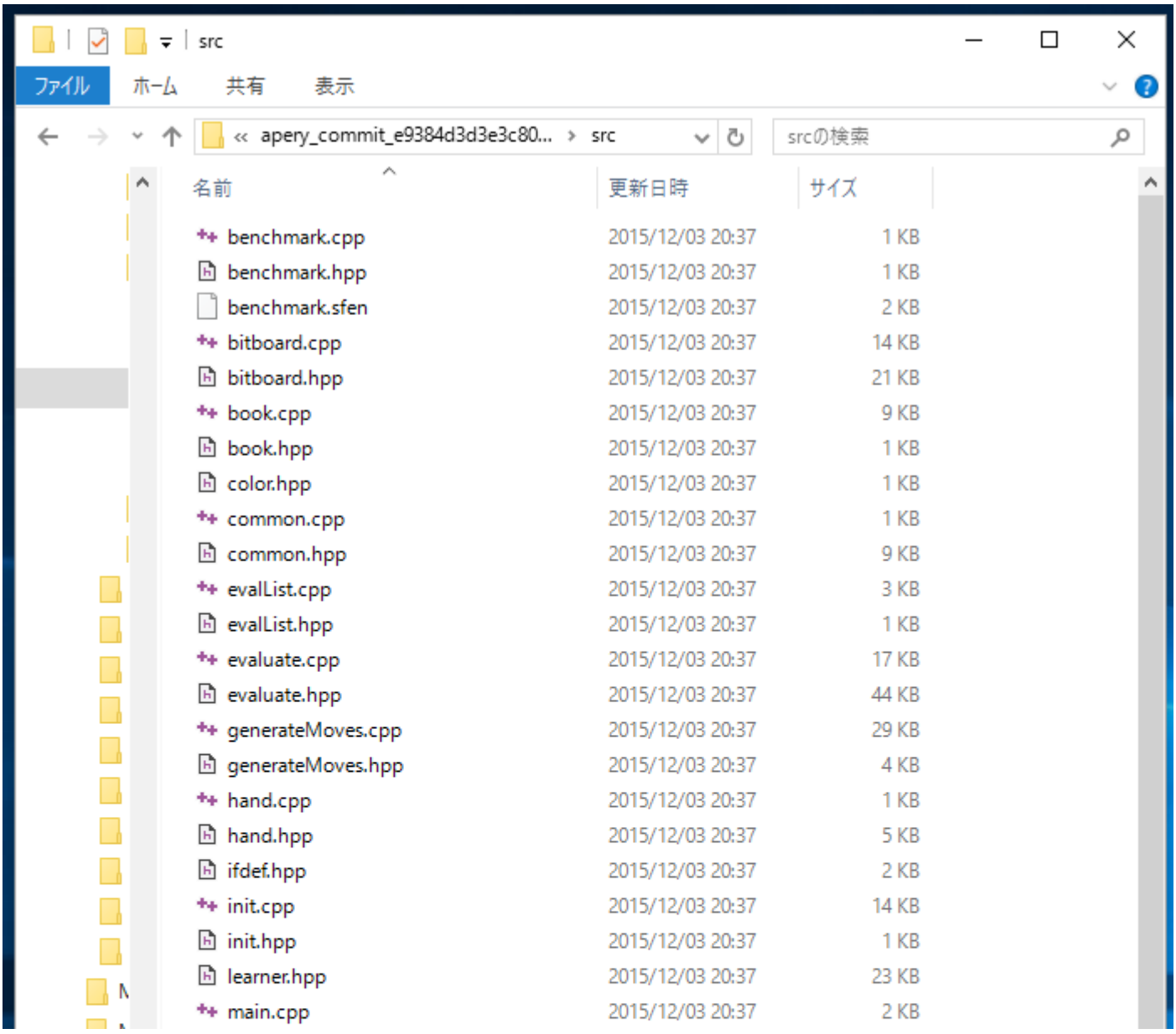
TAIJU NO EDA

エイプリー＿大樹の枝のソースコードを紹介するぜ☆

「どこを変えたの、を説明するために、
エイプリー＿大樹の枝　が　どういうものか説明しておくぜ☆」

「三行で頼むぜ☆」

Ａｐｅｒｙ＿Ｔｗｉｇ　（エイプリー＿大樹の枝）



Makefile	2015/12/03 20:37	2 KB
move.cpp	2015/12/03 20:37	2 KB
move.hpp	2015/12/03 20:37	9 KB
movePicker.cpp	2015/12/03 20:37	8 KB
movePicker.hpp	2015/12/03 20:37	2 KB
mt64bit.cpp	2015/12/03 20:37	1 KB
mt64bit.hpp	2015/12/03 20:37	1 KB
overloadEnumOperators.hpp	2015/12/03 20:37	3 KB
piece.hpp	2015/12/03 20:37	3 KB
pieceScore.cpp	2015/12/03 20:37	2 KB
pieceScore.hpp	2015/12/03 20:37	3 KB
position.cpp	2015/12/03 20:37	69 KB
position.hpp	2015/12/03 20:37	18 KB
score.hpp	2015/12/03 20:37	2 KB
search.cpp	2015/12/03 20:37	49 KB
search.hpp	2015/12/03 20:37	5 KB
square.cpp	2015/12/03 20:37	1 KB
square.hpp	2015/12/03 20:37	10 KB
thread.cpp	2015/12/03 20:37	7 KB
thread.hpp	2015/12/03 20:37	3 KB
timeManager.cpp	2015/12/03 20:37	7 KB
timeManager.hpp	2015/12/03 20:37	1 KB
tt.cpp	2015/12/03 20:37	3 KB
tt.hpp	2015/12/03 20:37	4 KB
usi.cpp	2015/12/03 20:37	16 KB
usi.hpp	2015/12/03 20:37	2 KB

49 個の項目

「 これが　エイプリー＿大樹の枝　のソースコードだぜ☆
　レーティング３３００とか出るという　将棋ソフト　のソースコードはこれだけ☆」

「 あら、これだけ　だったら
　わたしにも　読めそうね」

「 わたしは読んだぜ☆」

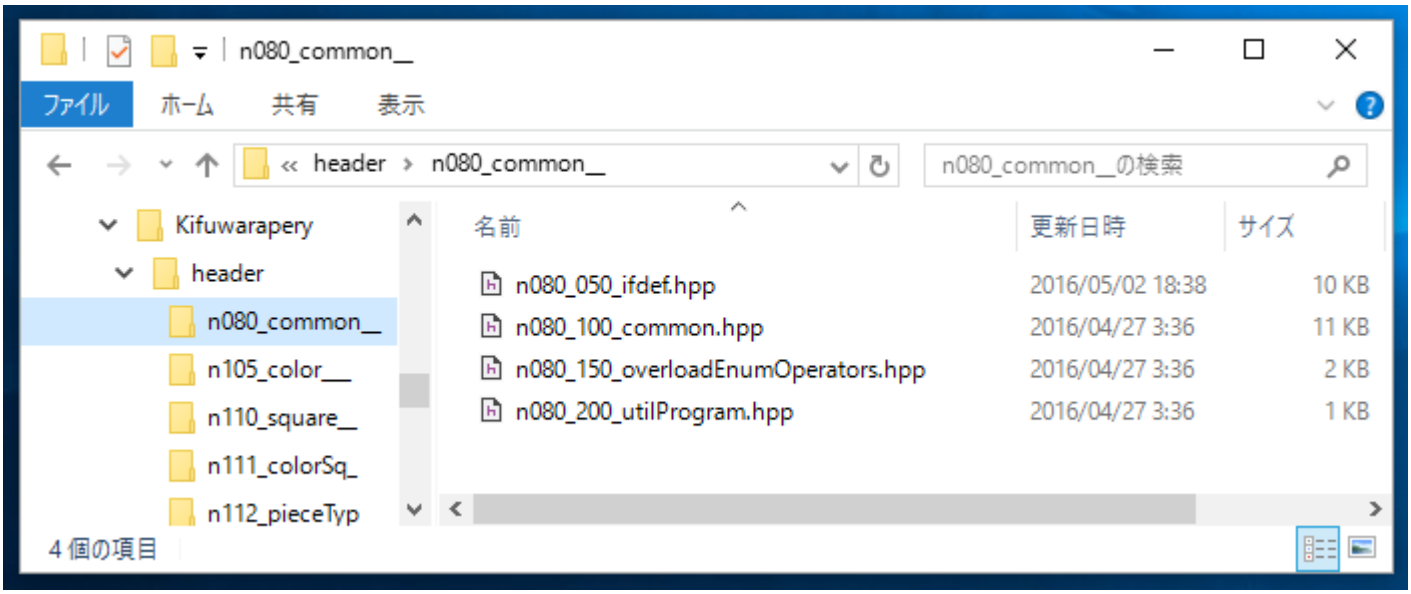
「 では、腹をくくってもらおうか☆」

「 ？」

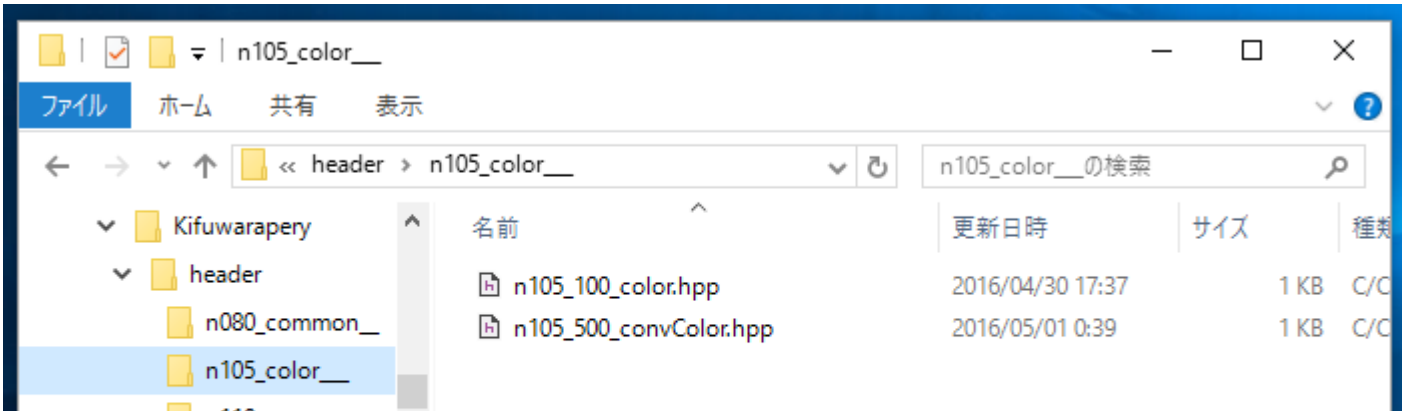
「 大会出場版　きふわらべ、　わたしは　きふわらぷりー　と呼んでいるんだが
ソースコードを紹介するぜ☆」

きふわらぷりー　（エイプリー_大樹の枝ベース）

1

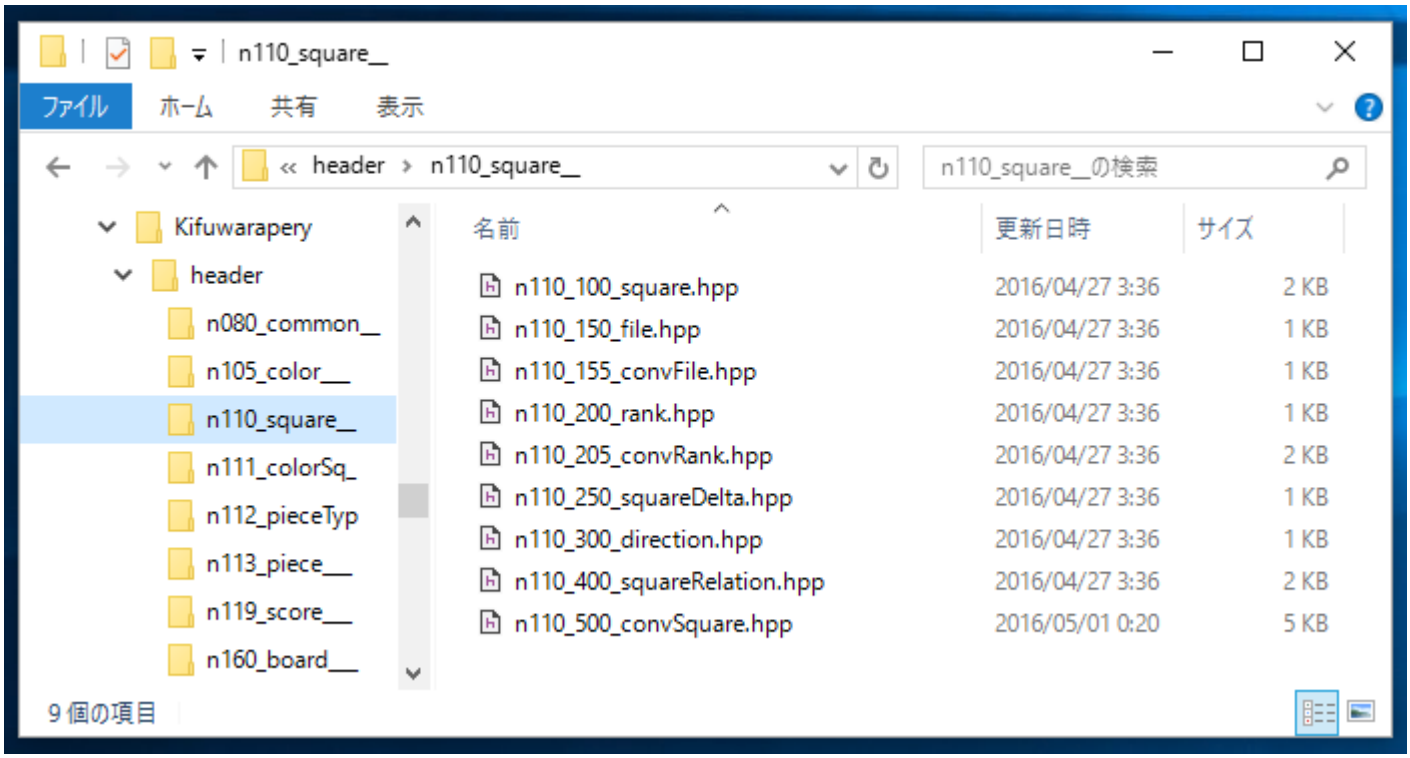


2

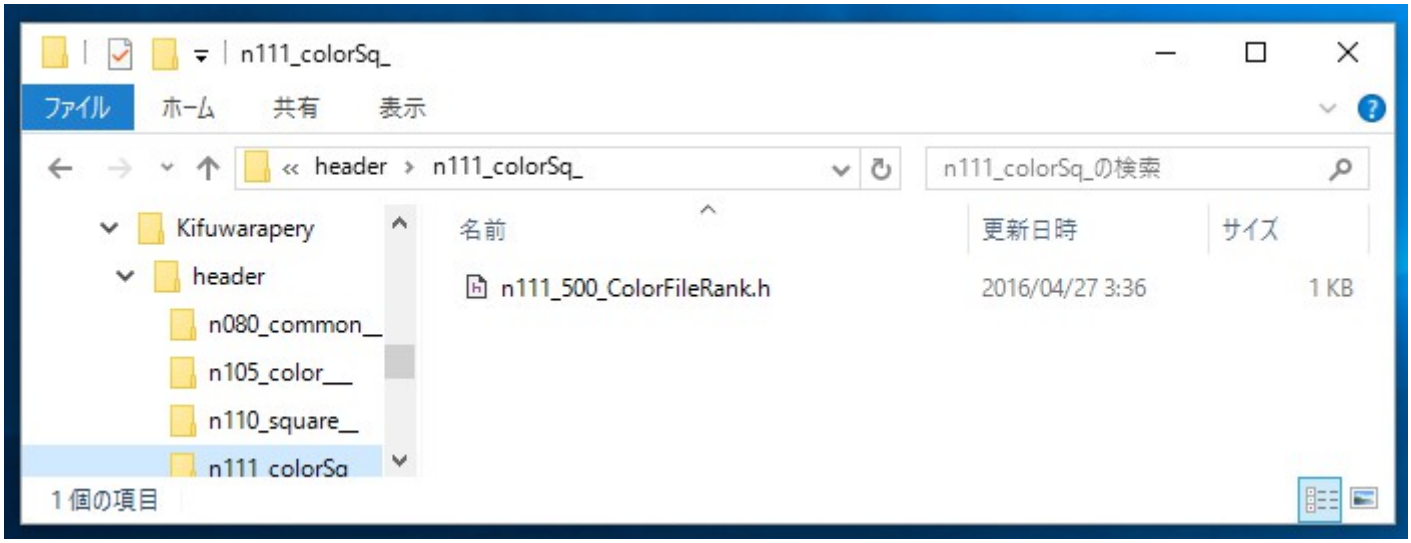


n110_square__
n111_colorSq_
n112_pieceTyp
2 個の項目

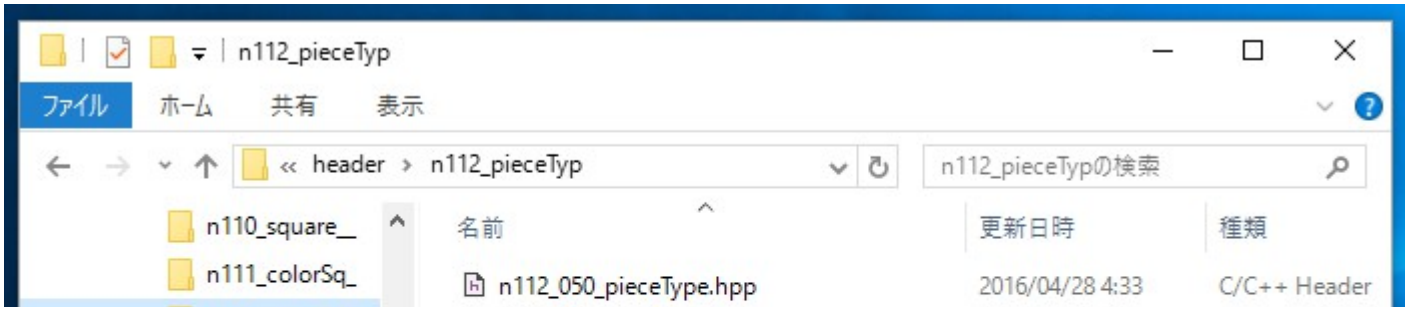
3


n110_square__
ファイル
ホーム
共有
表示
« header › n110_square__
n110_square__の検索
Kifuwarapery
header
n080_common__
n105_color___
n110_square__
n111_colorSq_
n112_pieceTyp
n113_piece___
n119_score___
n160_board___
名前
更新日時
サイズ
n110_100_square.hpp 2016/04/27 3:36 2 KB
n110_150_file.hpp 2016/04/27 3:36 1 KB
n110_155_convFile.hpp 2016/04/27 3:36 1 KB
n110_200_rank.hpp 2016/04/27 3:36 1 KB
n110_205_convRank.hpp 2016/04/27 3:36 2 KB
n110_250_squareDelta.hpp 2016/04/27 3:36 1 KB
n110_300_direction.hpp 2016/04/27 3:36 1 KB
n110_400_squareRelation.hpp 2016/04/27 3:36 2 KB
n110_500_convSquare.hpp 2016/05/01 0:20 5 KB
9 個の項目

4

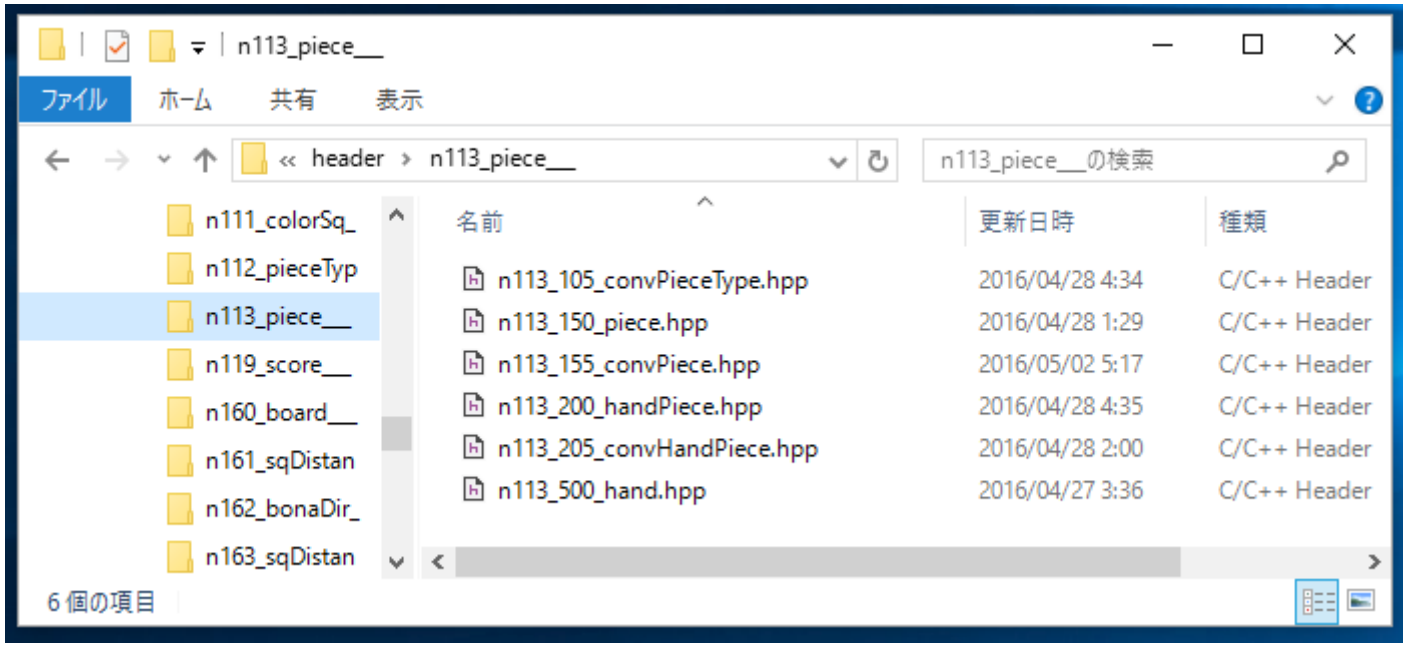

n111_colorSq_
ファイル
ホーム
共有
表示
« header › n111_colorSq_
n111_colorSq_の検索
Kifuwarapery
header
n080_common__
n105_color___
n110_square__
n111_colorSq
名前
更新日時
サイズ
n111_500_ColorFileRank.h 2016/04/27 3:36 1 KB
1 個の項目

5

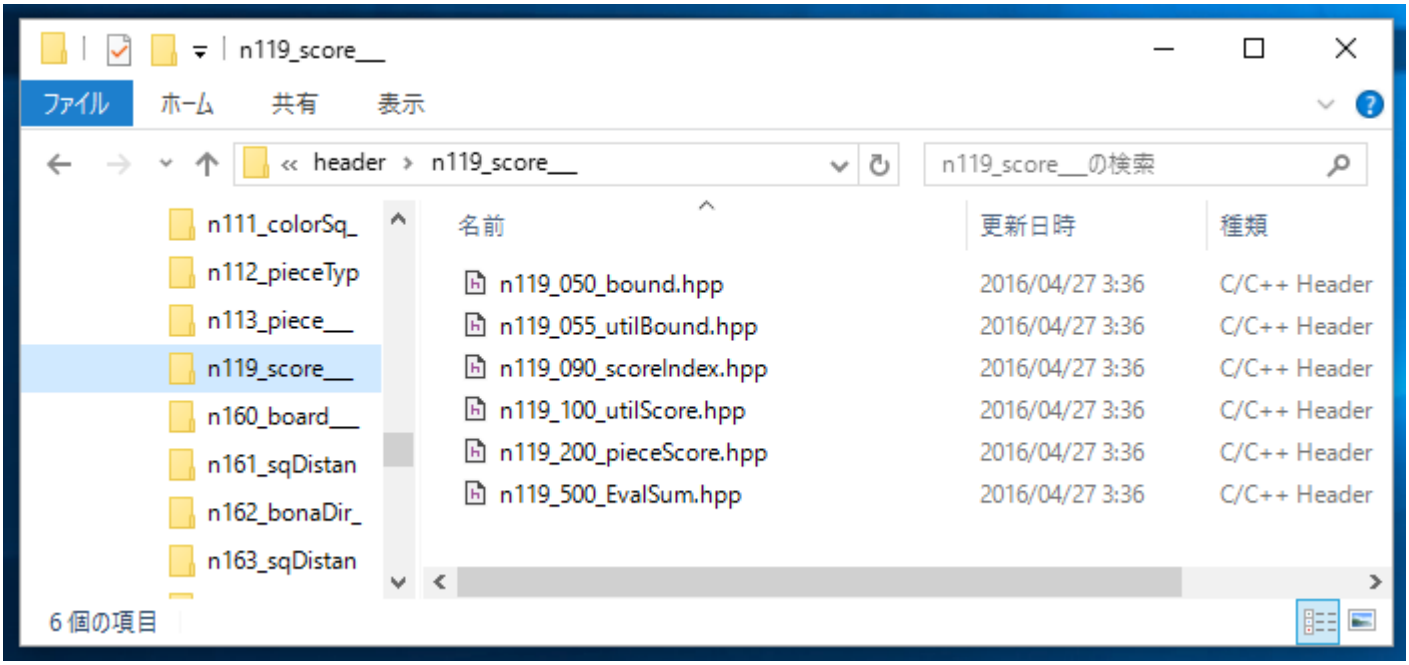

n112_pieceTyp
ファイル
ホーム
共有
表示
« header › n112_pieceTyp
n112_pieceTypの検索
n110_square__
n111_colorSq_
名前
更新日時
種類
n112_050_pieceType.hpp 2016/04/28 4:33 C/C++ Header

n112_pieceTyp
n113_piece__
n119_score__
n160 board
1 個の項目

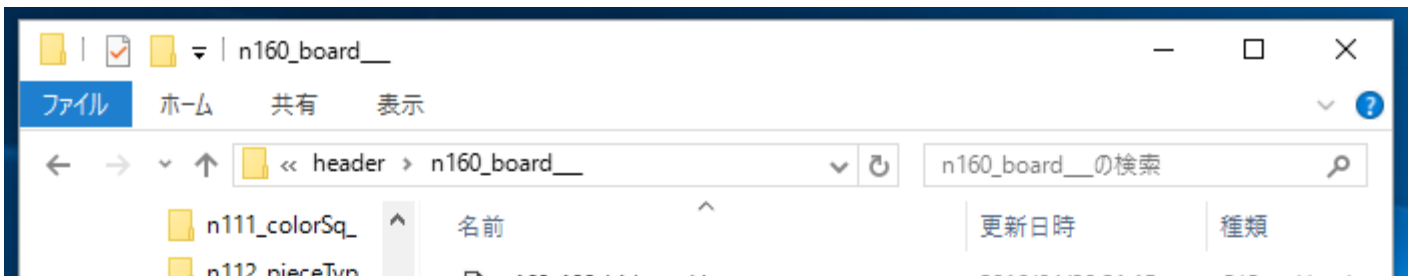
6

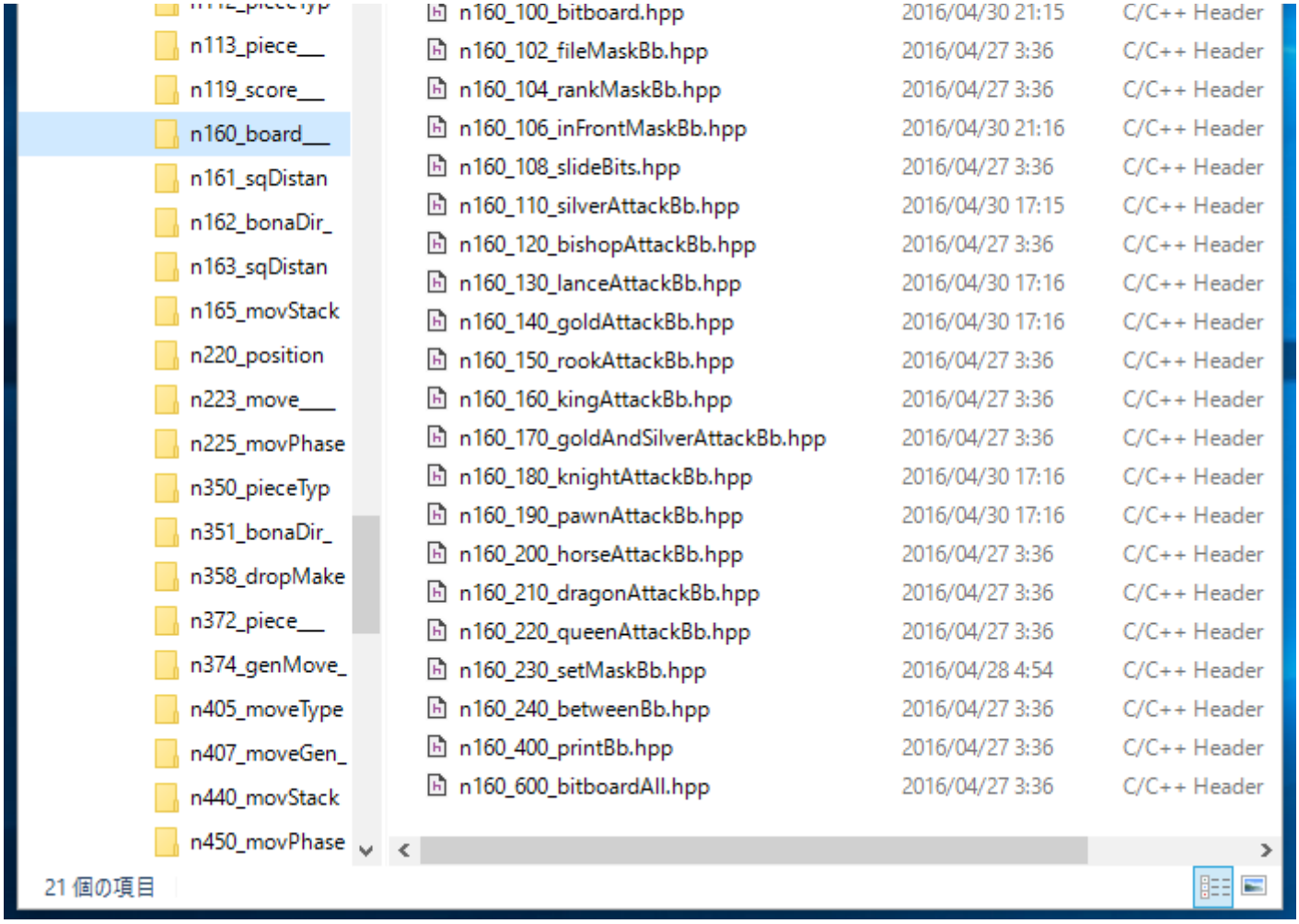

n113_piece__
ファイル
ホーム
共有
表示
« header › n113_piece__
n113_piece__の検索
n111_colorSq_
n112_pieceTyp
n113_piece__
n119_score__
n160_board__
n161_sqDistan
n162_bonaDir_
n163_sqDistan
名前
更新日時
種類
n113_105_convPieceType.hpp 2016/04/28 4:34 C/C++ Header
n113_150_piece.hpp 2016/04/28 1:29 C/C++ Header
n113_155_convPiece.hpp 2016/05/02 5:17 C/C++ Header
n113_200_handPiece.hpp 2016/04/28 4:35 C/C++ Header
n113_205_convHandPiece.hpp 2016/04/28 2:00 C/C++ Header
n113_500_hand.hpp 2016/04/27 3:36 C/C++ Header
6 個の項目

7

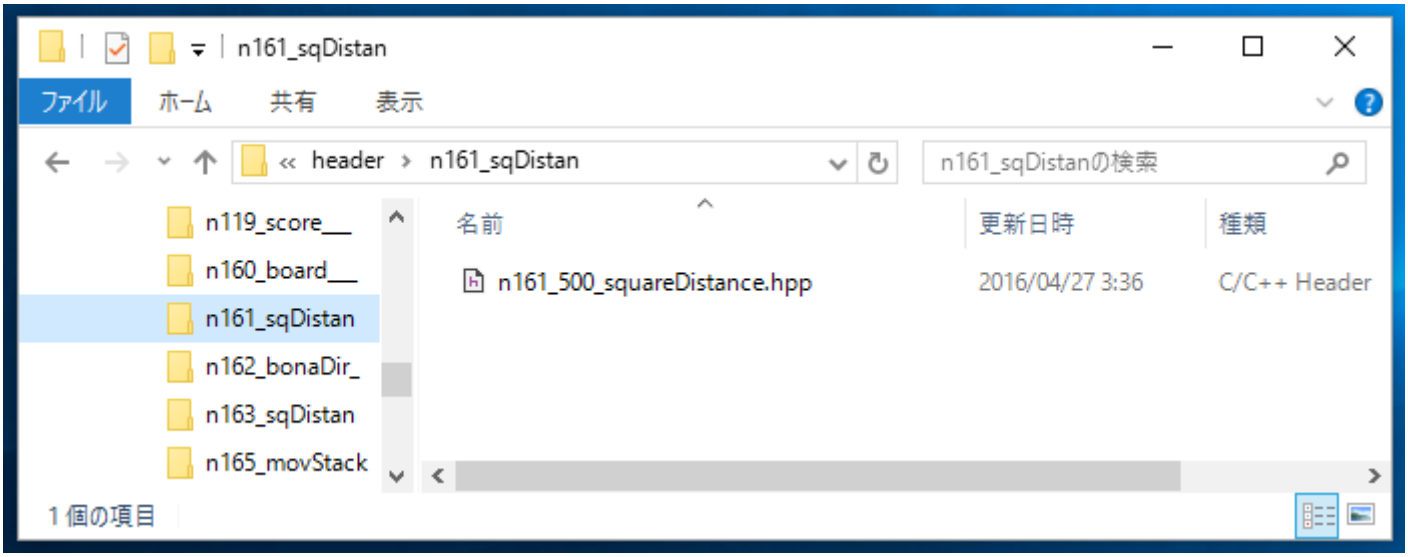

n119_score__
ファイル
ホーム
共有
表示
« header › n119_score__
n119_score__の検索
n111_colorSq_
n112_pieceTyp
n113_piece__
n119_score__
n160_board__
n161_sqDistan
n162_bonaDir_
n163_sqDistan
名前
更新日時
種類
n119_050_bound.hpp 2016/04/27 3:36 C/C++ Header
n119_055_utilBound.hpp 2016/04/27 3:36 C/C++ Header
n119_090_scoreIndex.hpp 2016/04/27 3:36 C/C++ Header
n119_100_utilScore.hpp 2016/04/27 3:36 C/C++ Header
n119_200_pieceScore.hpp 2016/04/27 3:36 C/C++ Header
n119_500_EvalSum.hpp 2016/04/27 3:36 C/C++ Header
6 個の項目

8

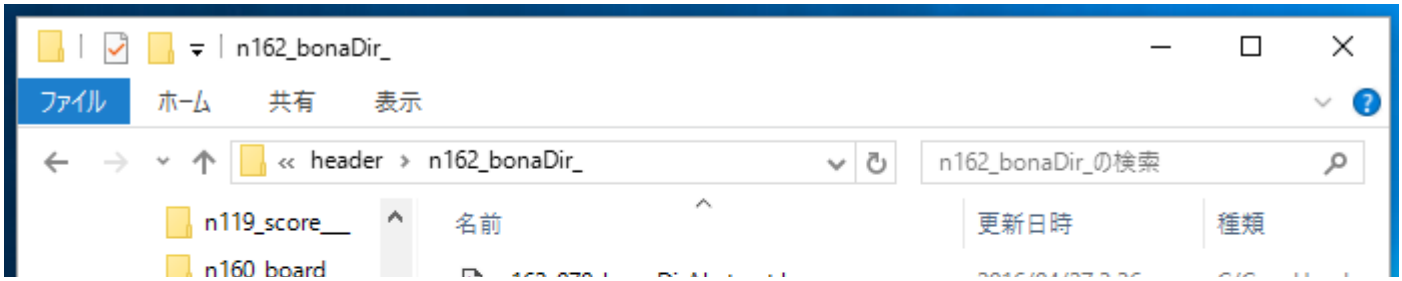

n160_board__
ファイル
ホーム
共有
表示
« header › n160_board__
n160_board__の検索
n111_colorSq_
名前
更新日時
種類

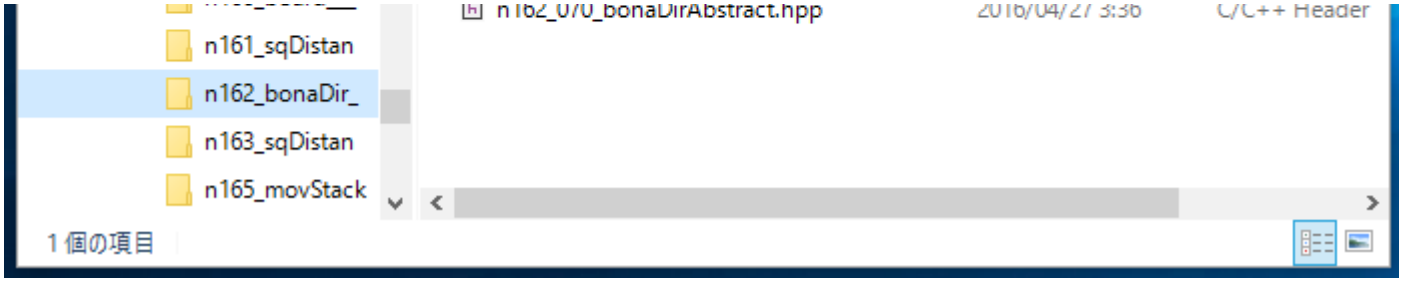

n113_piece___
n119_score___
n160_board___
n161_sqDistan
n162_bonaDir_
n163_sqDistan
n165_movStack
n220_position
n223_move____
n225_movPhase
n350_pieceTyp
n351_bonaDir_
n358_dropMake
n372_piece___
n374_genMove_
n405_moveType
n407_moveGen_
n440_movStack
n450_movPhase
n160_100_bitboard.hpp 2016/04/30 21:15 C/C++ Header
n160_102_fileMaskBb.hpp 2016/04/27 3:36 C/C++ Header
n160_104_rankMaskBb.hpp 2016/04/27 3:36 C/C++ Header
n160_106_inFrontMaskBb.hpp 2016/04/30 21:16 C/C++ Header
n160_108_slideBits.hpp 2016/04/27 3:36 C/C++ Header
n160_110_silverAttackBb.hpp 2016/04/30 17:15 C/C++ Header
n160_120_bishopAttackBb.hpp 2016/04/27 3:36 C/C++ Header
n160_130_lanceAttackBb.hpp 2016/04/30 17:16 C/C++ Header
n160_140_goldAttackBb.hpp 2016/04/30 17:16 C/C++ Header
n160_150_rookAttackBb.hpp 2016/04/27 3:36 C/C++ Header
n160_160_kingAttackBb.hpp 2016/04/27 3:36 C/C++ Header
n160_170_goldAndSilverAttackBb.hpp 2016/04/27 3:36 C/C++ Header
n160_180_knightAttackBb.hpp 2016/04/30 17:16 C/C++ Header
n160_190_pawnAttackBb.hpp 2016/04/30 17:16 C/C++ Header
n160_200_horseAttackBb.hpp 2016/04/27 3:36 C/C++ Header
n160_210_dragonAttackBb.hpp 2016/04/27 3:36 C/C++ Header
n160_220_queenAttackBb.hpp 2016/04/27 3:36 C/C++ Header
n160_230_setMaskBb.hpp 2016/04/28 4:54 C/C++ Header
n160_240_betweenBb.hpp 2016/04/27 3:36 C/C++ Header
n160_400_printBb.hpp 2016/04/27 3:36 C/C++ Header
n160_600_bitboardAll.hpp 2016/04/27 3:36 C/C++ Header
21 個の項目

9

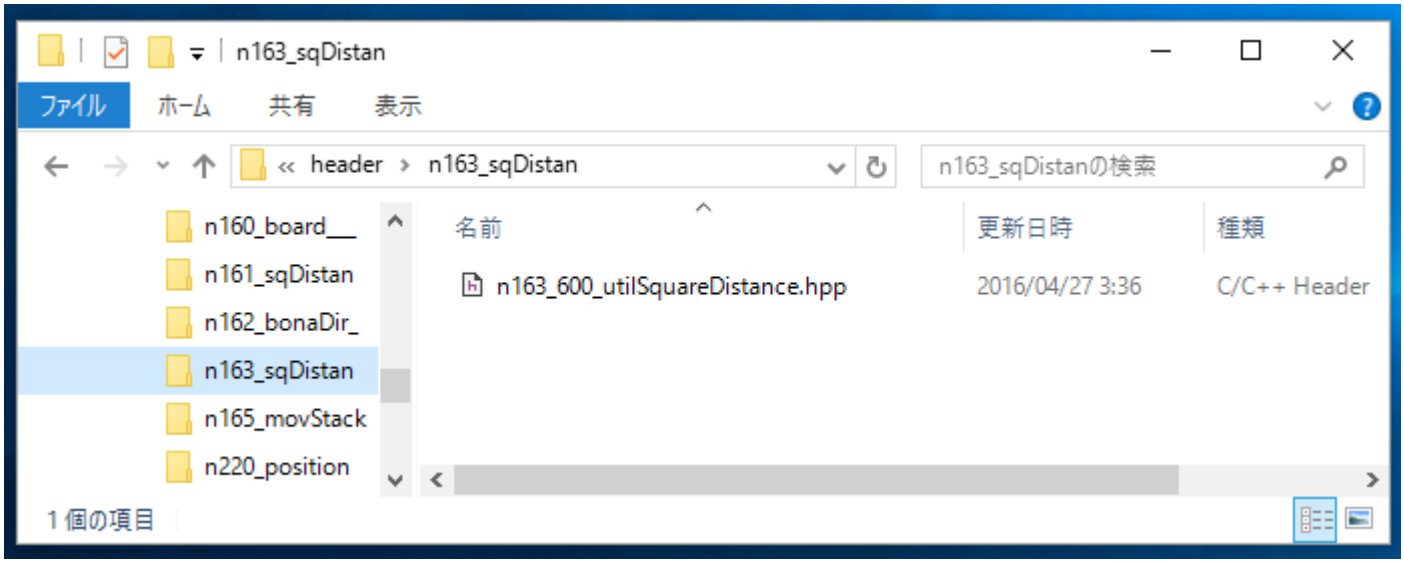

n161_sqDistan
ファイル
ホーム
共有
表示
« header › n161_sqDistan
n161_sqDistanの検索
n119_score___
n160_board___
n161_sqDistan
n162_bonaDir_
n163_sqDistan
n165_movStack
名前
更新日時
種類
n161_500_squareDistance.hpp 2016/04/27 3:36 C/C++ Header
1 個の項目

10

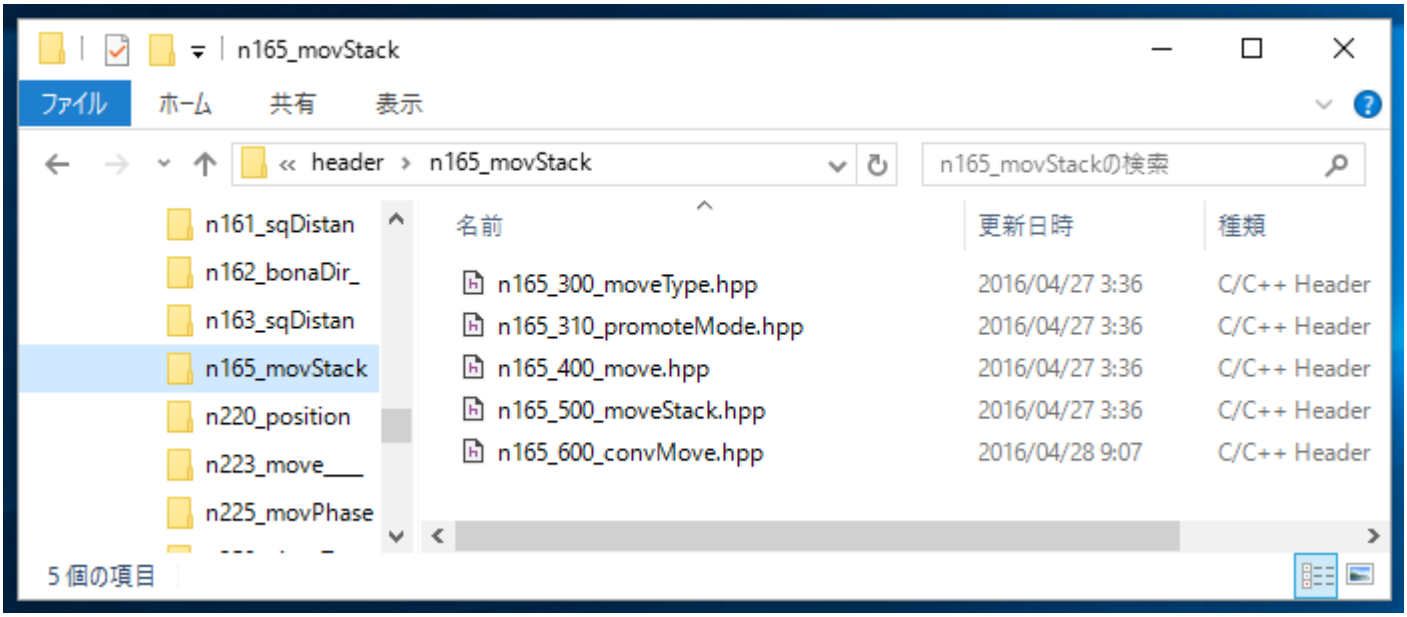

n162_bonaDir_
ファイル
ホーム
共有
表示
« header › n162_bonaDir_
n162_bonaDir_の検索
n119_score___
n160_board
名前
更新日時
種類


n161_sqDistan
n162_bonaDir_
n163_sqDistan
n165_movStack
1 個の項目

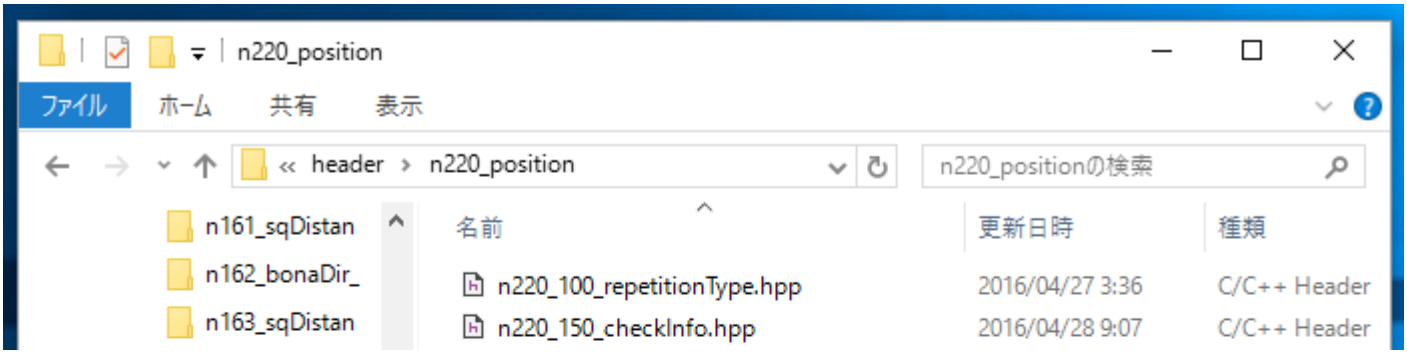
11

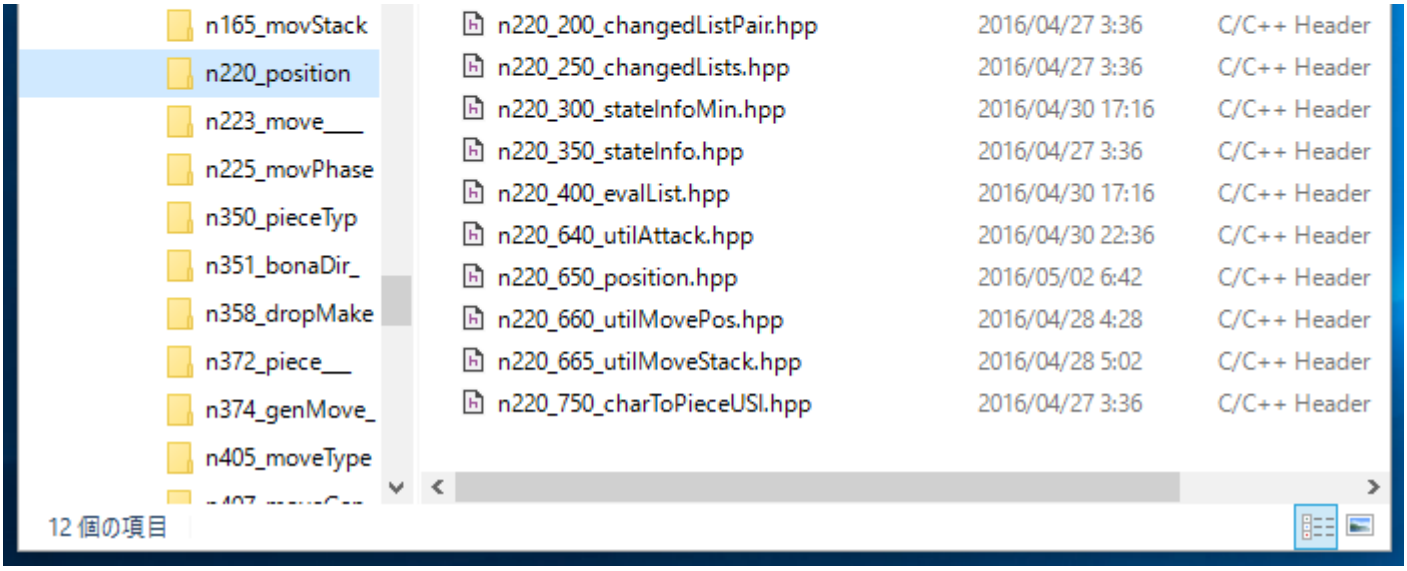

n163_sqDistan
ファイル
ホーム
共有
表示
« header › n163_sqDistan
n163_sqDistanの検索
n160_board___
n161_sqDistan
n162_bonaDir_
n163_sqDistan
n165_movStack
n220_position
名前
更新日時
種類
n163_600_utilSquareDistance.hpp
2016/04/27 3:36
C/C++ Header
1 個の項目

12

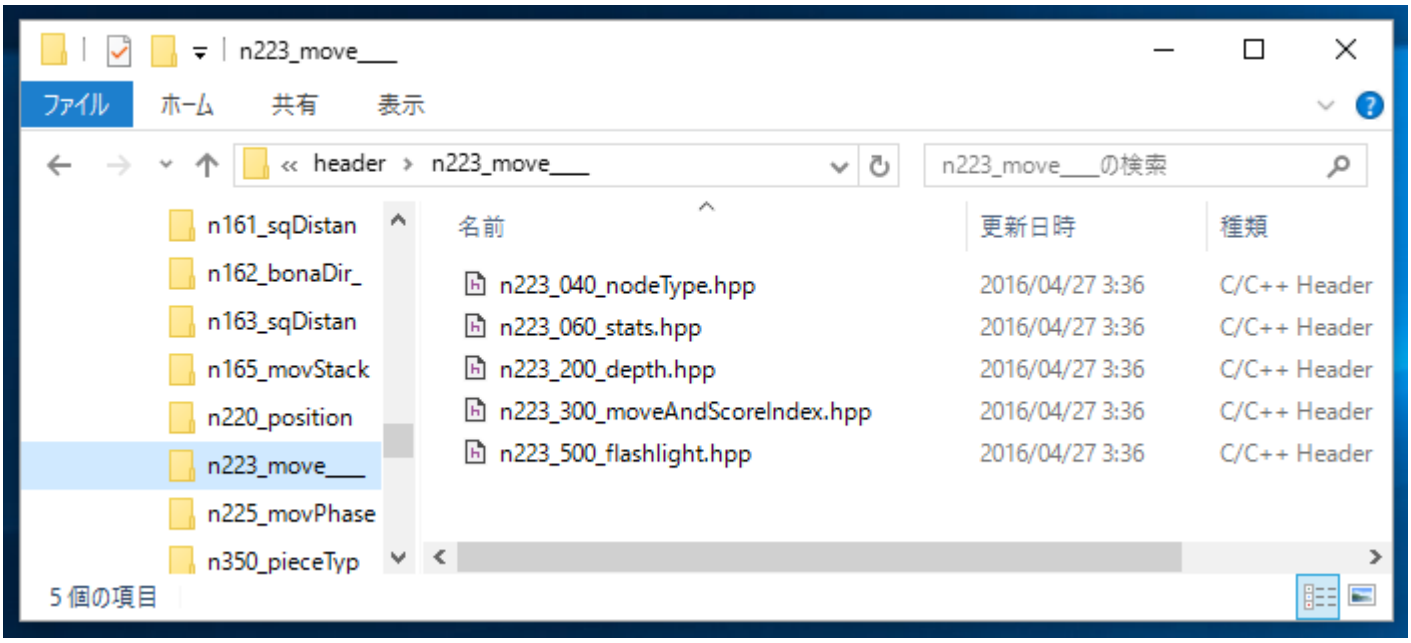

n165_movStack
ファイル
ホーム
共有
表示
« header › n165_movStack
n165_movStackの検索
n161_sqDistan
n162_bonaDir_
n163_sqDistan
n165_movStack
n220_position
n223_move___
n225_movPhase
名前
更新日時
種類
n165_300_moveType.hpp
2016/04/27 3:36
C/C++ Header
n165_310_promoteMode.hpp
2016/04/27 3:36
C/C++ Header
n165_400_move.hpp
2016/04/27 3:36
C/C++ Header
n165_500_moveStack.hpp
2016/04/27 3:36
C/C++ Header
n165_600_convMove.hpp
2016/04/28 9:07
C/C++ Header
5 個の項目

13

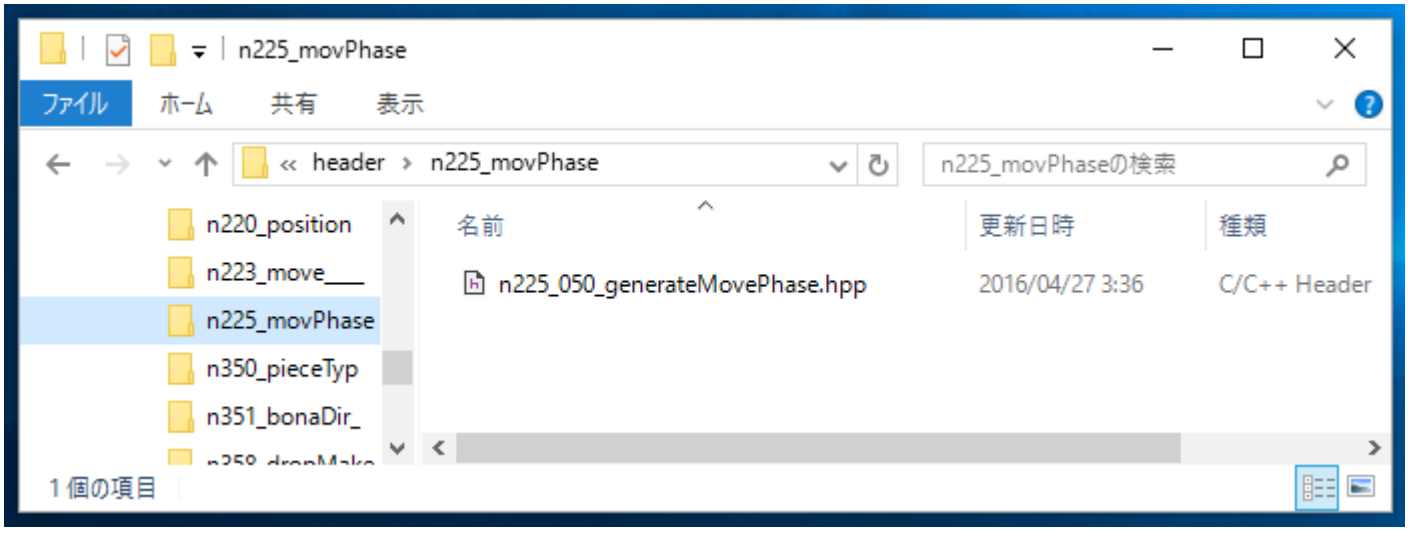

n220_position
ファイル
ホーム
共有
表示
« header › n220_position
n220_positionの検索
n161_sqDistan
n162_bonaDir_
n163_sqDistan
名前
更新日時
種類
n220_100_repetitionType.hpp
2016/04/27 3:36
C/C++ Header
n220_150_checkInfo.hpp
2016/04/28 9:07
C/C++ Header


n165_movStack
n220_position
n223_move___
n225_movPhase
n350_pieceTyp
n351_bonaDir_
n358_dropMake
n372_piece___
n374_genMove_
n405_moveType
n220_200_changedListPair.hpp 2016/04/27 3:36 C/C++ Header
n220_250_changedLists.hpp 2016/04/27 3:36 C/C++ Header
n220_300_stateInfoMin.hpp 2016/04/30 17:16 C/C++ Header
n220_350_stateInfo.hpp 2016/04/27 3:36 C/C++ Header
n220_400_evalList.hpp 2016/04/30 17:16 C/C++ Header
n220_640_utilAttack.hpp 2016/04/30 22:36 C/C++ Header
n220_650_position.hpp 2016/05/02 6:42 C/C++ Header
n220_660_utilMovePos.hpp 2016/04/28 4:28 C/C++ Header
n220_665_utilMoveStack.hpp 2016/04/28 5:02 C/C++ Header
n220_750_charToPieceUSI.hpp 2016/04/27 3:36 C/C++ Header
12 個の項目

14

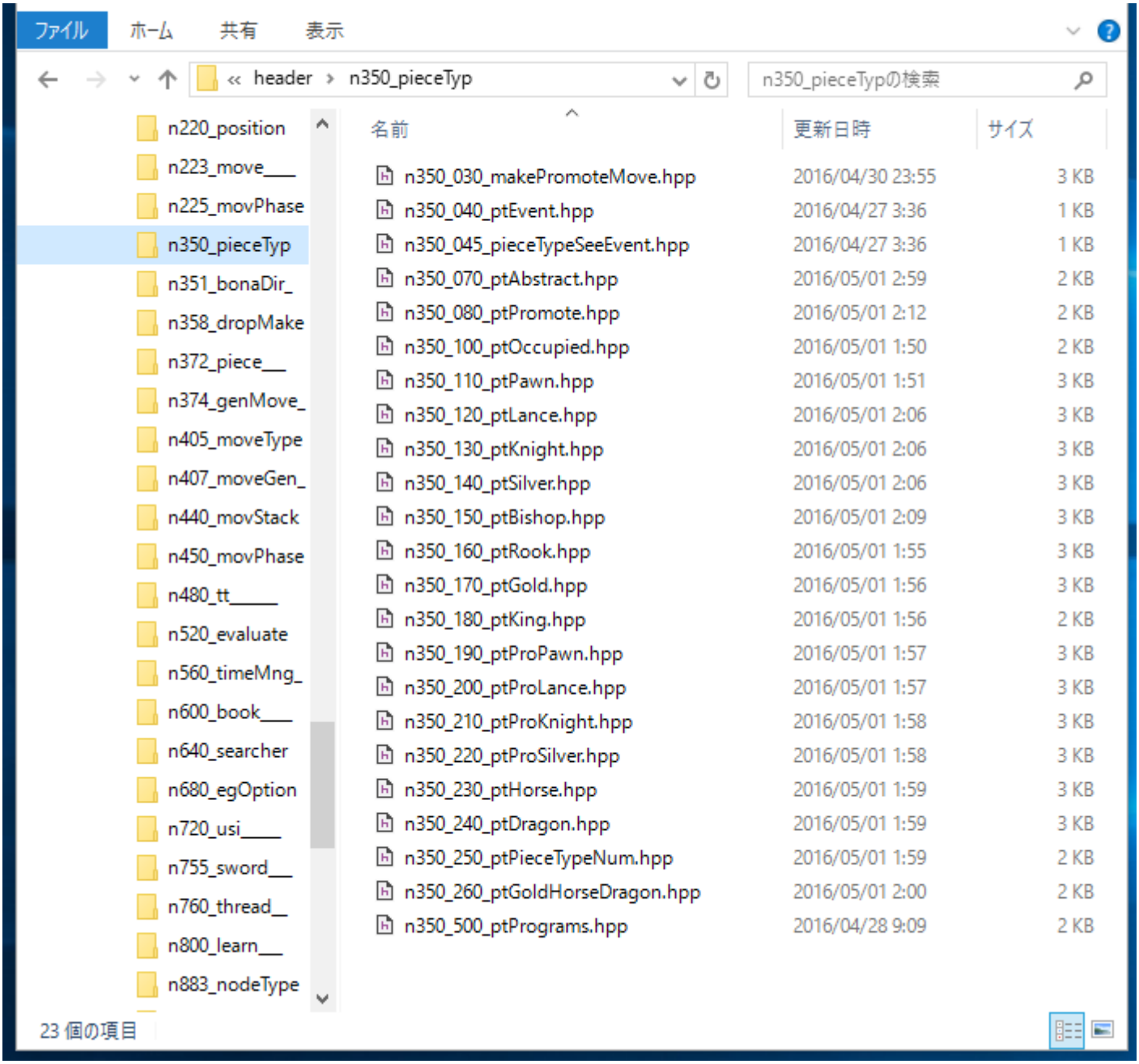

n223_move___
ファイル
ホーム
共有
表示
« header › n223_move___
n223_move___の検索
n161_sqDistan
n162_bonaDir_
n163_sqDistan
n165_movStack
n220_position
n223_move___
n225_movPhase
n350_pieceTyp
名前
更新日時
種類
n223_040_nodeType.hpp 2016/04/27 3:36 C/C++ Header
n223_060_stats.hpp 2016/04/27 3:36 C/C++ Header
n223_200_depth.hpp 2016/04/27 3:36 C/C++ Header
n223_300_moveAndScoreIndex.hpp 2016/04/27 3:36 C/C++ Header
n223_500_flashlight.hpp 2016/04/27 3:36 C/C++ Header
5 個の項目

15

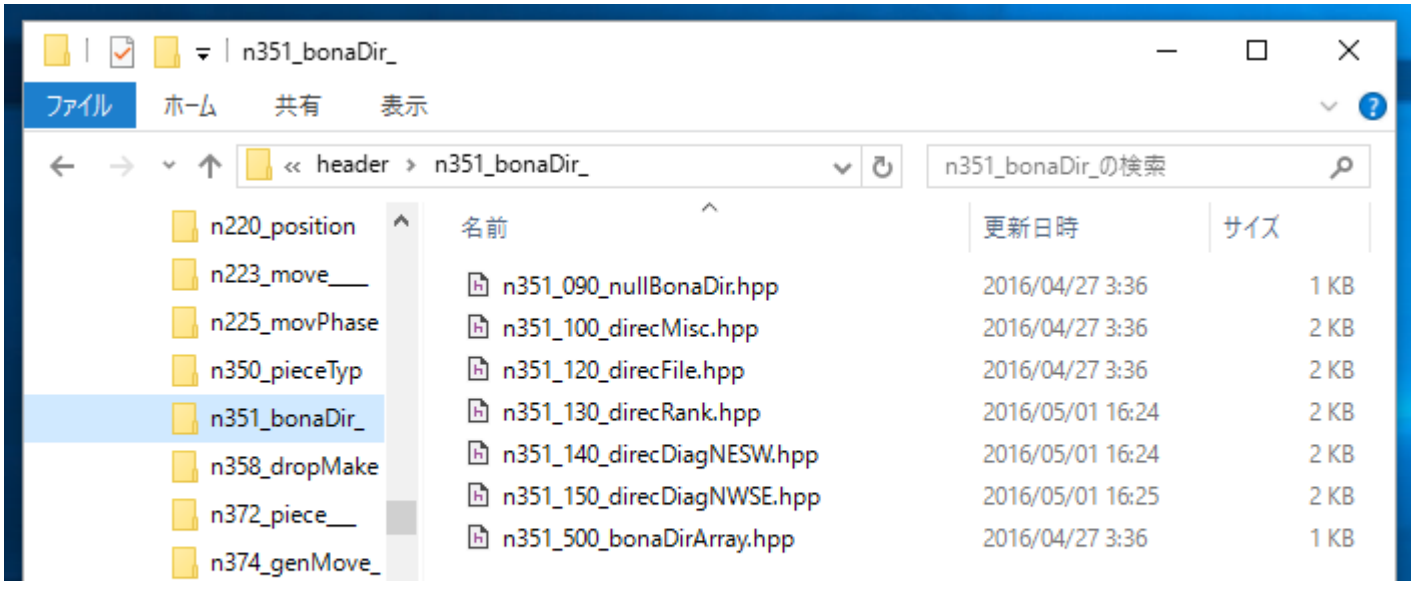

n225_movPhase
ファイル
ホーム
共有
表示
« header › n225_movPhase
n225_movPhaseの検索
n220_position
n223_move___
n225_movPhase
n350_pieceTyp
n351_bonaDir_
名前
更新日時
種類
n225_050_generateMovePhase.hpp 2016/04/27 3:36 C/C++ Header
1 個の項目

16

n350_pieceTyp

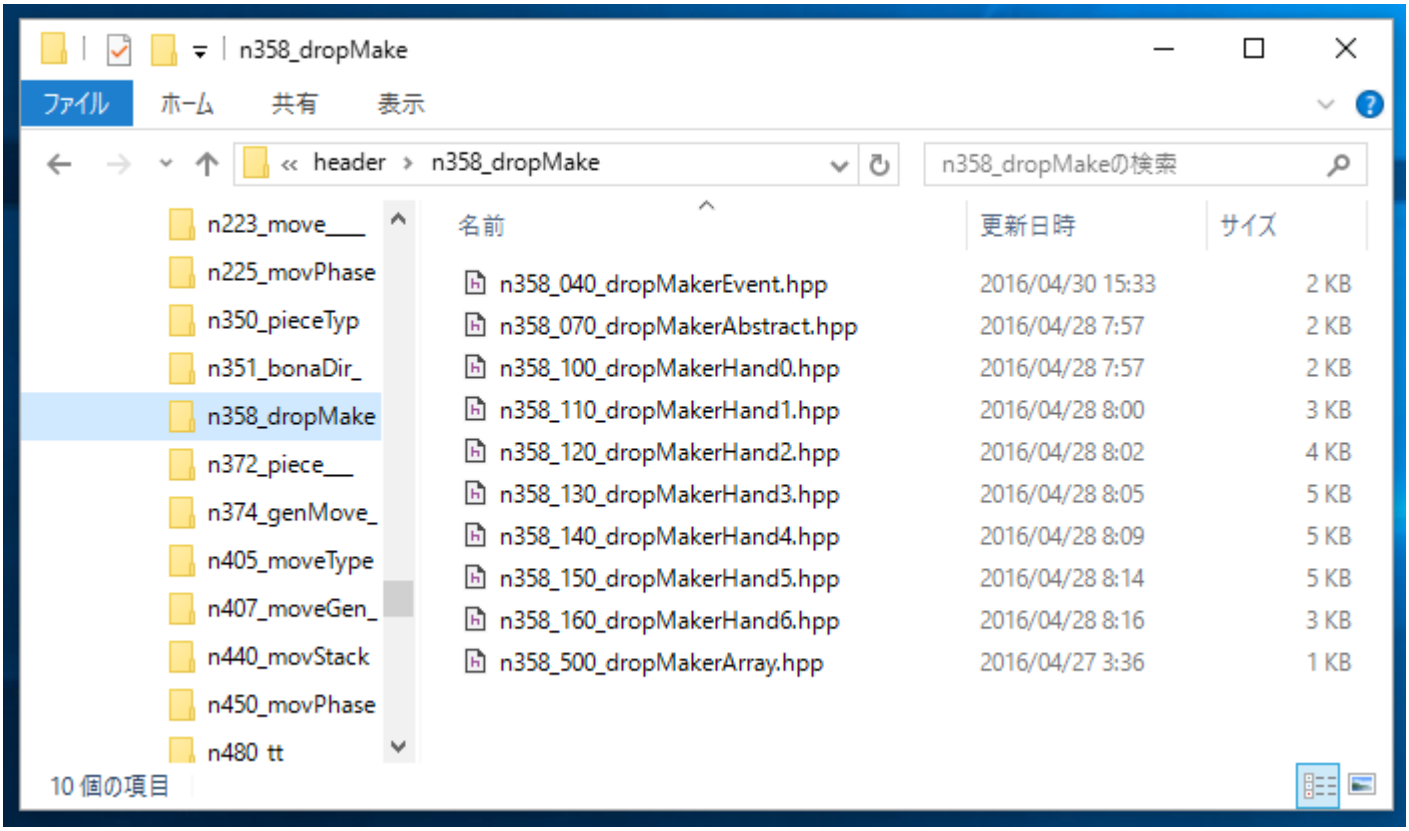

ファイル
ホーム
共有
表示
« header › n350_pieceTyp
n350_pieceTypの検索
n220_position
n223_move___
n225_movPhase
n350_pieceTyp
n351_bonaDir_
n358_dropMake
n372_piece___
n374_genMove_
n405_moveType
n407_moveGen_
n440_movStack
n450_movPhase
n480_tt_____
n520_evaluate
n560_timeMng_
n600_book___
n640_searcher
n680_egOption
n720_usi____
n755_sword___
n760_thread__
n800_learn___
n883_nodeType
名前
更新日時
サイズ
n350_030_makePromoteMove.hpp 2016/04/30 23:55 3 KB
n350_040_ptEvent.hpp 2016/04/27 3:36 1 KB
n350_045_pieceTypeSeeEvent.hpp 2016/04/27 3:36 1 KB
n350_070_ptAbstract.hpp 2016/05/01 2:59 2 KB
n350_080_ptPromote.hpp 2016/05/01 2:12 2 KB
n350_100_ptOccupied.hpp 2016/05/01 1:50 2 KB
n350_110_ptPawn.hpp 2016/05/01 1:51 3 KB
n350_120_ptLance.hpp 2016/05/01 2:06 3 KB
n350_130_ptKnight.hpp 2016/05/01 2:06 3 KB
n350_140_ptSilver.hpp 2016/05/01 2:06 3 KB
n350_150_ptBishop.hpp 2016/05/01 2:09 3 KB
n350_160_ptRook.hpp 2016/05/01 1:55 3 KB
n350_170_ptGold.hpp 2016/05/01 1:56 3 KB
n350_180_ptKing.hpp 2016/05/01 1:56 2 KB
n350_190_ptProPawn.hpp 2016/05/01 1:57 3 KB
n350_200_ptProLance.hpp 2016/05/01 1:57 3 KB
n350_210_ptProKnight.hpp 2016/05/01 1:58 3 KB
n350_220_ptProSilver.hpp 2016/05/01 1:58 3 KB
n350_230_ptHorse.hpp 2016/05/01 1:59 3 KB
n350_240_ptDragon.hpp 2016/05/01 1:59 3 KB
n350_250_ptPieceTypeNum.hpp 2016/05/01 1:59 2 KB
n350_260_ptGoldHorseDragon.hpp 2016/05/01 2:00 2 KB
n350_500_ptPrograms.hpp 2016/04/28 9:09 2 KB
23 個の項目

17

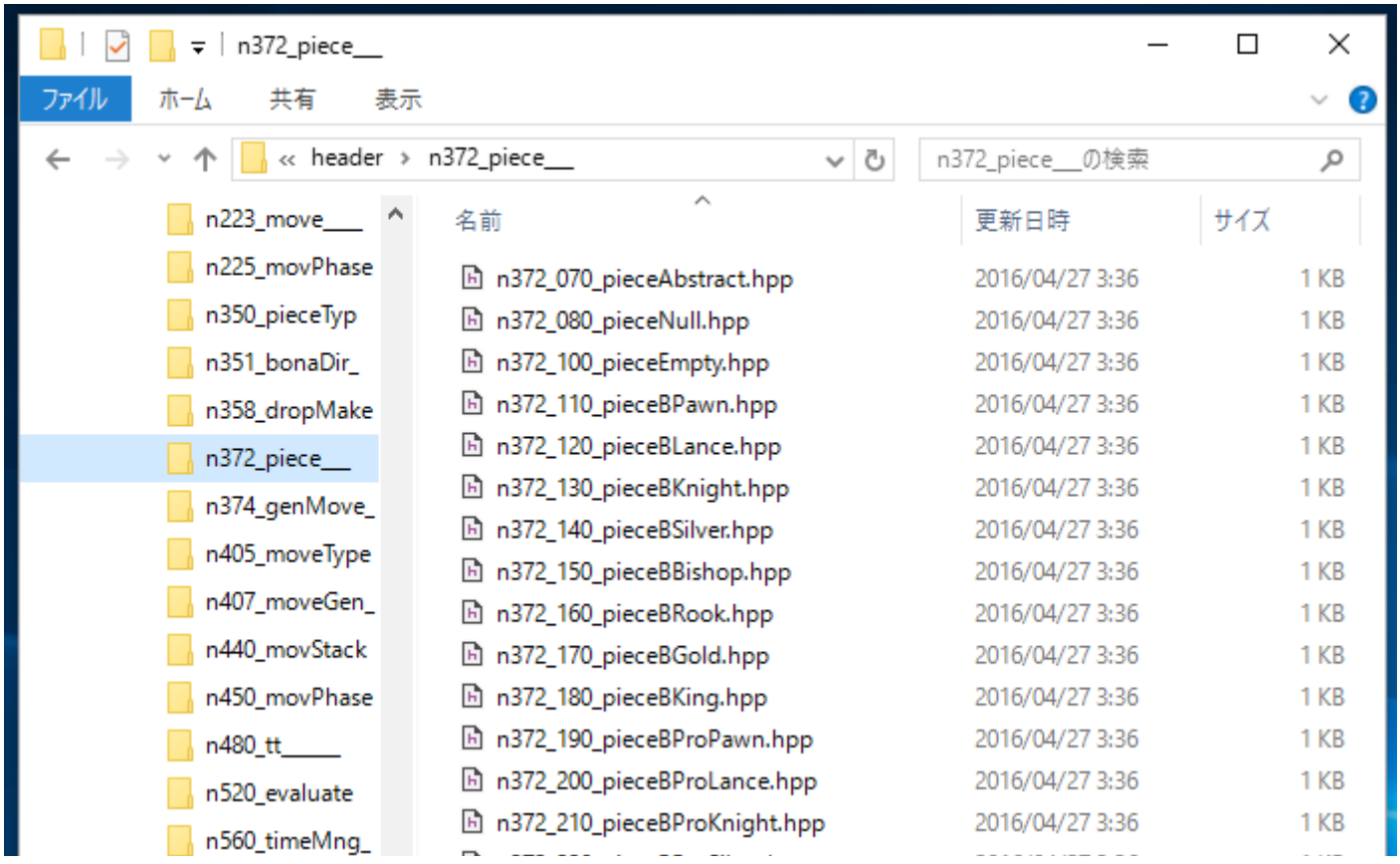

n351_bonaDir_
ファイル
ホーム
共有
表示
« header › n351_bonaDir_
n351_bonaDir_の検索
n220_position
n223_move___
n225_movPhase
n350_pieceTyp
n351_bonaDir_
n358_dropMake
n372_piece___
n374_genMove_
名前
更新日時
サイズ
n351_090_nullBonaDir.hpp 2016/04/27 3:36 1 KB
n351_100_direcMisc.hpp 2016/04/27 3:36 2 KB
n351_120_direcFile.hpp 2016/04/27 3:36 2 KB
n351_130_direcRank.hpp 2016/05/01 16:24 2 KB
n351_140_direcDiagNESW.hpp 2016/05/01 16:24 2 KB
n351_150_direcDiagNWSE.hpp 2016/05/01 16:25 2 KB
n351_500_bonaDirArray.hpp 2016/04/27 3:36 1 KB

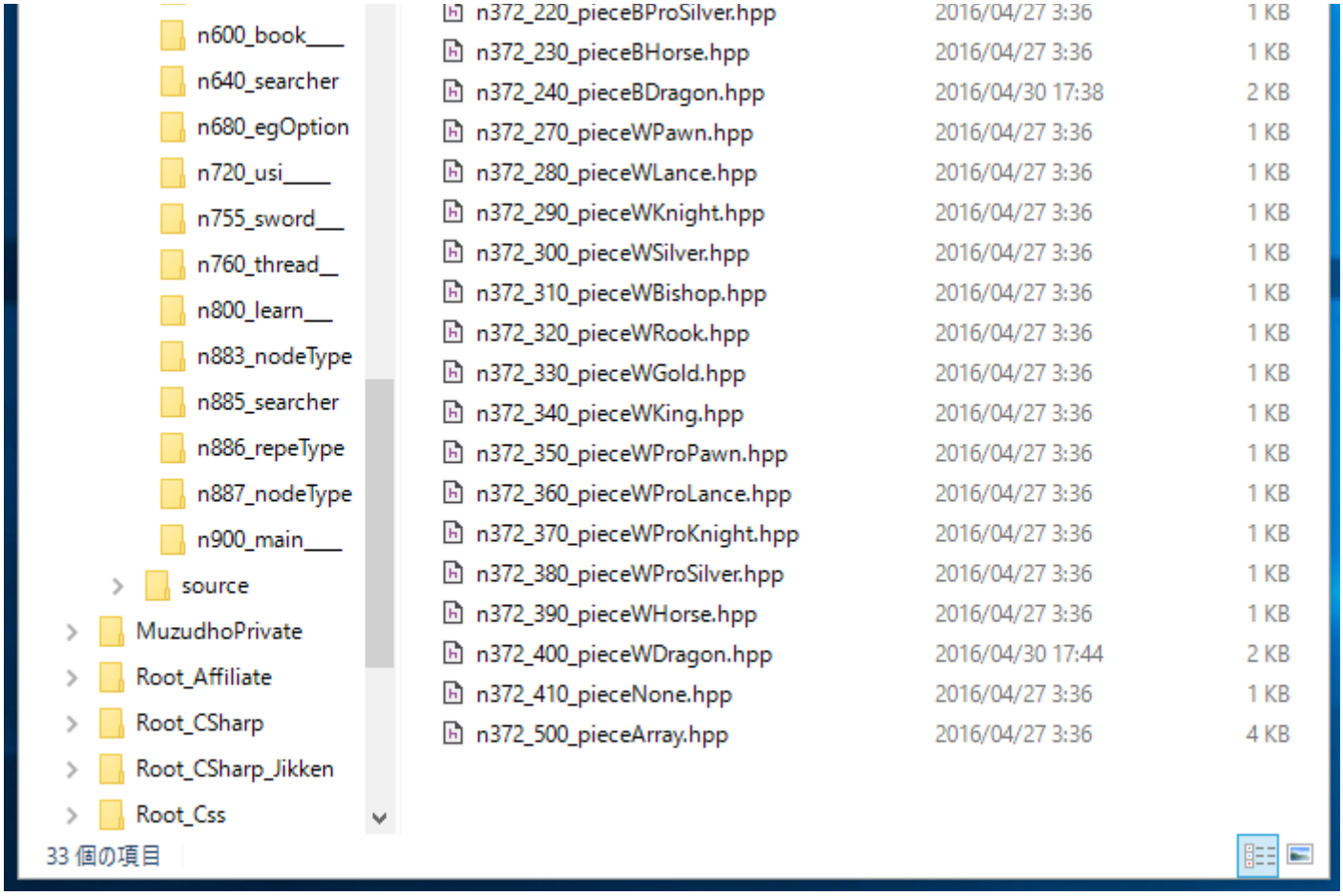
n405_moveType
7 個の項目

18

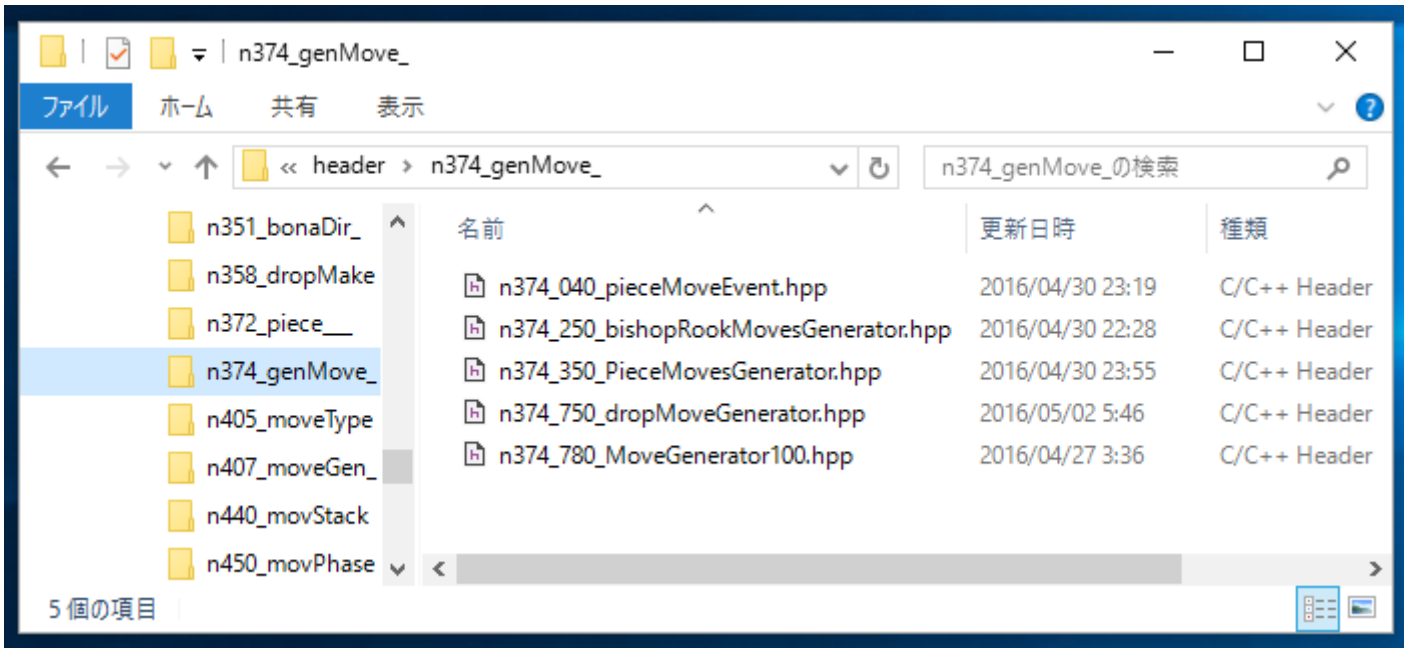

n358_dropMake
ファイル
ホーム
共有
表示
« header › n358_dropMake
n358_dropMakeの検索
n223_move___
n225_movPhase
n350_pieceTyp
n351_bonaDir_
n358_dropMake
n372_piece___
n374_genMove_
n405_moveType
n407_moveGen_
n440_movStack
n450_movPhase
n480 tt
名前
更新日時
サイズ
n358_040_dropMakerEvent.hpp 2016/04/30 15:33 2 KB
n358_070_dropMakerAbstract.hpp 2016/04/28 7:57 2 KB
n358_100_dropMakerHand0.hpp 2016/04/28 7:57 2 KB
n358_110_dropMakerHand1.hpp 2016/04/28 8:00 3 KB
n358_120_dropMakerHand2.hpp 2016/04/28 8:02 4 KB
n358_130_dropMakerHand3.hpp 2016/04/28 8:05 5 KB
n358_140_dropMakerHand4.hpp 2016/04/28 8:09 5 KB
n358_150_dropMakerHand5.hpp 2016/04/28 8:14 5 KB
n358_160_dropMakerHand6.hpp 2016/04/28 8:16 3 KB
n358_500_dropMakerArray.hpp 2016/04/27 3:36 1 KB
10 個の項目

19

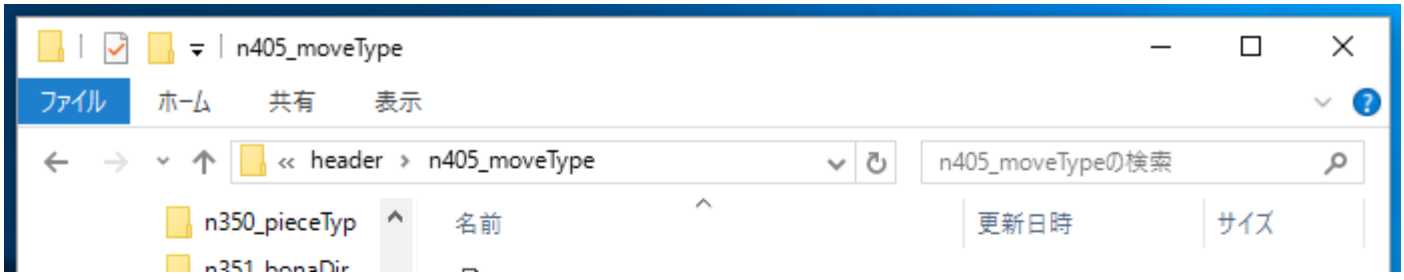

n372_piece___
ファイル
ホーム
共有
表示
« header › n372_piece___
n372_piece___の検索
n223_move___
n225_movPhase
n350_pieceTyp
n351_bonaDir_
n358_dropMake
n372_piece___
n374_genMove_
n405_moveType
n407_moveGen_
n440_movStack
n450_movPhase
n480_tt_____
n520_evaluate
n560_timeMng_
名前
更新日時
サイズ
n372_070_pieceAbstract.hpp 2016/04/27 3:36 1 KB
n372_080_pieceNull.hpp 2016/04/27 3:36 1 KB
n372_100_pieceEmpty.hpp 2016/04/27 3:36 1 KB
n372_110_pieceBPawn.hpp 2016/04/27 3:36 1 KB
n372_120_pieceBLance.hpp 2016/04/27 3:36 1 KB
n372_130_pieceBKnight.hpp 2016/04/27 3:36 1 KB
n372_140_pieceBSilver.hpp 2016/04/27 3:36 1 KB
n372_150_pieceBBishop.hpp 2016/04/27 3:36 1 KB
n372_160_pieceBRook.hpp 2016/04/27 3:36 1 KB
n372_170_pieceBGold.hpp 2016/04/27 3:36 1 KB
n372_180_pieceBKing.hpp 2016/04/27 3:36 1 KB
n372_190_pieceBProPawn.hpp 2016/04/27 3:36 1 KB
n372_200_pieceBProLance.hpp 2016/04/27 3:36 1 KB
n372_210_pieceBProKnight.hpp 2016/04/27 3:36 1 KB

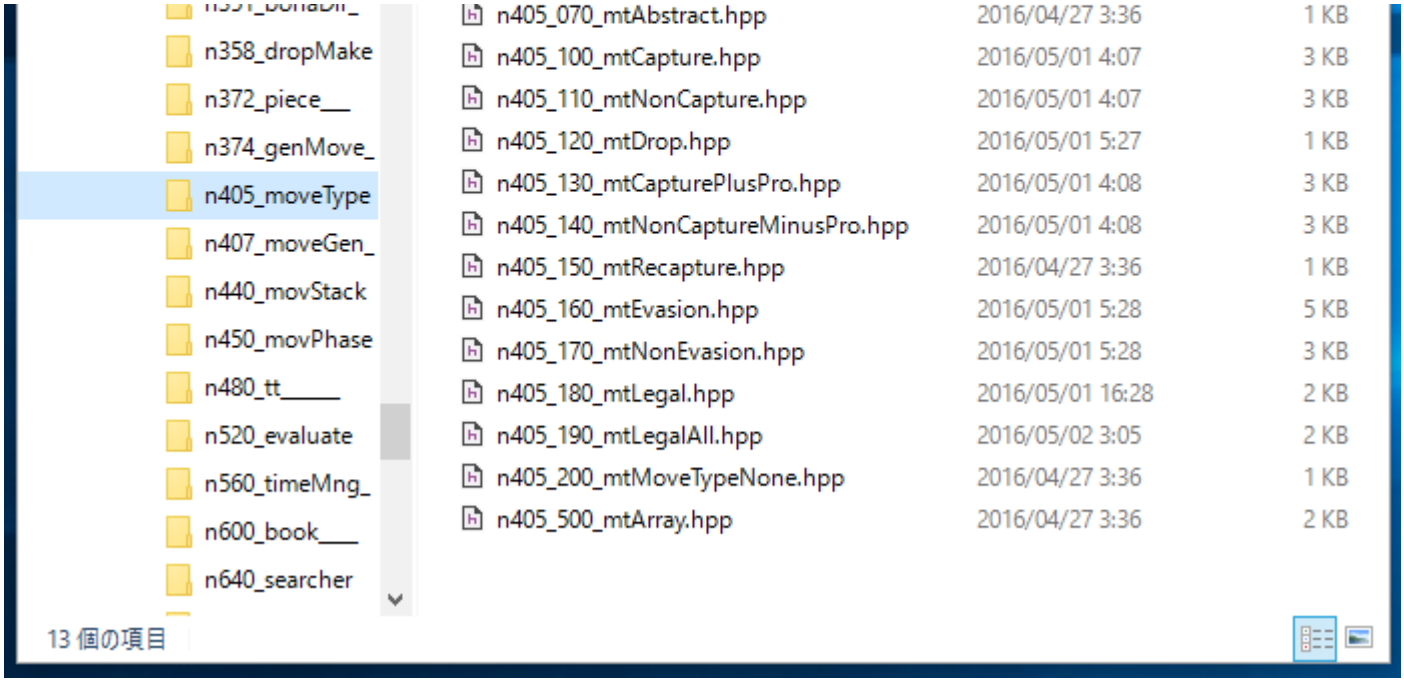

n600_book___
n640_searcher
n680_egOption
n720_usi____
n755_sword___
n760_thread__
n800_learn___
n883_nodeType
n885_searcher
n886_repeType
n887_nodeType
n900_main___
source
MuzudhoPrivate
Root_Affiliate
Root_CSharp
Root_CSharp_Jikken
Root_Css
33 個の項目
n372_220_pieceBProSilver.hpp 2016/04/27 3:36 1 KB
n372_230_pieceBHorse.hpp 2016/04/27 3:36 1 KB
n372_240_pieceBDragon.hpp 2016/04/30 17:38 2 KB
n372_270_pieceWPawn.hpp 2016/04/27 3:36 1 KB
n372_280_pieceWLance.hpp 2016/04/27 3:36 1 KB
n372_290_pieceWKnight.hpp 2016/04/27 3:36 1 KB
n372_300_pieceWSilver.hpp 2016/04/27 3:36 1 KB
n372_310_pieceWBishop.hpp 2016/04/27 3:36 1 KB
n372_320_pieceWRook.hpp 2016/04/27 3:36 1 KB
n372_330_pieceWGold.hpp 2016/04/27 3:36 1 KB
n372_340_pieceWKing.hpp 2016/04/27 3:36 1 KB
n372_350_pieceWProPawn.hpp 2016/04/27 3:36 1 KB
n372_360_pieceWProLance.hpp 2016/04/27 3:36 1 KB
n372_370_pieceWProKnight.hpp 2016/04/27 3:36 1 KB
n372_380_pieceWProSilver.hpp 2016/04/27 3:36 1 KB
n372_390_pieceWHorse.hpp 2016/04/27 3:36 1 KB
n372_400_pieceWDragon.hpp 2016/04/30 17:44 2 KB
n372_410_pieceNone.hpp 2016/04/27 3:36 1 KB
n372_500_pieceArray.hpp 2016/04/27 3:36 4 KB

20

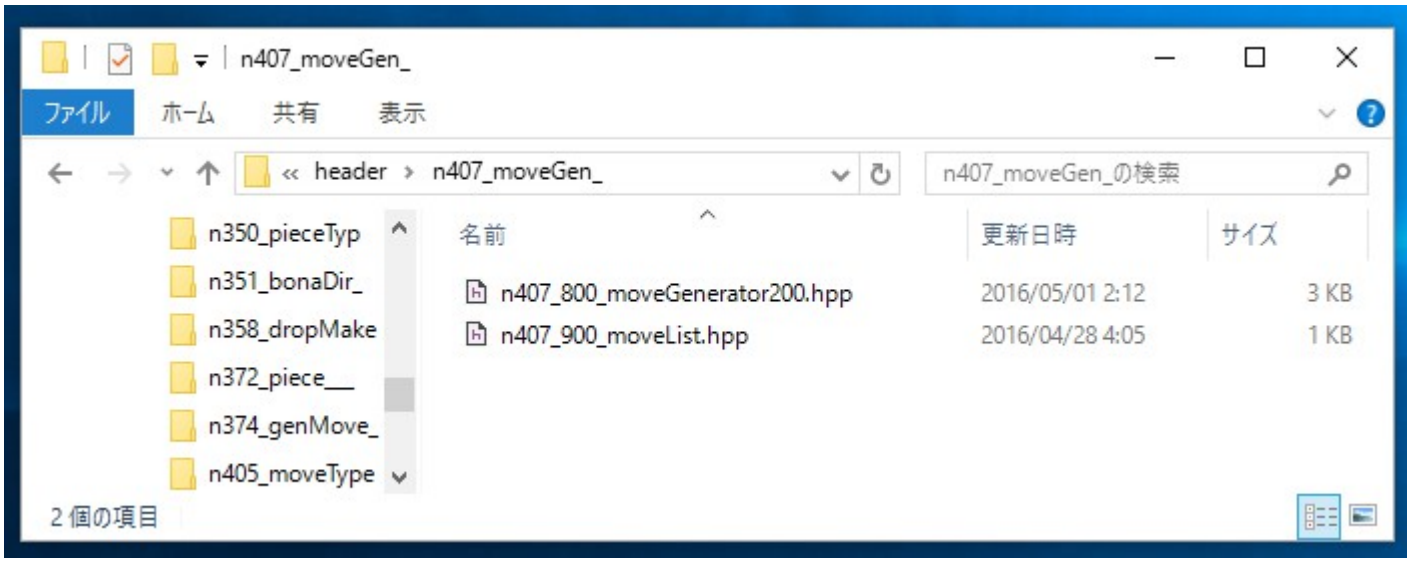

n374_genMove_
ファイル
ホーム
共有
表示
« header › n374_genMove_
n374_genMove_の検索
n351_bonaDir_
n358_dropMake
n372_piece___
n374_genMove_
n405_moveType
n407_moveGen_
n440_movStack
n450_movPhase
5 個の項目
名前
更新日時
種類
n374_040_pieceMoveEvent.hpp 2016/04/30 23:19 C/C++ Header
n374_250_bishopRookMovesGenerator.hpp 2016/04/30 22:28 C/C++ Header
n374_350_PieceMovesGenerator.hpp 2016/04/30 23:55 C/C++ Header
n374_750_dropMoveGenerator.hpp 2016/05/02 5:46 C/C++ Header
n374_780_MoveGenerator100.hpp 2016/04/27 3:36 C/C++ Header

21

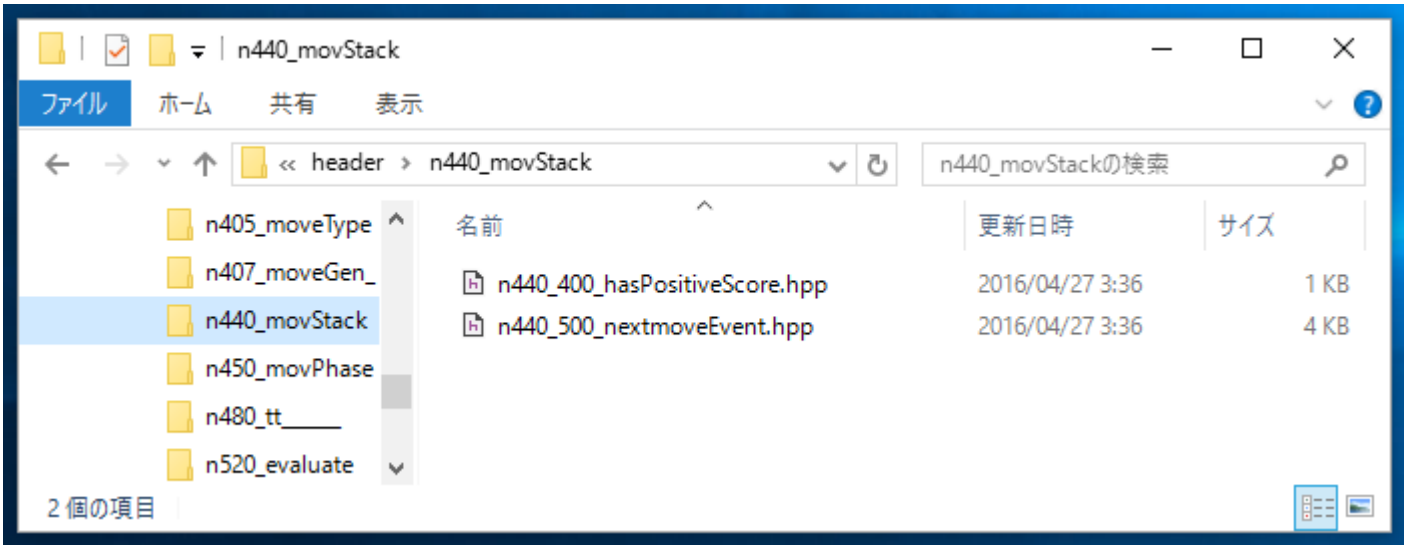

n405_moveType
ファイル
ホーム
共有
表示
« header › n405_moveType
n405_moveTypeの検索
n350_pieceTyp
名前
更新日時
サイズ


n358_dropMake
n372_piece___
n374_genMove_
n405_moveType
n407_moveGen_
n440_movStack
n450_movPhase
n480_tt_____
n520_evaluate
n560_timeMng_
n600_book___
n640_searcher
13 個の項目
n405_070_mtAbstract.hpp 2016/04/27 3:36 1 KB
n405_100_mtCapture.hpp 2016/05/01 4:07 3 KB
n405_110_mtNonCapture.hpp 2016/05/01 4:07 3 KB
n405_120_mtDrop.hpp 2016/05/01 5:27 1 KB
n405_130_mtCapturePlusPro.hpp 2016/05/01 4:08 3 KB
n405_140_mtNonCaptureMinusPro.hpp 2016/05/01 4:08 3 KB
n405_150_mtRecapture.hpp 2016/04/27 3:36 1 KB
n405_160_mtEvasion.hpp 2016/05/01 5:28 5 KB
n405_170_mtNonEvasion.hpp 2016/05/01 5:28 3 KB
n405_180_mtLegal.hpp 2016/05/01 16:28 2 KB
n405_190_mtLegalAll.hpp 2016/05/02 3:05 2 KB
n405_200_mtMoveTypeNone.hpp 2016/04/27 3:36 1 KB
n405_500_mtArray.hpp 2016/04/27 3:36 2 KB

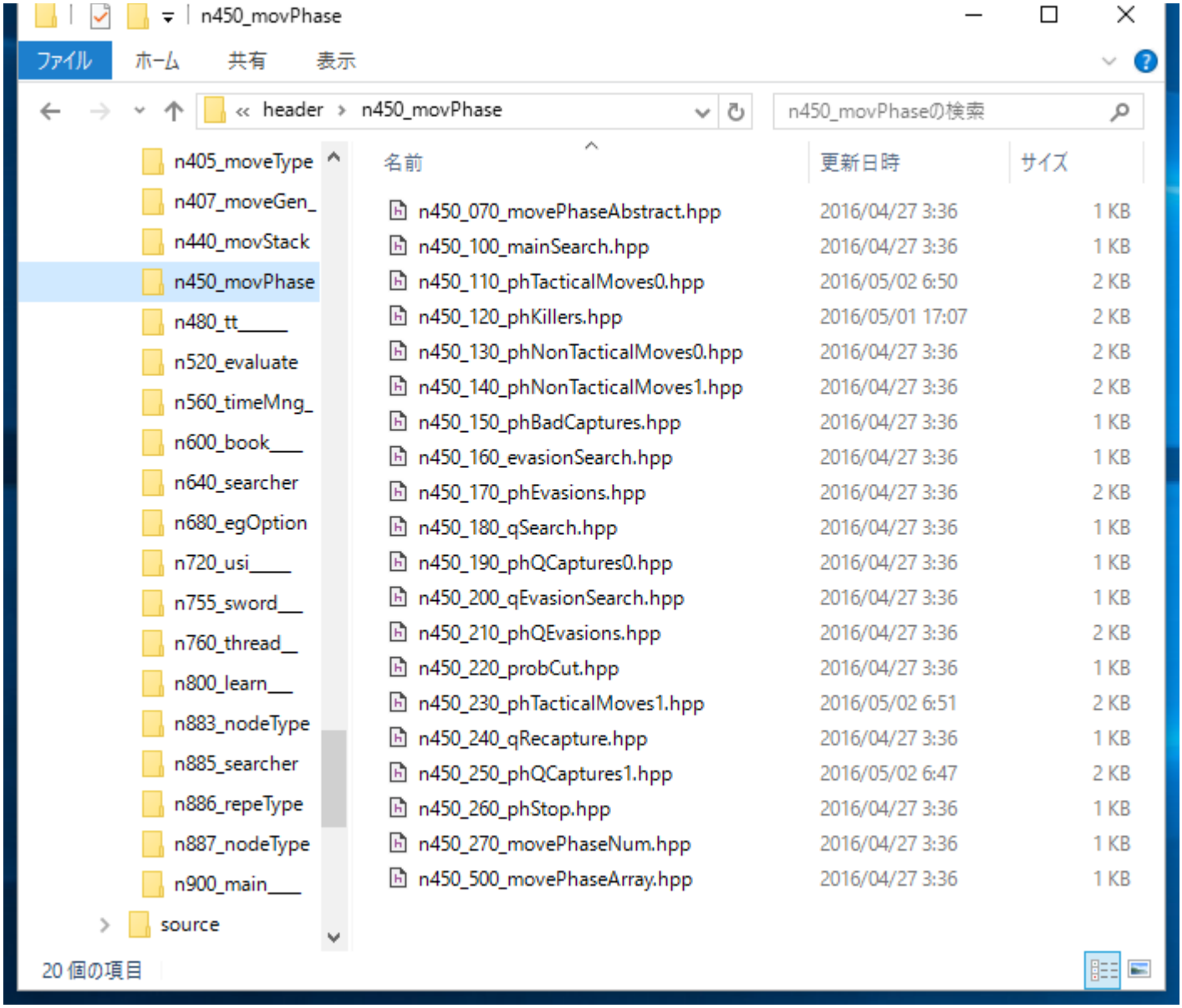
22


n407_moveGen_
ファイル
ホーム
共有
表示
« header › n407_moveGen_
n407_moveGen_の検索
n350_pieceTyp
n351_bonaDir_
n358_dropMake
n372_piece___
n374_genMove_
n405_moveType
名前
更新日時
サイズ
n407_800_moveGenerator200.hpp 2016/05/01 2:12 3 KB
n407_900_moveList.hpp 2016/04/28 4:05 1 KB
2 個の項目

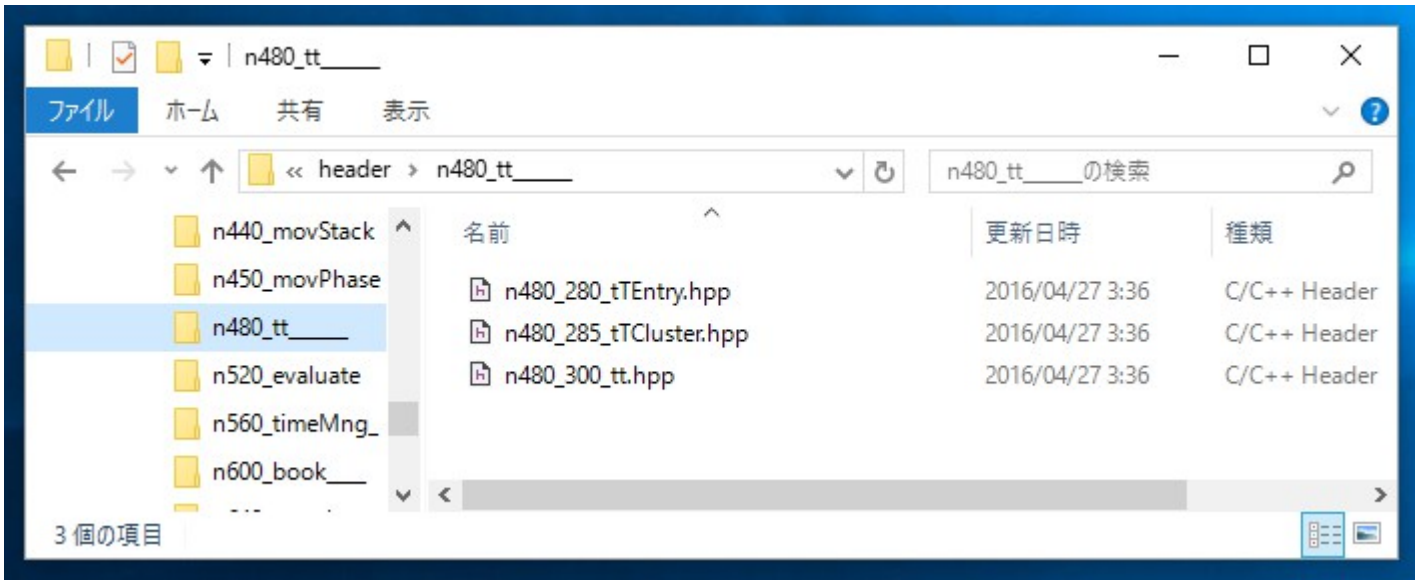
23


n440_movStack
ファイル
ホーム
共有
表示
« header › n440_movStack
n440_movStackの検索
n405_moveType
n407_moveGen_
n440_movStack
n450_movPhase
n480_tt_____
n520_evaluate
名前
更新日時
サイズ
n440_400_hasPositiveScore.hpp 2016/04/27 3:36 1 KB
n440_500_nextmoveEvent.hpp 2016/04/27 3:36 4 KB
2 個の項目

24

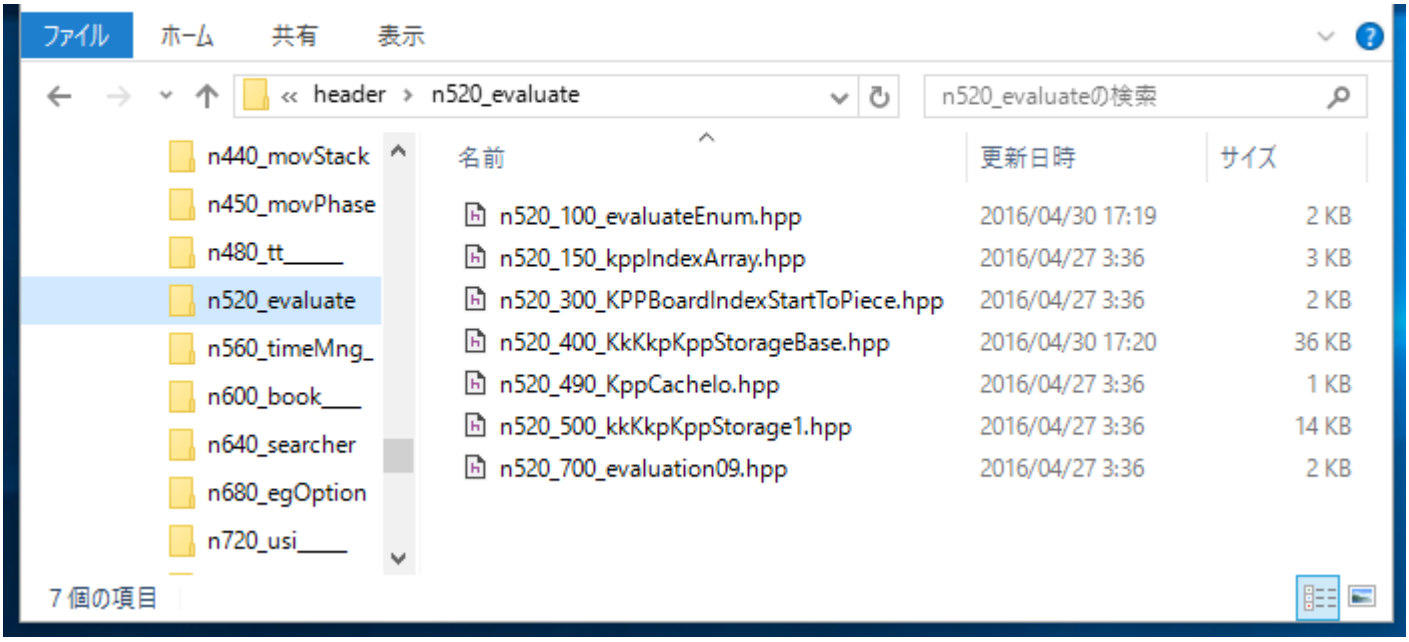

n450_movPhase
ファイル
ホーム
共有
表示
« header › n450_movPhase
n450_movPhaseの検索
n405_moveType
n407_moveGen_
n440_movStack
n450_movPhase
n480_tt_____
n520_evaluate
n560_timeMng_
n600_book___
n640_searcher
n680_egOption
n720_usi____
n755_sword__
n760_thread_
n800_learn__
n883_nodeType
n885_searcher
n886_repeType
n887_nodeType
n900_main___
source
名前
更新日時
サイズ
n450_070_movePhaseAbstract.hpp 2016/04/27 3:36 1 KB
n450_100_mainSearch.hpp 2016/04/27 3:36 1 KB
n450_110_phTacticalMoves0.hpp 2016/05/02 6:50 2 KB
n450_120_phKillers.hpp 2016/05/01 17:07 2 KB
n450_130_phNonTacticalMoves0.hpp 2016/04/27 3:36 2 KB
n450_140_phNonTacticalMoves1.hpp 2016/04/27 3:36 2 KB
n450_150_phBadCaptures.hpp 2016/04/27 3:36 1 KB
n450_160_evasionSearch.hpp 2016/04/27 3:36 1 KB
n450_170_phEvasions.hpp 2016/04/27 3:36 2 KB
n450_180_qSearch.hpp 2016/04/27 3:36 1 KB
n450_190_phQCaptures0.hpp 2016/04/27 3:36 1 KB
n450_200_qEvasionSearch.hpp 2016/04/27 3:36 1 KB
n450_210_phQEvasions.hpp 2016/04/27 3:36 2 KB
n450_220_probCut.hpp 2016/04/27 3:36 1 KB
n450_230_phTacticalMoves1.hpp 2016/05/02 6:51 2 KB
n450_240_qRecapture.hpp 2016/04/27 3:36 1 KB
n450_250_phQCaptures1.hpp 2016/05/02 6:47 2 KB
n450_260_phStop.hpp 2016/04/27 3:36 1 KB
n450_270_movePhaseNum.hpp 2016/04/27 3:36 1 KB
n450_500_movePhaseArray.hpp 2016/04/27 3:36 1 KB
20 個の項目

25

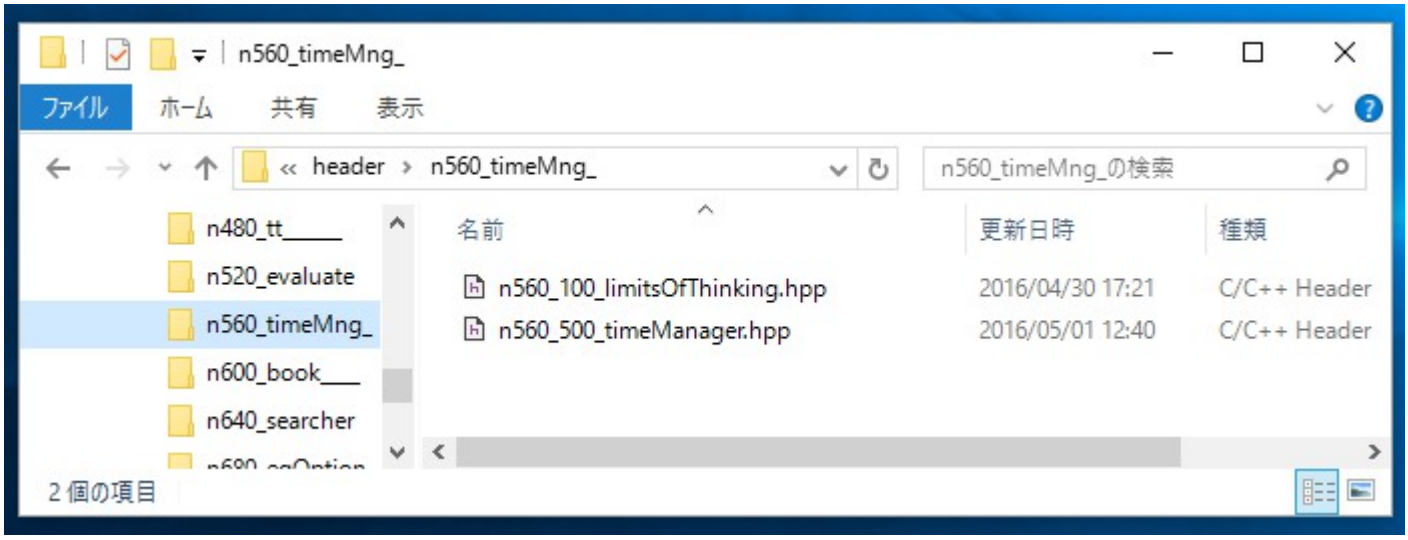

n480_tt_____
ファイル
ホーム
共有
表示
« header › n480_tt_____
n480_tt_____の検索
n440_movStack
n450_movPhase
n480_tt_____
n520_evaluate
n560_timeMng_
n600_book___
名前
更新日時
種類
n480_280_tTEntry.hpp 2016/04/27 3:36 C/C++ Header
n480_285_tTCluster.hpp 2016/04/27 3:36 C/C++ Header
n480_300_tt.hpp 2016/04/27 3:36 C/C++ Header
3 個の項目

26

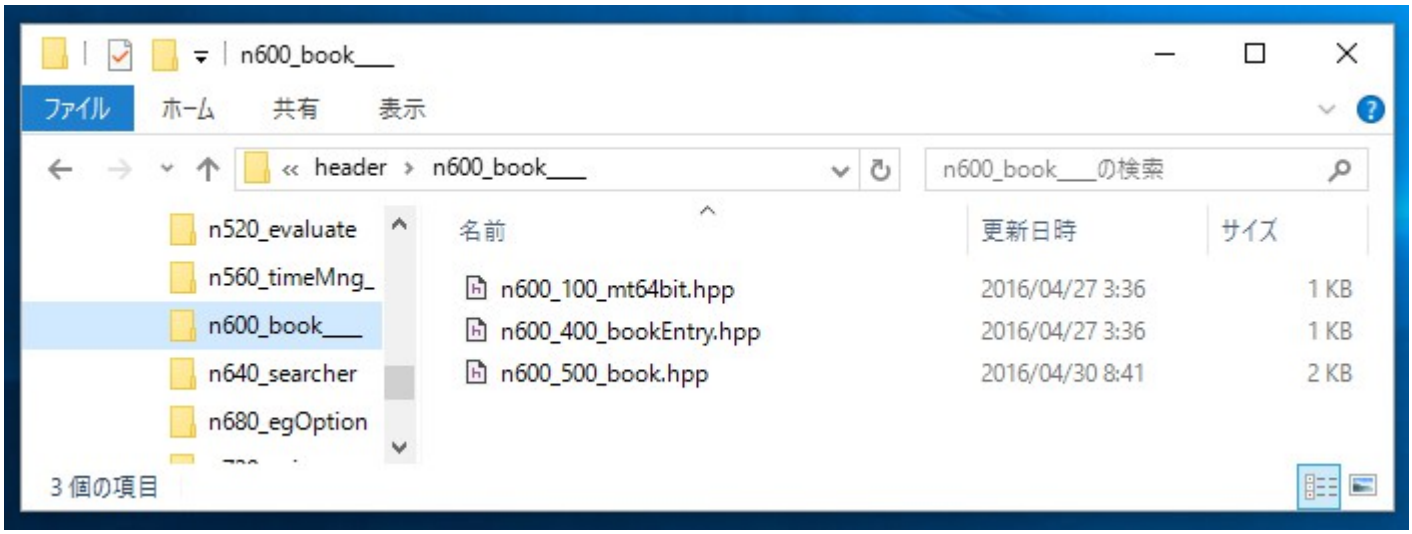
n520_evaluate


ファイル
ホーム
共有
表示
« header › n520_evaluate
n520_evaluateの検索
n440_movStack
n450_movPhase
n480_tt_____
n520_evaluate
n560_timeMng_
n600_book___
n640_searcher
n680_egOption
n720_usi____
名前
更新日時
サイズ
n520_100_evaluateEnum.hpp 2016/04/30 17:19 2 KB
n520_150_kppIndexArray.hpp 2016/04/27 3:36 3 KB
n520_300_KPPBoardIndexStartToPiece.hpp 2016/04/27 3:36 2 KB
n520_400_KkKkpKppStorageBase.hpp 2016/04/30 17:20 36 KB
n520_490_KppCachelo.hpp 2016/04/27 3:36 1 KB
n520_500_kkKkpKppStorage1.hpp 2016/04/27 3:36 14 KB
n520_700_evaluation09.hpp 2016/04/27 3:36 2 KB
7 個の項目

27

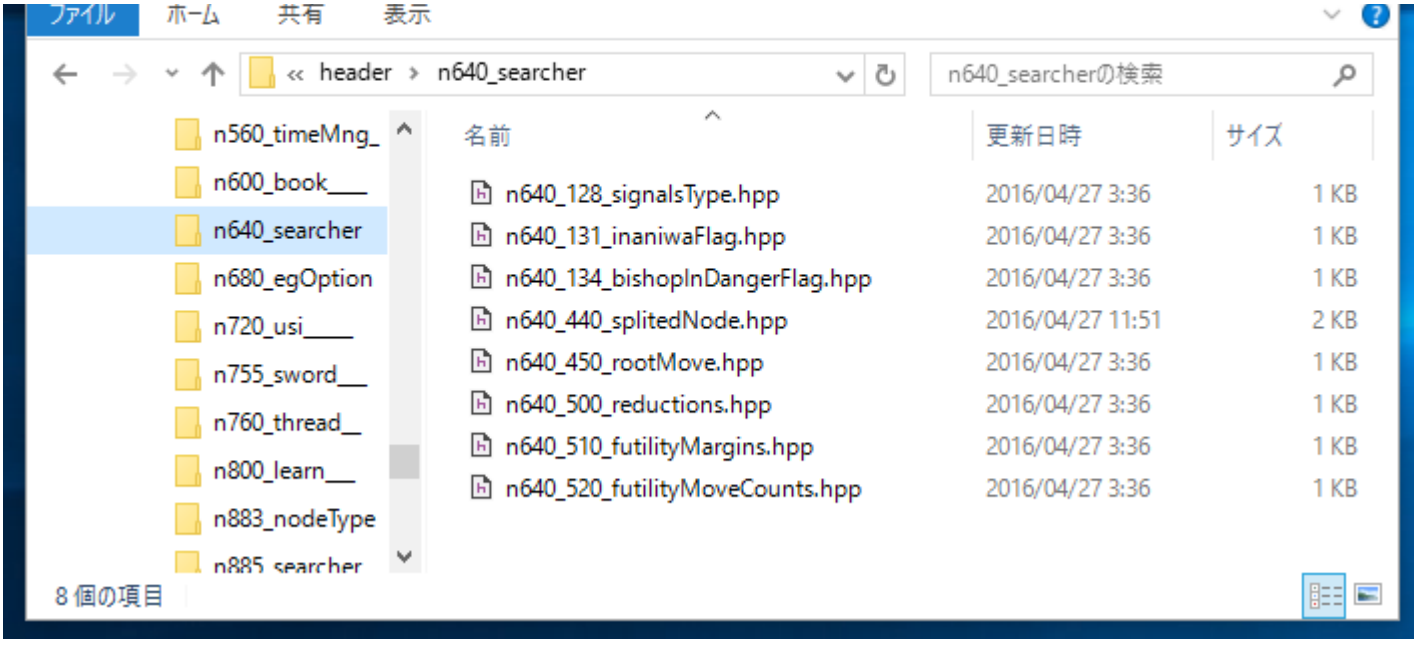

n560_timeMng_
ファイル
ホーム
共有
表示
« header › n560_timeMng_
n560_timeMng_の検索
n480_tt_____
n520_evaluate
n560_timeMng_
n600_book___
n640_searcher
名前
更新日時
種類
n560_100_limitsOfThinking.hpp 2016/04/30 17:21 C/C++ Header
n560_500_timeManager.hpp 2016/05/01 12:40 C/C++ Header
2 個の項目

28

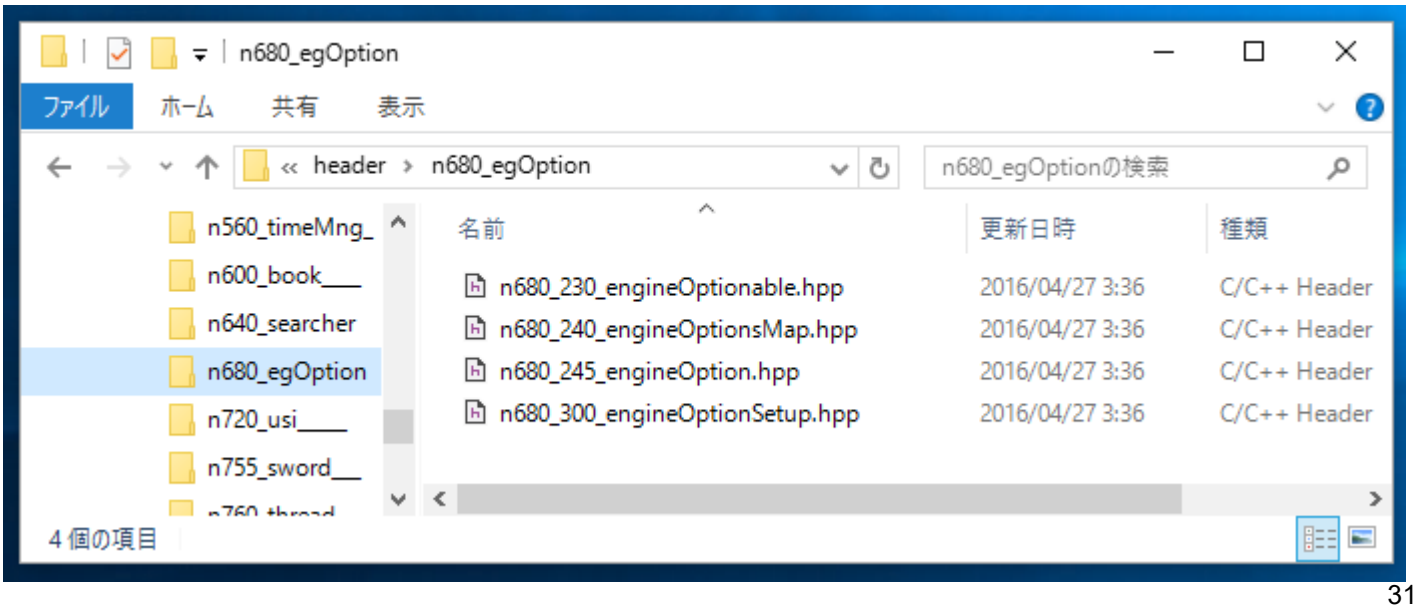

n600_book___
ファイル
ホーム
共有
表示
« header › n600_book___
n600_book___の検索
n520_evaluate
n560_timeMng_
n600_book___
n640_searcher
n680_egOption
名前
更新日時
サイズ
n600_100_mt64bit.hpp 2016/04/27 3:36 1 KB
n600_400_bookEntry.hpp 2016/04/27 3:36 1 KB
n600_500_book.hpp 2016/04/30 8:41 2 KB
3 個の項目

29

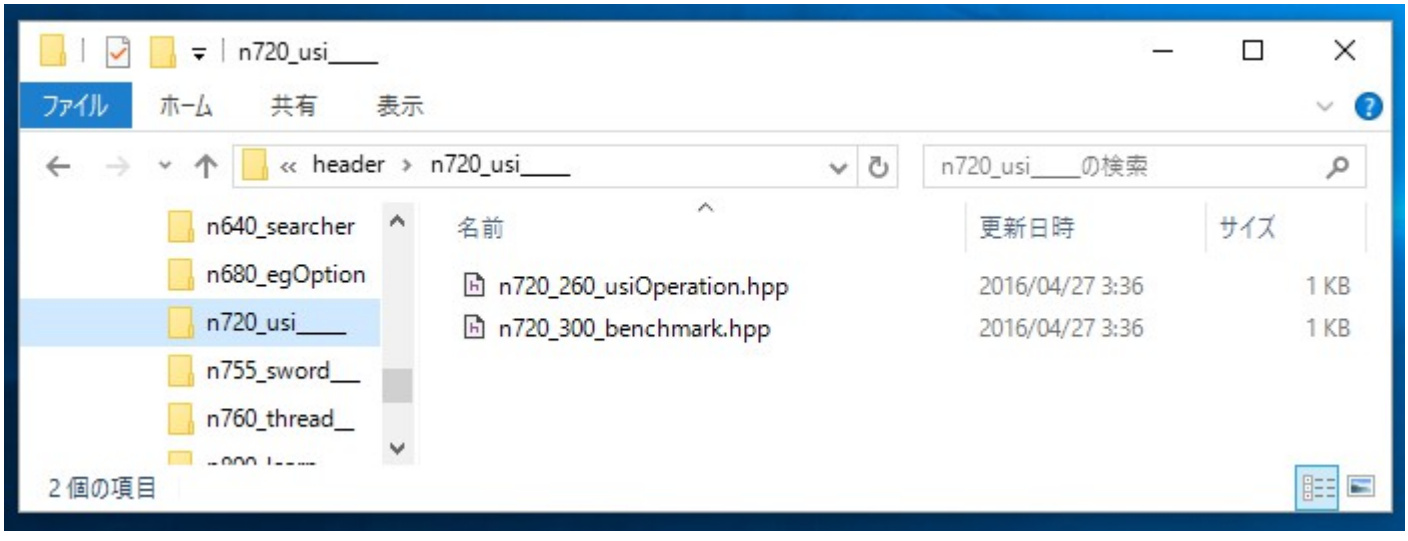
n640_searcher


ファイル
ホーム
共有
表示
« header › n640_searcher
n640_searcherの検索
n560_timeMng_
n600_book___
n640_searcher
n680_egOption
n720_usi____
n755_sword__
n760_thread_
n800_learn__
n883_nodeType
名前
更新日時
サイズ
n640_128_signalsType.hpp 2016/04/27 3:36 1 KB
n640_131_inaniwaFlag.hpp 2016/04/27 3:36 1 KB
n640_134_bishopInDangerFlag.hpp 2016/04/27 3:36 1 KB
n640_440_splitedNode.hpp 2016/04/27 11:51 2 KB
n640_450_rootMove.hpp 2016/04/27 3:36 1 KB
n640_500_reductions.hpp 2016/04/27 3:36 1 KB
n640_510_futilityMargins.hpp 2016/04/27 3:36 1 KB
n640_520_futilityMoveCounts.hpp 2016/04/27 3:36 1 KB
8 個の項目

30

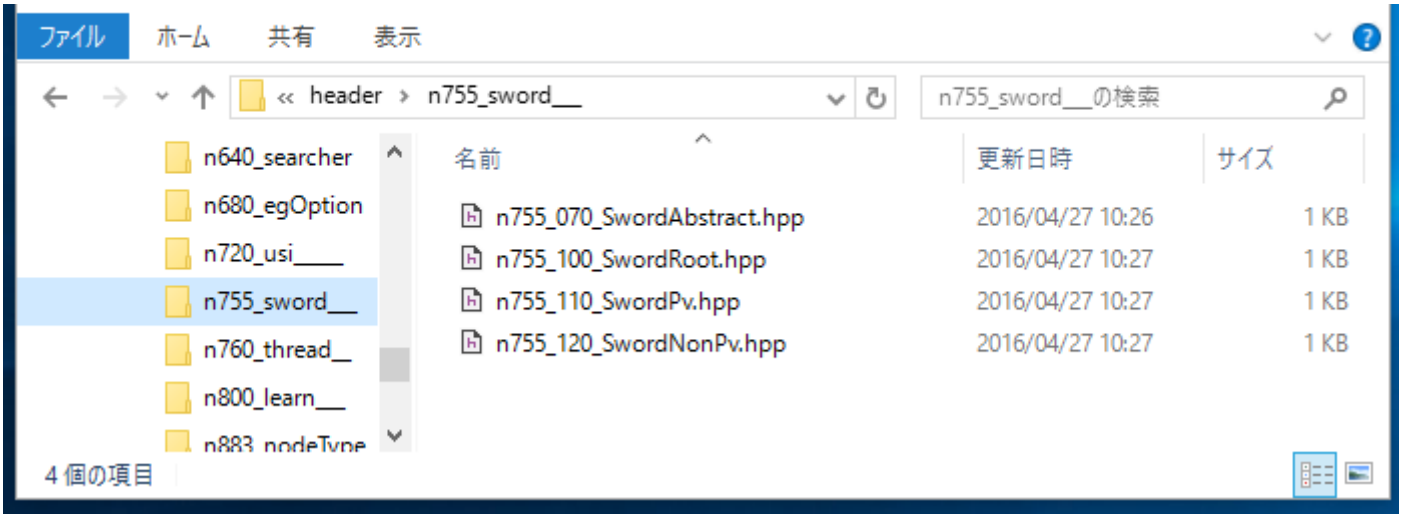

n680_egOption
ファイル
ホーム
共有
表示
« header › n680_egOption
n680_egOptionの検索
n560_timeMng_
n600_book___
n640_searcher
n680_egOption
n720_usi____
n755_sword__
名前
更新日時
種類
n680_230_engineOptionable.hpp 2016/04/27 3:36 C/C++ Header
n680_240_engineOptionsMap.hpp 2016/04/27 3:36 C/C++ Header
n680_245_engineOption.hpp 2016/04/27 3:36 C/C++ Header
n680_300_engineOptionSetup.hpp 2016/04/27 3:36 C/C++ Header
4 個の項目

31

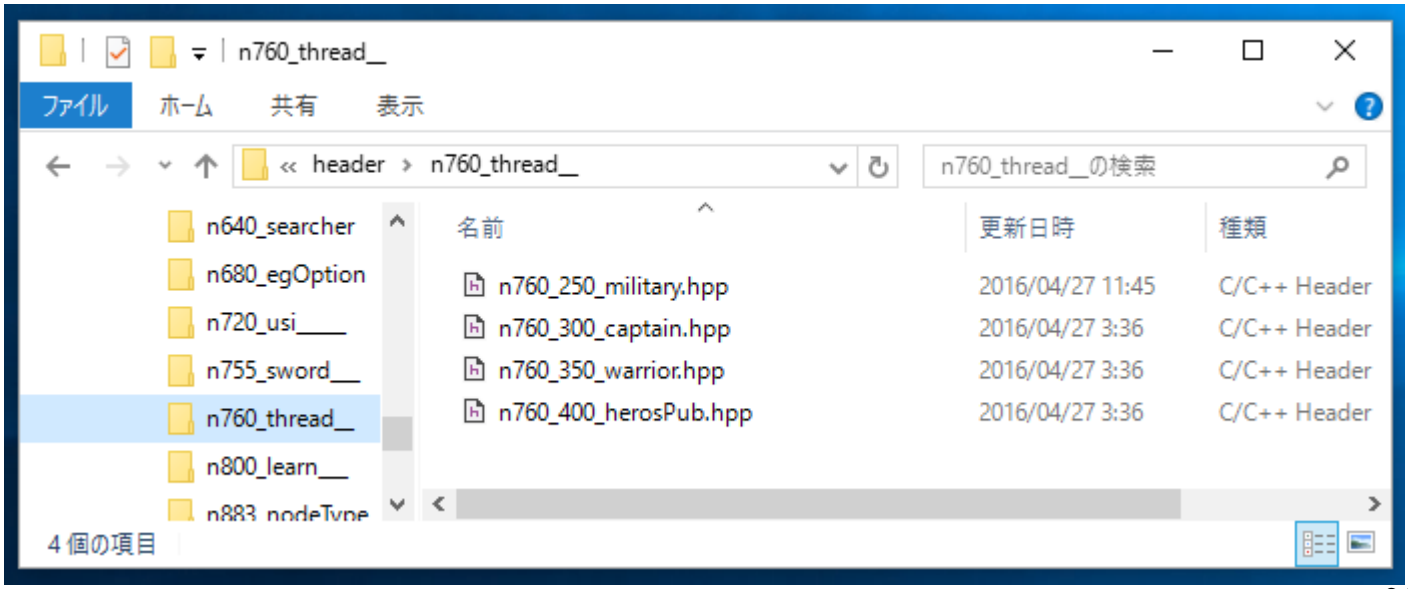

n720_usi____
ファイル
ホーム
共有
表示
« header › n720_usi____
n720_usi____の検索
n640_searcher
n680_egOption
n720_usi____
n755_sword__
n760_thread_
名前
更新日時
サイズ
n720_260_usiOperation.hpp 2016/04/27 3:36 1 KB
n720_300_benchmark.hpp 2016/04/27 3:36 1 KB
2 個の項目

32

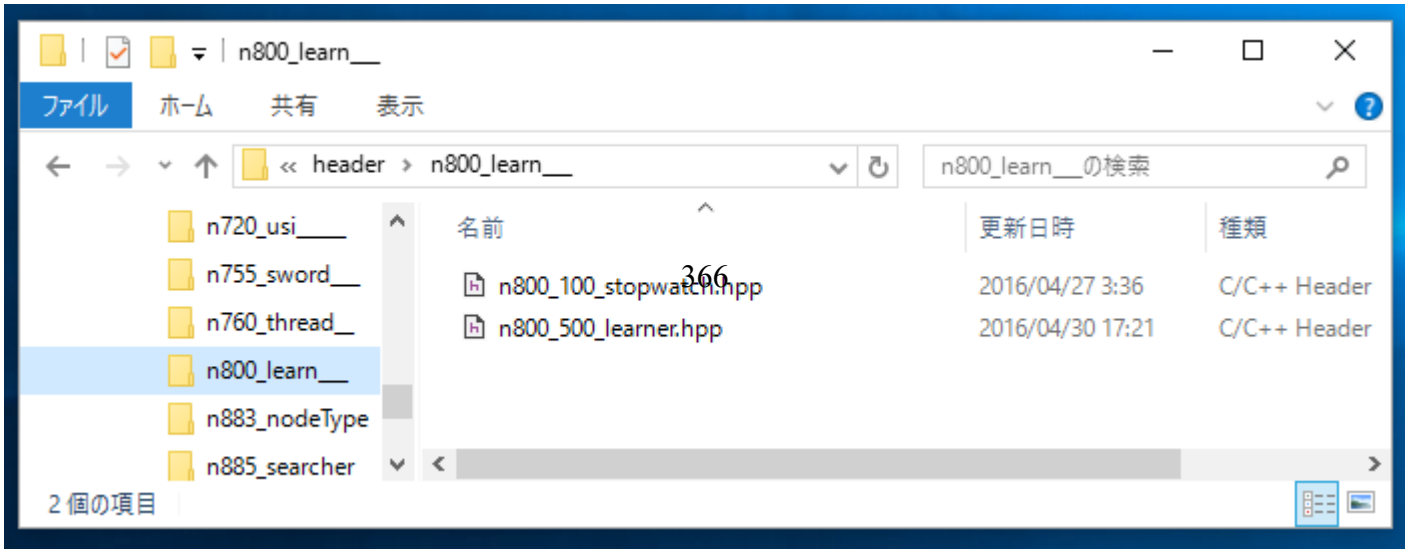
n755_sword__


ファイル
ホーム
共有
表示
« header › n755_sword__
n755_sword__の検索
n640_searcher
n680_egOption
n720_usi____
n755_sword__
n760_thread__
n800_learn__
名前
更新日時
サイズ
n755_070_SwordAbstract.hpp 2016/04/27 10:26 1 KB
n755_100_SwordRoot.hpp 2016/04/27 10:27 1 KB
n755_110_SwordPv.hpp 2016/04/27 10:27 1 KB
n755_120_SwordNonPv.hpp 2016/04/27 10:27 1 KB
4 個の項目

33

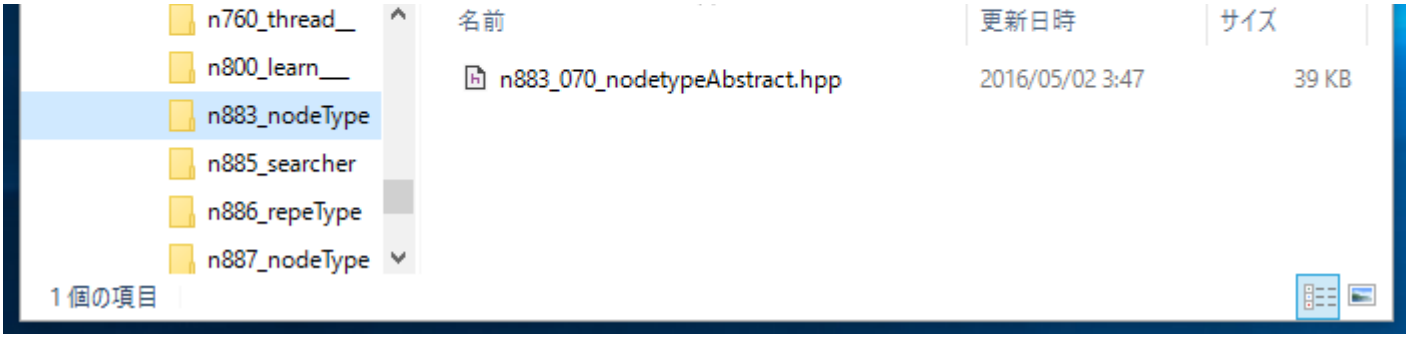

n760_thread__
ファイル
ホーム
共有
表示
« header › n760_thread__
n760_thread__の検索
n640_searcher
n680_egOption
n720_usi____
n755_sword__
n760_thread__
n800_learn__
名前
更新日時
種類
n760_250_military.hpp 2016/04/27 11:45 C/C++ Header
n760_300_captain.hpp 2016/04/27 3:36 C/C++ Header
n760_350_warrior.hpp 2016/04/27 3:36 C/C++ Header
n760_400_herosPub.hpp 2016/04/27 3:36 C/C++ Header
4 個の項目

34

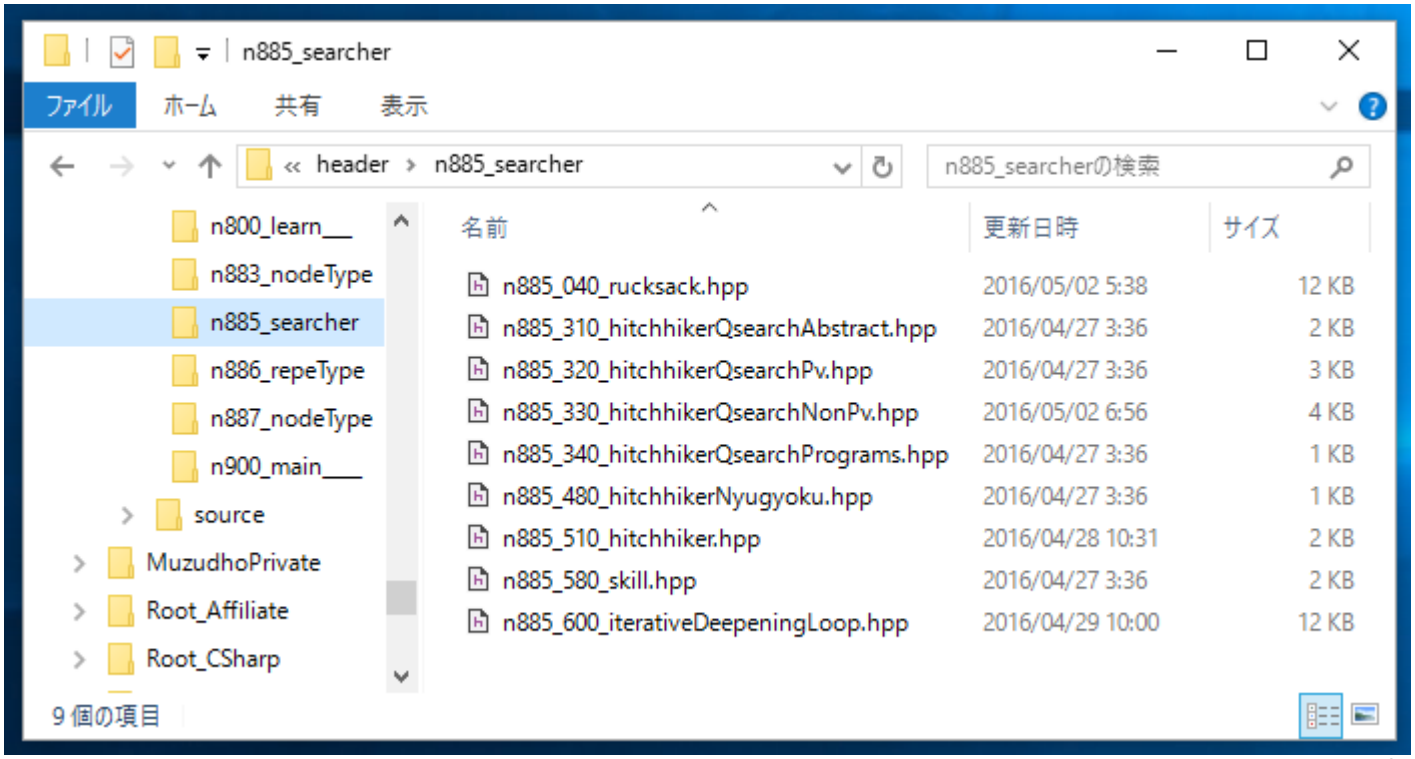

n800_learn__
ファイル
ホーム
共有
表示
« header › n800_learn__
n800_learn__の検索
n720_usi____
n755_sword__
n760_thread__
n800_learn__
n883_nodeType
n885_searcher
名前
更新日時
種類
n800_100_stopwatch.hpp 2016/04/27 3:36 C/C++ Header
n800_500_learner.hpp 2016/04/30 17:21 C/C++ Header
2 個の項目

35

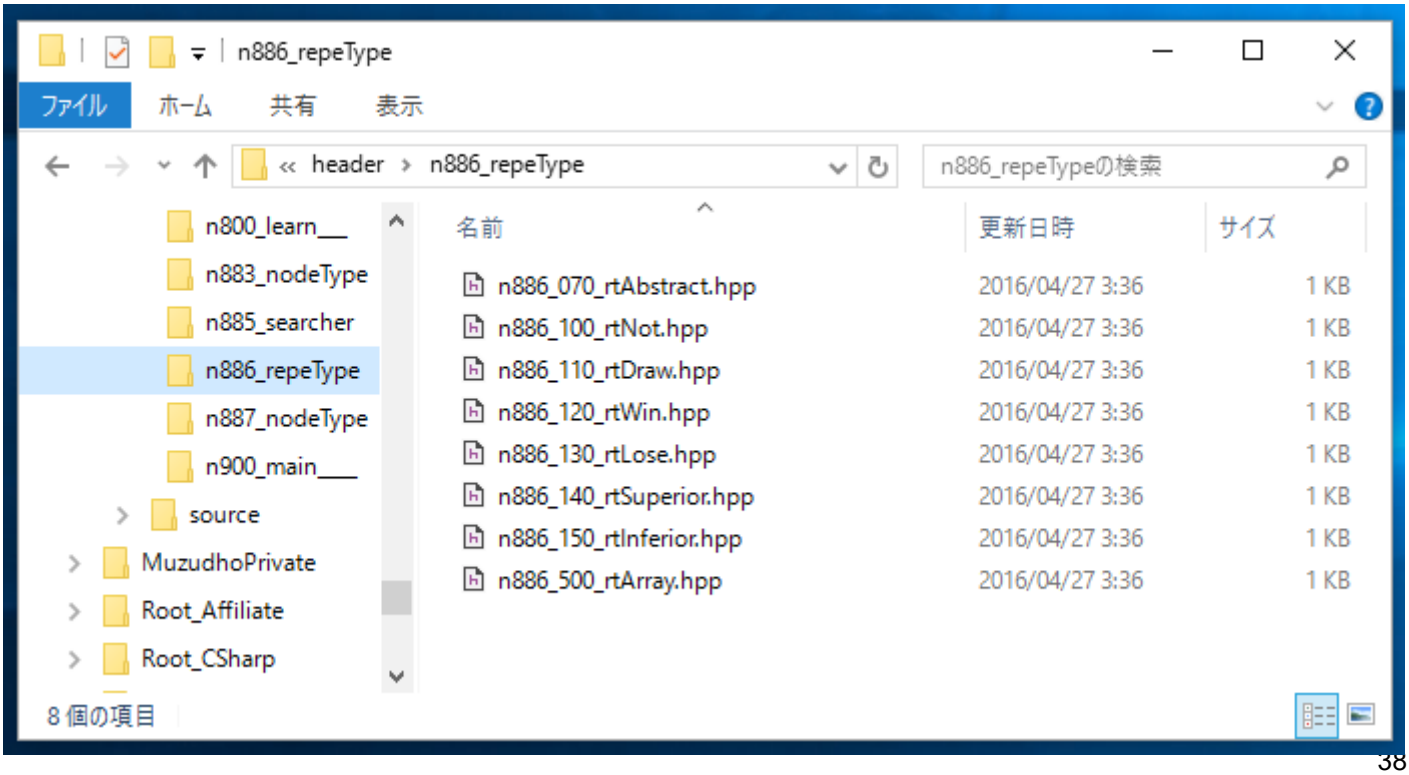
n883_nodeType
ファイル
ホーム
共有
表示
« header › n883_nodeType
n883_nodeTypeの検索


n760_thread__
n800_learn___
n883_nodeType
n885_searcher
n886_repeType
n887_nodeType
名前
更新日時
サイズ
n883_070_nodetypeAbstract.hpp
2016/05/02 3:47
39 KB
1 個の項目

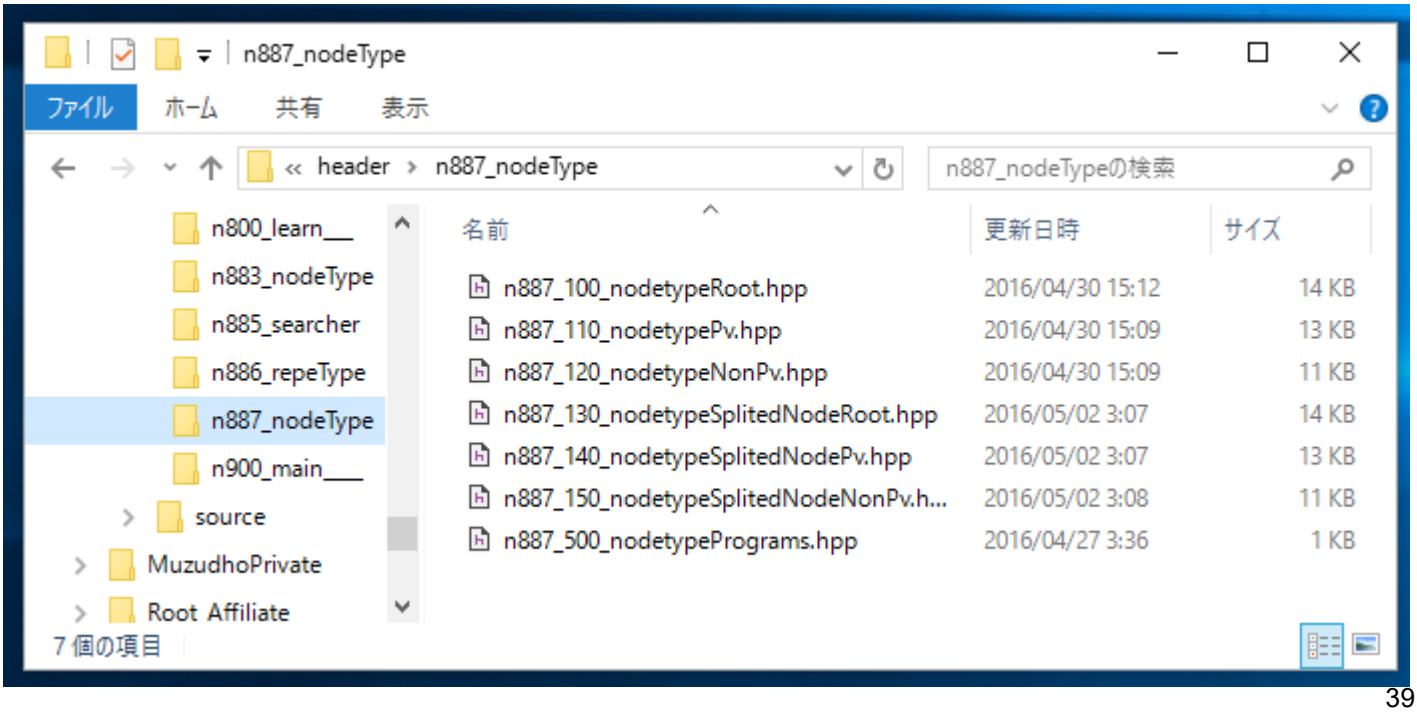
36


n885_searcher
ファイル
ホーム
共有
表示
« header › n885_searcher
n885_searcherの検索
n800_learn___
n883_nodeType
n885_searcher
n886_repeType
n887_nodeType
n900_main___
source
MuzudhoPrivate
Root_Affiliate
Root_CSharp
名前
更新日時
サイズ
n885_040_rucksack.hpp 2016/05/02 5:38 12 KB
n885_310_hitchhikerQsearchAbstract.hpp 2016/04/27 3:36 2 KB
n885_320_hitchhikerQsearchPv.hpp 2016/04/27 3:36 3 KB
n885_330_hitchhikerQsearchNonPv.hpp 2016/05/02 6:56 4 KB
n885_340_hitchhikerQsearchPrograms.hpp 2016/04/27 3:36 1 KB
n885_480_hitchhikerNyugyoku.hpp 2016/04/27 3:36 1 KB
n885_510_hitchhiker.hpp 2016/04/28 10:31 2 KB
n885_580_skill.hpp 2016/04/27 3:36 2 KB
n885_600_iterativeDeepeningLoop.hpp 2016/04/29 10:00 12 KB
9 個の項目

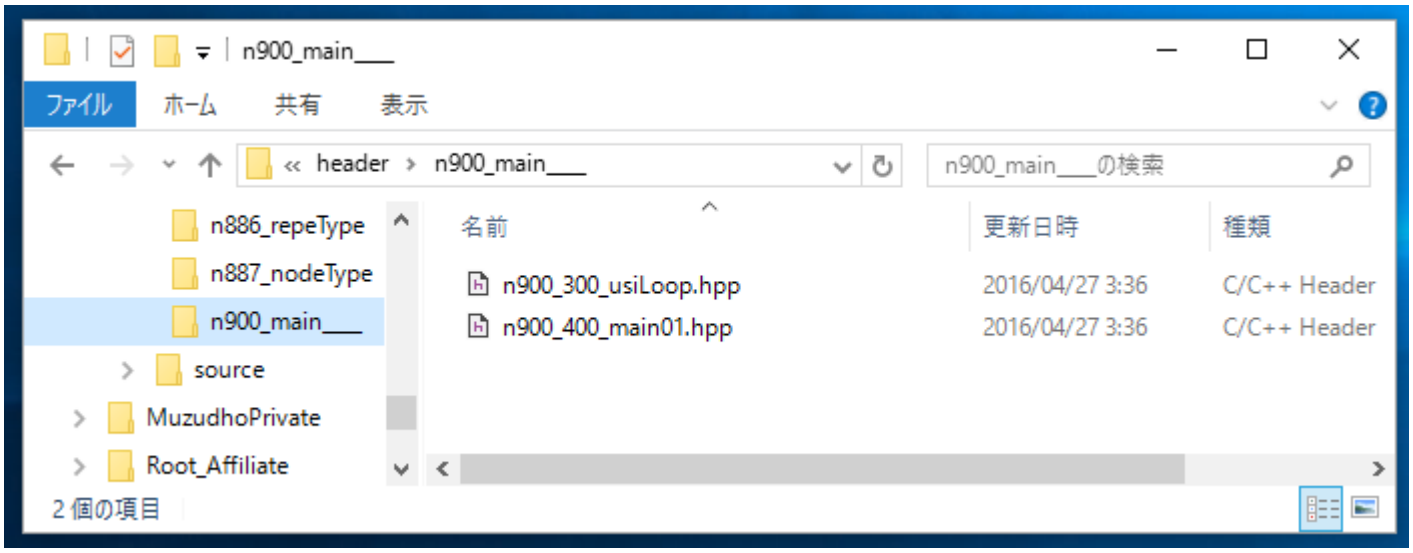
37


n886_repeType
ファイル
ホーム
共有
表示
« header › n886_repeType
n886_repeTypeの検索
n800_learn___
n883_nodeType
n885_searcher
n886_repeType
n887_nodeType
n900_main___
source
MuzudhoPrivate
Root_Affiliate
Root_CSharp
名前
更新日時
サイズ
n886_070_rtAbstract.hpp 2016/04/27 3:36 1 KB
n886_100_rtNot.hpp 2016/04/27 3:36 1 KB
n886_110_rtDraw.hpp 2016/04/27 3:36 1 KB
n886_120_rtWin.hpp 2016/04/27 3:36 1 KB
n886_130_rtLose.hpp 2016/04/27 3:36 1 KB
n886_140_rtSuperior.hpp 2016/04/27 3:36 1 KB
n886_150_rtInferior.hpp 2016/04/27 3:36 1 KB
n886_500_rtArray.hpp 2016/04/27 3:36 1 KB
8 個の項目

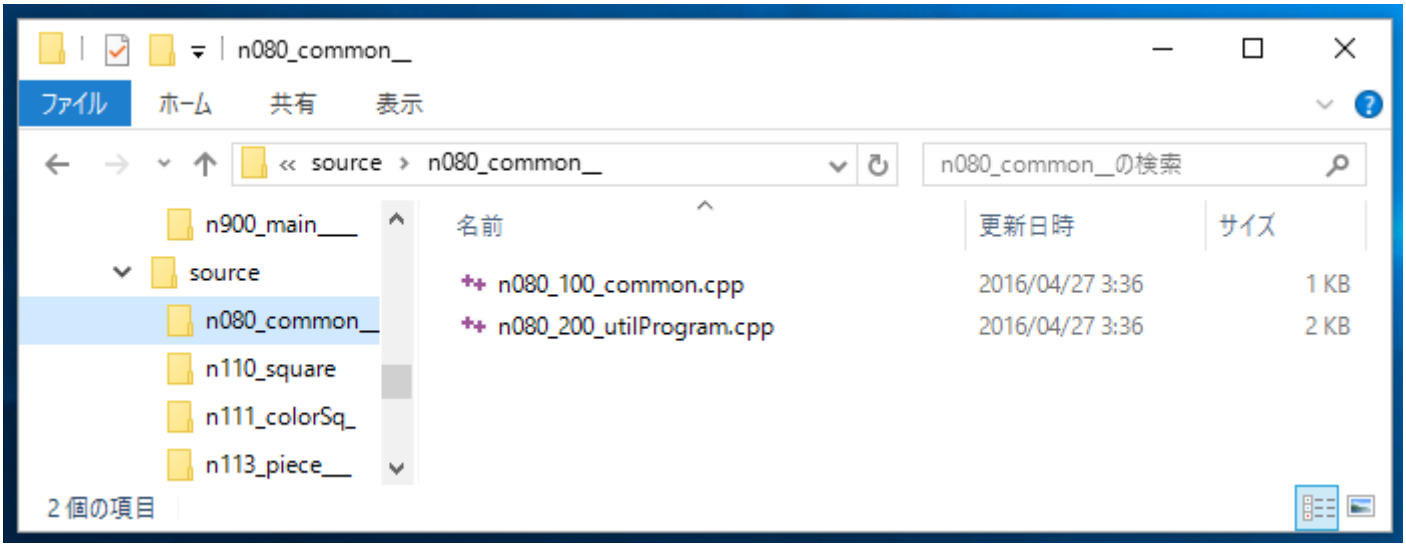
38


n887_nodeType
ファイル
ホーム
共有
表示
« header › n887_nodeType
n887_nodeTypeの検索
n800_learn__
n883_nodeType
n885_searcher
n886_repeType
n887_nodeType
n900_main___
source
MuzudhoPrivate
Root_Affiliate
名前
更新日時
サイズ
n887_100_nodetypeRoot.hpp 2016/04/30 15:12 14 KB
n887_110_nodetypePv.hpp 2016/04/30 15:09 13 KB
n887_120_nodetypeNonPv.hpp 2016/04/30 15:09 11 KB
n887_130_nodetypeSplitedNodeRoot.hpp 2016/05/02 3:07 14 KB
n887_140_nodetypeSplitedNodePv.hpp 2016/05/02 3:07 13 KB
n887_150_nodetypeSplitedNodeNonPv.h... 2016/05/02 3:08 11 KB
n887_500_nodetypePrograms.hpp 2016/04/27 3:36 1 KB
7 個の項目

39


n900_main___
ファイル
ホーム
共有
表示
« header › n900_main___
n900_main___の検索
n886_repeType
n887_nodeType
n900_main___
source
MuzudhoPrivate
Root_Affiliate
名前
更新日時
種類
n900_300_usiLoop.hpp 2016/04/27 3:36 C/C++ Header
n900_400_main01.hpp 2016/04/27 3:36 C/C++ Header
2 個の項目

40

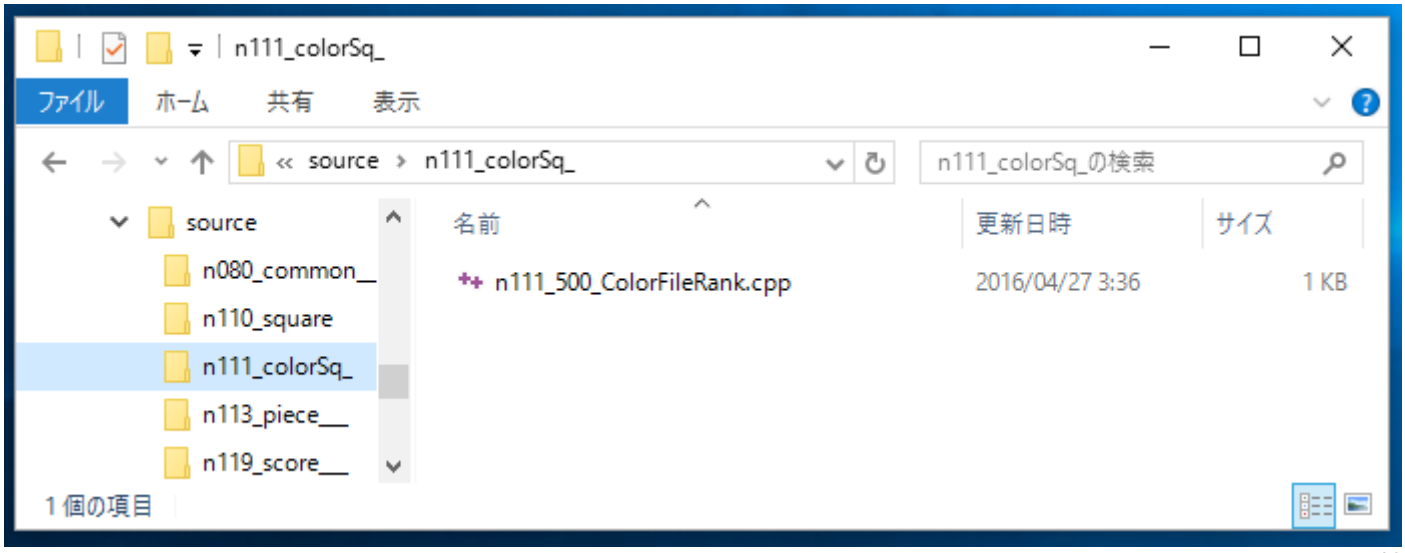

n080_common__
ファイル
ホーム
共有
表示
« source › n080_common__
n080_common__の検索
n900_main___
source
n080_common__
n110_square
n111_colorSq_
n113_piece___
名前
更新日時
サイズ
n080_100_common.cpp 2016/04/27 3:36 1 KB
n080_200_utilProgram.cpp 2016/04/27 3:36 2 KB
2 個の項目

41

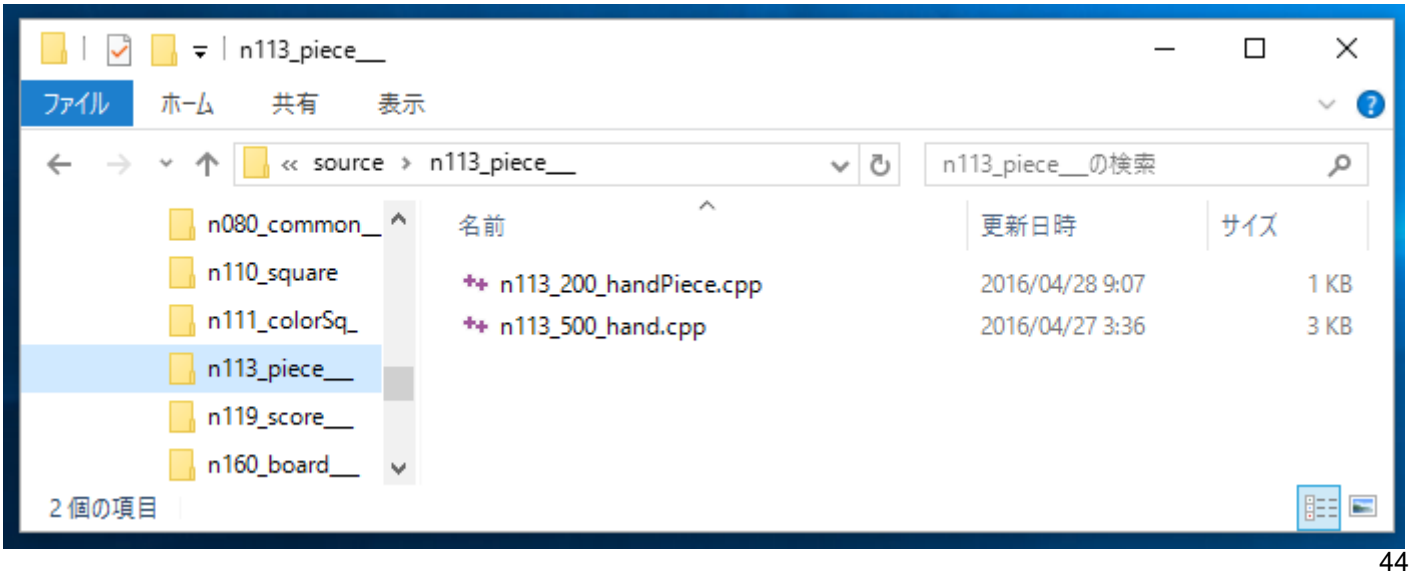
n110_square

ファイル
ホーム
共有
表示
« source › n110_square
n110_squareの検索
n900_main___
source
n080_common__
n110_square
n111_colorSq_
n113_piece___
名前
更新日時
サイズ
n110_150_file.cpp
2016/04/27 3:36
1 KB
n110_200_rank.cpp
2016/04/27 3:36
1 KB
n110_400_squareRelation.cpp
2016/04/27 3:36
2 KB
3 個の項目

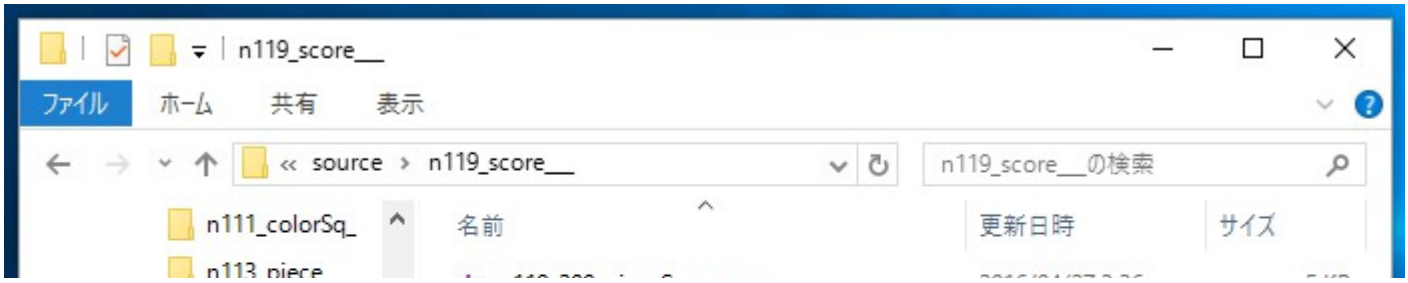
42

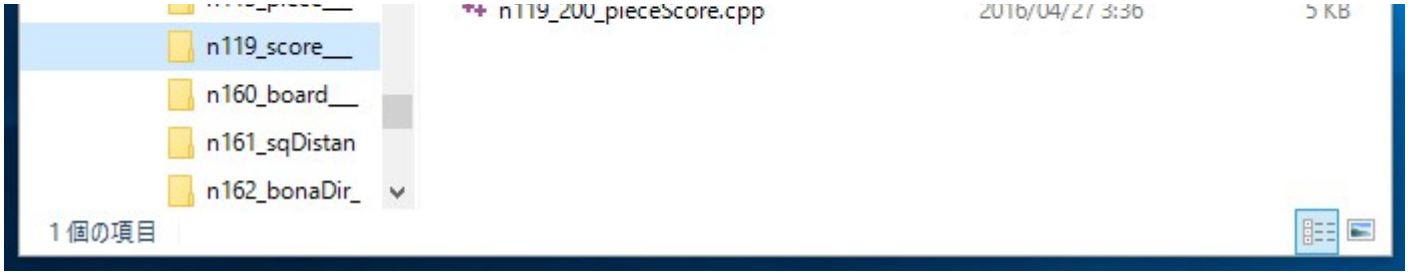

n111_colorSq_
ファイル
ホーム
共有
表示
« source › n111_colorSq_
n111_colorSq_の検索
source
n080_common__
n110_square
n111_colorSq_
n113_piece___
n119_score___
名前
更新日時
サイズ
n111_500_ColorFileRank.cpp
2016/04/27 3:36
1 KB
1 個の項目

43

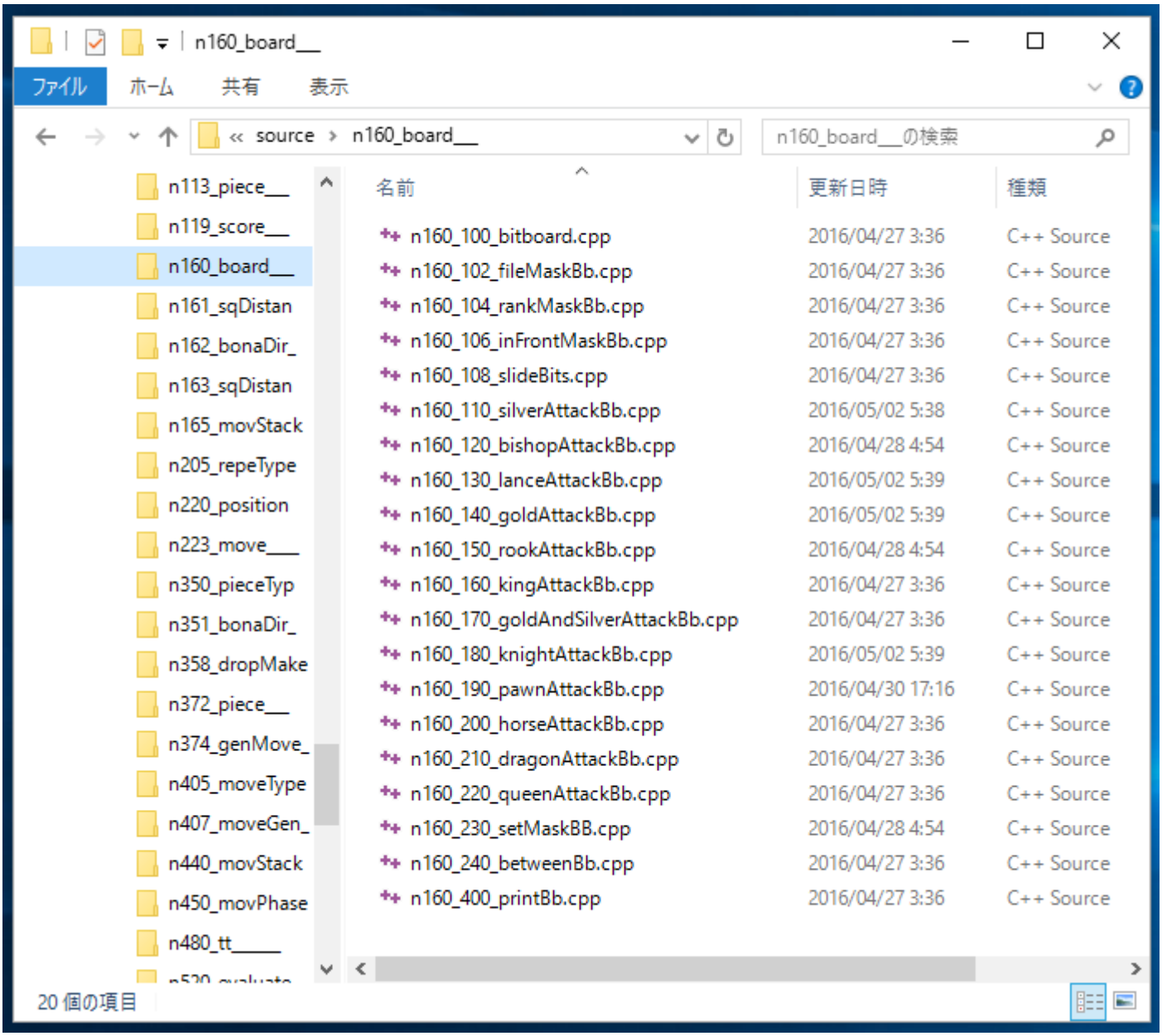

n113_piece___
ファイル
ホーム
共有
表示
« source › n113_piece___
n113_piece___の検索
n080_common__
n110_square
n111_colorSq_
n113_piece___
n119_score___
n160_board___
名前
更新日時
サイズ
n113_200_handPiece.cpp
2016/04/28 9:07
1 KB
n113_500_hand.cpp
2016/04/27 3:36
3 KB
2 個の項目

44

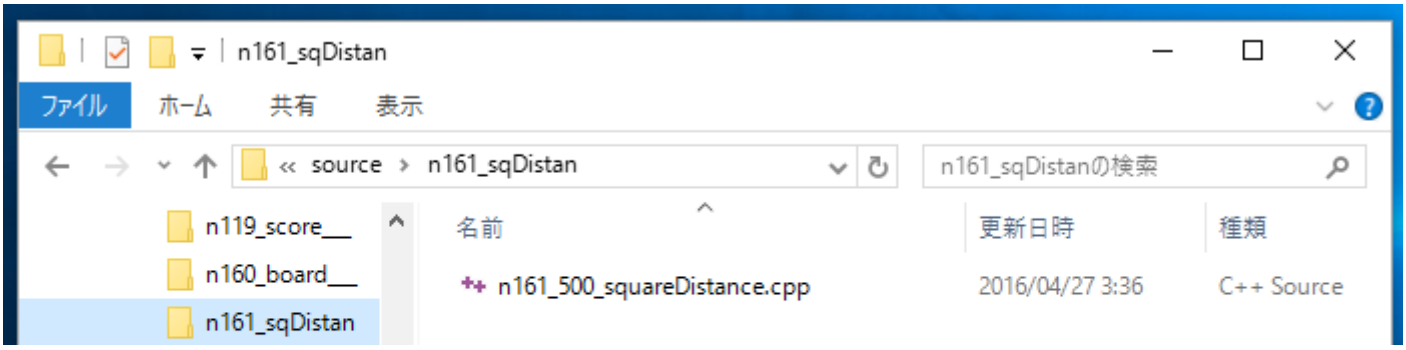

n119_score___
ファイル
ホーム
共有
表示
« source › n119_score___
n119_score___の検索
n111_colorSq_
名前
更新日時
サイズ


n119_score__
n160_board__
n161_sqDistan
n162_bonaDir_
1 個の項目

45

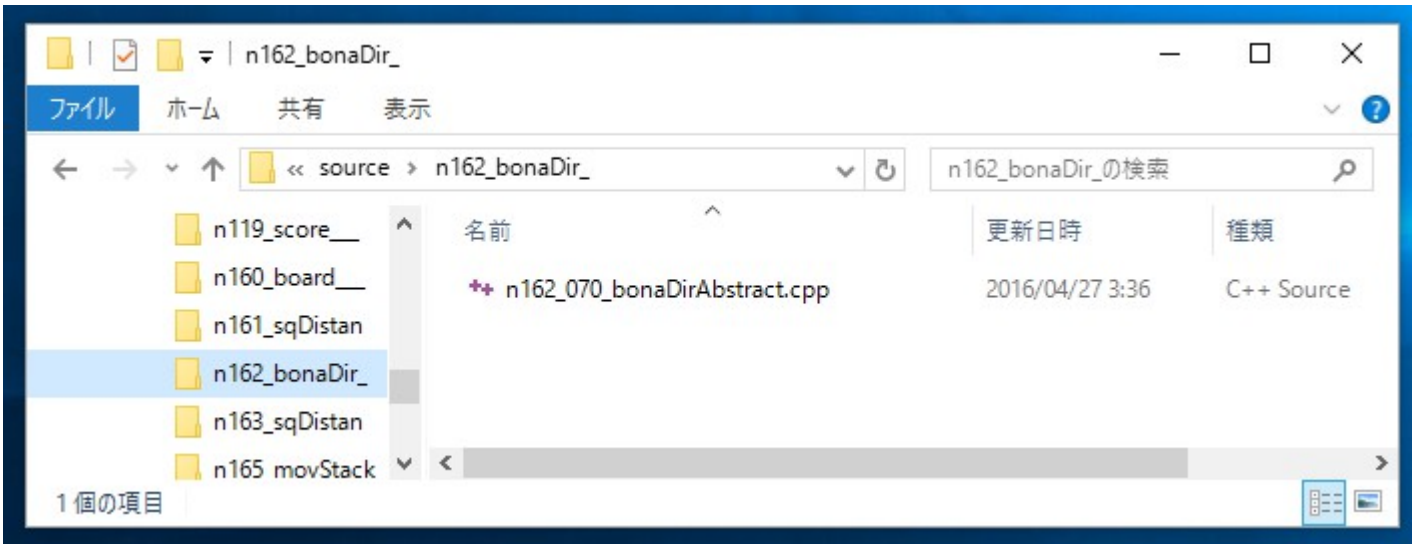

n160_board__
ファイル
ホーム
共有
表示
« source › n160_board__
n160_board__の検索
n113_piece__
n119_score__
n160_board__
n161_sqDistan
n162_bonaDir_
n163_sqDistan
n165_movStack
n205_repeType
n220_position
n223_move___
n350_pieceTyp
n351_bonaDir_
n358_dropMake
n372_piece__
n374_genMove_
n405_moveType
n407_moveGen_
n440_movStack
n450_movPhase
n480_tt_____
名前
更新日時
種類
n160_100_bitboard.cpp 2016/04/27 3:36 C++ Source
n160_102_fileMaskBb.cpp 2016/04/27 3:36 C++ Source
n160_104_rankMaskBb.cpp 2016/04/27 3:36 C++ Source
n160_106_inFrontMaskBb.cpp 2016/04/27 3:36 C++ Source
n160_108_slideBits.cpp 2016/04/27 3:36 C++ Source
n160_110_silverAttackBb.cpp 2016/05/02 5:38 C++ Source
n160_120_bishopAttackBb.cpp 2016/04/28 4:54 C++ Source
n160_130_lanceAttackBb.cpp 2016/05/02 5:39 C++ Source
n160_140_goldAttackBb.cpp 2016/05/02 5:39 C++ Source
n160_150_rookAttackBb.cpp 2016/04/28 4:54 C++ Source
n160_160_kingAttackBb.cpp 2016/04/27 3:36 C++ Source
n160_170_goldAndSilverAttackBb.cpp 2016/04/27 3:36 C++ Source
n160_180_knightAttackBb.cpp 2016/05/02 5:39 C++ Source
n160_190_pawnAttackBb.cpp 2016/04/30 17:16 C++ Source
n160_200_horseAttackBb.cpp 2016/04/27 3:36 C++ Source
n160_210_dragonAttackBb.cpp 2016/04/27 3:36 C++ Source
n160_220_queenAttackBb.cpp 2016/04/27 3:36 C++ Source
n160_230_setMaskBB.cpp 2016/04/28 4:54 C++ Source
n160_240_betweenBb.cpp 2016/04/27 3:36 C++ Source
n160_400_printBb.cpp 2016/04/27 3:36 C++ Source
20 個の項目

46

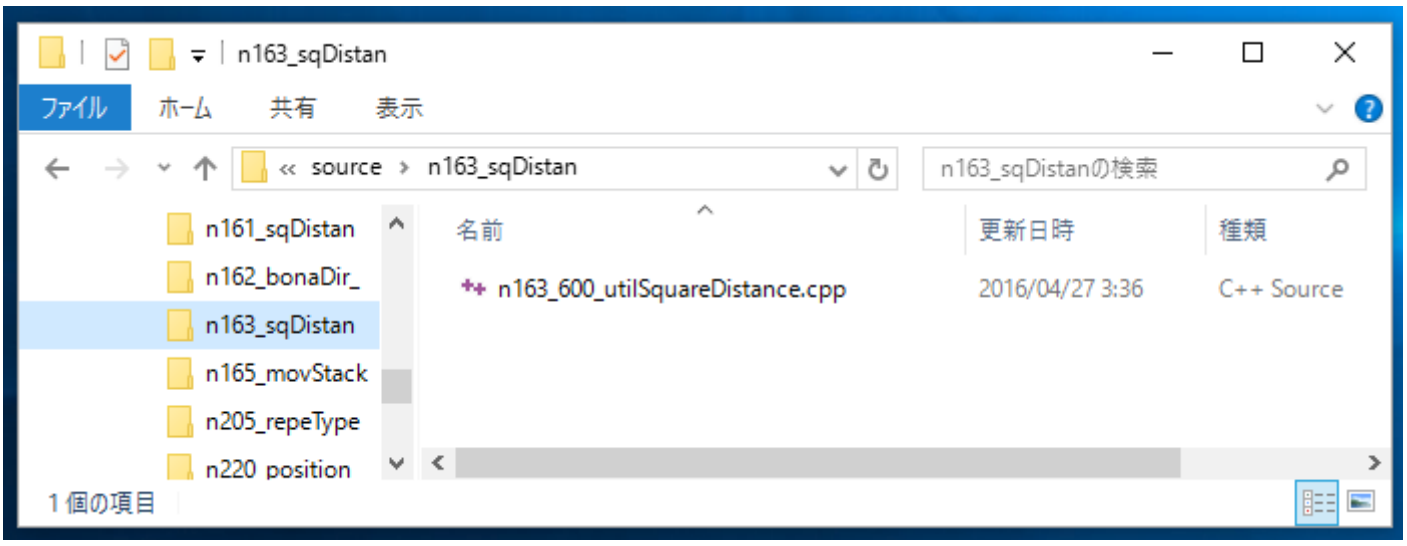

n161_sqDistan
ファイル
ホーム
共有
表示
« source › n161_sqDistan
n161_sqDistanの検索
n119_score__
n160_board__
n161_sqDistan
名前
更新日時
種類
n161_500_squareDistance.cpp 2016/04/27 3:36 C++ Source

n162_bonaDir_
n163_sqDistan
n165 movStack
1 個の項目

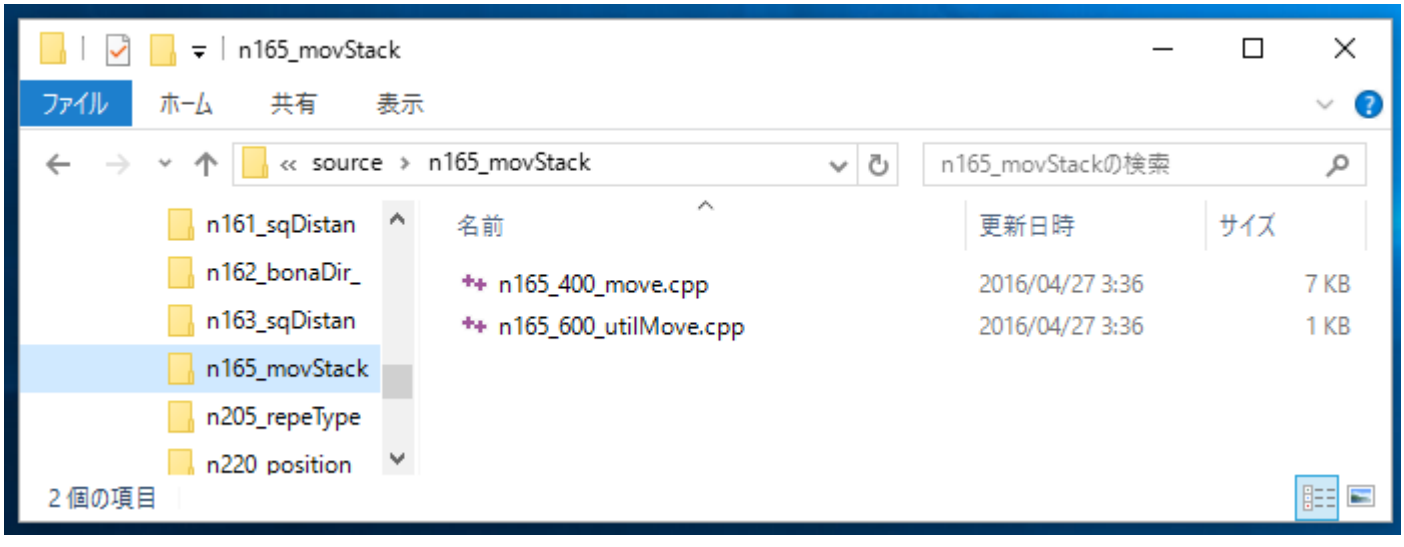
47


n162_bonaDir_
ファイル
ホーム
共有
表示
« source › n162_bonaDir_
n162_bonaDir_の検索
n119_score__
n160_board__
n161_sqDistan
n162_bonaDir_
n163_sqDistan
n165 movStack
名前
更新日時
種類
n162_070_bonaDirAbstract.cpp
2016/04/27 3:36
C++ Source
1 個の項目

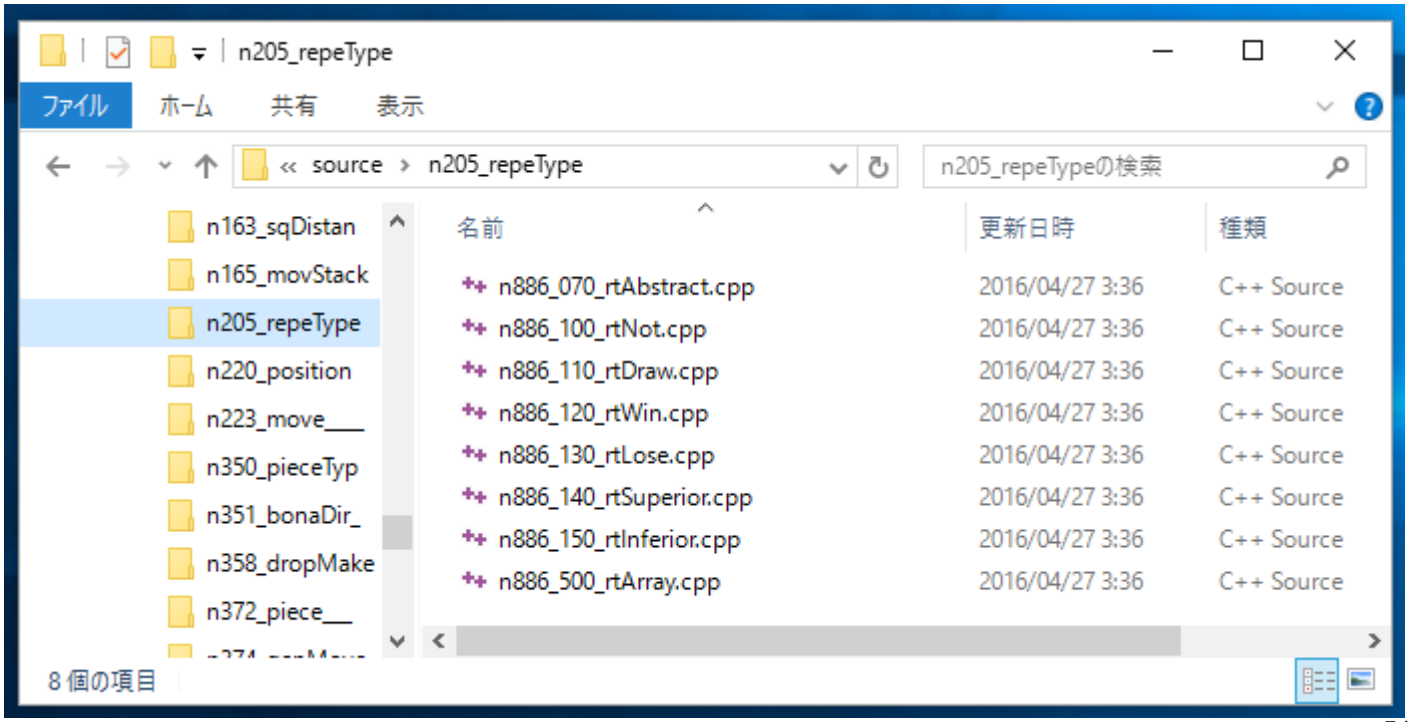
48


n163_sqDistan
ファイル
ホーム
共有
表示
« source › n163_sqDistan
n163_sqDistanの検索
n161_sqDistan
n162_bonaDir_
n163_sqDistan
n165_movStack
n205_repeType
n220 position
名前
更新日時
種類
n163_600_utilSquareDistance.cpp
2016/04/27 3:36
C++ Source
1 個の項目

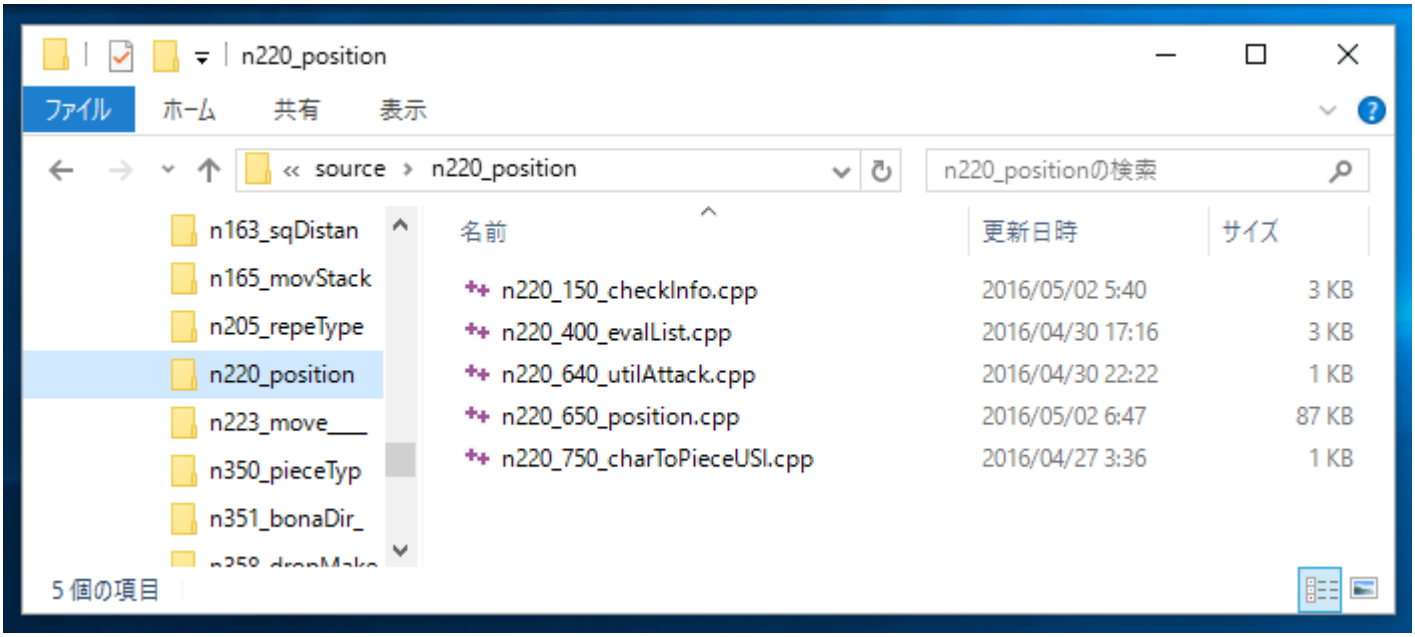
49


n165_movStack
ファイル
ホーム
共有
表示
« source › n165_movStack
n165_movStackの検索
n161_sqDistan
n162_bonaDir_
n163_sqDistan
n165_movStack
n205_repeType
n220 position
名前
更新日時
サイズ
n165_400_move.cpp
2016/04/27 3:36
7 KB
n165_600_utilMove.cpp
2016/04/27 3:36
1 KB
2 個の項目

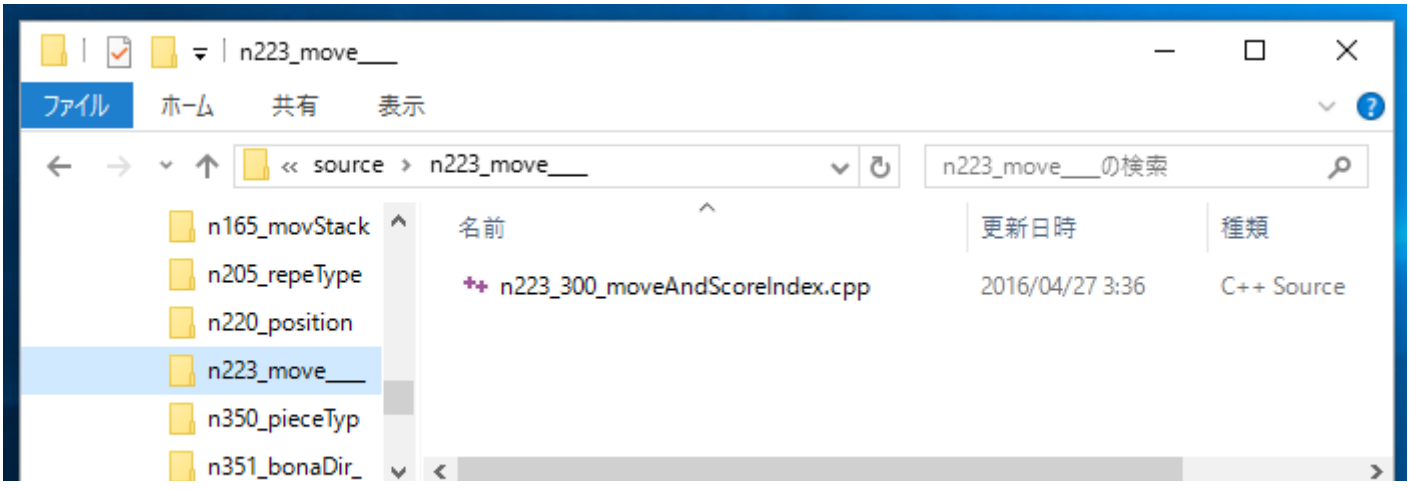
50


n205_repeType
ファイル
ホーム
共有
表示
« source › n205_repeType
n205_repeTypeの検索
n163_sqDistan
n165_movStack
n205_repeType
n220_position
n223_move___
n350_pieceTyp
n351_bonaDir_
n358_dropMake
n372_piece___
名前
更新日時
種類
n886_070_rtAbstract.cpp 2016/04/27 3:36 C++ Source
n886_100_rtNot.cpp 2016/04/27 3:36 C++ Source
n886_110_rtDraw.cpp 2016/04/27 3:36 C++ Source
n886_120_rtWin.cpp 2016/04/27 3:36 C++ Source
n886_130_rtLose.cpp 2016/04/27 3:36 C++ Source
n886_140_rtSuperior.cpp 2016/04/27 3:36 C++ Source
n886_150_rtInferior.cpp 2016/04/27 3:36 C++ Source
n886_500_rtArray.cpp 2016/04/27 3:36 C++ Source
8 個の項目

51

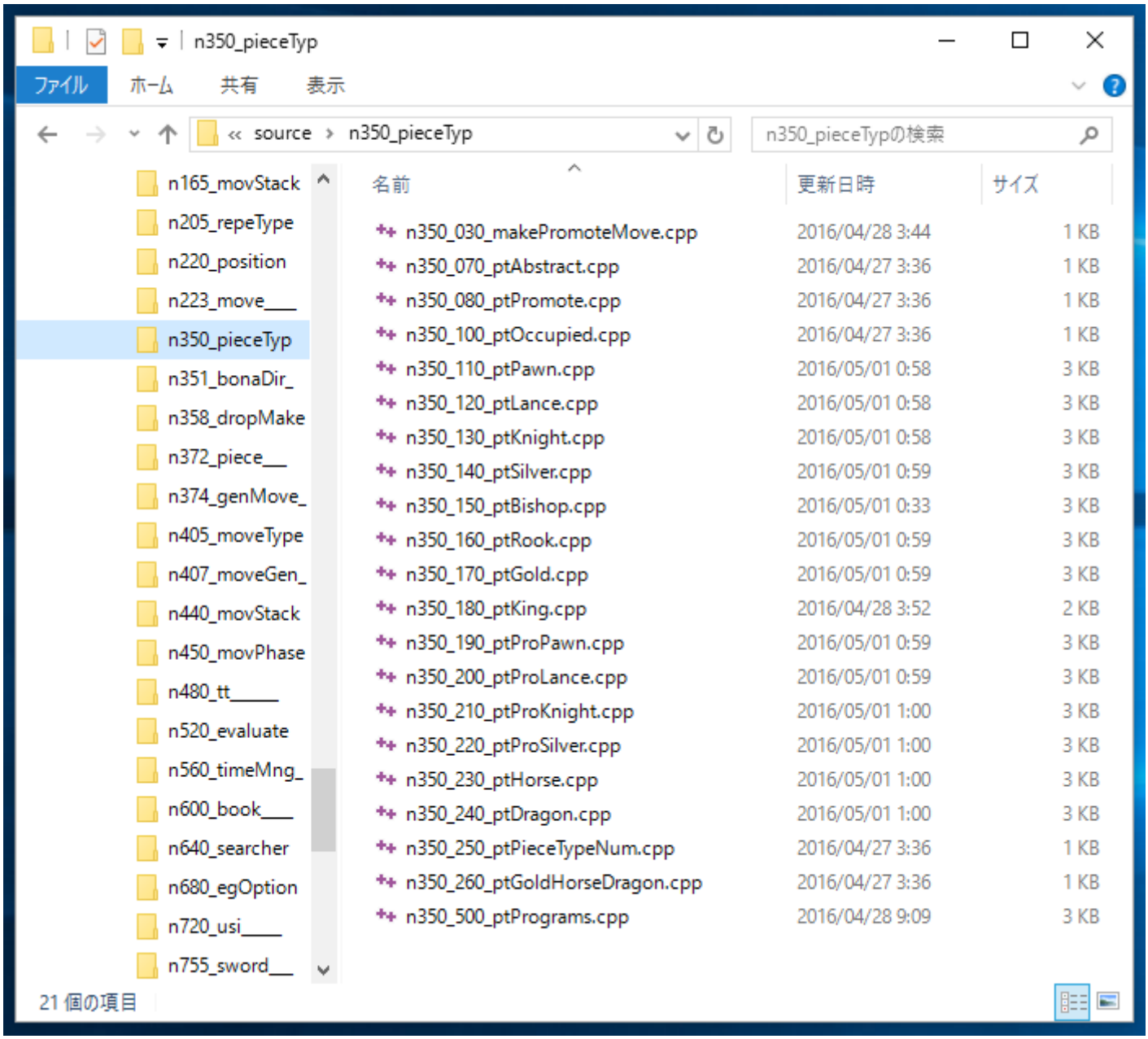

n220_position
ファイル
ホーム
共有
表示
« source › n220_position
n220_positionの検索
n163_sqDistan
n165_movStack
n205_repeType
n220_position
n223_move___
n350_pieceTyp
n351_bonaDir_
名前
更新日時
サイズ
n220_150_checkInfo.cpp 2016/05/02 5:40 3 KB
n220_400_evalList.cpp 2016/04/30 17:16 3 KB
n220_640_utilAttack.cpp 2016/04/30 22:22 1 KB
n220_650_position.cpp 2016/05/02 6:47 87 KB
n220_750_charToPieceUSI.cpp 2016/04/27 3:36 1 KB
5 個の項目

52

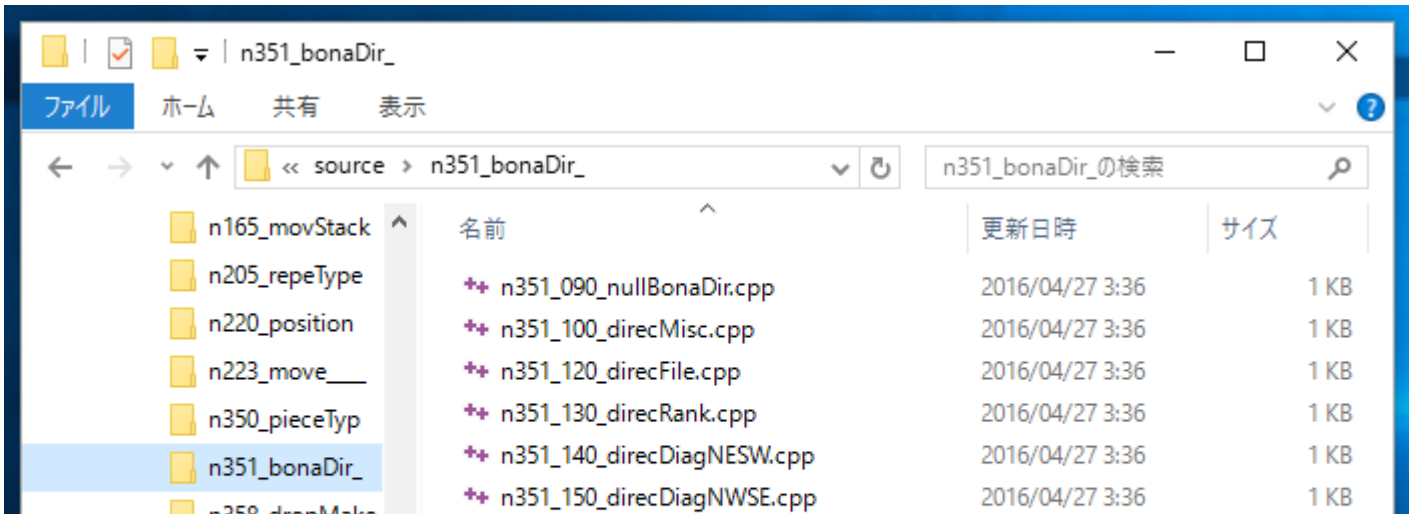

n223_move___
ファイル
ホーム
共有
表示
« source › n223_move___
n223_move___の検索
n165_movStack
n205_repeType
n220_position
n223_move___
n350_pieceTyp
n351_bonaDir_
名前
更新日時
種類
n223_300_moveAndScoreIndex.cpp 2016/04/27 3:36 C++ Source

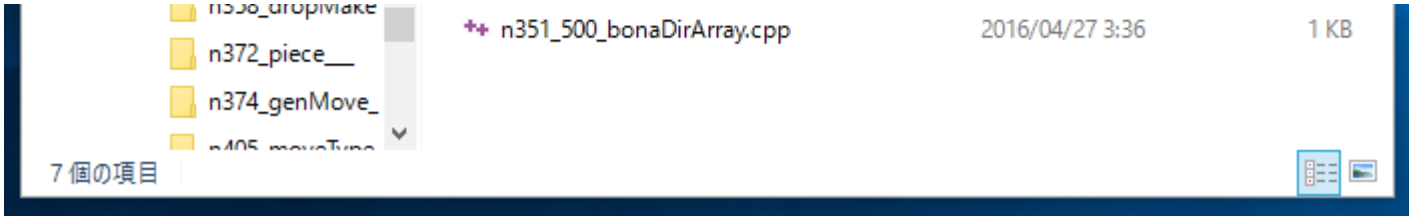
1 個の項目

53

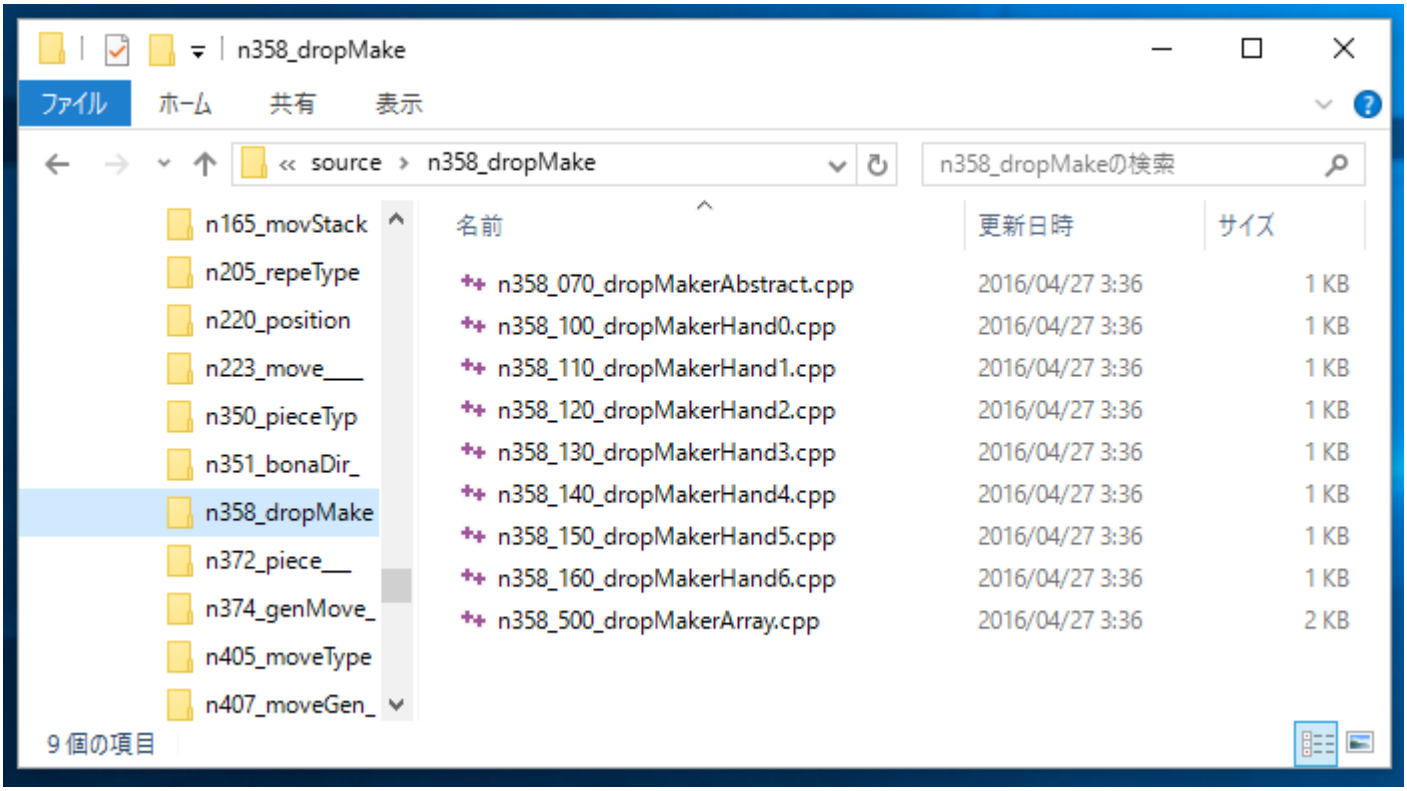

n350_pieceTyp
ファイル
ホーム
共有
表示
« source › n350_pieceTyp
n350_pieceTypの検索
n165_movStack
n205_repeType
n220_position
n223_move___
n350_pieceTyp
n351_bonaDir_
n358_dropMake
n372_piece___
n374_genMove_
n405_moveType
n407_moveGen_
n440_movStack
n450_movPhase
n480_tt_____
n520_evaluate
n560_timeMng_
n600_book___
n640_searcher
n680_egOption
n720_usi____
n755_sword___
名前
更新日時
サイズ
n350_030_makePromoteMove.cpp 2016/04/28 3:44 1 KB
n350_070_ptAbstract.cpp 2016/04/27 3:36 1 KB
n350_080_ptPromote.cpp 2016/04/27 3:36 1 KB
n350_100_ptOccupied.cpp 2016/04/27 3:36 1 KB
n350_110_ptPawn.cpp 2016/05/01 0:58 3 KB
n350_120_ptLance.cpp 2016/05/01 0:58 3 KB
n350_130_ptKnight.cpp 2016/05/01 0:58 3 KB
n350_140_ptSilver.cpp 2016/05/01 0:59 3 KB
n350_150_ptBishop.cpp 2016/05/01 0:33 3 KB
n350_160_ptRook.cpp 2016/05/01 0:59 3 KB
n350_170_ptGold.cpp 2016/05/01 0:59 3 KB
n350_180_ptKing.cpp 2016/04/28 3:52 2 KB
n350_190_ptProPawn.cpp 2016/05/01 0:59 3 KB
n350_200_ptProLance.cpp 2016/05/01 0:59 3 KB
n350_210_ptProKnight.cpp 2016/05/01 1:00 3 KB
n350_220_ptProSilver.cpp 2016/05/01 1:00 3 KB
n350_230_ptHorse.cpp 2016/05/01 1:00 3 KB
n350_240_ptDragon.cpp 2016/05/01 1:00 3 KB
n350_250_ptPieceTypeNum.cpp 2016/04/27 3:36 1 KB
n350_260_ptGoldHorseDragon.cpp 2016/04/27 3:36 1 KB
n350_500_ptPrograms.cpp 2016/04/28 9:09 3 KB
21 個の項目

54

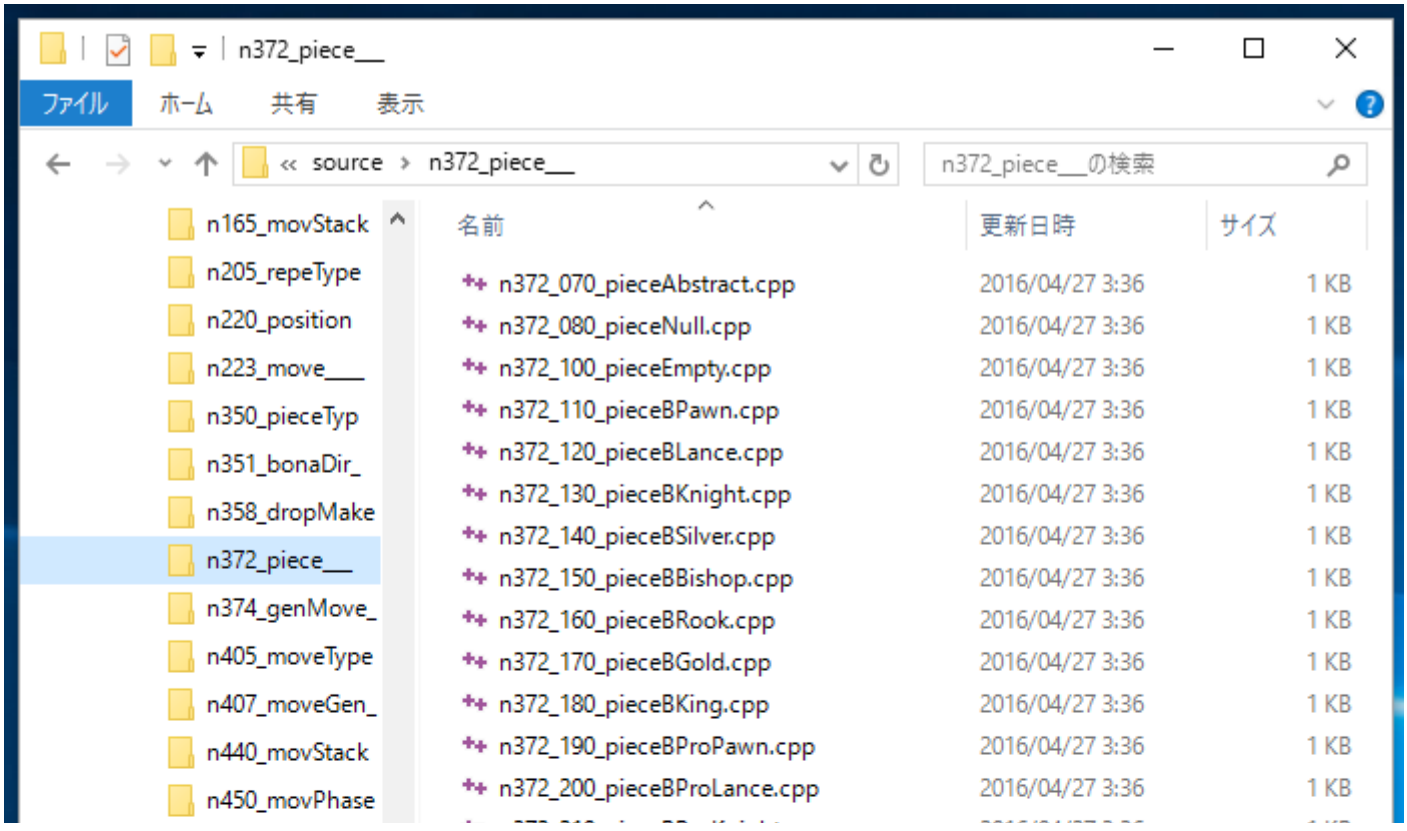

n351_bonaDir_
ファイル
ホーム
共有
表示
« source › n351_bonaDir_
n351_bonaDir_の検索
n165_movStack
n205_repeType
n220_position
n223_move___
n350_pieceTyp
n351_bonaDir_
名前
更新日時
サイズ
n351_090_nullBonaDir.cpp 2016/04/27 3:36 1 KB
n351_100_direcMisc.cpp 2016/04/27 3:36 1 KB
n351_120_direcFile.cpp 2016/04/27 3:36 1 KB
n351_130_direcRank.cpp 2016/04/27 3:36 1 KB
n351_140_direcDiagNESW.cpp 2016/04/27 3:36 1 KB
n351_150_direcDiagNWSE.cpp 2016/04/27 3:36 1 KB

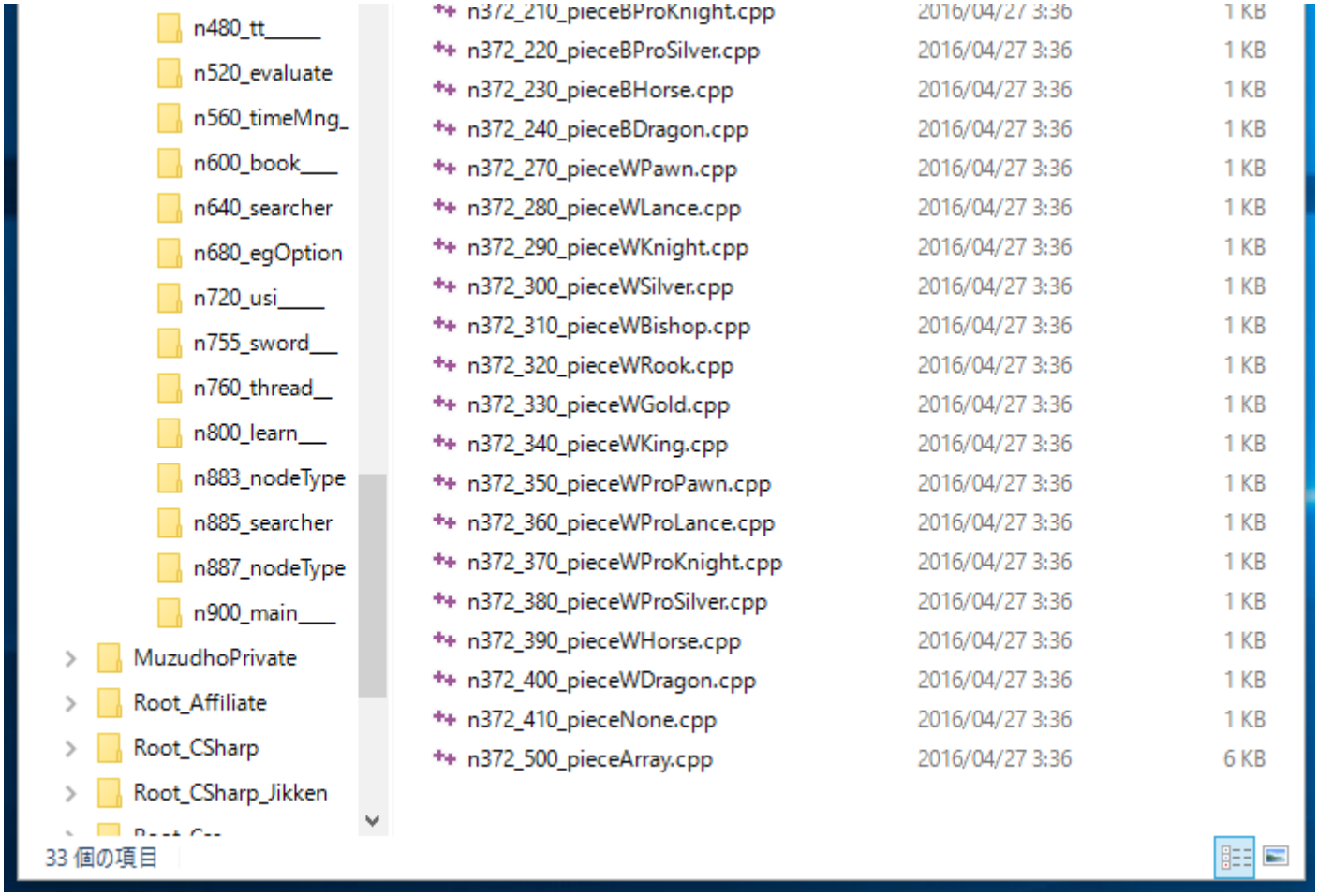

n372_piece___
n374_genMove_
n351_500_bonaDirArray.cpp 2016/04/27 3:36 1 KB
7 個の項目

55

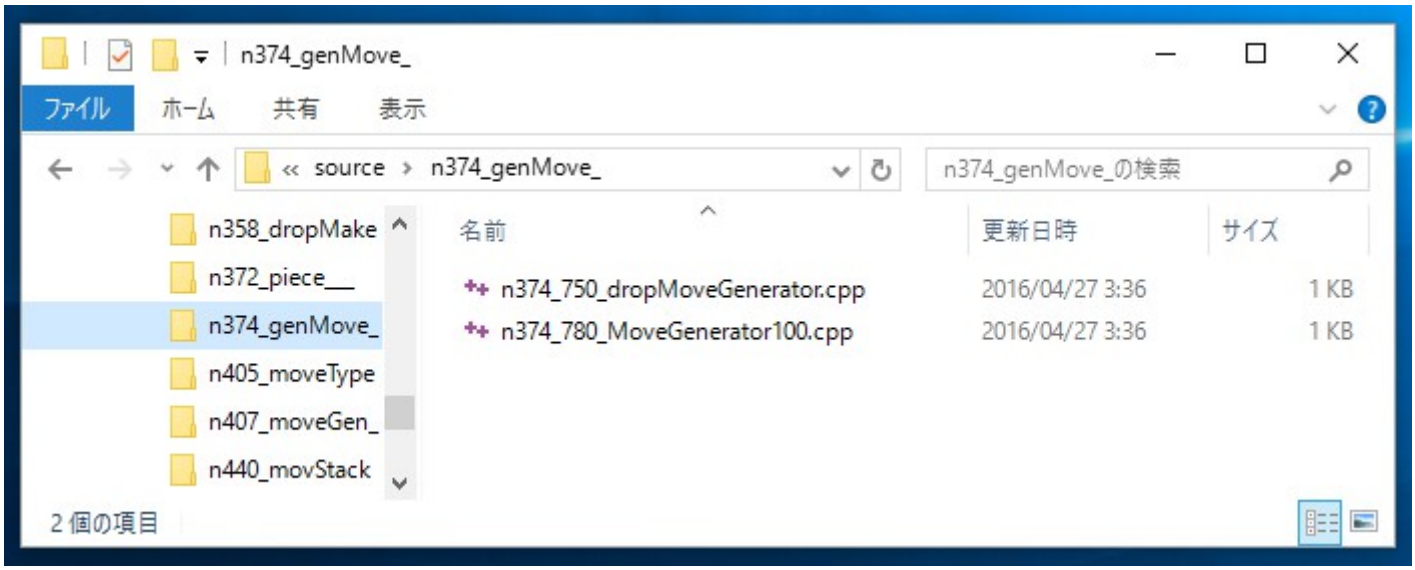

n358_dropMake
ファイル ホーム 共有 表示
« source › n358_dropMake
n358_dropMakeの検索
n165_movStack
n205_repeType
n220_position
n223_move___
n350_pieceTyp
n351_bonaDir_
n358_dropMake
n372_piece___
n374_genMove_
n405_moveType
n407_moveGen_
名前 更新日時 サイズ
n358_070_dropMakerAbstract.cpp 2016/04/27 3:36 1 KB
n358_100_dropMakerHand0.cpp 2016/04/27 3:36 1 KB
n358_110_dropMakerHand1.cpp 2016/04/27 3:36 1 KB
n358_120_dropMakerHand2.cpp 2016/04/27 3:36 1 KB
n358_130_dropMakerHand3.cpp 2016/04/27 3:36 1 KB
n358_140_dropMakerHand4.cpp 2016/04/27 3:36 1 KB
n358_150_dropMakerHand5.cpp 2016/04/27 3:36 1 KB
n358_160_dropMakerHand6.cpp 2016/04/27 3:36 1 KB
n358_500_dropMakerArray.cpp 2016/04/27 3:36 2 KB
9 個の項目

56

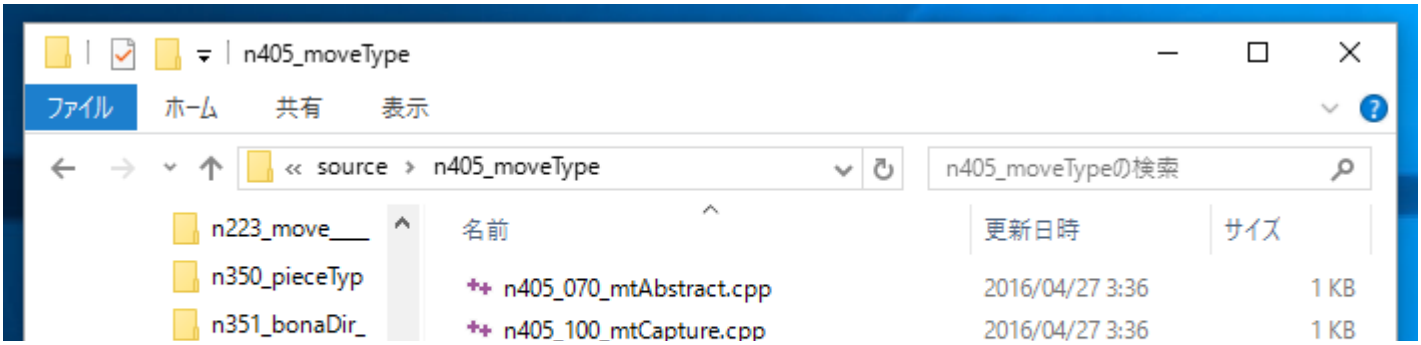

n372_piece___
ファイル ホーム 共有 表示
« source › n372_piece___
n372_piece___の検索
n165_movStack
n205_repeType
n220_position
n223_move___
n350_pieceTyp
n351_bonaDir_
n358_dropMake
n372_piece___
n374_genMove_
n405_moveType
n407_moveGen_
n440_movStack
n450_movPhase
名前 更新日時 サイズ
n372_070_pieceAbstract.cpp 2016/04/27 3:36 1 KB
n372_080_pieceNull.cpp 2016/04/27 3:36 1 KB
n372_100_pieceEmpty.cpp 2016/04/27 3:36 1 KB
n372_110_pieceBPawn.cpp 2016/04/27 3:36 1 KB
n372_120_pieceBLance.cpp 2016/04/27 3:36 1 KB
n372_130_pieceBKnight.cpp 2016/04/27 3:36 1 KB
n372_140_pieceBSilver.cpp 2016/04/27 3:36 1 KB
n372_150_pieceBBishop.cpp 2016/04/27 3:36 1 KB
n372_160_pieceBRook.cpp 2016/04/27 3:36 1 KB
n372_170_pieceBGold.cpp 2016/04/27 3:36 1 KB
n372_180_pieceBKing.cpp 2016/04/27 3:36 1 KB
n372_190_pieceBProPawn.cpp 2016/04/27 3:36 1 KB
n372_200_pieceBProLance.cpp 2016/04/27 3:36 1 KB

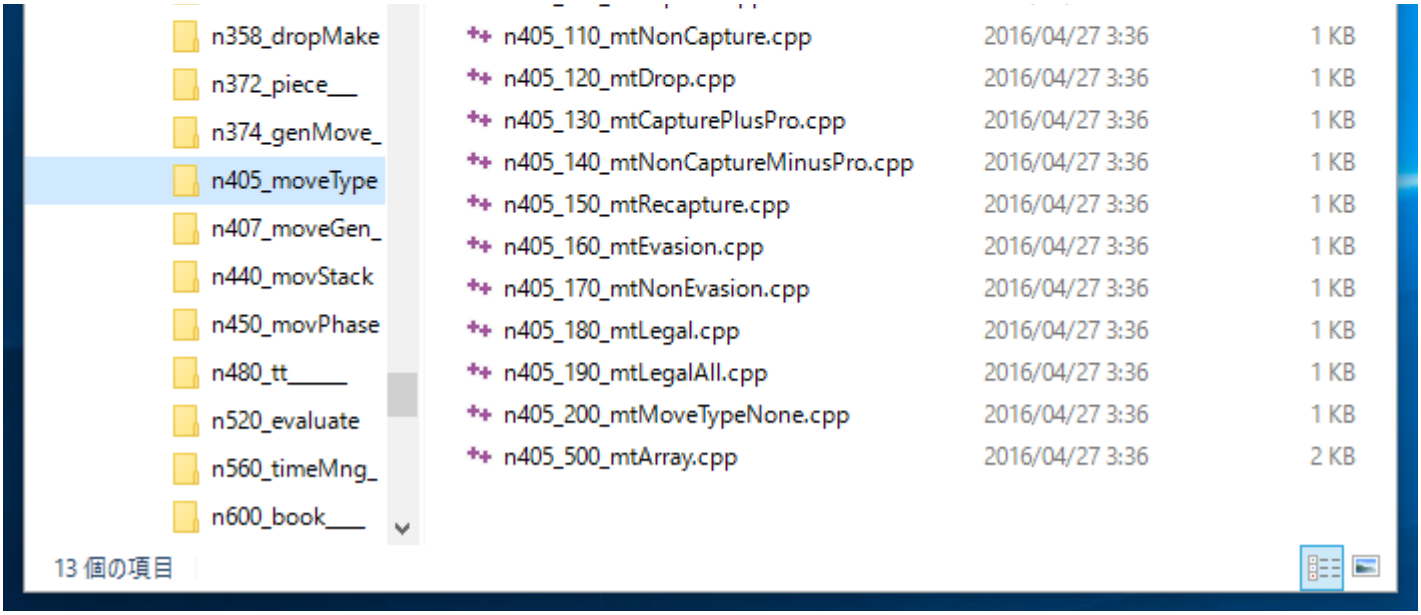

n372_210_pieceBProKnight.cpp 2016/04/27 3:36 1 KB
n372_220_pieceBProSilver.cpp 2016/04/27 3:36 1 KB
n372_230_pieceBHorse.cpp 2016/04/27 3:36 1 KB
n372_240_pieceBDragon.cpp 2016/04/27 3:36 1 KB
n372_270_pieceWPawn.cpp 2016/04/27 3:36 1 KB
n372_280_pieceWLance.cpp 2016/04/27 3:36 1 KB
n372_290_pieceWKnight.cpp 2016/04/27 3:36 1 KB
n372_300_pieceWSilver.cpp 2016/04/27 3:36 1 KB
n372_310_pieceWBishop.cpp 2016/04/27 3:36 1 KB
n372_320_pieceWRook.cpp 2016/04/27 3:36 1 KB
n372_330_pieceWGold.cpp 2016/04/27 3:36 1 KB
n372_340_pieceWKing.cpp 2016/04/27 3:36 1 KB
n372_350_pieceWProPawn.cpp 2016/04/27 3:36 1 KB
n372_360_pieceWProLance.cpp 2016/04/27 3:36 1 KB
n372_370_pieceWProKnight.cpp 2016/04/27 3:36 1 KB
n372_380_pieceWProSilver.cpp 2016/04/27 3:36 1 KB
n372_390_pieceWHorse.cpp 2016/04/27 3:36 1 KB
n372_400_pieceWDragon.cpp 2016/04/27 3:36 1 KB
n372_410_pieceNone.cpp 2016/04/27 3:36 1 KB
n372_500_pieceArray.cpp 2016/04/27 3:36 6 KB
n480_tt_____
n520_evaluate
n560_timeMng_
n600_book___
n640_searcher
n680_egOption
n720_usi____
n755_sword__
n760_thread__
n800_learn__
n883_nodeType
n885_searcher
n887_nodeType
n900_main___
MuzudhoPrivate
Root_Affiliate
Root_CSharp
Root_CSharp_Jikken
33 個の項目

57

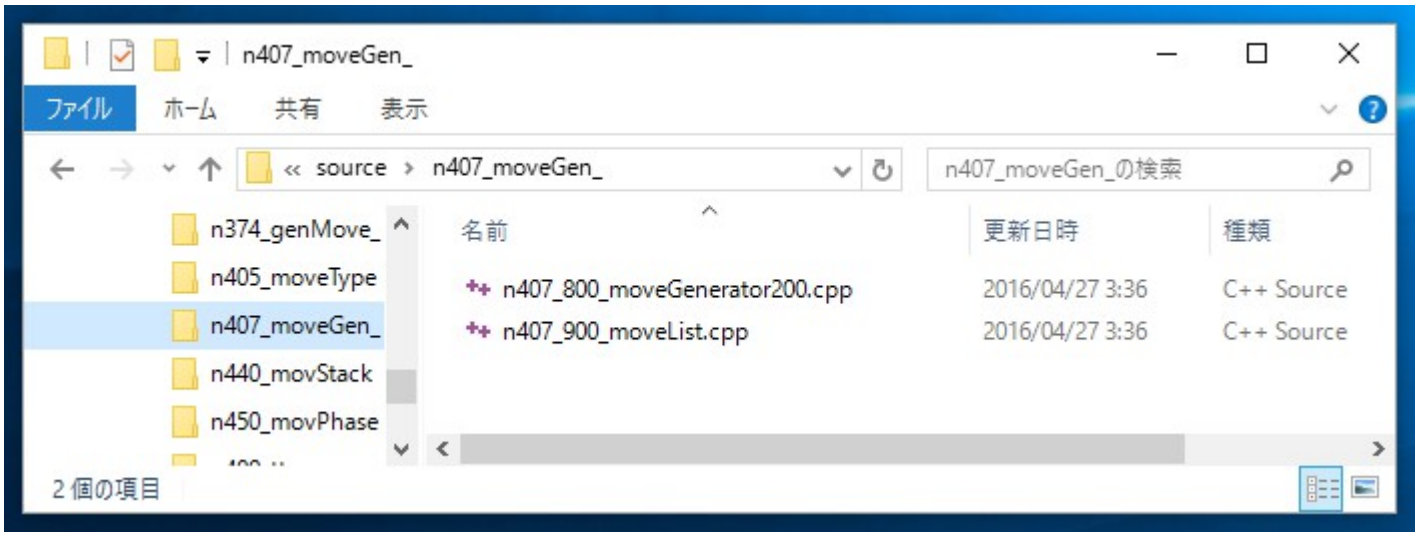

n374_genMove_
ファイル ホーム 共有 表示
« source › n374_genMove_
n374_genMove_の検索
名前 更新日時 サイズ
n374_750_dropMoveGenerator.cpp 2016/04/27 3:36 1 KB
n374_780_MoveGenerator100.cpp 2016/04/27 3:36 1 KB
n358_dropMake
n372_piece__
n374_genMove_
n405_moveType
n407_moveGen_
n440_movStack
2 個の項目

58

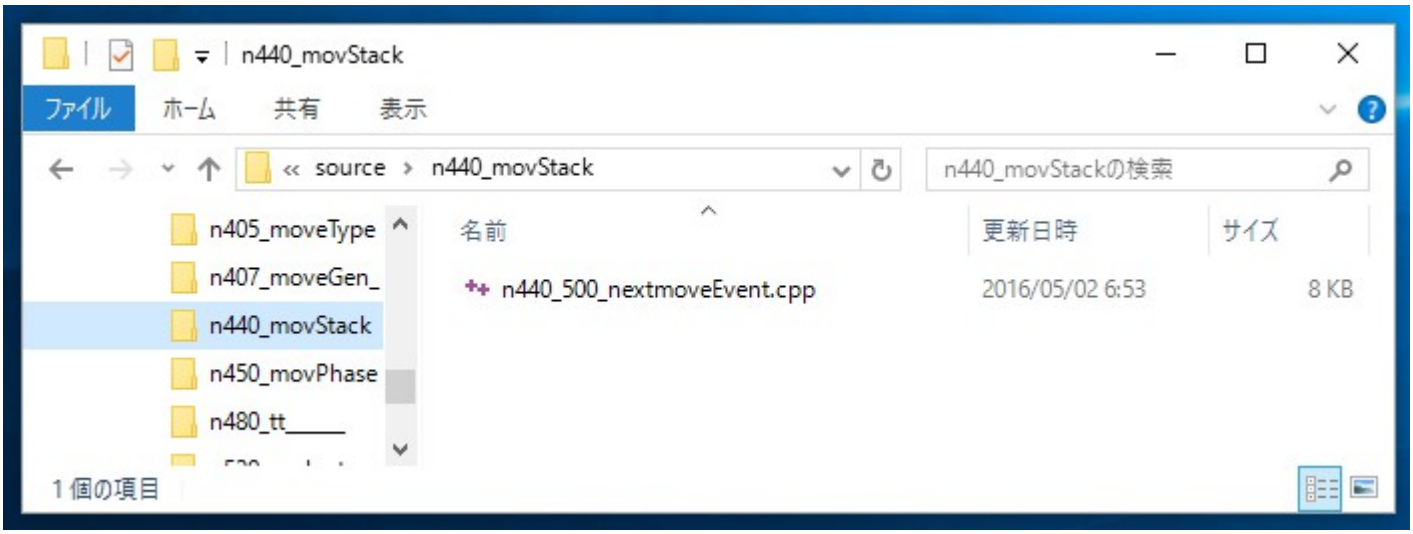

n405_moveType
ファイル ホーム 共有 表示
« source › n405_moveType
n405_moveTypeの検索
名前 更新日時 サイズ
n405_070_mtAbstract.cpp 2016/04/27 3:36 1 KB
n405_100_mtCapture.cpp 2016/04/27 3:36 1 KB
n223_move___
n350_pieceTyp
n351_bonaDir_


n358_dropMake
n372_piece___
n374_genMove_
n405_moveType
n407_moveGen_
n440_movStack
n450_movPhase
n480_tt_____
n520_evaluate
n560_timeMng_
n600_book___
n405_110_mtNonCapture.cpp 2016/04/27 3:36 1 KB
n405_120_mtDrop.cpp 2016/04/27 3:36 1 KB
n405_130_mtCapturePlusPro.cpp 2016/04/27 3:36 1 KB
n405_140_mtNonCaptureMinusPro.cpp 2016/04/27 3:36 1 KB
n405_150_mtRecapture.cpp 2016/04/27 3:36 1 KB
n405_160_mtEvasion.cpp 2016/04/27 3:36 1 KB
n405_170_mtNonEvasion.cpp 2016/04/27 3:36 1 KB
n405_180_mtLegal.cpp 2016/04/27 3:36 1 KB
n405_190_mtLegalAll.cpp 2016/04/27 3:36 1 KB
n405_200_mtMoveTypeNone.cpp 2016/04/27 3:36 1 KB
n405_500_mtArray.cpp 2016/04/27 3:36 2 KB
13 個の項目

59

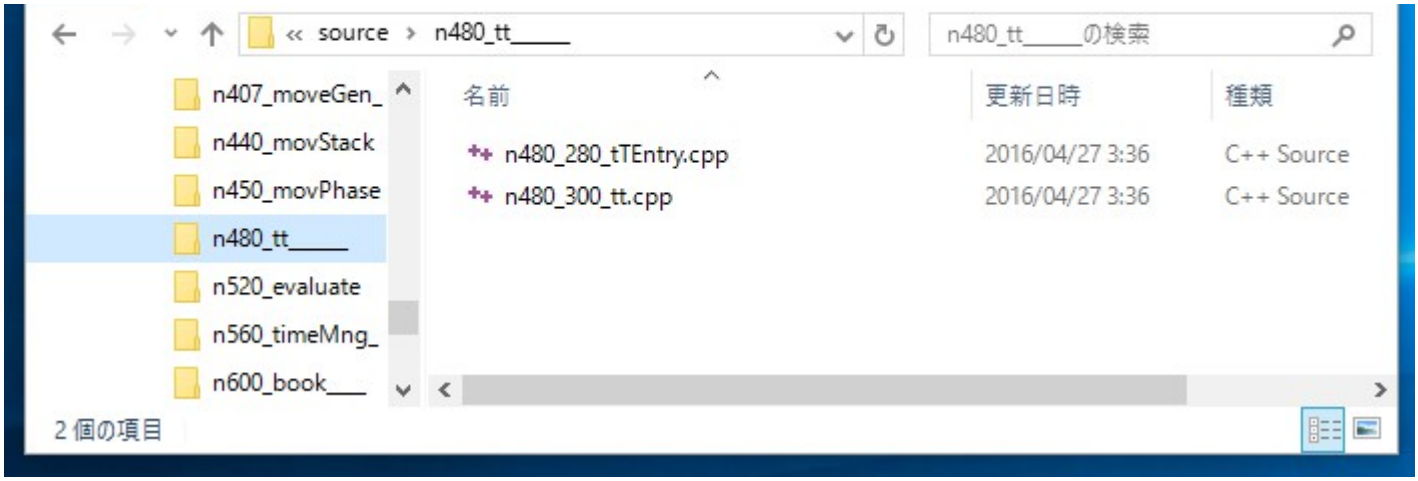

n407_moveGen_
ファイル ホーム 共有 表示
« source › n407_moveGen_
n407_moveGen_の検索
n374_genMove_
n405_moveType
n407_moveGen_
n440_movStack
n450_movPhase
名前 更新日時 種類
n407_800_moveGenerator200.cpp 2016/04/27 3:36 C++ Source
n407_900_moveList.cpp 2016/04/27 3:36 C++ Source
2 個の項目

60

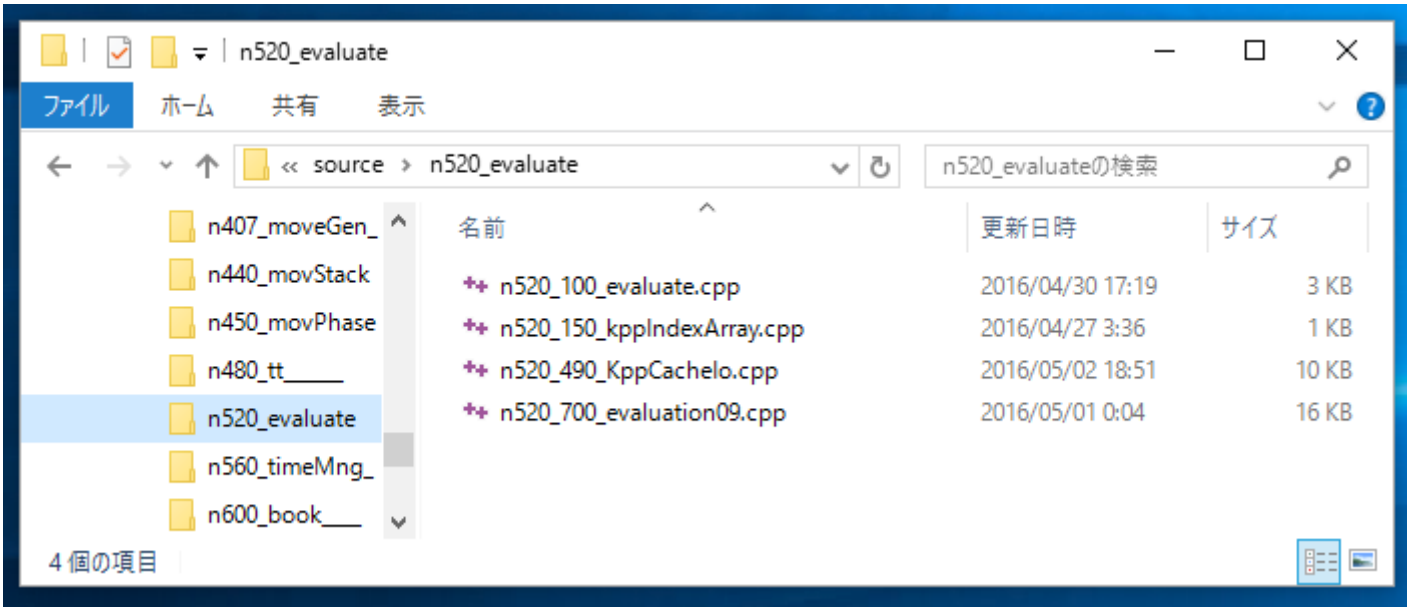

n440_movStack
ファイル ホーム 共有 表示
« source › n440_movStack
n440_movStackの検索
n405_moveType
n407_moveGen_
n440_movStack
n450_movPhase
n480_tt_____
名前 更新日時 サイズ
n440_500_nextmoveEvent.cpp 2016/05/02 6:53 8 KB
1 個の項目

61

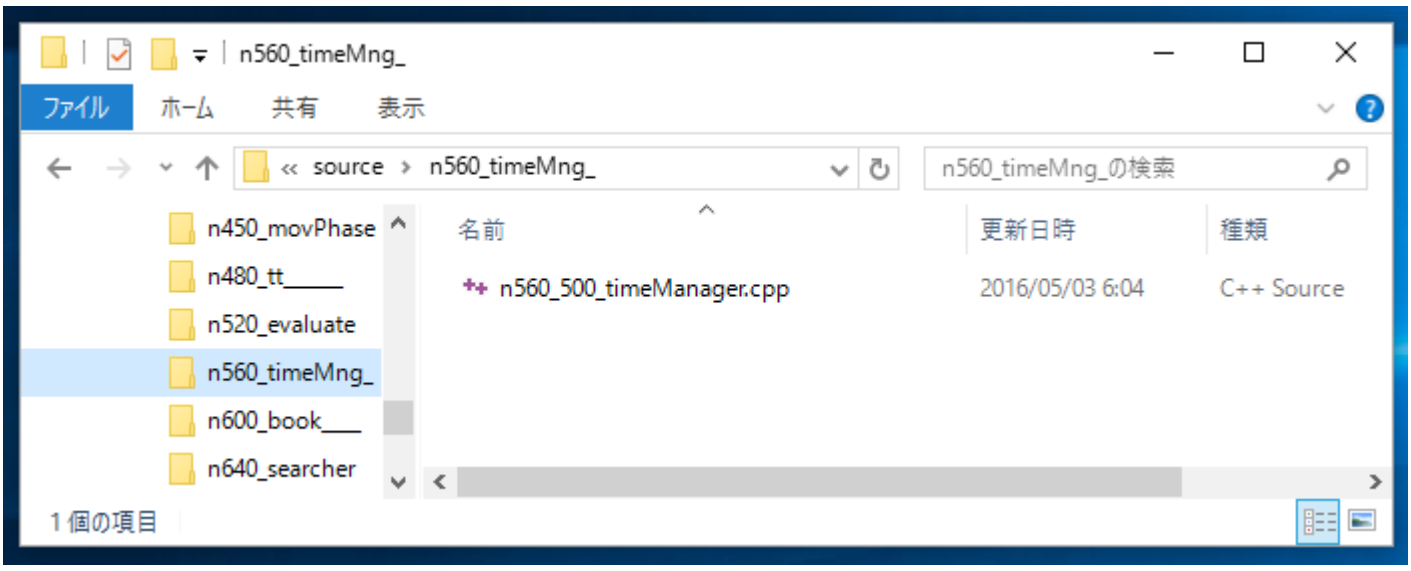
n480_tt_____
ファイル ホーム 共有 表示


« source › n480_tt_____
n480_tt_____の検索
n407_moveGen_
n440_movStack
n450_movPhase
n480_tt_____
n520_evaluate
n560_timeMng_
n600_book___
名前
更新日時
種類
n480_280_tTEntry.cpp
2016/04/27 3:36
C++ Source
n480_300_tt.cpp
2016/04/27 3:36
C++ Source
2 個の項目

62

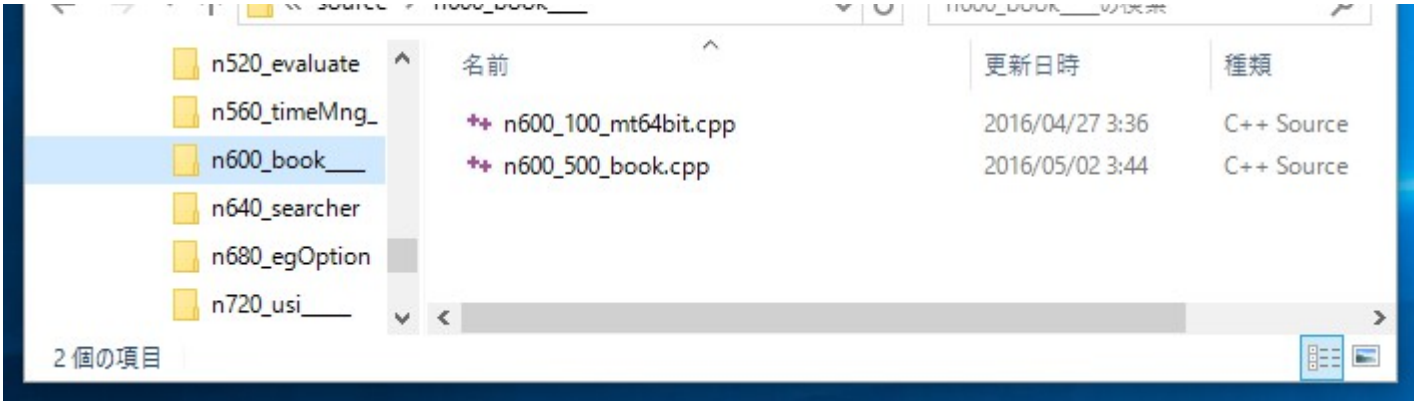

n520_evaluate
ファイル
ホーム
共有
表示
« source › n520_evaluate
n520_evaluateの検索
n407_moveGen_
n440_movStack
n450_movPhase
n480_tt_____
n520_evaluate
n560_timeMng_
n600_book___
名前
更新日時
サイズ
n520_100_evaluate.cpp
2016/04/30 17:19
3 KB
n520_150_kppIndexArray.cpp
2016/04/27 3:36
1 KB
n520_490_KppCachelo.cpp
2016/05/02 18:51
10 KB
n520_700_evaluation09.cpp
2016/05/01 0:04
16 KB
4 個の項目

63

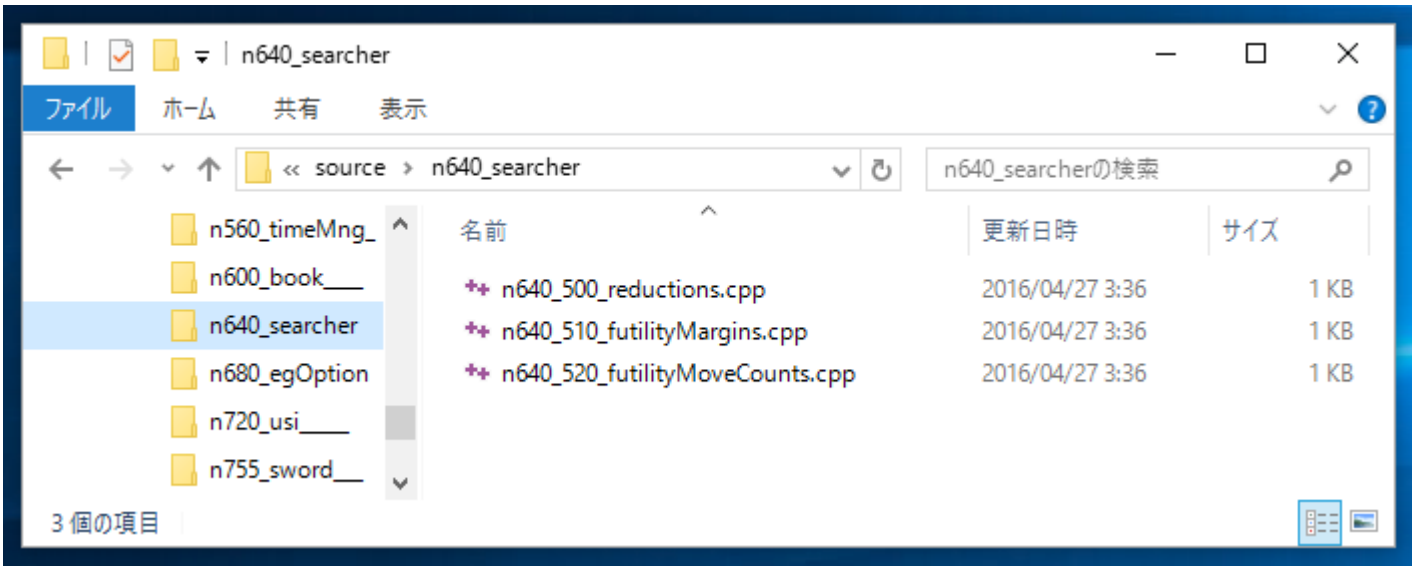

n560_timeMng_
ファイル
ホーム
共有
表示
« source › n560_timeMng_
n560_timeMng_の検索
n450_movPhase
n480_tt_____
n520_evaluate
n560_timeMng_
n600_book___
n640_searcher
名前
更新日時
種類
n560_500_timeManager.cpp
2016/05/03 6:04
C++ Source
1 個の項目

64

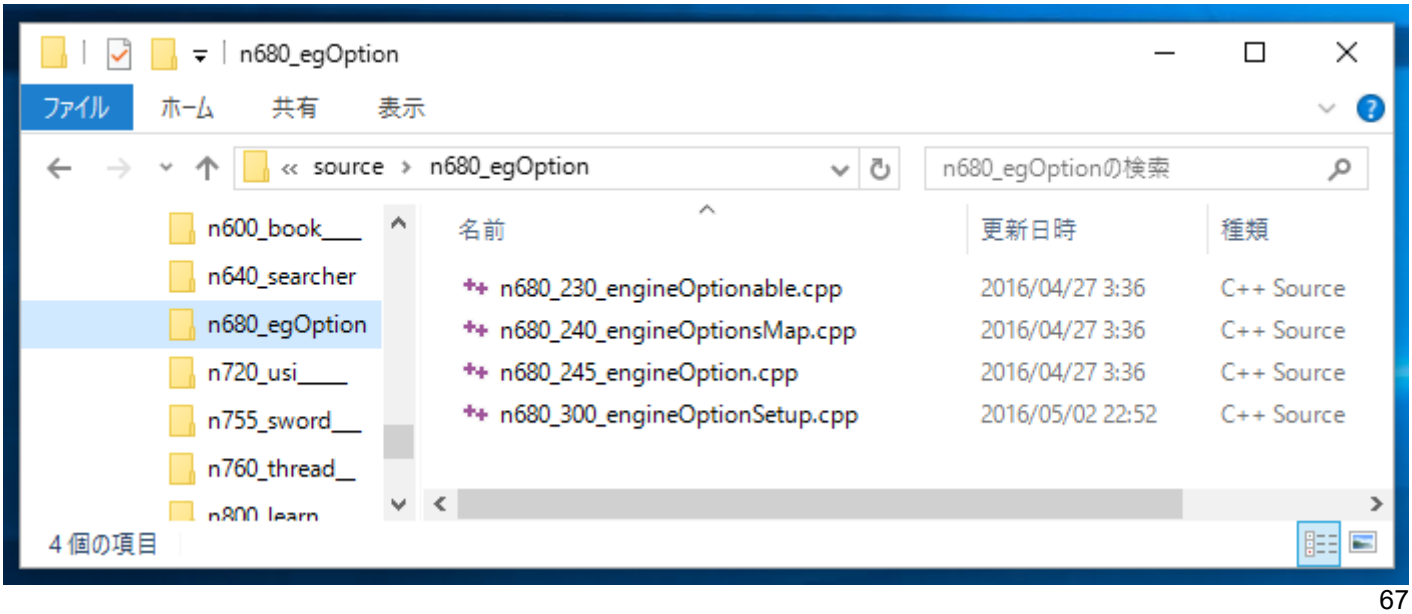
n600_book___
ファイル
ホーム
共有
表示


n520_evaluate
n560_timeMng_
n600_book___
n640_searcher
n680_egOption
n720_usi____
名前
更新日時
種類
n600_100_mt64bit.cpp
2016/04/27 3:36
C++ Source
n600_500_book.cpp
2016/05/02 3:44
C++ Source
2 個の項目

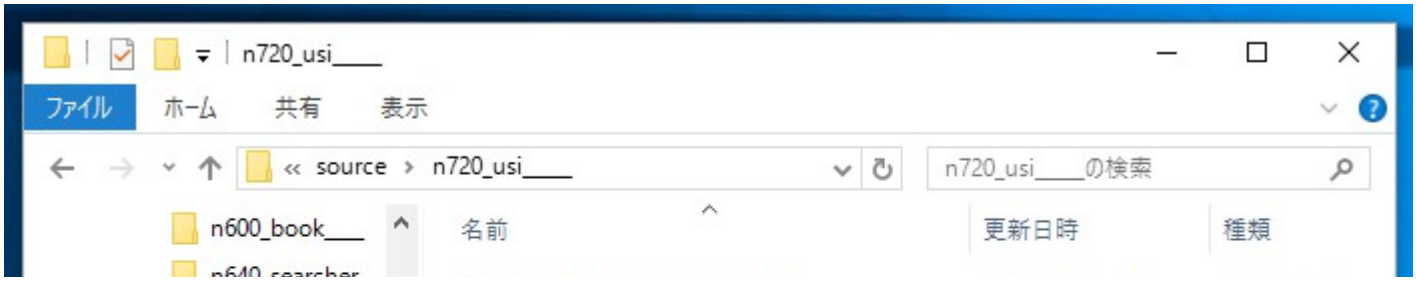
65

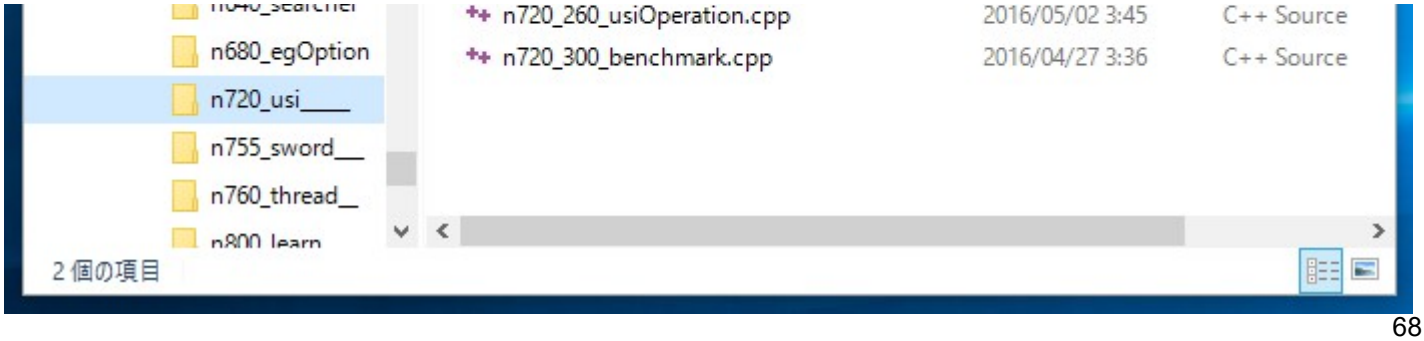

n640_searcher
ファイル
ホーム
共有
表示
« source › n640_searcher
n640_searcherの検索
n560_timeMng_
n600_book___
n640_searcher
n680_egOption
n720_usi____
n755_sword__
名前
更新日時
サイズ
n640_500_reductions.cpp
2016/04/27 3:36
1 KB
n640_510_futilityMargins.cpp
2016/04/27 3:36
1 KB
n640_520_futilityMoveCounts.cpp
2016/04/27 3:36
1 KB
3 個の項目

66

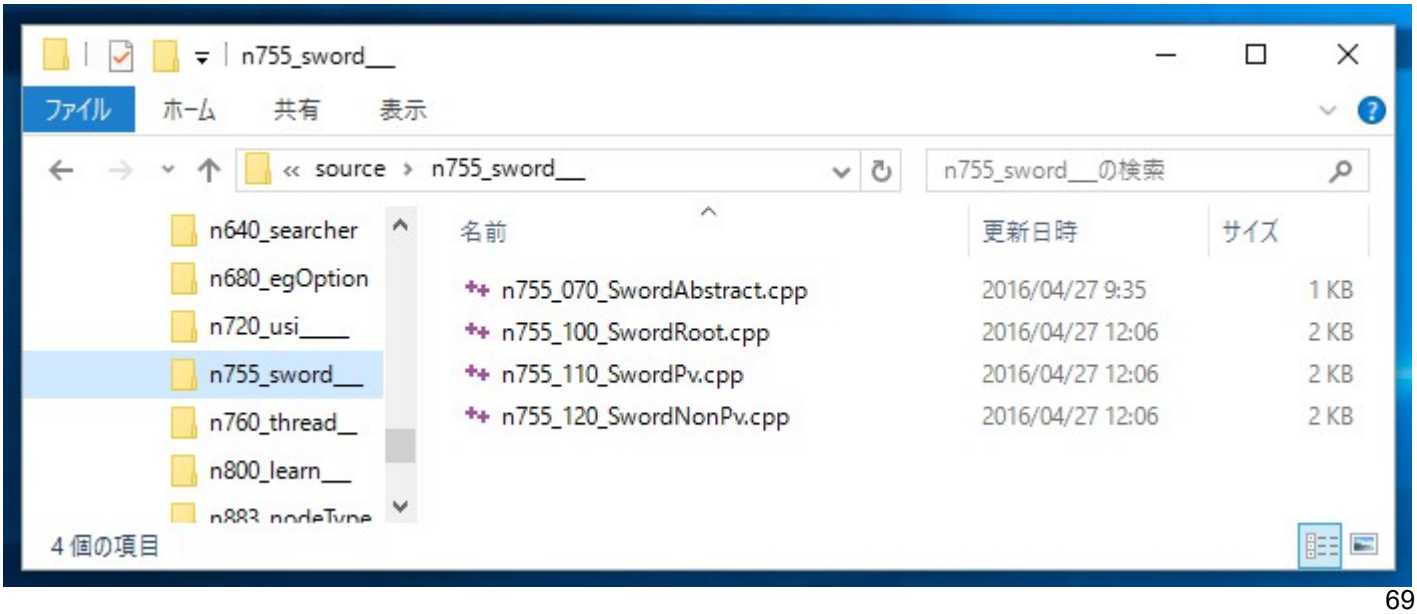

n680_egOption
ファイル
ホーム
共有
表示
« source › n680_egOption
n680_egOptionの検索
n600_book___
n640_searcher
n680_egOption
n720_usi____
n755_sword__
n760_thread__
名前
更新日時
種類
n680_230_engineOptionable.cpp
2016/04/27 3:36
C++ Source
n680_240_engineOptionsMap.cpp
2016/04/27 3:36
C++ Source
n680_245_engineOption.cpp
2016/04/27 3:36
C++ Source
n680_300_engineOptionSetup.cpp
2016/05/02 22:52
C++ Source
4 個の項目

67

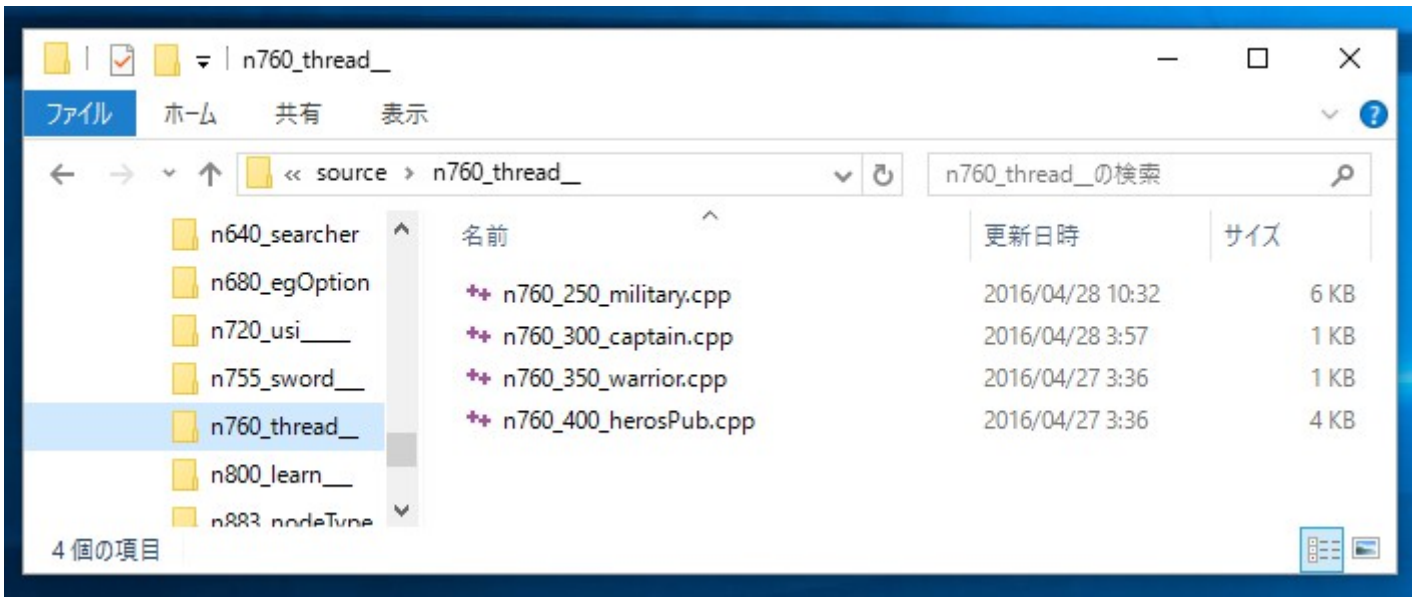

n720_usi____
ファイル
ホーム
共有
表示
« source › n720_usi____
n720_usi____の検索
n600_book___
名前
更新日時
種類


n680_egOption
n720_usi____
n755_sword__
n760_thread__
n720_260_usiOperation.cpp 2016/05/02 3:45 C++ Source
n720_300_benchmark.cpp 2016/04/27 3:36 C++ Source
2 個の項目

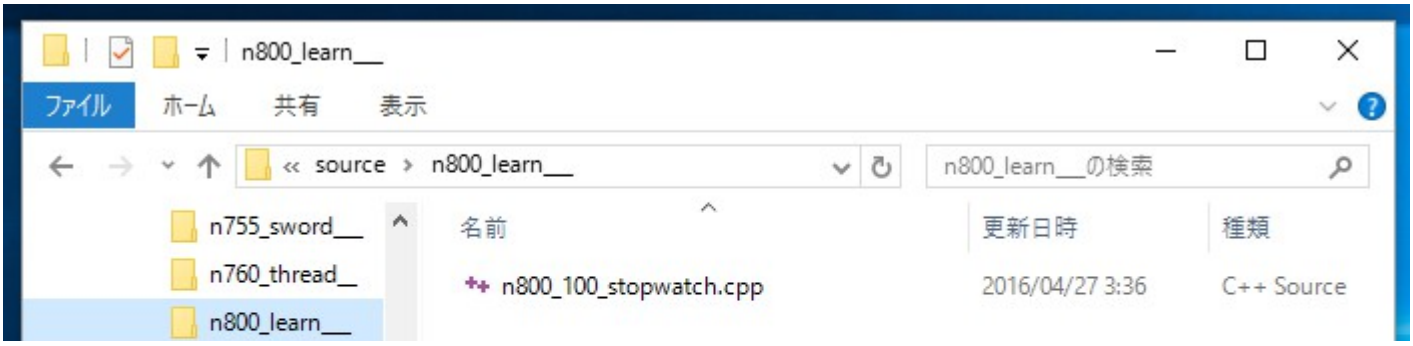
68


n755_sword__
ファイル ホーム 共有 表示
« source › n755_sword__
n755_sword__の検索
n640_searcher
n680_egOption
n720_usi____
n755_sword__
n760_thread__
n800_learn___
名前 更新日時 サイズ
n755_070_SwordAbstract.cpp 2016/04/27 9:35 1 KB
n755_100_SwordRoot.cpp 2016/04/27 12:06 2 KB
n755_110_SwordPv.cpp 2016/04/27 12:06 2 KB
n755_120_SwordNonPv.cpp 2016/04/27 12:06 2 KB
4 個の項目

69

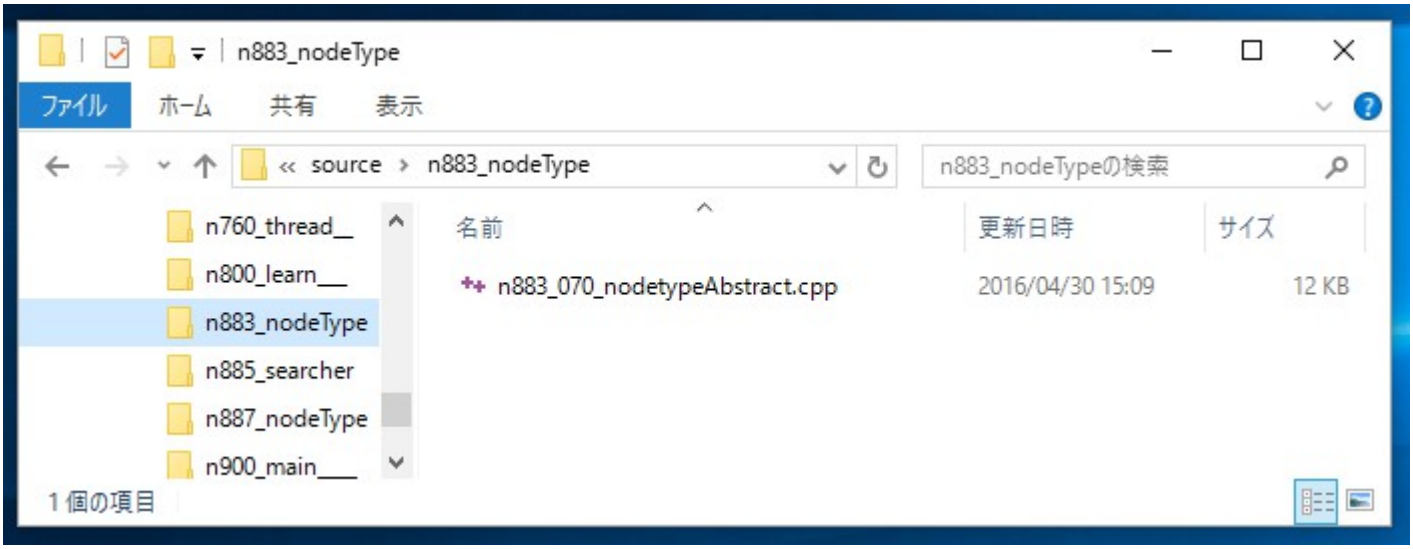

n760_thread__
ファイル ホーム 共有 表示
« source › n760_thread__
n760_thread__の検索
n640_searcher
n680_egOption
n720_usi____
n755_sword__
n760_thread__
n800_learn___
名前 更新日時 サイズ
n760_250_military.cpp 2016/04/28 10:32 6 KB
n760_300_captain.cpp 2016/04/28 3:57 1 KB
n760_350_warrior.cpp 2016/04/27 3:36 1 KB
n760_400_herosPub.cpp 2016/04/27 3:36 4 KB
4 個の項目

70

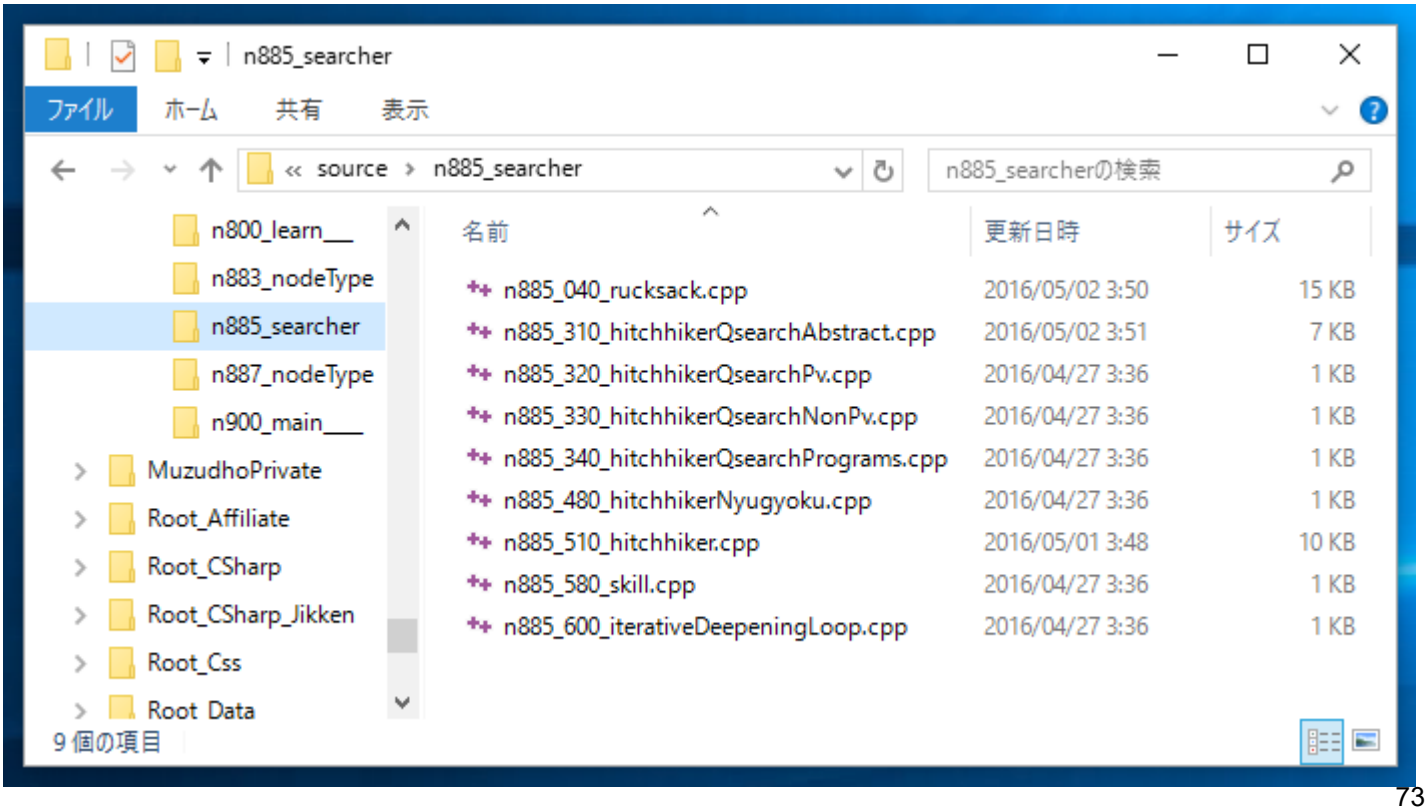

n800_learn___
ファイル ホーム 共有 表示
« source › n800_learn___
n800_learn___の検索
n755_sword__
n760_thread__
n800_learn___
名前 更新日時 種類
n800_100_stopwatch.cpp 2016/04/27 3:36 C++ Source

n883_nodeType
n885_searcher
n887_nodeType
1 個の項目

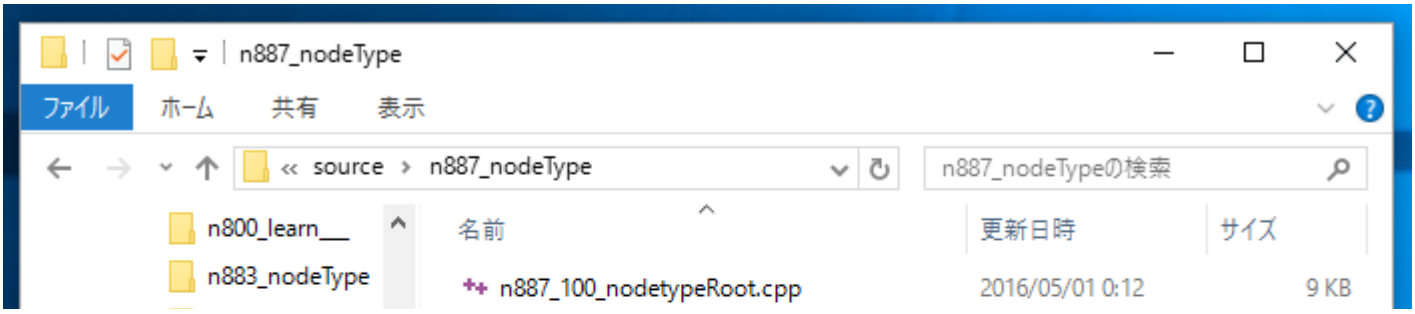
71

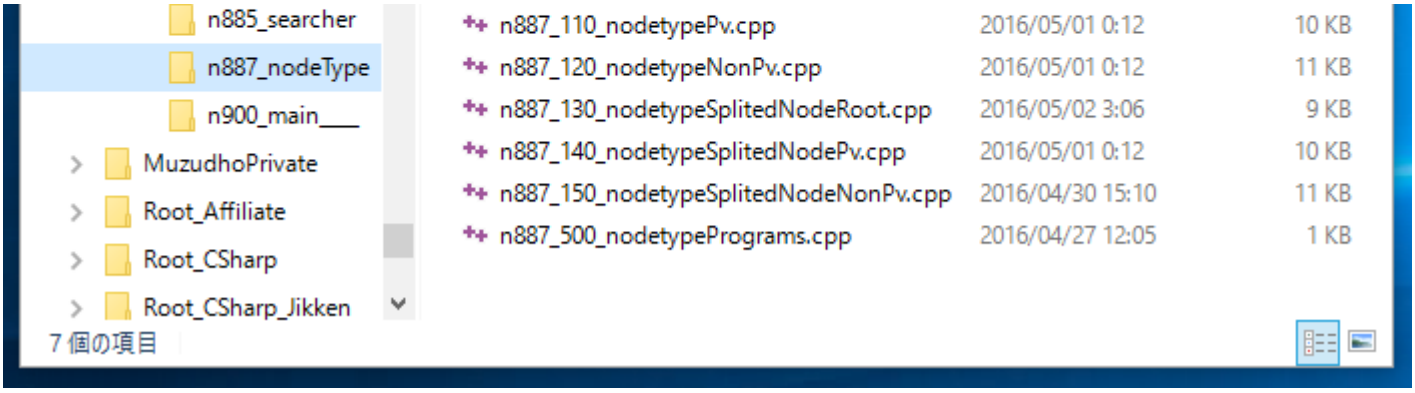

n883_nodeType
ファイル
ホーム
共有
表示
« source › n883_nodeType
n883_nodeTypeの検索
n760_thread__
n800_learn___
n883_nodeType
n885_searcher
n887_nodeType
n900_main___
名前
更新日時
サイズ
n883_070_nodetypeAbstract.cpp
2016/04/30 15:09
12 KB
1 個の項目

72

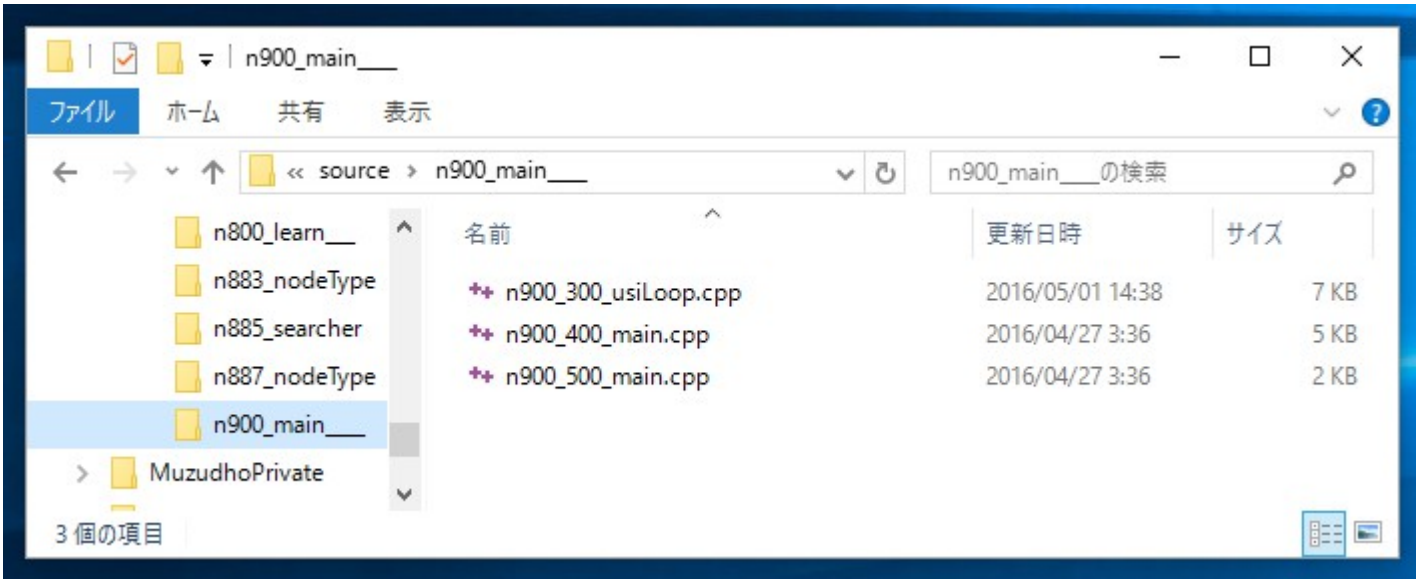

n885_searcher
ファイル
ホーム
共有
表示
« source › n885_searcher
n885_searcherの検索
n800_learn___
n883_nodeType
n885_searcher
n887_nodeType
n900_main___
MuzudhoPrivate
Root_Affiliate
Root_CSharp
Root_CSharp_Jikken
Root_Css
Root_Data
名前
更新日時
サイズ
n885_040_rucksack.cpp 2016/05/02 3:50 15 KB
n885_310_hitchhikerQsearchAbstract.cpp 2016/05/02 3:51 7 KB
n885_320_hitchhikerQsearchPv.cpp 2016/04/27 3:36 1 KB
n885_330_hitchhikerQsearchNonPv.cpp 2016/04/27 3:36 1 KB
n885_340_hitchhikerQsearchPrograms.cpp 2016/04/27 3:36 1 KB
n885_480_hitchhikerNyugyoku.cpp 2016/04/27 3:36 1 KB
n885_510_hitchhiker.cpp 2016/05/01 3:48 10 KB
n885_580_skill.cpp 2016/04/27 3:36 1 KB
n885_600_iterativeDeepeningLoop.cpp 2016/04/27 3:36 1 KB
9 個の項目

73


n887_nodeType
ファイル
ホーム
共有
表示
« source › n887_nodeType
n887_nodeTypeの検索
n800_learn___
n883_nodeType
名前
更新日時
サイズ
n887_100_nodetypeRoot.cpp
2016/05/01 0:12
9 KB



74



「 これが　きふわらぶりー　のソースコードだぜ☆！」

「 元に戻せだぜ☆」

「 無理☆」

「 こんなこと　しちゃったら　読む手間が　いっぱい増えた　じゃない！
　解散！　かいさ～ん！」

「 ざっくり　ファイルの説明を　しておくぜ☆」

始まり

ｍａｉｎ	エンジンを起動すると　ここから始まるぜ☆
ｕｓｉ	将棋所との通信に使う☆

大きな将棋の表現

ｐｏｓｉｔｉｏｎ	盤上の駒の位置☆　一局面☆
ｈａｎｄ	持ち駒の数
ｂｏｏｋ	定跡ファイルを読込む

細かな将棋の表現

ｍｏｖｅ	指し手の符号
ｓｑｕａｒｅ	将棋盤のマスに振ってある番号☆
ｃｏｌｏｒ	先後（黒先、白後）
ｐｉｅｃｅ	▲歩　△歩　▲香　△香……、と　先後付きの　駒の種類☆ 歩　香　桂……、など　先後が付かないのは　ＰｉｅｃｅＴｙｐｅ☆

学習

ｌｅａｒｎｅｒ	棋譜を読込んで　評価値データ　をせっせと調整する☆ 読み／書き　には　ｅｖａｌｕａｔｅ　を使う☆

形勢判断

ｅｖａｌｕａｔｅ	評価値データを　読み／書き
ｓｃｏｒｅ	評価値。 詰みまで何手、という数字も　スコアの端っこの数字を使う☆
ｐｉｅｃｅＳｃｏｒｅ	駒の種類の点数☆

時間管理

timeManager　使っていい時間を　探索部　に教えるもの☆

探索

search　ゲームツリーの探索☆

thread　ゲームツリーを探索するもの☆

tt　トランスポジション・テーブル。千日手の発見などに使う☆

指し手生成

movePicker　将棋盤の現局面から　指し手を　探し出すぜ☆
駒を取る手から探させたりするなど、
優先順を付けているすぐれもの☆

generateMoves　指し手を　符号（Move）に変換するぜ☆

bitboard　将棋盤８１マスの
ON／OFF　チェックシート☆

init　エンジン起動時に、たくさんの
ビットボードを　せっせと　作っている。

その他

benchmark　対局中に頻出するプログラムを調べるのに使う☆
common　細々したものを詰め込んである☆
ifdef　稲庭対策のON／OFF　など　いろいろON／OFF☆
mt64bit　乱数☆
overloadEnumOperator　Enum型を操作しやすくするもの☆

学習（ｌｅａｒｎ）の説明はスキップするぜ☆

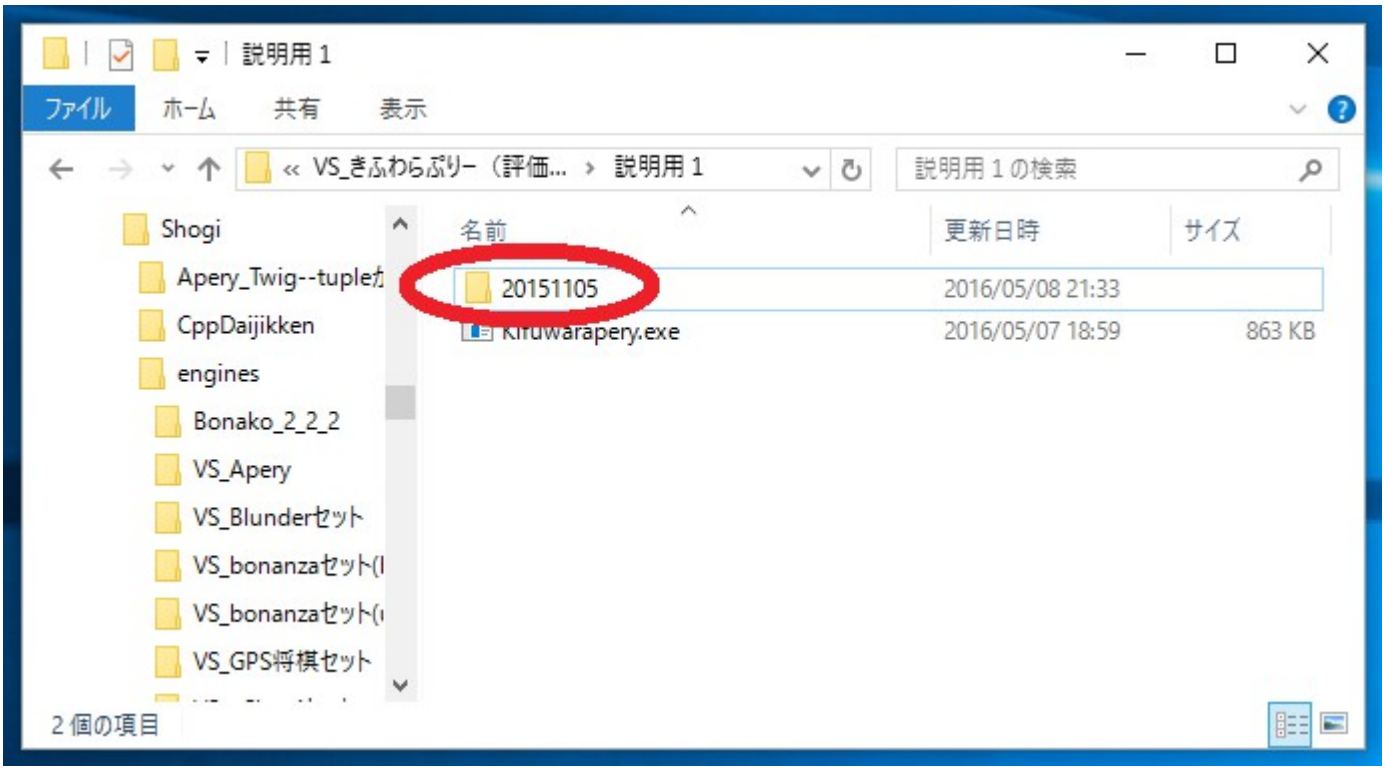
GAKUSYU

学習はやってないぜ☆！

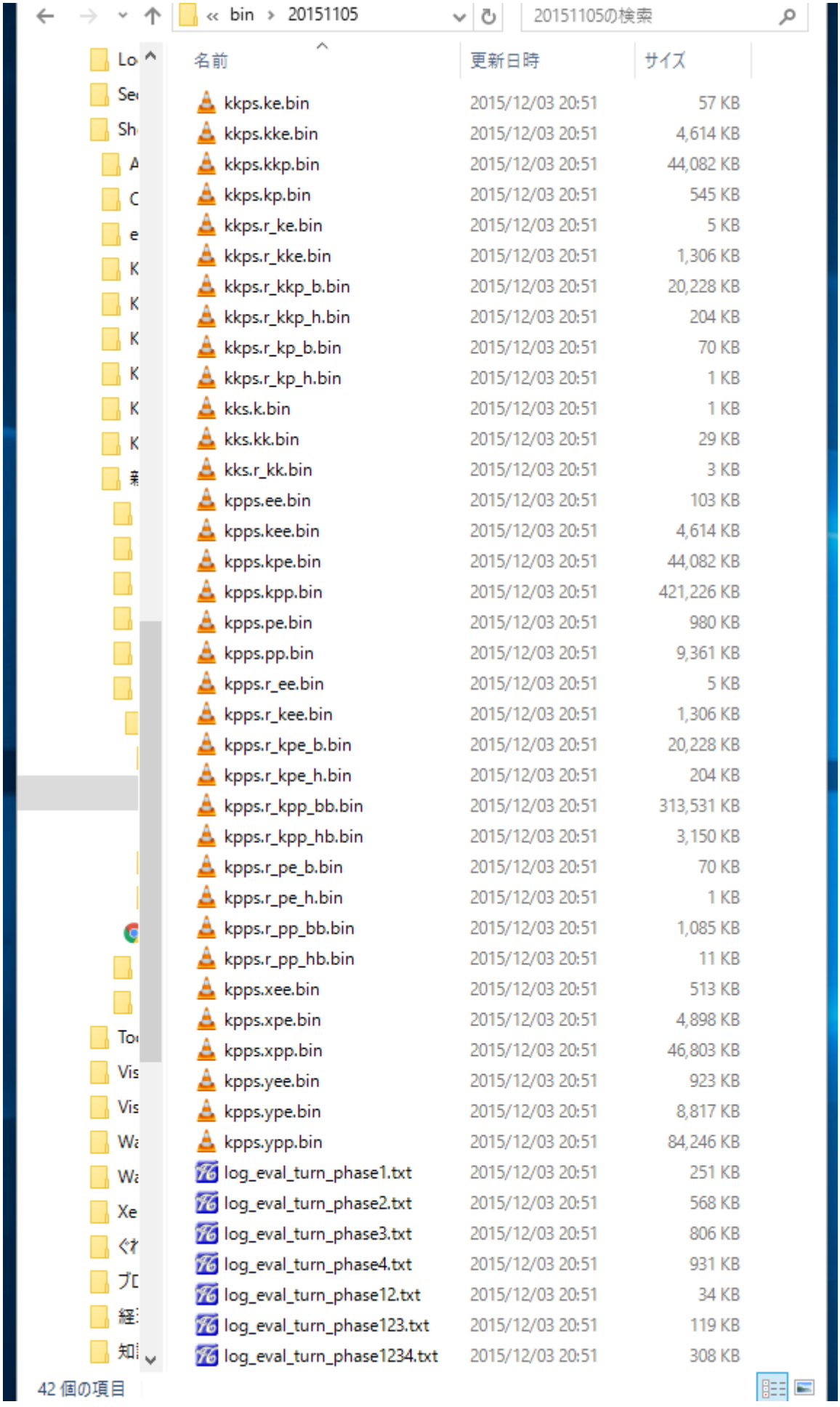
「 今大会は　機械学習はやらない、　と決めていたので
機械学習の説明は　スキップするぜ☆」

「 機械学習抜きで　レーティングを　６００も下げたのね」

「 やったことないんだが、機械学習をすると、
毎月日ファイルが出来上がると　推測するぜ☆」



「 ２０１５年１１月５日に　学習でもやったのだろう☆
中身を見てみようぜ☆」


« bin › 20151105
201511105の検索
名前
更新日時
サイズ
kkps.ke.bin 2015/12/03 20:51 57 KB
kkps.kke.bin 2015/12/03 20:51 4,614 KB
kkps.kkp.bin 2015/12/03 20:51 44,082 KB
kkps.kp.bin 2015/12/03 20:51 545 KB
kkps.r_ke.bin 2015/12/03 20:51 5 KB
kkps.r_kke.bin 2015/12/03 20:51 1,306 KB
kkps.r_kkp_b.bin 2015/12/03 20:51 20,228 KB
kkps.r_kkp_h.bin 2015/12/03 20:51 204 KB
kkps.r_kp_b.bin 2015/12/03 20:51 70 KB
kkps.r_kp_h.bin 2015/12/03 20:51 1 KB
kks.k.bin 2015/12/03 20:51 1 KB
kks.kk.bin 2015/12/03 20:51 29 KB
kks.r_kk.bin 2015/12/03 20:51 3 KB
kpps.ee.bin 2015/12/03 20:51 103 KB
kpps.kee.bin 2015/12/03 20:51 4,614 KB
kpps.kpe.bin 2015/12/03 20:51 44,082 KB
kpps.kpp.bin 2015/12/03 20:51 421,226 KB
kpps.pe.bin 2015/12/03 20:51 980 KB
kpps.pp.bin 2015/12/03 20:51 9,361 KB
kpps.r_ee.bin 2015/12/03 20:51 5 KB
kpps.r_kee.bin 2015/12/03 20:51 1,306 KB
kpps.r_kpe_b.bin 2015/12/03 20:51 20,228 KB
kpps.r_kpe_h.bin 2015/12/03 20:51 204 KB
kpps.r_kpp_bb.bin 2015/12/03 20:51 313,531 KB
kpps.r_kpp_hb.bin 2015/12/03 20:51 3,150 KB
kpps.r_pe_b.bin 2015/12/03 20:51 70 KB
kpps.r_pe_h.bin 2015/12/03 20:51 1 KB
kpps.r_pp_bb.bin 2015/12/03 20:51 1,085 KB
kpps.r_pp_hb.bin 2015/12/03 20:51 11 KB
kpps.xee.bin 2015/12/03 20:51 513 KB
kpps.xpe.bin 2015/12/03 20:51 4,898 KB
kpps.xpp.bin 2015/12/03 20:51 46,803 KB
kpps.yee.bin 2015/12/03 20:51 923 KB
kpps.ype.bin 2015/12/03 20:51 8,817 KB
kpps.ypp.bin 2015/12/03 20:51 84,246 KB
log_eval_turn_phase1.txt 2015/12/03 20:51 251 KB
log_eval_turn_phase2.txt 2015/12/03 20:51 568 KB
log_eval_turn_phase3.txt 2015/12/03 20:51 806 KB
log_eval_turn_phase4.txt 2015/12/03 20:51 931 KB
log_eval_turn_phase12.txt 2015/12/03 20:51 34 KB
log_eval_turn_phase123.txt 2015/12/03 20:51 119 KB
log_eval_turn_phase1234.txt 2015/12/03 20:51 308 KB
42 個の項目

「ｂｉｎファイルがいっぱいあるだろ☆
ファイルの名前を見ると、三駒関係は、ＫＫＰ、ＫＫ、ＫＰＰ　の
３種類を使っているのかもしれないな☆」

「ｓとか、ｒ＿とか　ｘとか　ｙとか　ｅとか　ｈとか　ｂとか、
ｈｂとか　ｂｂ　って何よ！」

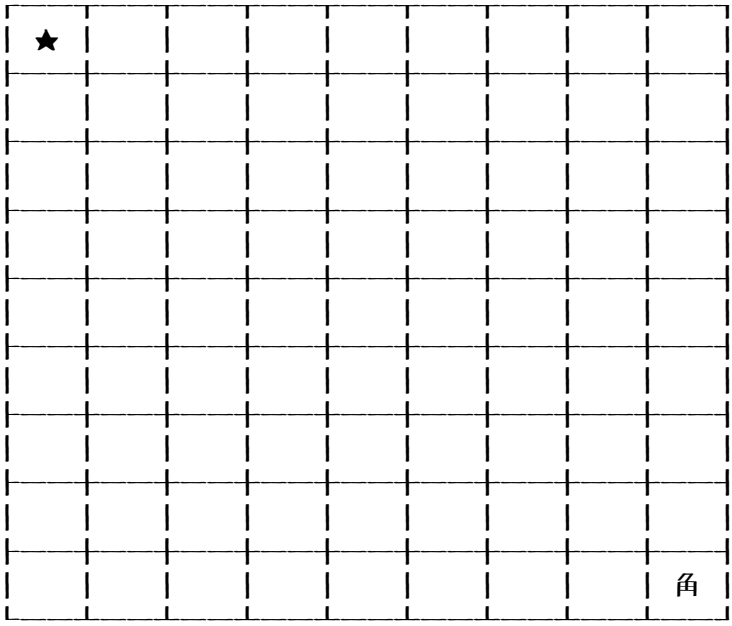
「ｋｋｐｓ．　ｋｋｓ．　ｋｐｐｓ．　というのは、
ＫＫＰ、　ＫＫ、　ＫＰＰ　に対応しているぜ☆
その後ろに　ｒ＿、　ｋ、　ｐ、　ｅ、　ｘ、　ｙ、　ｈ、　ｂ
というのがあるな☆　説明するぜ☆」

r＿

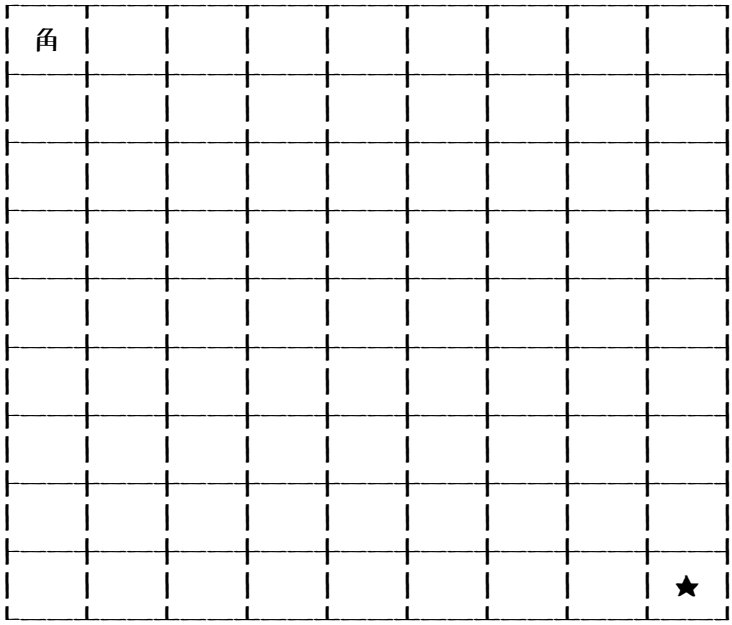
Relative

リレーティブのRなのかだぜ☆！？

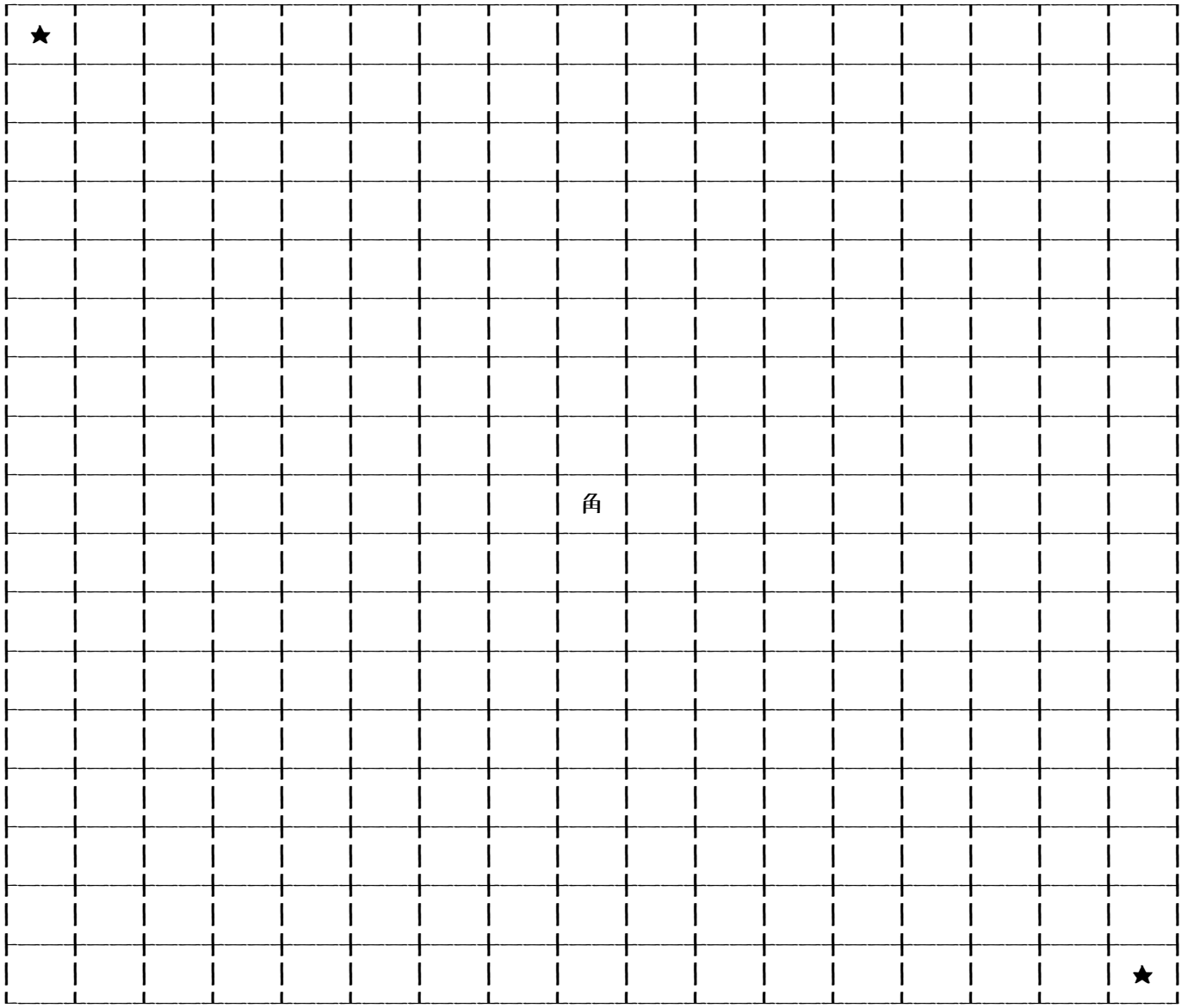
「ｒ＿　というのは　相対位置だぜ☆
　将棋盤というのは　９×９　のマスがあるんだが……☆」



「１九の角から見て　９一の地点は　上に９個、左に９個☆
　そして……☆」



「９ーの角から見て　１九の地点は　下に９個、右に９個☆
つまり……☆」



「１７×１７　サイズの将棋盤があれば、
自分の位置を中心とした、周りの駒について　点数をメモできるわけだぜ☆」

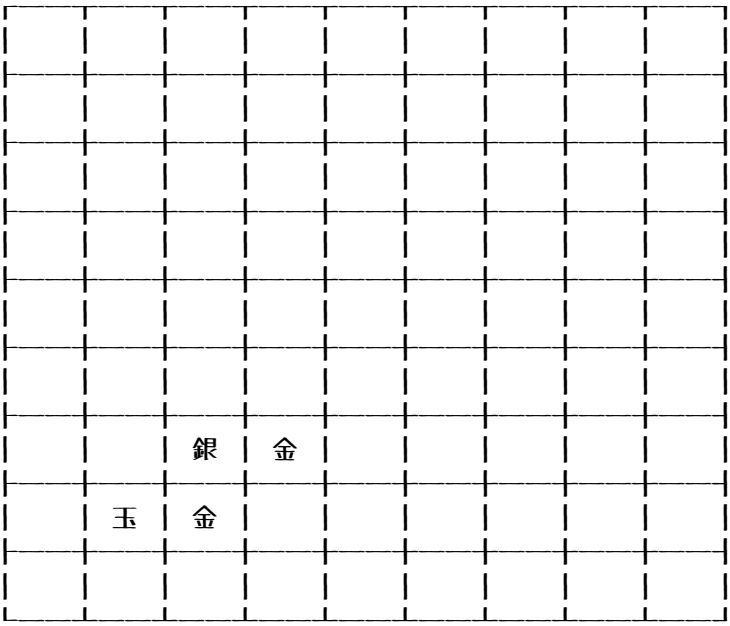
「横着するなだぜ☆　ペイントツールを使って　お絵描きしようぜ☆？」

「エイプリー＿大樹の枝　は　相対位置を使って　評価値を付けているわけ？」

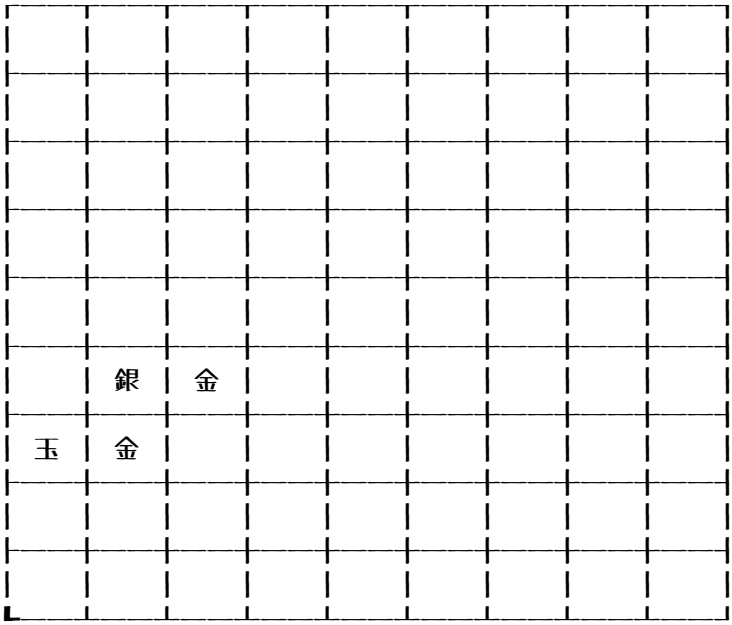
「そうなんじゃないか☆？」

「なんで？
　ぜーんぶ　9×9　の将棋盤の　絶対位置　で評価を付けないわけ？」

「宇宙サイズのハナシになるんだが……☆
　棋譜を使った学習で　開発者が　頭を悩ますのは、
　棋譜が少ないということなんだぜ☆
　例えば……☆」



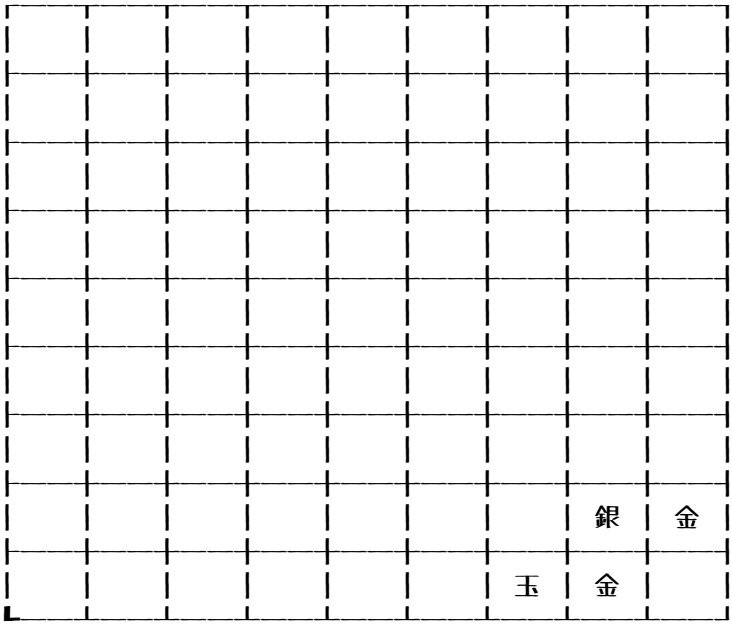
「こんな矢倉の形は　山ほど　棋譜があるので　勉強は十分なんだぜ☆
　次に……☆」



「 こんな　実戦で出てきそうもないけど　矢倉と同じ形☆
　絶対位置で　学習していると　勉強不足になりそうなんだが、
　相対位置で　学習しておけば、矢倉ということがわかるだろ☆」

「 相対位置は　練習不足の局面を補うための　応用力を　付けているのね」

「 じゃあ全部　相対位置にしようぜ☆？」



「 相対位置だけだと問題があるだろ☆」

k

King

キングのKなのかだぜ☆！？

「Kは　王様の位置だぜ☆
絶対位置なら　9×9　の将棋盤の位置を　0～80　で表すぜ☆
相対位置なら　17×17　の将棋盤の位置を　0～288　で表すぜ☆」

「絶対位置のKK　なら、
片方のKの位置［0～80］　×　もう片方のKの位置［0～80］だぜ☆
相対位置のKK　なら、Kを中心に　もう片方のKの位置が
［0～16］×［0～16］だぜ☆」

「きふわらちゃんは　偉いのねぇ」

「わたしも　偉いんだぜ☆！」

p

Ｐｉｅｃｅ

ピースのＰなのかだぜ☆！？

「だいたい、玉　以外のものを　Ｐ　と呼ぶぜ☆
盤上の歩や　香車を　Ｐ　と呼ぶことも当然のことながら、
持ち駒の枚数も　Ｐ　だぜ☆」

「歩を１枚持っているのもＰ、
歩を２枚持っているのもＰ　とみなすことがあるぜ☆
▲玉　×　△玉　×　▲香　　も三駒関係なら、
▲玉　×　△玉　×　▲歩１枚　も三駒関係、
▲玉　×　△玉　×　▲歩２枚　も三駒関係だぜ☆
比べる項目なら　なんでも　Ｐ　だぜ☆」

「きふわらちゃんは　よく勉強しているのねぇ」

「教えている　わたしの方が　もっと勉強しているぜ☆」

「なんで　玉　だけ　特別扱いなの？」

「Ｐ×Ｐ×Ｐ　にしたとき　玉×玉×玉　になったら無駄臭いだろ☆
飛×飛×飛　になっても　無駄臭いぜ☆
ＫＰＰと　玉だけ別勘定で決め打ちしておけば　最悪でも　玉飛飛　の組み合わせで
無駄にならないぜ☆
だから　二駒関係でやるなら　Ｋは分けなくていいぜ☆　これは私の推測だぜ☆」

「ふぅ～ん」

「褒めてくれてもいいんだぜ☆？」

e

Ｅｆｆｅｃｔ

エフェクトのｅなのかだぜ☆！？

「ｅ　って何よ？」

「利きだぜ☆　チェスでは　Ｃｏｎｔｒｏｌ　の　ｃ　になるな☆
　コンピュータ将棋は　日本人英語　で、効くの　ｅ　だぜ☆」

「利きには、どの駒かは関係ないぜ☆
　先手番がどこに利いているか、後手番がどこに利いているか、だけ覚えていて、
　絶対位置なら　２×８１、
　相対位置なら　２×１７×１７　と、１マス１マス覚えておくことになるぜ☆」

「きふわらべちゃんは　偉いわねぇ」

「……☆！」

ｘ　と　ｙ

YOKO　JIKU　　TATE　JIKU

横軸と、縦軸だぜ☆

「ｘ　って何よ？」

「ｘ　と　ｙ　は　対になっているんだぜ☆
ｘ　は　筋、　ｙ　は　段　だぜ☆
三駒関係のＫでは　５九　の位置に玉があるとき、と調べるんだが、
ｘとｙでは　５の筋に玉があるとき、　九の段に玉があるとき、
と２つに分けて　覚えることもあるんだぜ☆」

「ｘ　と　ｙ　は　玉　の位置の代わりに使っているぜ☆
ここで　段は　１～９　とそのまんまなんだが、
筋は　鏡面　で考えていて　５の筋を　中心として、
５～１、５～９　の左右は区別していないぜ☆
推測なんだが、ｘとｙ　と分けたのは、ｘを鏡面に考えることで
左半分の学習を　右半分にも共用できて、
学習した棋譜の数の２倍、応用できる局面を広げていると思うんだぜ☆」

「ふぅーん。　きふわらべちゃんは　偉いわねぇ」

「……☆！」

ｈ　と　ｂ

HAND　　BAN

ハンドと、盤だろうか……☆？

「ｈ　って何よ？」

「ｈ　は　ハンドで」

「持ち駒☆　ｂ　は　恐らく　盤上の駒で」

「ｂ　かだぜ☆　ｈｈ　は」

「ｐｐ　に対応していて、持ち駒Ｐ　と　持ち駒Ｐ」

「ではないかと推測しているんだぜ☆　同じく　ｂｂ　も」

「ＰＰ　に対応していて、盤上の駒Ｐ　と　盤上の駒Ｐ　ではないかと
推測しているんだぜ☆」

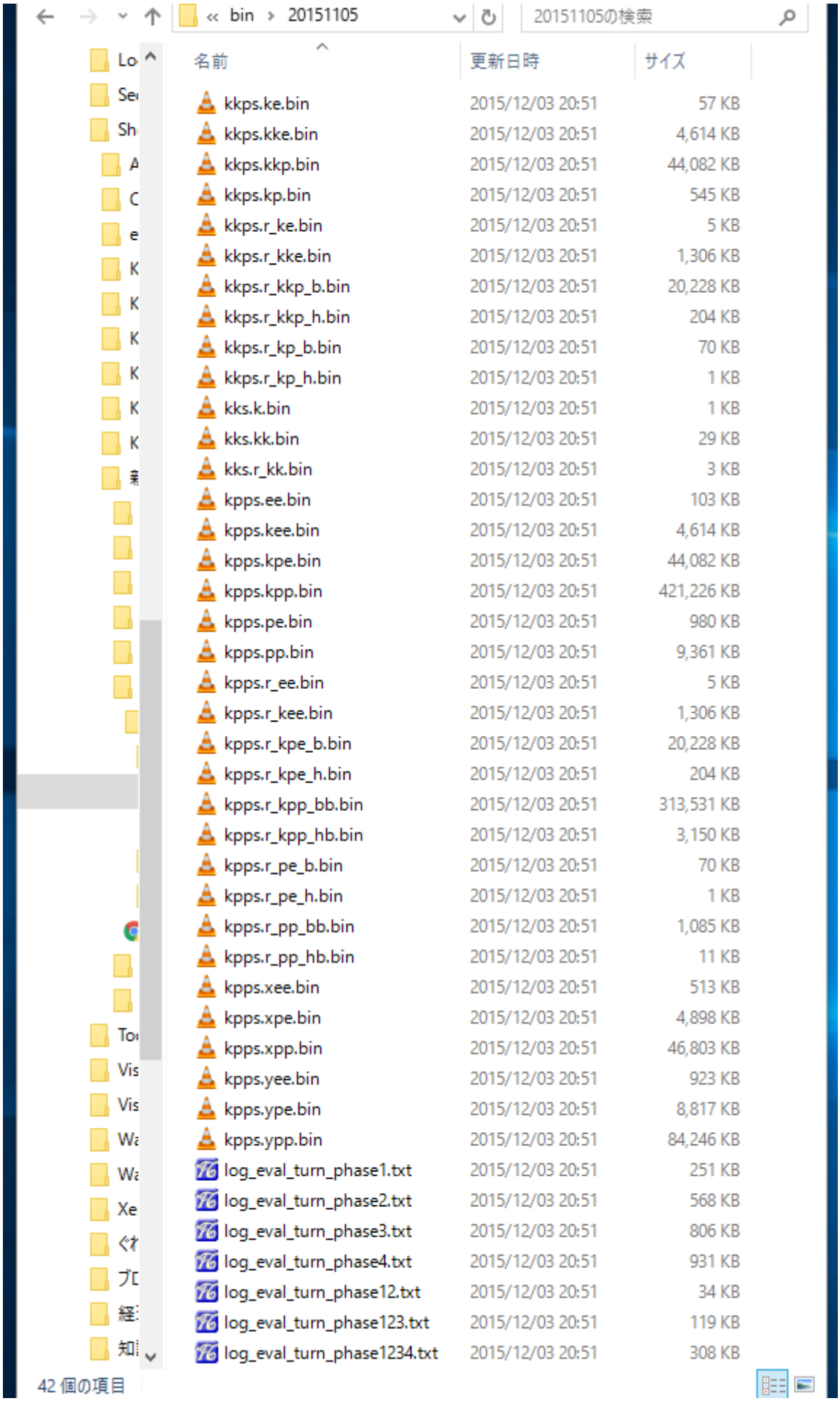
「なんで　２スレッドで　声が聞こえてくるのよ！」

「 ……☆」

「 きふわらべちゃんは　偉いわねぇ」

「 ……☆」

「 ここで、20151105 フォルダーの中の .bin ファイルを再掲するぜ☆」


« bin › 20151105
20151105の検索
名前
更新日時
サイズ
kkps.ke.bin 2015/12/03 20:51 57 KB
kkps.kke.bin 2015/12/03 20:51 4,614 KB
kkps.kkp.bin 2015/12/03 20:51 44,082 KB
kkps.kp.bin 2015/12/03 20:51 545 KB
kkps.r_ke.bin 2015/12/03 20:51 5 KB
kkps.r_kke.bin 2015/12/03 20:51 1,306 KB
kkps.r_kkp_b.bin 2015/12/03 20:51 20,228 KB
kkps.r_kkp_h.bin 2015/12/03 20:51 204 KB
kkps.r_kp_b.bin 2015/12/03 20:51 70 KB
kkps.r_kp_h.bin 2015/12/03 20:51 1 KB
kks.k.bin 2015/12/03 20:51 1 KB
kks.kk.bin 2015/12/03 20:51 29 KB
kks.r_kk.bin 2015/12/03 20:51 3 KB
kpps.ee.bin 2015/12/03 20:51 103 KB
kpps.kee.bin 2015/12/03 20:51 4,614 KB
kpps.kpe.bin 2015/12/03 20:51 44,082 KB
kpps.kpp.bin 2015/12/03 20:51 421,226 KB
kpps.pe.bin 2015/12/03 20:51 980 KB
kpps.pp.bin 2015/12/03 20:51 9,361 KB
kpps.r_ee.bin 2015/12/03 20:51 5 KB
kpps.r_kee.bin 2015/12/03 20:51 1,306 KB
kpps.r_kpe_b.bin 2015/12/03 20:51 20,228 KB
kpps.r_kpe_h.bin 2015/12/03 20:51 204 KB
kpps.r_kpp_bb.bin 2015/12/03 20:51 313,531 KB
kpps.r_kpp_hb.bin 2015/12/03 20:51 3,150 KB
kpps.r_pe_b.bin 2015/12/03 20:51 70 KB
kpps.r_pe_h.bin 2015/12/03 20:51 1 KB
kpps.r_pp_bb.bin 2015/12/03 20:51 1,085 KB
kpps.r_pp_hb.bin 2015/12/03 20:51 11 KB
kpps.xee.bin 2015/12/03 20:51 513 KB
kpps.xpe.bin 2015/12/03 20:51 4,898 KB
kpps.xpp.bin 2015/12/03 20:51 46,803 KB
kpps.yee.bin 2015/12/03 20:51 923 KB
kpps.ype.bin 2015/12/03 20:51 8,817 KB
kpps.ypp.bin 2015/12/03 20:51 84,246 KB
log_eval_turn_phase1.txt 2015/12/03 20:51 251 KB
log_eval_turn_phase2.txt 2015/12/03 20:51 568 KB
log_eval_turn_phase3.txt 2015/12/03 20:51 806 KB
log_eval_turn_phase4.txt 2015/12/03 20:51 931 KB
log_eval_turn_phase12.txt 2015/12/03 20:51 34 KB
log_eval_turn_phase123.txt 2015/12/03 20:51 119 KB
log_eval_turn_phase1234.txt 2015/12/03 20:51 308 KB
42 個の項目

「 なるほど……、ｋｋｐｓ．ｋｅ．ｂｉｎ　ファイルは
　ＫＫＰの三駒関係で、玉と、利き……、」

「 ｋｅじゃ　二駒関係じゃないのよ！」

「 でも実際、ＫＥ　は　ＫＫＰ　の計算の途中で　足しこまれているんだぜ☆
　これだと、自玉の位置がどこにあるかに関わらず、
　敵玉と敵金の二駒関係の点数は同じだとみなして
　二駒関係の点数を　三駒関係に重ね合わせているように見えるぜ☆」

「 よく分かんないけど　二駒関係の点数も　三駒関係に入れてるのね」

「 多分　そう思う☆」

「 お父んは　『多分そう思う駆動』　だからな☆」

「 なぜ確認しなかったんだ、報告しろ☆！　って　言われるよな☆
　上司の評価はナイアガラだぜ☆
　わたしには　分かっていない　というタイミングは　分かる途中なので
　分かっていないから聞く　という動作がないぜ☆」

「 分かっていないと　自己認識　できた時点で相談するのではなく、
　５分経った時点で　分かってなかったら　相談しなさいよ！」

「 ……☆！」

学習部の実装を見てみようぜ☆？

RENDOU

ファイル名と　配列の名前は　連動☆

「 ここで、エイプリー＿大樹の枝　の実装を見てみようぜ☆」

evaluate.hpp

E:¥...¥src¥evaluate.hpp

ファイル(F) 編集(E) 検索(S) 整形(P) 表示(V) ツール(T) フォルダ(L) HTMLタグ(M) ウインドウ(W) ヘルプ(H)

```
160    // 例えば kpp だったら、k が優先的に小さくなるようする。左右の対称も含めてアクセス位
161    // ただし、kkp に関する項目 (kkp, r_kkp_b, r_kkp_h) のみ、p は味方の駒として扱うので、
162    struct KPPElements {
163        KPPType dummy; // 一次元配列に変換したとき、符号で += を表すようにしているが、inc
164        KPPType kpp[SquareNoLeftNum][fe_end][fe_end];
165        // 相対位置は[file][rank]の順
166        KPPType r_kpp_bb[PieceNone][17][17][PieceNone][17][17];
167        KPPType r_kpp_hb[fe_hand_end][PieceNone][17][17];
168        KPPType xpp[FileNoLeftNum][fe_end][fe_end];
169        KPPType ypp[RankNum][fe_end][fe_end];
170        KPPType pp[fe_end][fe_end];
171        KPPType r_pp_bb[PieceNone][PieceNone][17][17];
172        KPPType r_pp_hb[fe_hand_end][PieceNone];
173
174        // e は Effect の頭文字で利きを表す。(Control = 利き という説もあり。)
175        // todo: 玉の利きは全く無視しているけれど、それで良いのか？
176        KPPType kpe[SquareNoLeftNum][fe_end][ColorNum][SquareNum];
177        KPPType kee[SquareNoLeftNum][ColorNum][SquareNum][ColorNum][SquareNum];
178        KPPType r_kpe_b[PieceNone][17][17][ColorNum][17][17];
179        KPPType r_kpe_h[fe_hand_end][ColorNum][17][17];
180        KPPType r_kee[ColorNum][17][17][ColorNum][17][17];
181        KPPType xpe[FileNoLeftNum][fe_end][ColorNum][SquareNum];
182        KPPType xee[FileNoLeftNum][ColorNum][SquareNum][ColorNum][SquareNum];
183        KPPType ype[RankNum][fe_end][ColorNum][SquareNum];
184        KPPType yee[RankNum][ColorNum][SquareNum][ColorNum][SquareNum];
185        KPPType pe[fe_end][ColorNum][SquareNum];
186        KPPType ee[ColorNum][SquareNum][ColorNum][SquareNum];
187        KPPType r_pe_b[PieceNone][ColorNum][17][17];
188        KPPType r_pe_h[fe_hand_end][ColorNum];
189        KPPType r_ee[ColorNum][ColorNum][17][17];
190    };
191    KPPElements kpps;
192
193    struct KKPElements {
194        KKPType dummy; // 一次元配列に変換したとき、符号で += を表すようにしているが、inc
195        KKPType kkp[SquareNoLeftNum][SquareNum][fe_end];
196        KKPType kp[SquareNoLeftNum][fe_end];
197        KKPType r_kkp_b[17][17][PieceNone][17][17];
198        KKPType r_kkp_h[17][17][fe_hand_end];
199        KKPType r_kp_b[PieceNone][17][17];
200        KKPType r_kp_h[fe_hand_end];
201
202        KKPType kke[SquareNoLeftNum][SquareNum][ColorNum][SquareNum];
203        KKPType ke[SquareNoLeftNum][ColorNum][SquareNum];
```

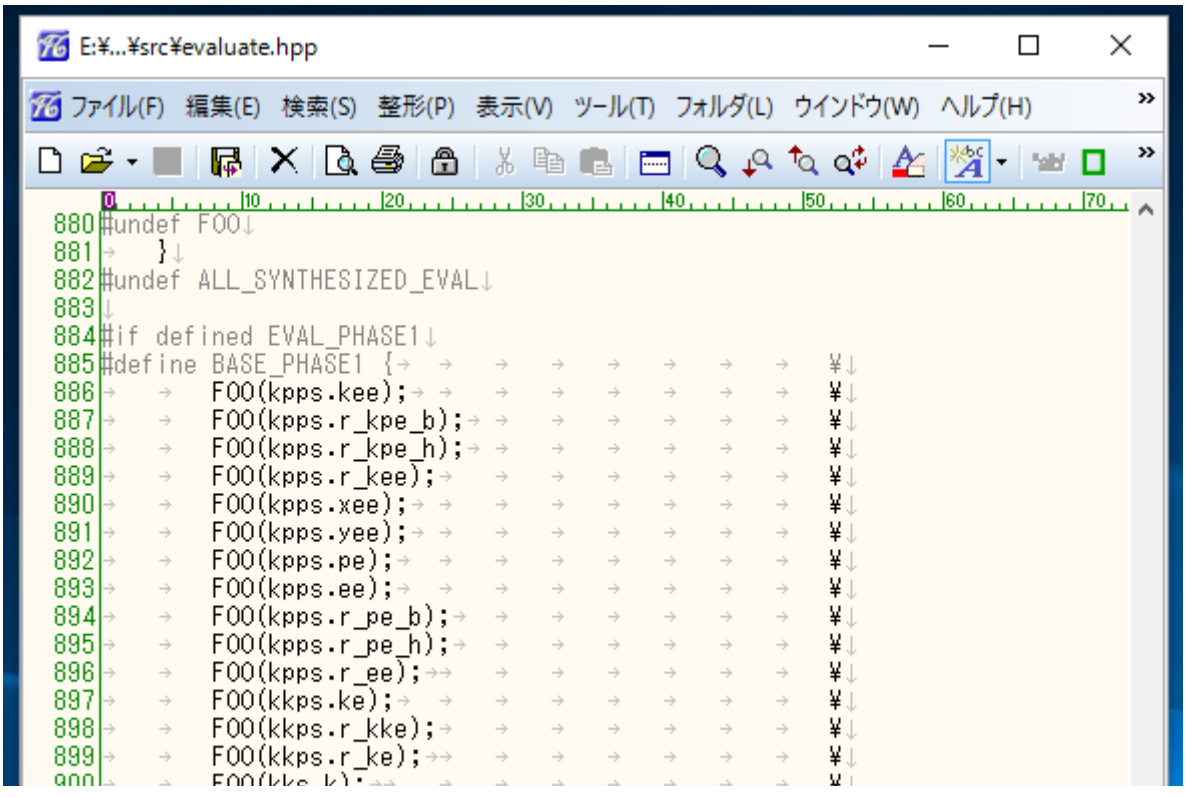
```
204→   →   KKPType r_kke[17][17][ColorNum][17][17];↓
205→   →   KKPType r_ke[ColorNum][17][17];↓
206→   };↓
207→   KKPElements kkps;↓
208↓
209→   struct KKElements {↓
210→   →   KKType dummy; // 一次元配列に変換したとき、符号で += を表すようにしているが、inde
211→   →   KKType kk[SquareNoLeftNum][SquareNum];↓
212→   →   KKType k[SquareNoLeftNum];↓
213→   →   KKType r_kk[17][17];↓
214→   };↓
215→   KKElements kks;↓
216↓
217→   // これらは↑のメンバ変数に一次元配列としてアクセスする為のもの。↓
```

「ＫＰＰＥｌｅｍｅｎｔｓ　とか書いてある　ｓｔｒｕｃｔ　のブロックに
ファイル名で見たことのある名前の　配列が並んでるだろ☆」

「ｋｋｐｓ．ｋｅ．ｂｉｎ　に対応するのは……、
２０７行目　構造体変数　ｋｋｐｓ　の、
２０３行目　ｋｅメンバー配列……、あるわね」

「お察しの通り、マクロで　ファイル名と　配列の名前は　連動しているんだぜ☆
その部分を　見ておこうぜ☆」

ｅｖａｌｕａｔｅ．ｈｐｐ



```
901    }
902#else
903#define BASE_PHASE1
904#endif
905
906#if defined EVAL_PHASE2
907#define BASE_PHASE2 {                                   \
908        FOO(kpps.r_pp_bb);                              \
909        FOO(kpps.r_pp_hb);                              \
910        FOO(kkps.r_kp_b);                               \
911        FOO(kkps.r_kp_h);                               \
912        FOO(kks.r_kk);                                  \
913    }
914#else
915#define BASE_PHASE2
916#endif
917
918#if defined EVAL_PHASE3
919#define BASE_PHASE3 {                                   \
920        FOO(kpps.r_kpp_bb);                             \
921        FOO(kpps.r_kpp_hb);                             \
922        FOO(kpps.pp);                                   \
923        FOO(kpps.kpe);                                  \
924        FOO(kpps.xpe);                                  \
925        FOO(kpps.ype);                                  \
926        FOO(kkps.kp);                                   \
927        FOO(kkps.r_kkp_b);                              \
928        FOO(kkps.r_kkp_h);                              \
929        FOO(kkps.kke);                                  \
930        FOO(kks.kk);                                    \
931    }
932#else
933#define BASE_PHASE3
934#endif
935
936#if defined EVAL_PHASE4
937#define BASE_PHASE4 {                                   \
```

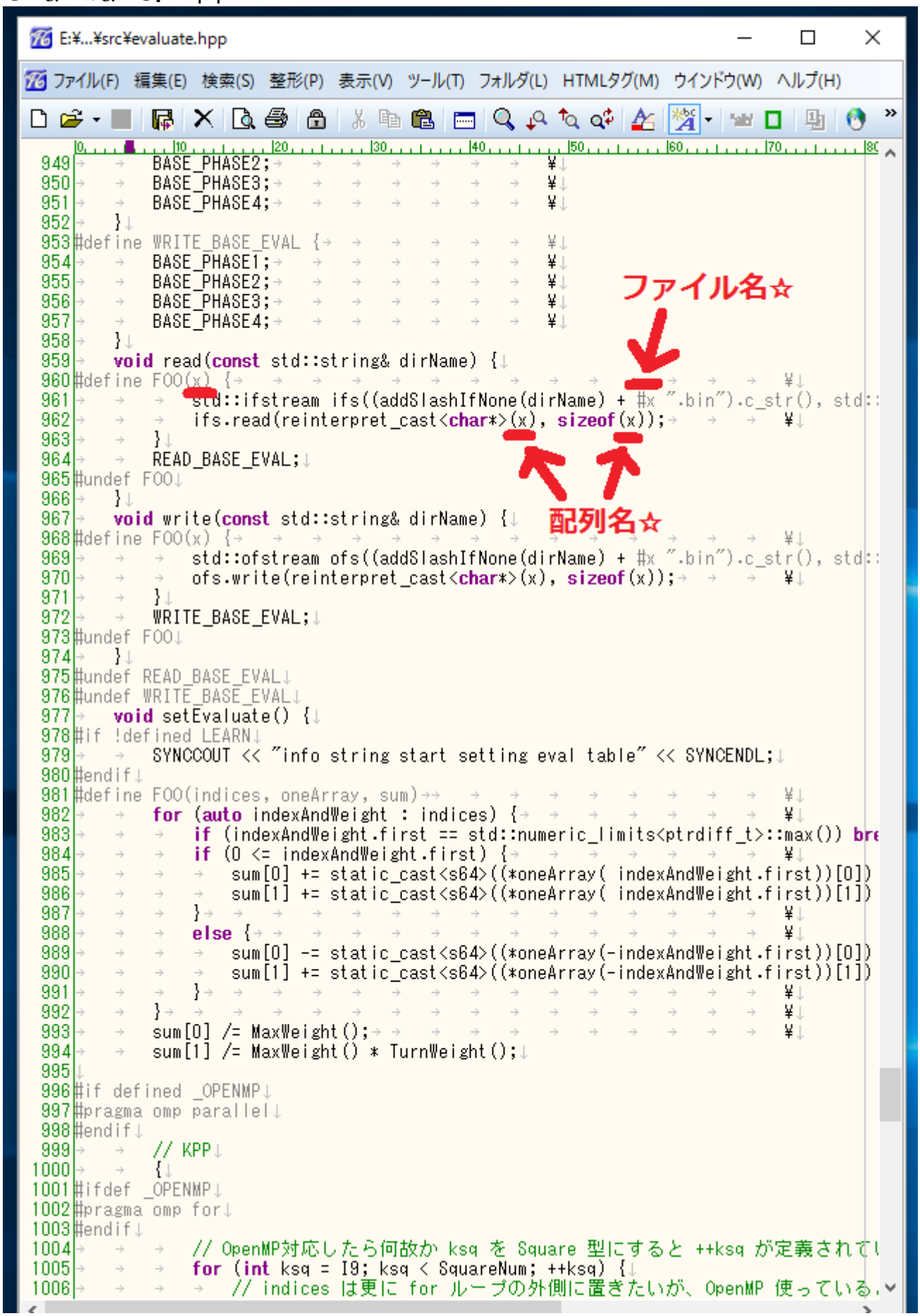
「ＦＯＯ　というのがマクロで、丸かっこ　の中に書いてある
　ｋｐｐｓ．ｋｅｅ　といった文字は、
　ファイル名にもなるし、変数名にもなるんだぜ☆」

「フー　って何だぜ☆？」

「意味のない単語なんじゃないの？　フー、バー、フガ、ホゲ」

「わたしは　これを見て　平岡さんは名前を考えるのもめんどくさくなったのか、
　と、ずっと　思っているぜ☆
　マクロの内容を見ておこうぜ☆」

evaluate.hpp



「 マクロの中では、引数の頭に＃を付けると　変数名ではなく
　文字列になるんだぜ☆
　Ｃ＃言語使いの　わたしには　最初、プログラムを読み解く直観が
　Ｃ＋＋言語に対して　まったく働かなかったんだが、今は　働くぜ☆」

「 ＃ｄｅｆｉｎｅ　と　＃ＵＮＤＥＦ　の連打ね」

「 やってることは、キータイピング・ミスをせずに
　ｋｐｐｓ．ｋｐｐ．ｂｉｎ　ファイルの内容を　ｋｐｐｓ．ｋｐｐ　配列に
　きちんと入れたい、とか　そういうことだぜ☆
　確かに　ｋｋｐｓ．ｋｐｐ　と　ｋｐｐｓ．ｋｐｐ　の区別とか
　やりたくないぜ☆」

「 つまり　お父んは　魔改造して　キータイピング・ミスをしたわけだな☆」

「 まさか　ファイル名と　配列名が　連動しているとは
　思いもしなかったんだぜ☆！
　そして　なぜ　連動させたのか、
　ああ、多分　キータイピング・ミスに苦しんだのだろう☆
　そんなことを　自分の言葉として　語れるのは
　魔界に足を踏み入れた人間だけだぜ☆」

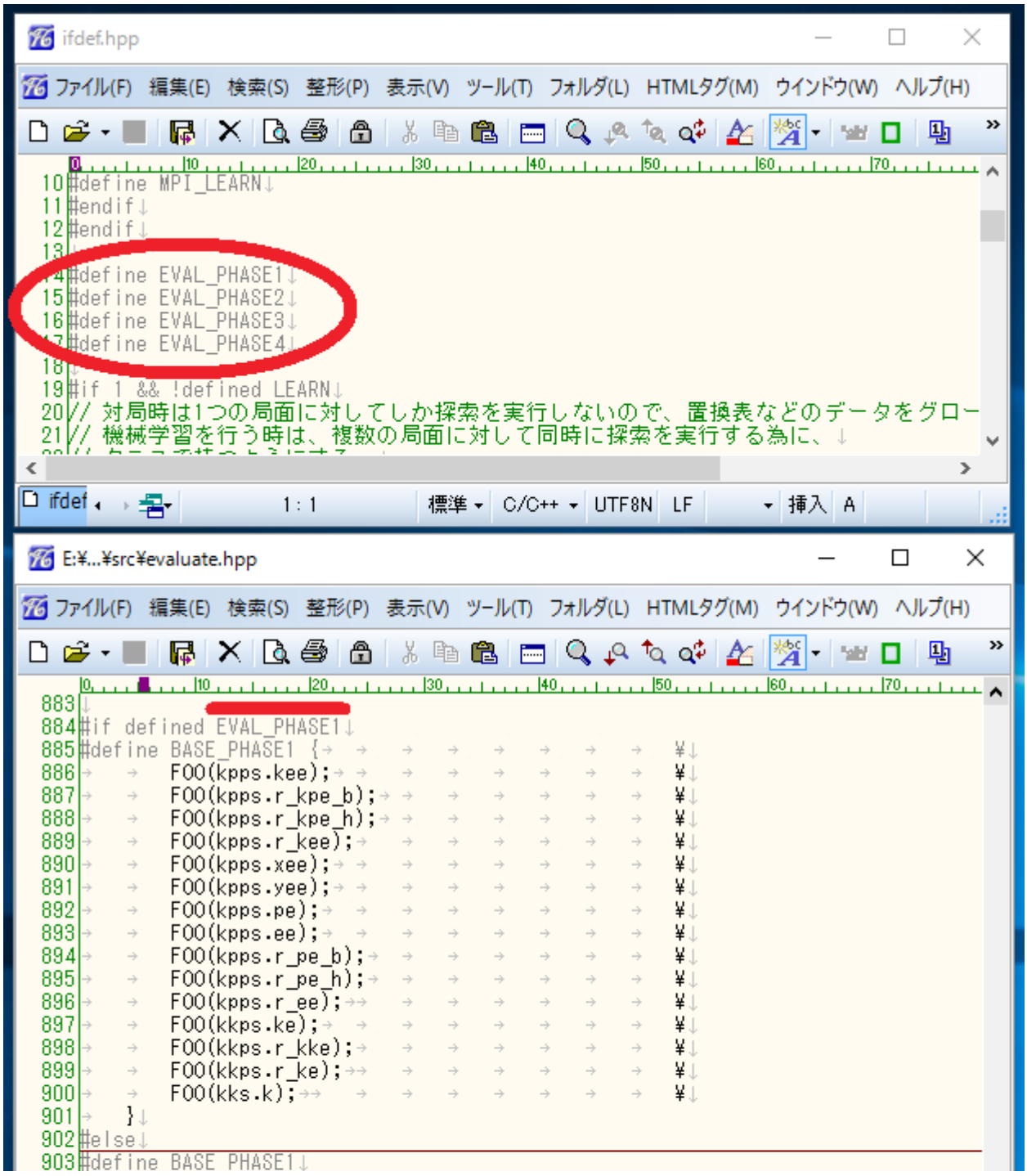
シンセサイズド☆！

SYNTHESIZED

学習の成果を、評価値ファイルに統合しようぜ☆！

「で、　～．ｂｉｎ　ファイルなんだが、
使うかどうかは　ｉｆｄｅｆ．ｈｐｐ　のマクロで
４グループに分けられていて、　切り替えできるみたいだぜ☆」

ｉｆｄｅｆ．ｈｐｐ　と、　ｅｖａｌｕａｔｅ．ｈｐｐ



```
904 #endif↓
905 ↓
906 #if defined EVAL_PHASE2↓
907 #define BASE_PHASE2 {→  →   →   →   →   →   →   →   ¥↓
908 →   →   FOO(kpps.r_pp_bb);→ →   →   →   →   →   →   ¥↓
909 →   →   FOO(kpps.r_pp_hb);→ →   →   →   →   →   →   ¥↓
910 →   →   FOO(kkps.r_kp_b);→  →   →   →   →   →   →   ¥↓
911 →   →   FOO(kkps.r_kp_h);→  →   →   →   →   →   →   ¥↓
912 →   →   FOO(kks.r_kk);→ →   →   →   →   →   →   →   ¥↓
913 →   }↓
914 #else↓
915 #define BASE_PHASE2↓
916 #endif↓
917 ↓
918 #if defined EVAL_PHASE3↓
919 #define BASE_PHASE3 {→  →   →   →   →   →   →   →   ¥
```

```
917 ↓
918 #if defined EVAL_PHASE3↓
919 #define BASE_PHASE3 {→  →   →   →   →   →   →   →   ¥↓
920 →   →   FOO(kpps.r_kpp_bb);→→  →   →   →   →   →   ¥↓
921 →   →   FOO(kpps.r_kpp_hb);→→  →   →   →   →   →   ¥↓
922 →   →   FOO(kpps.pp);→   →   →   →   →   →   →   ¥↓
923 →   →   FOO(kpps.kpe);→ →   →   →   →   →   →   ¥↓
924 →   →   FOO(kpps.xpe);→ →   →   →   →   →   →   ¥↓
925 →   →   FOO(kpps.ype);→ →   →   →   →   →   →   ¥↓
926 →   →   FOO(kkps.kp);→   →   →   →   →   →   →   ¥↓
927 →   →   FOO(kkps.r_kkp_b);→ →   →   →   →   →   ¥↓
928 →   →   FOO(kkps.r_kkp_h);→ →   →   →   →   →   ¥↓
929 →   →   FOO(kkps.kke);→ →   →   →   →   →   →   ¥↓
930 →   →   FOO(kks.kk);→   →   →   →   →   →   →   ¥↓
931 →   }↓
932 #else↓
933 #define BASE_PHASE3↓
934 #endif↓
935 ↓
936 #if defined EVAL_PHASE4↓
937 #define BASE_PHASE4 {→  →   →   →   →   →   →   →   ¥↓
938 →   →   FOO(kpps.kpp);→ →   →   →   →   →   →   ¥↓
939 →   →   FOO(kpps.xpp);→ →   →   →   →   →   →   ¥↓
940 →   →   FOO(kpps.ypp);→ →   →   →   →   →   →   ¥↓
941 →   →   FOO(kkps.kkp);→ →   →   →   →   →   →   ¥↓
942 →   }↓
943 #else↓
944 #define BASE_PHASE4↓
945 #endif↓
```

「こうやって見ると、Ａｐｅｒｙは
フェーズ１　で　利き　を　重視　していて、
フェーズ２　で　二駒関係　を　見ていて、
フェーズ３　で　三駒関係　を　見ていて、
フェーズ４　で　玉と、利きを除いた二駒の関係　を　鏡像で　見ている、
と、だいたい　どこまでが鉄板で　どこからが半信半疑と思って
創造神ＨＩＲＡＯＫＡが　評価を信頼して　積み上げてきたのか　想像できるな☆
まあ結局　全フェーズ　使うのが一番いいと思うんだが☆」

「エイプリー＿大樹の枝　のソースコードを読むまで、
お父んの中では　平岡さんは　ヨーグルトだけ　食べている人だったからな☆
がちがちのプログラマーと認識し始めたのも　このへんから　だっただろう☆」

「 コンピュータ将棋を始めるまで　自分が一番　プログラムができるとっていたので……☆」

「 今は何番なのよ？」

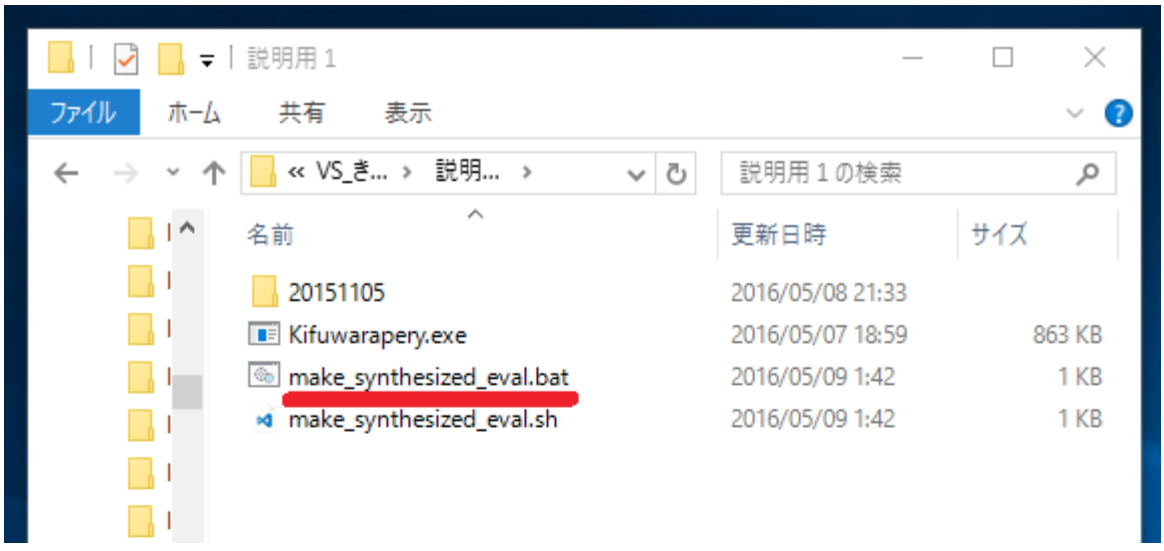
「 今は　埼玉県新座市で一番だと思っているぜ☆」

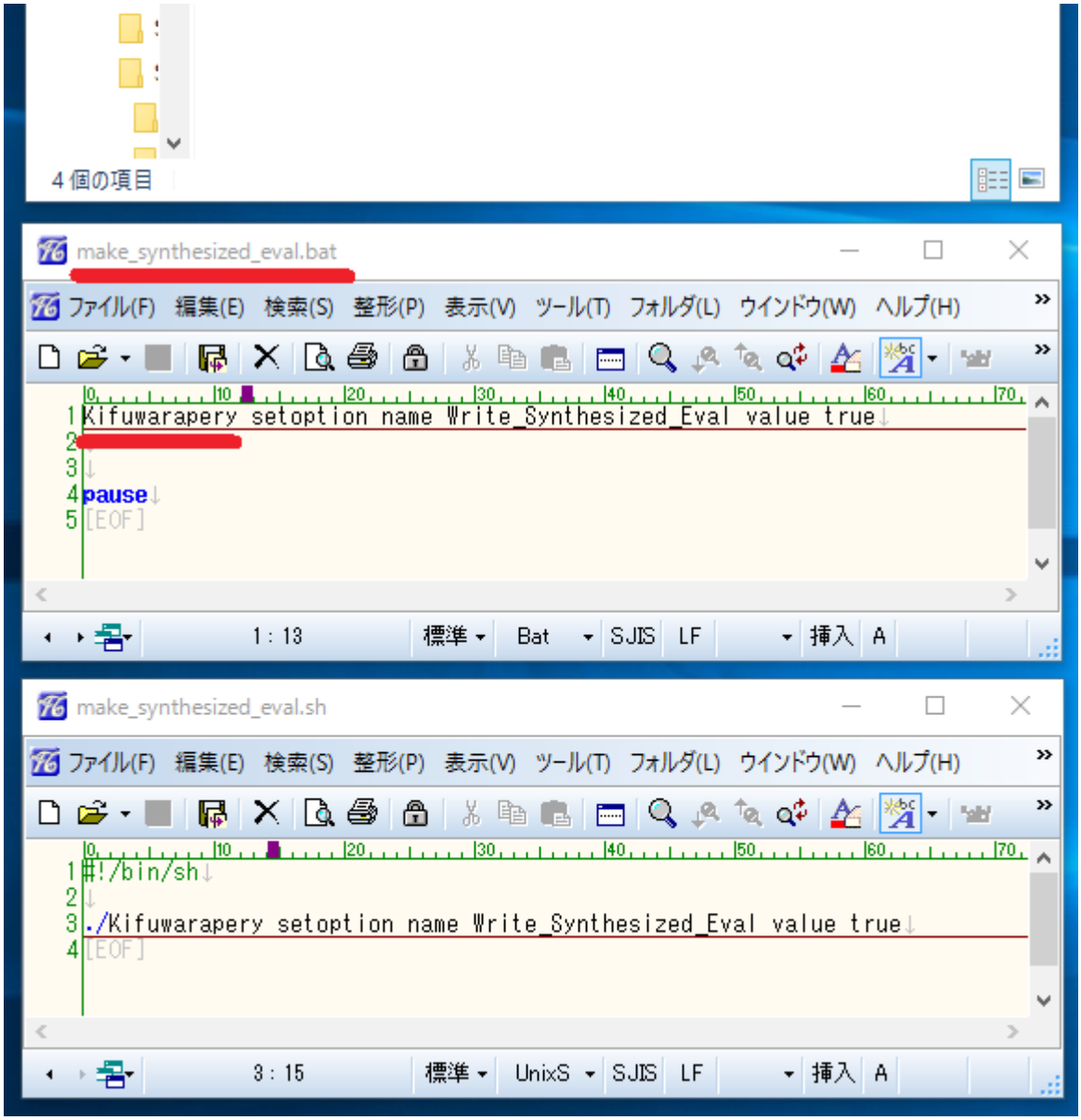
「 ｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗ」

「 で、　．ｂｉｎ　ファイルなんだが、
　これは　対局中に使っていないんだぜ☆　対局には要らない☆」

「 ．ｂｉｎ　ファイルって　なんに使ってるのよ！」

ｍａｋｅ＿ｓｙｎｔｈｅｓｉｚｅｄ＿ｅｖａｌ．ｂａｔ





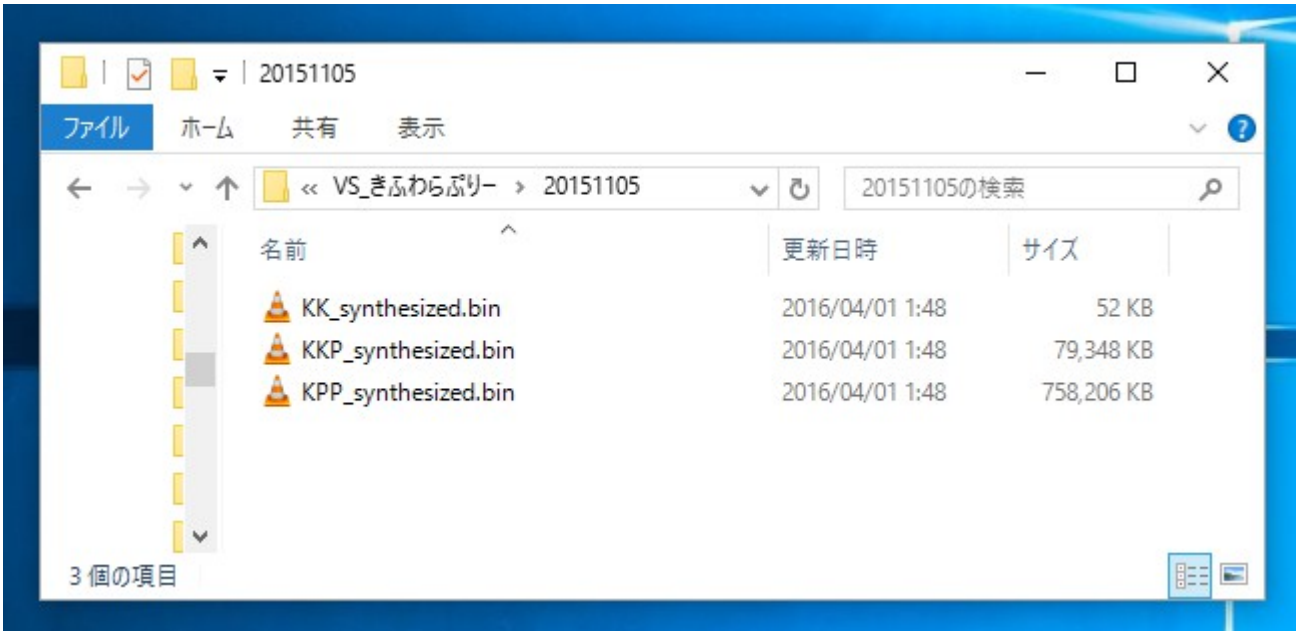
「～．ｂａｔ　が　　Ｗｉｎｄｏｗｓ用で、
～．ｓｈ　は多分　Ｌｉｎｕｘ　用なのだろう☆
やってることは　ａｐｅｒｙ．ｅｘｅを、
設定　Ｗｒｉｔｅ＿Ｓｙｎｔｈｅｓｉｚｅｄ＿Ｅｖａｌ　＝　ｔｒｕｅ
にして実行しているだけだぜ☆」

「．ｂａｔ　というのは、バットで　ぶんなぐるのかだぜ☆？」

「Ｗｉｎｄｏｗｓのファイル操作を　プログラムしておいて　実行できるのよ。
．ｅｘｅ　に注文を付けて　実行できるのが便利ね」

「わたしは　テキストエディタで開いて　Ｋｉｆｕｗａｒａｐｒｅｙ用に
編集したぜ☆
ｍａｋｅ＿ｓｙｎｔｈｅｓｉｚｅｄ＿ｅｖａｌ．ｂａｔ　を
ダブルクリックして　数分　待った結果を　紹介するぜ☆」

20151105　フォルダー☆



「ほんとは　ｋｋｐｓ．ｋｅ．ｂｉｎ　ファイルとかも
フォルダーに残ったままなんだが、対局には使わないので　削除　してあるぜ☆」

「～．ｂｉｎ　ファイルは　３つの　．ｂｉｎ　ファイルに統合されたんだぜ☆」

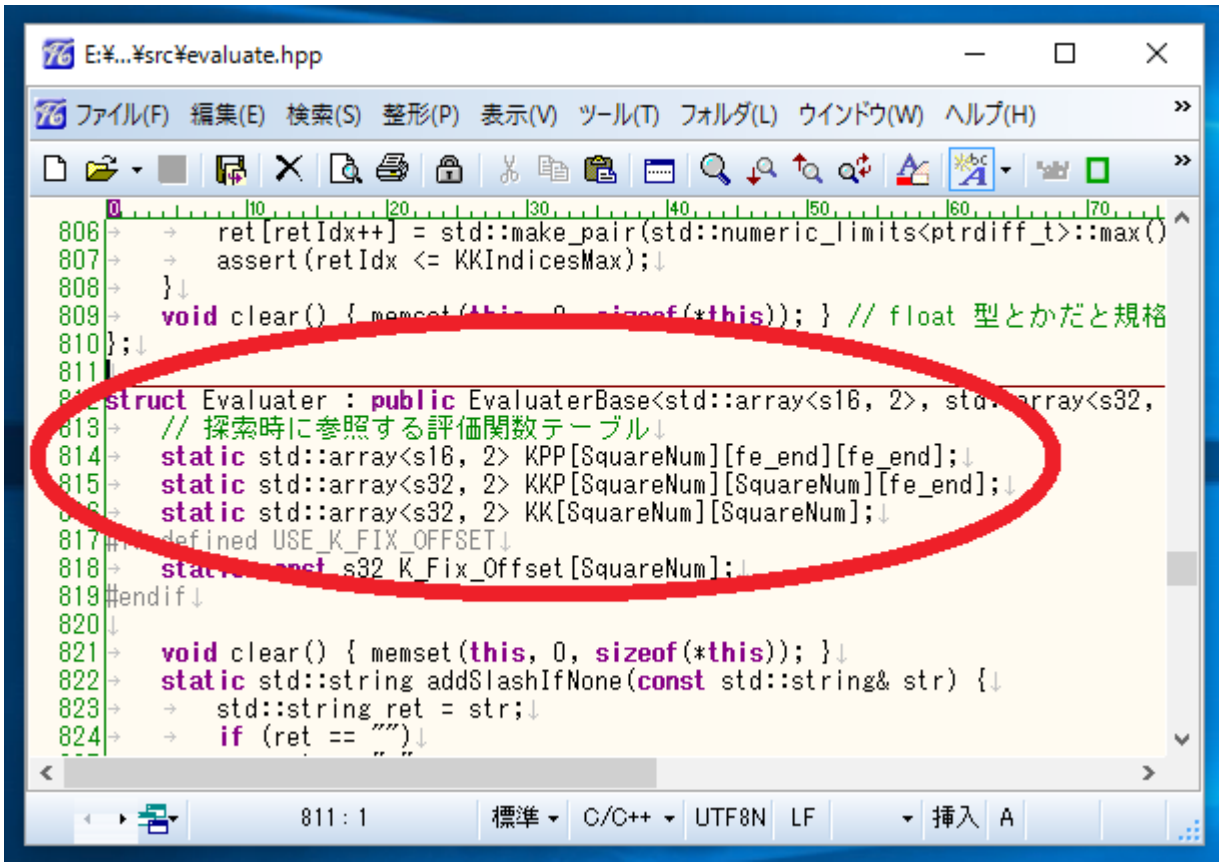
「これが　エイプリー＿大樹の枝　の評価値ファイルだぜ☆」

「Ｂｏｎａｎｚａ６．０の　ｆｖ．ｂｉｎ　や、
昔のＡｐｅｒｙの　ｆｖ３８．ｂｉｎ　じゃないのね」

「Ｂｏｎａｎｚａ６．０の　ｆｖ．ｂｉｎ　は、
駒割りの点数がまず最初に並んでいて、
そのあと、ＫＫ、ＫＫＰ、ＫＰＰ　の点数が
組み合わせ表になっていて　重複が出ないように　表の半分だけを
１つのファイルに　並べて　詰め込んでいたんだが、

エイプリー＿大樹の枝　の評価値ファイルは
ＫＫ配列、ＫＫＰ配列、ＫＰＰ配列　の３つに分割したみたいだぜ☆
そして　組み合わせ表の重複とか気にせず
表全体を　どかんっ☆　と　入れている感じ☆」

ｅｖａｌｕａｔｅ．ｈｐｐ



「このＫＰＰ配列と、ＫＰＰ＿ｓｙｎｔｈｅｓｉｚｅｄ．ｂｉｎ　ファイルと
いったように、　ファイル名と　配列名が　連動していると思うんだぜ☆」

評価値の謎☆

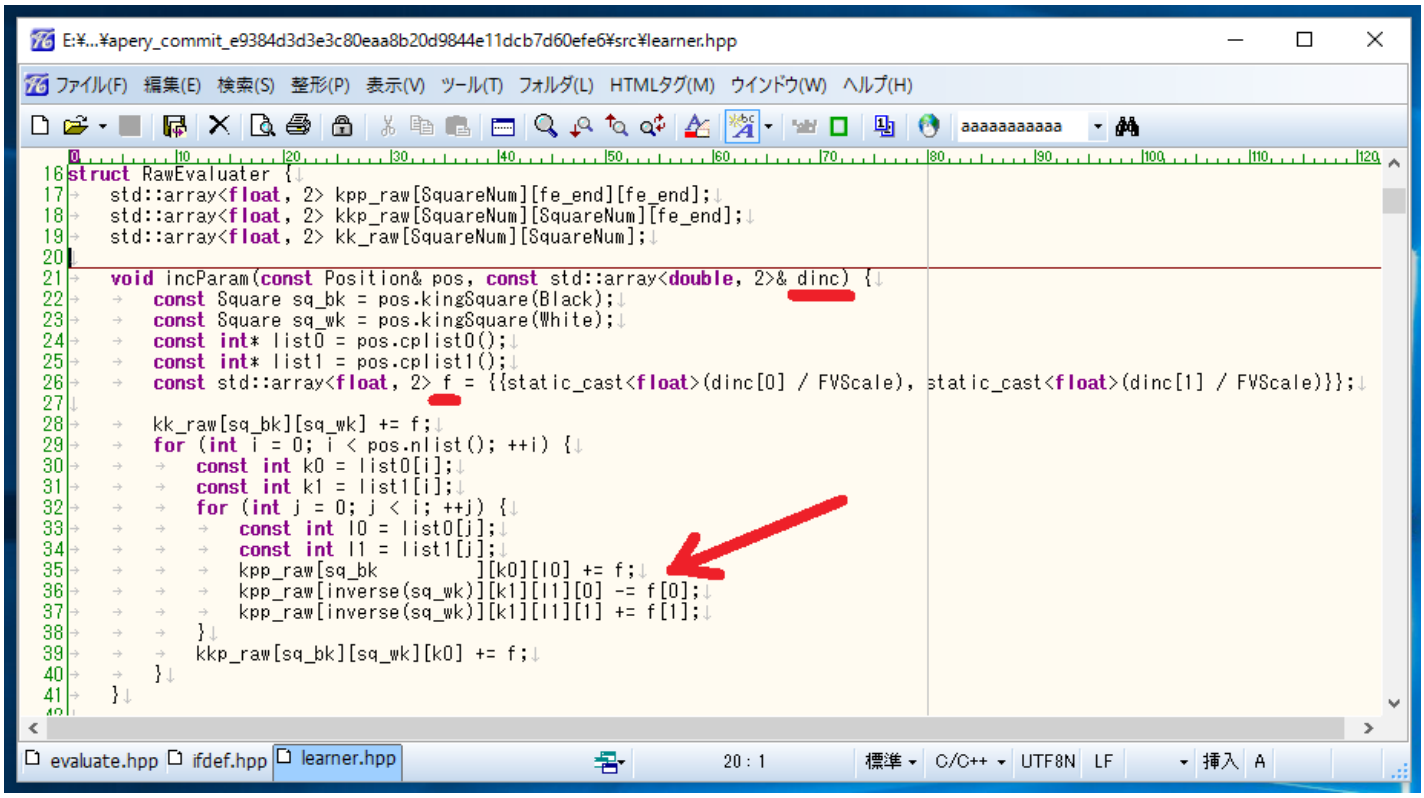
三駒関係のパラメーターは　要素［２］の配列☆？

YOUSO　NI

「ＫＰＰ配列の要素は　１６ビット値　２つなの？
ＫＫＰ配列と　ＫＫ配列の要素は　３２ビット値　２つなの？」

「そうなんだぜ☆
評価値は　なんか　１つの項目に　２つ　入っているんだぜ☆」

learner.hpp



「後手のグッド値を　マイナスで表示するのか、
先手・後手関わらず　自分のグッド値を　プラスにするのか、
ぐらいの違いかと思っていたんだが、
ソースコードを読んでいる内に　だんだん　わからなくなってきたので、
この判断は　保留だぜ☆」

「ダメだな、お父んは」

evaluate.hpp

```
977    void setEvaluate() {
978  #if !defined LEARN
979      SYNCCOUT << "info string start setting eval table" << SYNCENDL;
980  #endif
981  #define FOO(indices, oneArray, sum)                                    \
982      for (auto indexAndWeight : indices) {                              \
983        if (indexAndWeight.first == std::numeric_limits<ptrdiff_t>::max()) break; \
984        if (0 <= indexAndWeight.first) {                                 \
985          sum[0] += static_cast<s64>((*oneArray[ indexAndWeight.first])[0]) * indexAndWeight.second; \
986          sum[1] += static_cast<s64>((*oneArray[ indexAndWeight.first])[1]) * indexAndWeight.second; \
987        }                                                                \
988        else {                                                           \
989          sum[0] -= static_cast<s64>((*oneArray[-indexAndWeight.first])[0]) * indexAndWeight.second; \
990          sum[1] += static_cast<s64>((*oneArray[-indexAndWeight.first])[1]) * indexAndWeight.second; \
991        }                                                                \
992      }                                                                  \
993      sum[0] /= MaxWeight();                                             \
994      sum[1] /= MaxWeight() * TurnWeight();
995
```

「ペアの［0］の方は　何かと　符号が反転していて、
ペアの［1］の方は　重み８倍で　割られているんだぜ☆
何をやっているんだろうな☆？」

「これも　分かっている途中　なの？」

「そうだぜ☆」

evaluate.hpp

```
139
140 template <typename Tl, typename Tr>
141 inline std::array<Tl, 2> operator += (std::array<Tl, 2>& lhs, const std::array<Tr, 2>& rhs) {
142     lhs[0] += rhs[0];
143     lhs[1] += rhs[1];
144     return lhs;
145 }
146 template <typename Tl, typename Tr>
147 inline std::array<Tl, 2> operator -= (std::array<Tl, 2>& lhs, const std::array<Tr, 2>& rhs) {
```

```
148    lhs[0] -= rhs[0];
149    lhs[1] -= rhs[1];
150    return lhs;
151}
152
153template <typename KPPType, typename KKPType, typename KKType> struct EvaluaterBase {
154    static const int R_Mid = 8; // 相対位置の中心のindex
155    constexpr int MaxWeight() const { return 1 << 22; } // KPE自体が1/32の寄与。更にKPEの遠隔駒の
156                                                        // 更に重みを下げる場合、MaxWeightを更に
157                                                        // なぜか clang で static const int MaxWe
158    constexpr int TurnWeight() const { return 8; }
159    // 冗長に配列を確保しているが、対称な関係にある時は常に若いindexの方にアクセスすることにする.
160    // 例えば kpp だったら、k が優先的に小さくなるようする。左右の対称も含めてアクセス位置を決め
161    // ただし、kkp に関する項目 (kkp, r_kkp_b, r_kkp_h) のみ、p は味方の駒として扱うので、k0 < k
162    struct KPPElements {
```

evaluate.hpp　142 : 5　標準　C/C++　UTF8N　LF　挿入　A

「 演算子のオーバーロードを見ても、この２要素は　ずっと別個に
　加減算を行われているぜ☆」

「 平岡さんに　聞いたらいいんじゃないの？」

「 まだ分かっている途中なので……☆
　わたしの中では　まだ　分かっていない と確定していないので……☆」

「 これは負けられない勝負なのかなんだぜ☆？」

「 こんなところで質問していては、残りの全部　質問だろ☆」

きふわらぷりーと、Ａｐｅｒｙの　評価値は一緒☆

HYOKACHI

評価値はいじってないぜ☆

「で、第２６回世界コンピュータ将棋選手権に出場した　きふわらべ　と、
新ＣＳＡライブラリの　エイプリー＿大樹の枝　の評価値は　一緒だぜ☆
いじってないからな☆」

「長い３行だったんだぜ☆」

「じゃあ　どこを　いじったのよ？」

「レーティングを下げたところの解説をしても
解散、かいさーん　だと思ったから
いじってないところの　ガクジュツテキ　に有用な解説を　したんだが……☆」

「エイプリー＿大樹の枝　と　どれだけ変わったのか書かなきゃ、
きふわらべちゃんの　ＰＲ文書　にならないじゃないの」

「そうだぜ☆？
エイプリー＿大樹の枝　のＰＲ文書になってしまうぜ☆」

「主にいじったのは、指し手生成と、時間管理　だぜ☆
あとは　微妙☆」

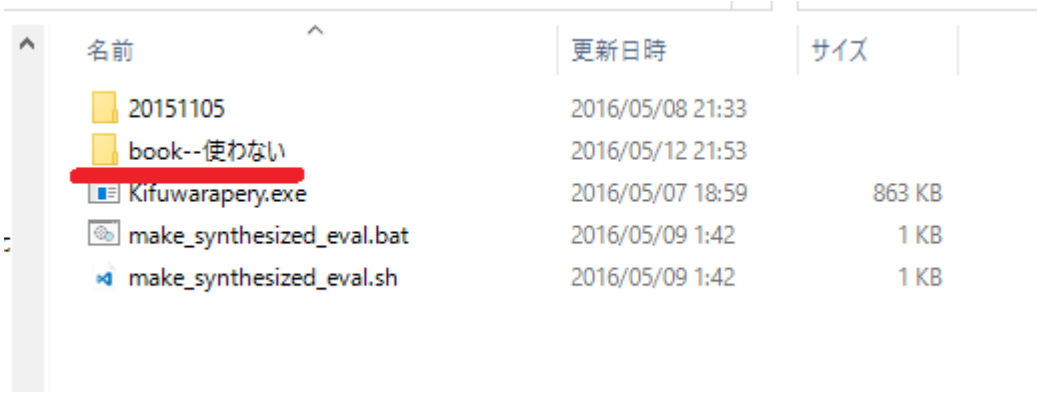
定跡☆

JOSEKI

定跡は使わなかったぜ☆

「ｗｃｓｃ２６　（第２６回世界コンピュータ将棋選手権）　では、
きふわらべ　は　定跡を　使っていないぜ☆」

定跡を使わないようにする方法の例



「ｂｏｏｋファイルの名前を変えておけば、使われなくなるぜ☆」

「なんで定跡を使わないわけ？
定跡を使う相手と当たったとき、
きふわらべちゃんが　時間を使いすぎて　損してるじゃない」

「そうだぜ☆？」

「初手から考えさせて　観戦を　楽しもうと思ったんだぜ☆」

指し手生成をいじろうぜ☆

YOCHI

どこに　いじる余地が　あったのか☆？

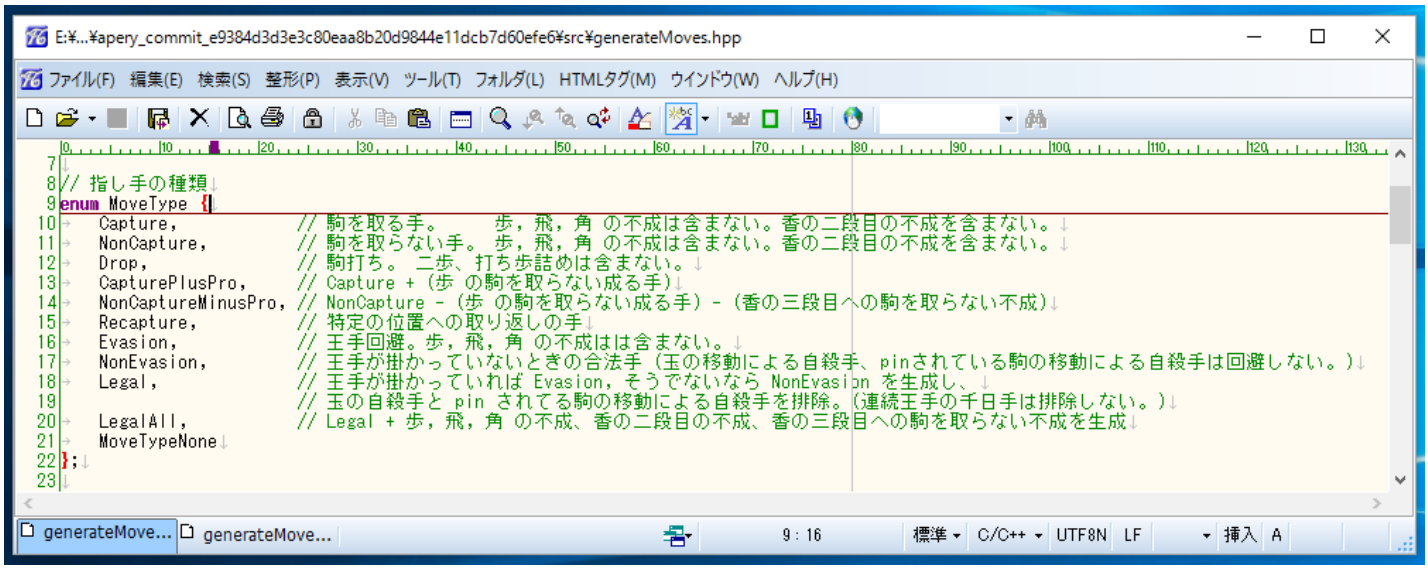
「まず、指し手生成とは何かなんだが、
指せる手　を一覧することだぜ☆
歩は　１つ前へ進むとか、
先手から見て　桂馬は　１段目、２段目には　打ち込めないけど、
３段目から下は打ち込めるとかだぜ☆」

「そうなのね」

「で、駒を取る手だけ　調べたいとか、
駒を取ったばかりの駒を、取り返される手だけ　調べたいとか、
優先して　調べたい　指し手　があるだろ☆」

「いろんなパターンがありそうね」

generateMoves.hpp



「およそ　１１パターン　あるんだぜ☆
そして、必ずしも　将棋で　指せる手　を全部読んでいるわけじゃないんだぜ☆

エイプリー＿大樹の枝　が　指し手生成を　省くことがある　合法手には、
次のようなものがあるぜ☆」

（１）歩の　不成
（２）飛の　不成
（３）角の　不成

「さらに、次の１つも同じ理由だぜ☆　利きの数が損なので　見ることもないだろ☆」

（４）香の　二段目の不成

「あと、駒を取る手を　キャプチャー、　駒を取らない手を　ノン・キャプチャー　と呼んでいるようで、
指し手は　どっちかになるかと思うんだぜ☆」

ムーブタイプ

キャプチャー	ノン・キャプチャー

「そうね」

「さらに　踏み込んで、
次のような分け方も　使われているぜ☆」

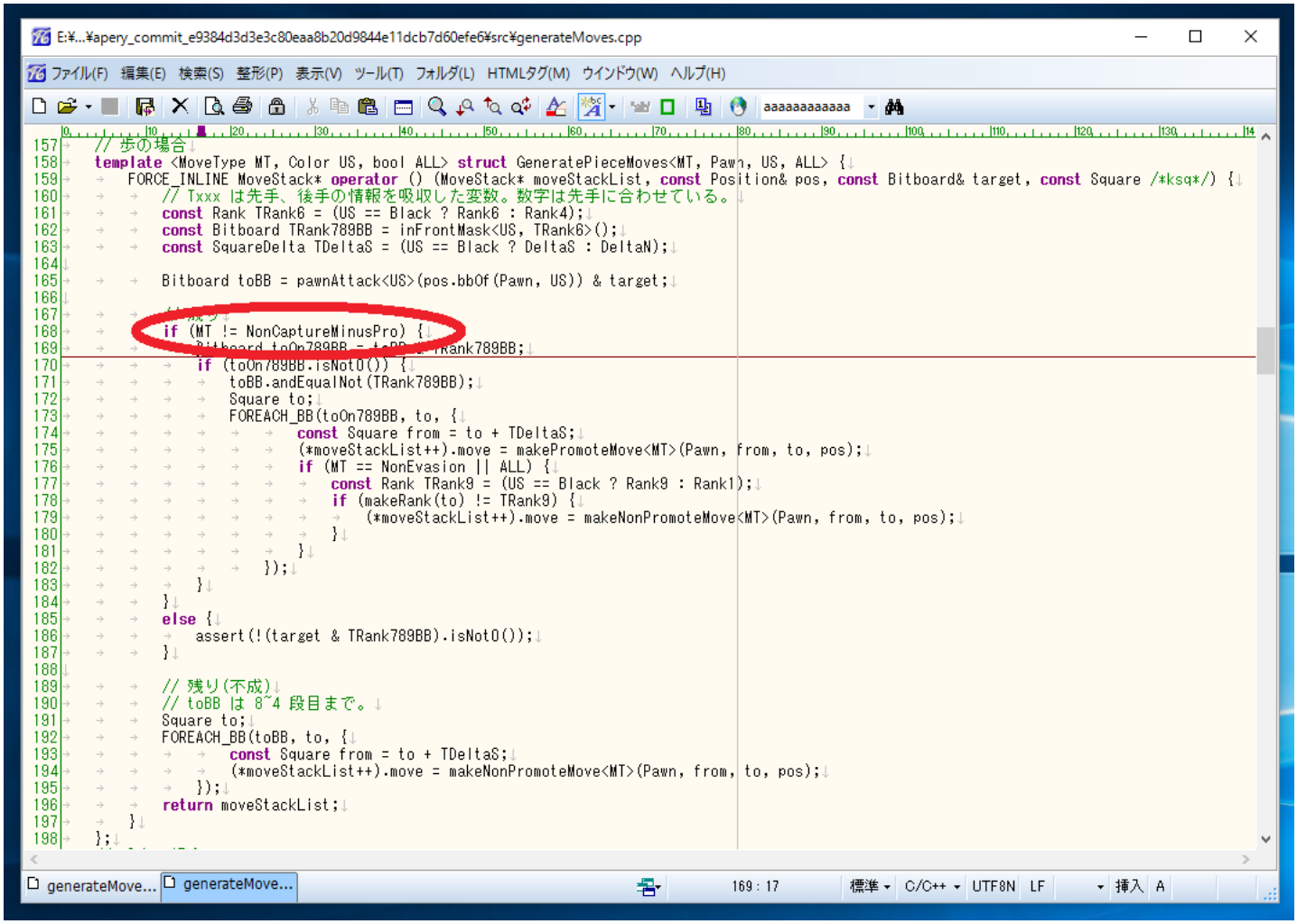
ムーブタイプ

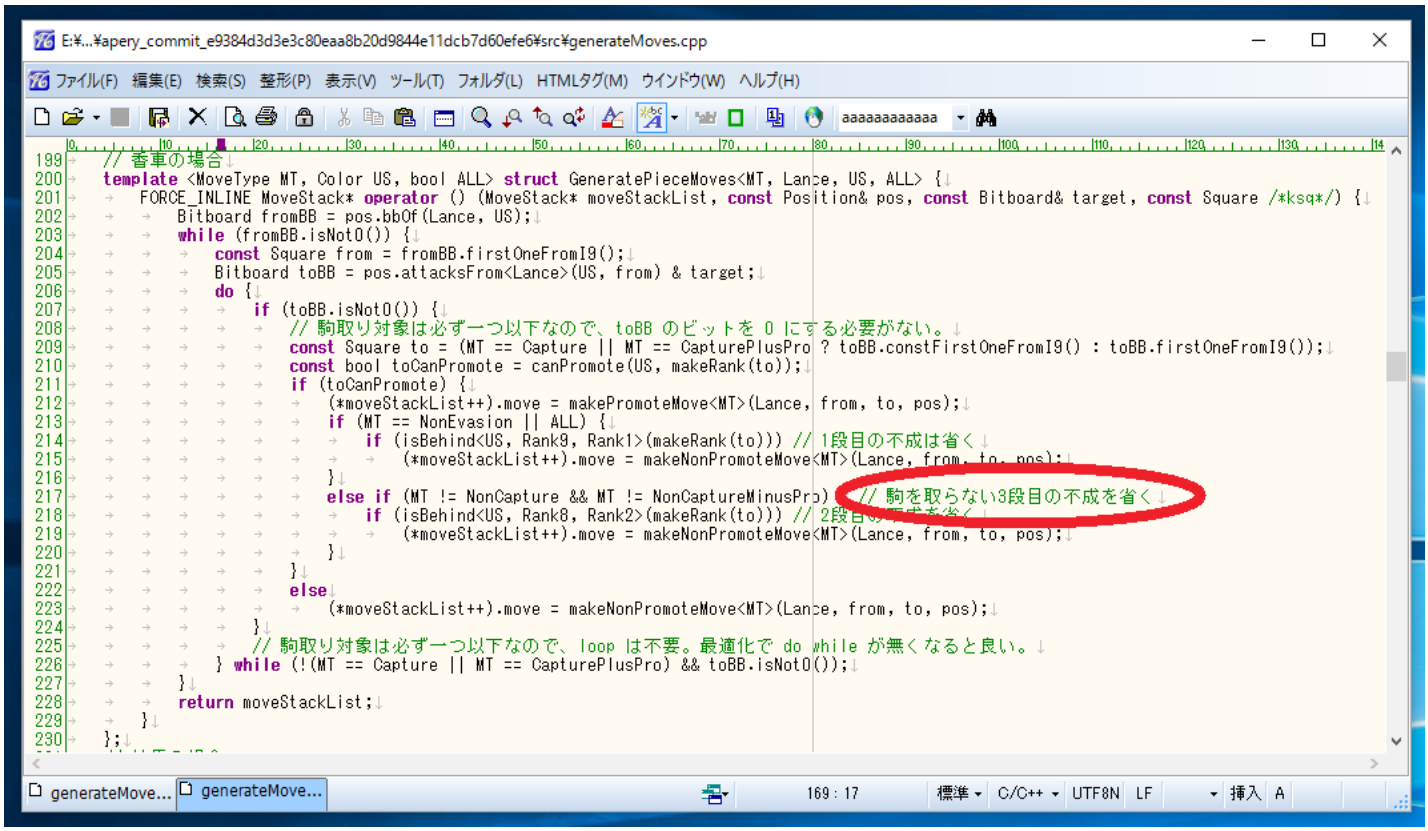
キャプチャー	ノン・キャプチャー
＋　ノン・キャプチャー歩成	－　ノン・キャプチャー歩成
	－　3段目ノン・キャプチャー香成

「なんなの、この　わけわかんない　分け方？
駒を取らなくても　成る歩は　キャプチャーで、
駒を取っていないのに、成る歩は　ノン・キャプチャーに入れてくれないの？

駒を取っていないのに、3段目に成る香はノン・キャプチャーに入れてくれないの？
左右対称じゃないのも　気になるわね！」

generateMoves.hpp





「 なぜ　こうなっているのかは分からないが、
　何をやっているかが　分かれば　十分だぜ☆

　歩成りは、
　駒の取り合い、たたき合いに含まれるような、何か特別視されているんだぜ☆
　３役目の香成りも　駒を取っていないとは思われないような　何かあるのだろう☆」

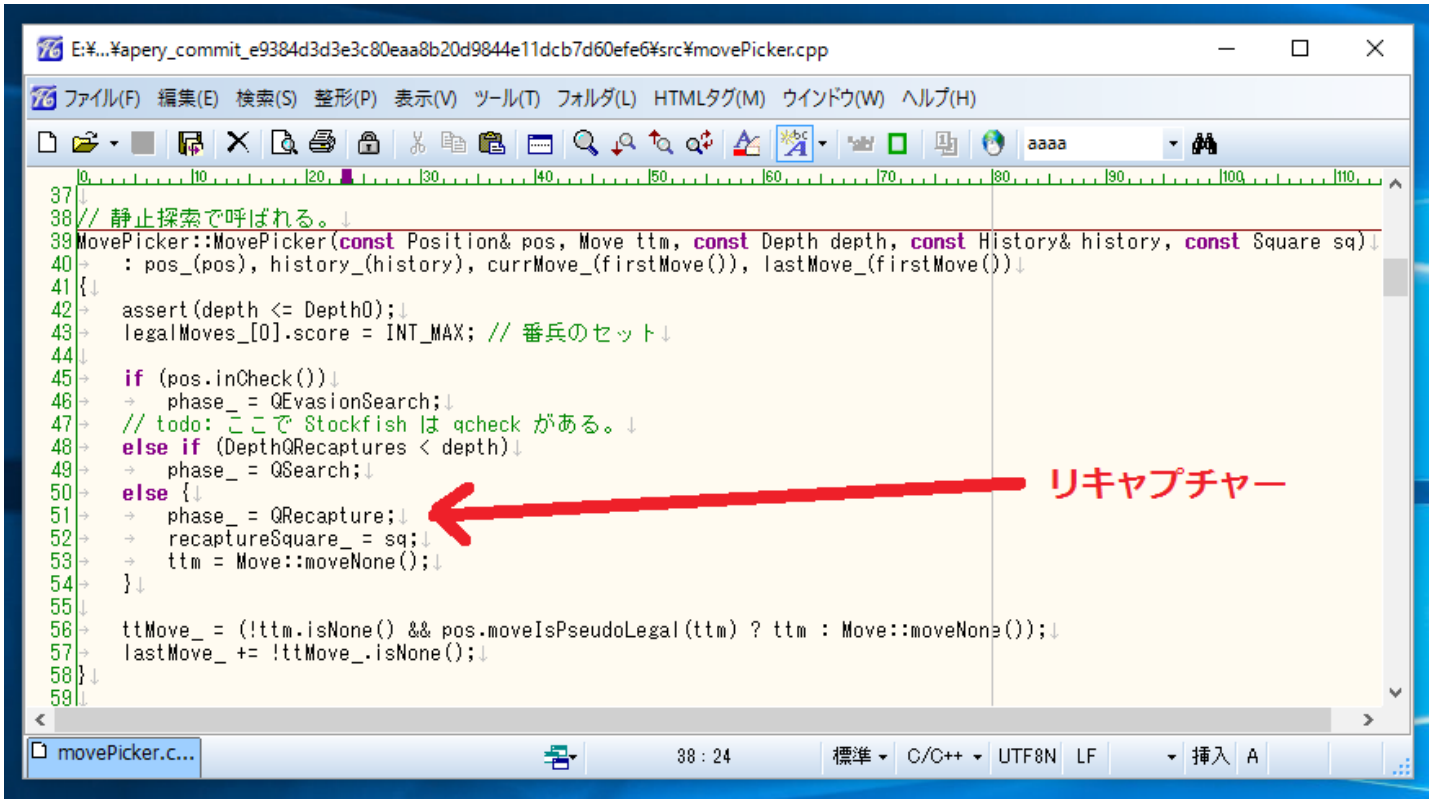
「 ほかにも、大きな分類があるぜ☆」
　駒を取った駒を取り返す、たたき合いを　リキャプチャー　と呼ぶと思うんだぜ☆」

ムーブタイプ

リキャプチャー	ノン・リキャプチャー

「 静止探索で　リキャプチャーの指し手生成を　見た気がするぜ☆」

movePicker.cpp



「 王手を回避する指し手だけを　調べたいのを、エヴァーション　と呼ぶようだぜ☆」

ムーブタイプ

エヴァーション	ノン・エヴァーション

「 エヴァーションは、静止探索で使われているのは確実だが、
それ以外でも　複数カ所で　使われているようだぜ☆

ノン・エヴァーションでは　玉や合い駒が移動することで自殺手に
なってしまうような　王手回避漏れ　があるように見えたんだが、
正確さがいるところと、そうでないところで
メリハリを付けているのかもしれないななんだぜ☆」

「 ほかには、
角不成りも含む　　全ての合法手、
角不成りなどを含まない　実戦的な合法手、
王が相手の利きに飛び込む自殺手も含む　指し手、という区別があるぜ☆」

ムーブタイプ

リーガル・オール	リーガル・ムーブ	スイード・リーガル

「 この指し手生成を　いじったの？」

「 ９９．９％　いじってないぜ☆」

「 ……☆」

「 ０．１％　は　どこを　いじったんだぜ☆？」

指し手生成も　ほとんど　いじっていないんだぜ☆

IJITTA

どこを　いじったのか☆？

「 指し手は、将棋の駒の種類によって　違うだろ☆」

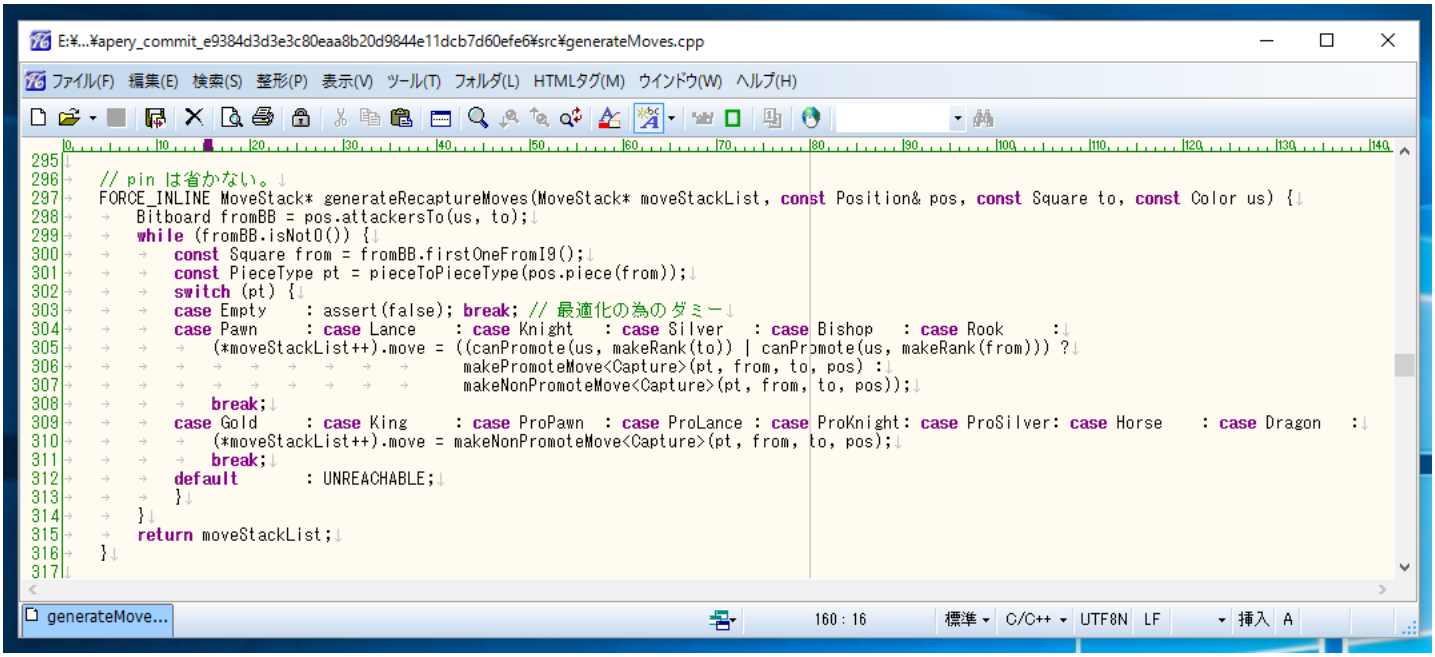
ピースタイプ

歩	香	桂	銀	角	飛	金	玉	と	杏	圭	全	馬	竜

「 そりゃそうね」

「 じゃあ、これらの　駒の種類が　たたき合うところを見るぜ☆」

generateMoves.hpp



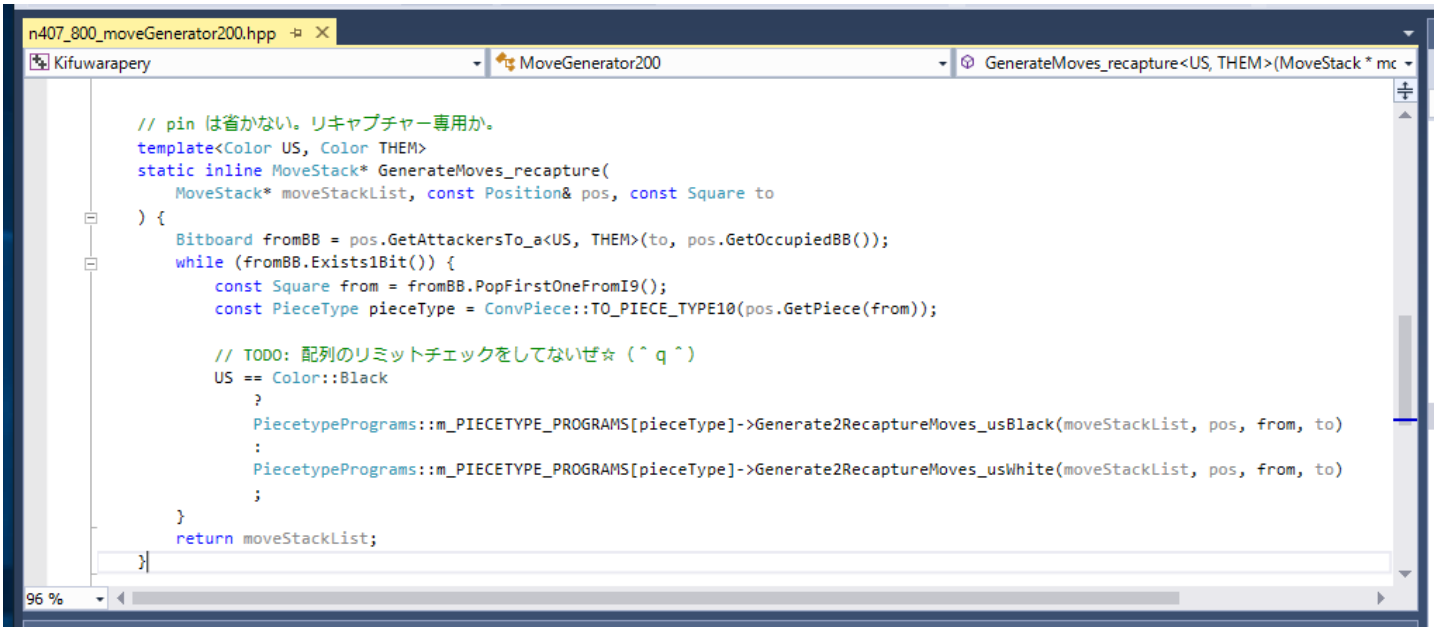
「 ｓｗｉｔｃｈ文があるんだが、
ｃａｓｅ　Ｅｍｐｔｙ：　とか、
ｃａｓｅ　Ｐａｗｎ：　とか書いてあるだろ☆
これを　わたしは　いじって……☆」

「ようやく　エイプリー＿大樹の枝と　きふわらぷりー　の違いがでてくるのね！」

「……☆」

「下図のように、ｓｗｉｔｃｈ文を　配列に　変えたんだぜ☆」

n407_800_moveGenerator200.hpp



「んむー☆」

「ここで　ポイントとなる場所は……、」

generateMoves.cpp

```
const PieceType pt = pieceToPieceType(pos.piece(from));
switch (pt) {
case Empty    : assert(false); break; // 最適化の為のダミー
case Pawn     : case Lance    : case Knight   : case Silver   : case Bishop   : case Ro
    (*moveStackList++).move = ((canPromote(us, makeRank(to)) | canPromote(us, makeRank(
                               makePromoteMove<Capture>(pt, from, to, pos) :
                               makeNonPromoteMove<Capture>(pt, from, to, pos));
    break;
case Gold     : case King     : case ProPawn  : case ProLance : case ProKnight: case Pr
    (*moveStackList++).move = makeNonPromoteMove<Capture>(pt, from, to, pos);
    break;
default       : UNREACHABLE;
}

turn moveStackList;
```

「第一引数　ｐｔ　だぜ☆！」

「……☆」

「……☆」

「ピーティー　というのは　ピースタイプのことで、
歩とか、香とか、馬　といった　駒の種類　の変数だぜ☆

エイプリー＿大樹の枝では　ピースタイプが　関数の引数　になっていて、
きふわらぷりー　　では　ピースタイプが　配列の添え字になっているぜ☆」

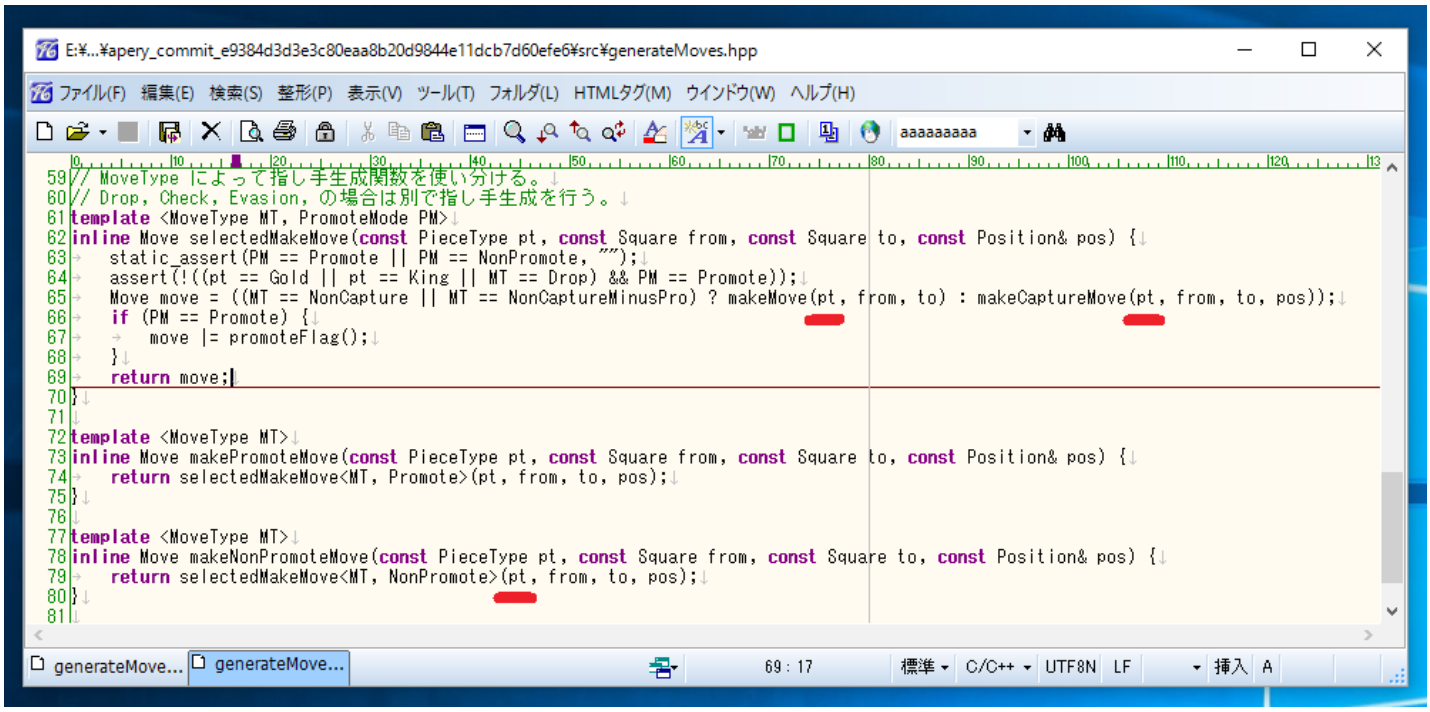
n407_800_moveGenerator200.hpp

```
// TODO: 配列のリミットチェックをしてないぜ☆（＾ｑ＾）
US == Color::Black
    ?
    PiecetypePrograms::m_PIECETYPE_PROGRAMS[pieceType]->Generate2RecaptureMoves_usBlack(moveStackList, pos, from, to)
    :
    PiecetypePrograms::m_PIECETYPE_PROGRAMS[pieceType]->Generate2RecaptureMoves_usWhite(moveStackList, pos, from, to)
    ;

eturn moveStackList;
```

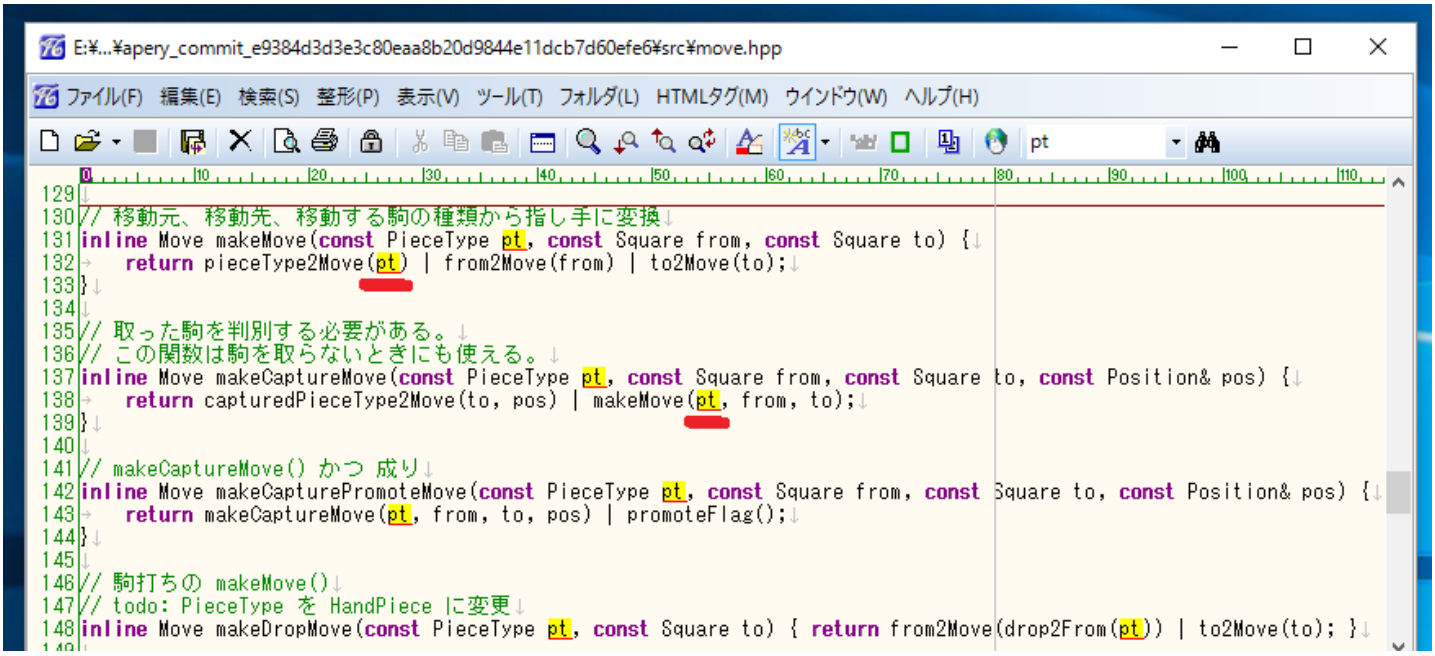
「 どういう利点の主張があるのかしら？」

「 エイプリー＿大樹の枝では、この　ピースタイプ　を引数として
さらに関数を呼び出して」

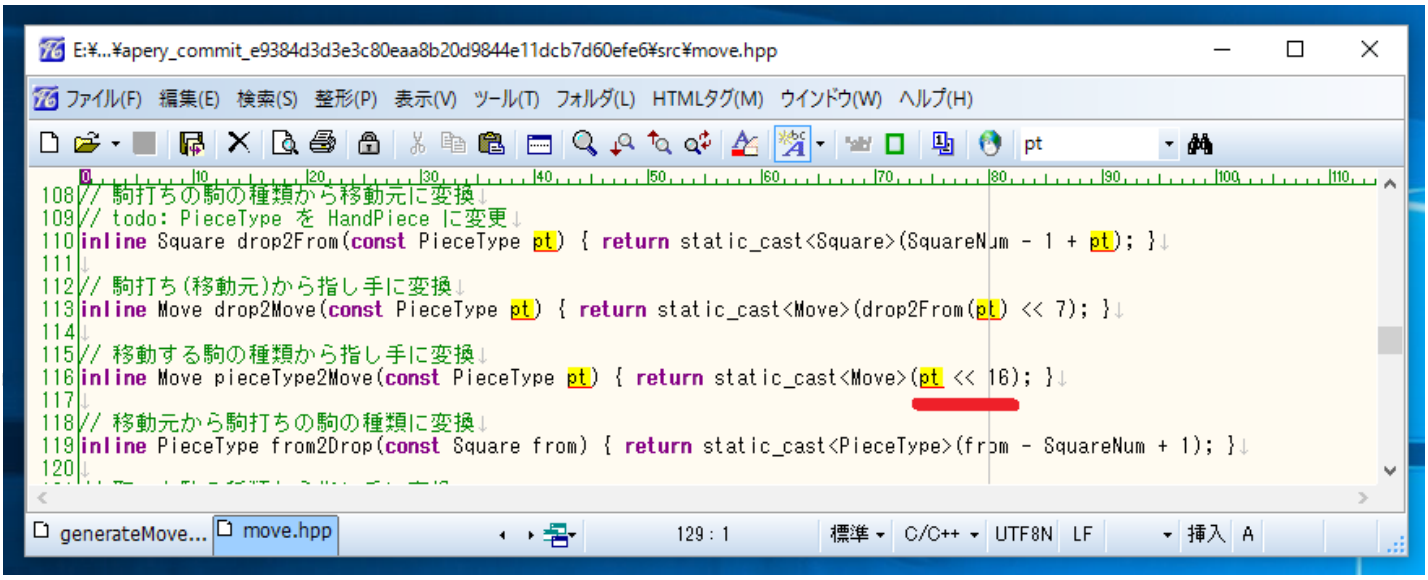
generateMoves.hpp



move.hpp



「この　ピースタイプ　という引数は　どんどん　奥の方に渡されて……☆」



「最終的に、　１６ビット　左シフト　されているんだぜ☆」

「？　ピースタイプを　１６ビット　左へ　ずらしたの？」
　何が行われたの？」

「１秒　で理解できる方法があるぜ☆」

generateMoves.cpp

```
switch (pt) {
case Empty   : assert(false); break; // 最適化の為のダミー
case Pawn    : case Lance   : case Knight  : case Silver  : case Bishop  : case Rook    :
  (*moveStackList++).move = ((canPromote(us, makeRank(to)) | canPromote(us, makeRank(from))) ?
                             makePromoteMove<Capture>(pt, from, to, pos) :
                             makeNonPromoteMove<Capture>(pt, from, to, pos));
  break;
case Gold    : case King    : case ProPawn  : case ProLance : case ProKnight: case ProSilver: case
  (*moveStackList++).move = makeNonPromoteMove<Capture>(pt, from, to, pos);
default      : UNREACHABLE;
```

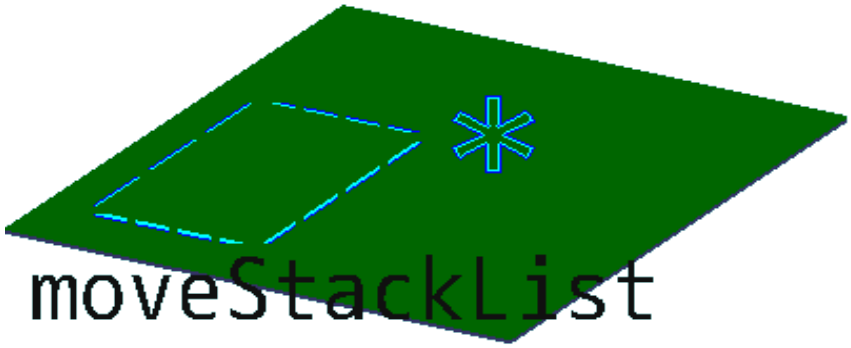
「 左にｍｏｖｅＳｔａｃｋＬｉｓｔ、　右にｍａｋｅなんとかＭｏｖｅ、
　というのは　指し手生成で　山ほど　出てくるんだが、
　説明しよう☆」

「 ３行で頼むぜ☆」

スタックは手札、ムーブは１枚のカードの読み方☆

NITEIRU

指し手生成は、カードゲームに似ている☆

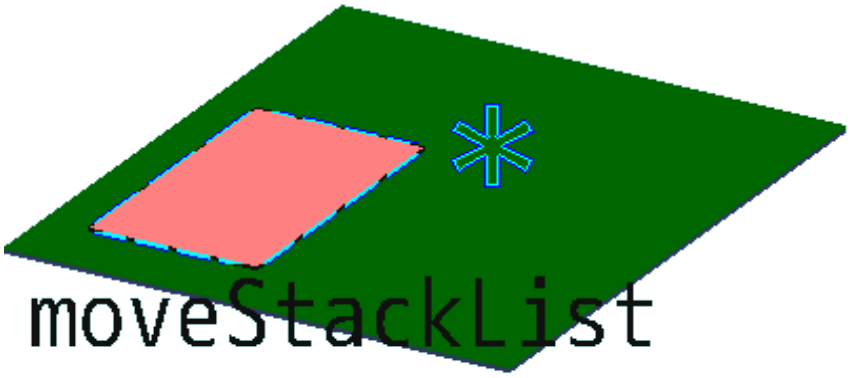


「 これ、手積み麻雀とかで敷くマットな☆
　マットの上には　見えないカーソル（図中＊印）　があって、
　ここに　カードを置くんだぜ☆」
　最初は　〇枚　な☆」

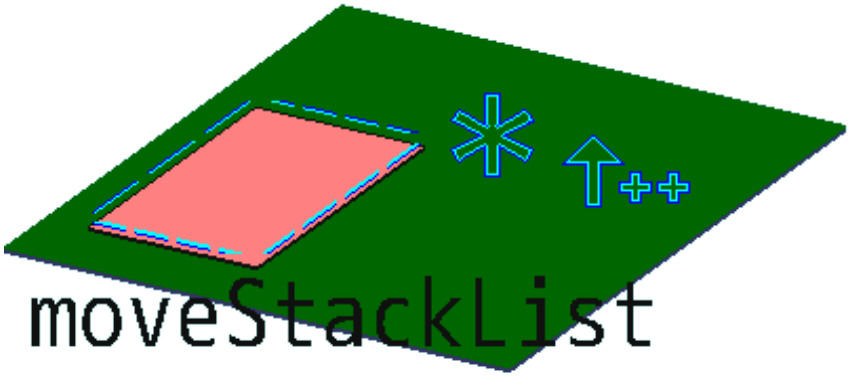
「 １秒経ったわよ！」

「 ……☆！」

「 ダメだな、お父んは☆」



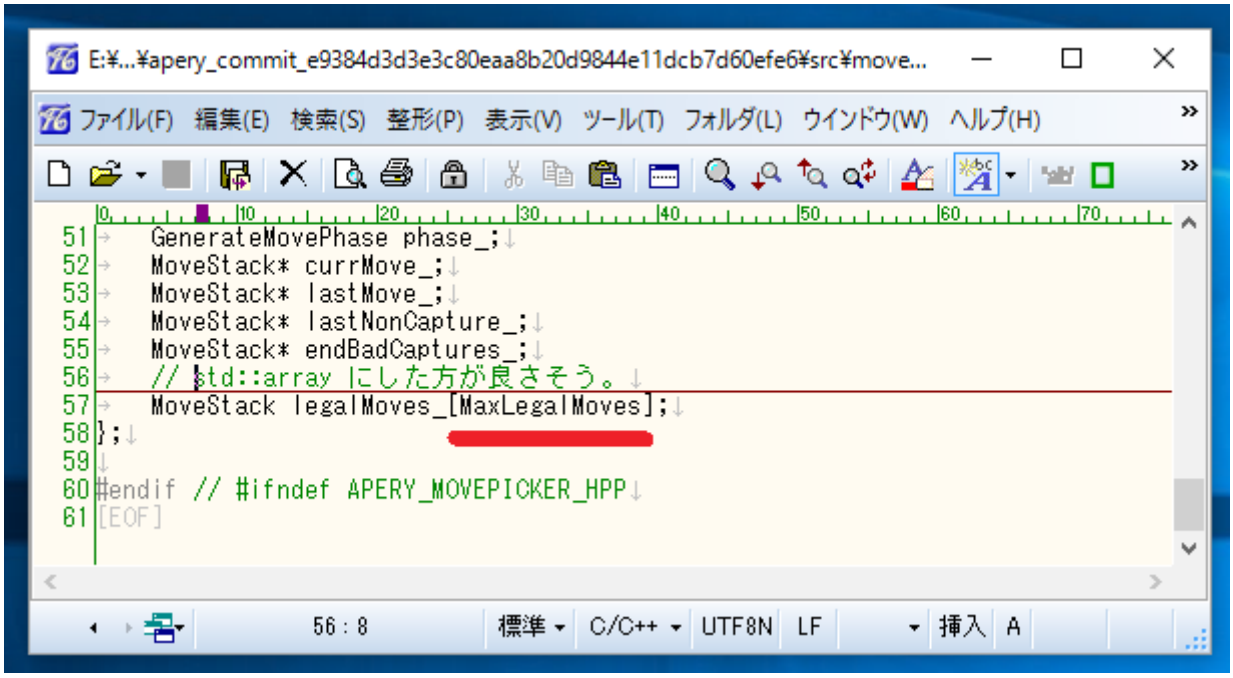
「 カードを１枚置いて……☆」



「 見えないカーソル（図中＊印）を　上空に　１枚分　上げる☆
　この　１個上げる操作を　＋＋（プラスプラス）と書くとしようぜ☆」

「 この　見えないカーソル（図中＊印）　は、
　天井を突き破って　月まで届くの？」

movePicker.hpp



position.hpp

E:¥...¥src¥position.hpp

ファイル(F) 編集(E) 検索(S) 整形(P) 表示(V) ツール(T) フォルダ(L) ウインドウ(W) ヘルプ(H)

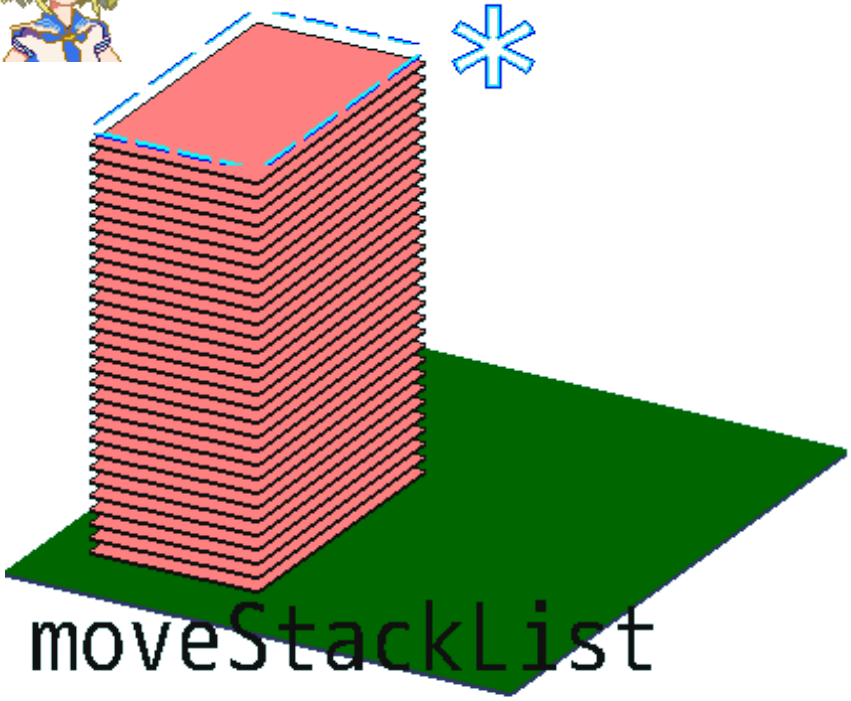
```
396 template <> inline Bitboard Position::attacksFrom<Horse >(const Color  , cons
397 template <> inline Bitboard Position::attacksFrom<Dragon>(const Color  , cons
398
399 // position sfen R8/2K1S1SSk/4B4/9/9/9/9/9/1L1L1L3 b PLNSGBR17p3n3g 1
400 // の局面が最大合法手局面で 593 手。番兵の分、+ 1 しておく。
401 const int MaxLegalMoves = 593 + 1;
402
403 class CharToPieceUSI : public std::map<char, Piece> {
404 public:
405     CharToPieceUSI() {
406         (*this)['P'] = BPawn;   (*this)['p'] = WPawn;
```

406 : 1 | 標準 | C/C++ | UTF8N | LF | 挿入 | A

「 見えないカーソル　は、５９４枚分までしか　上がらないぜ☆
　一番高く　カードが積みあがるケースでは、
　カードは　５９３枚まで　積みあがって、
　見えないカーソルが　５９４枚目で　昼寝する感じだぜ☆」

「 無限じゃないのね」

「 ３０枚ぐらい　繰り返すと、こうなるぜ☆」

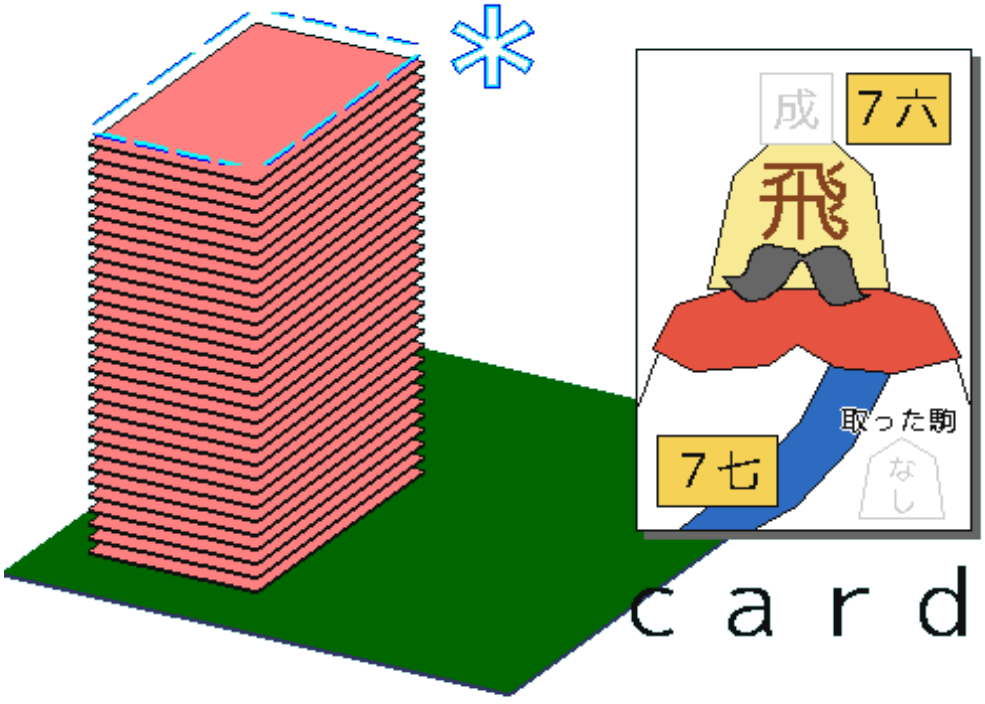


「 これが　ムーブ・スタック・リストな☆
　指し手生成　というのは、
　これからゲームを始めるためにに
　カードゲームの手札を配っているところに
　似ているぜ☆」

「 カード１枚が　指し手１つか☆？」

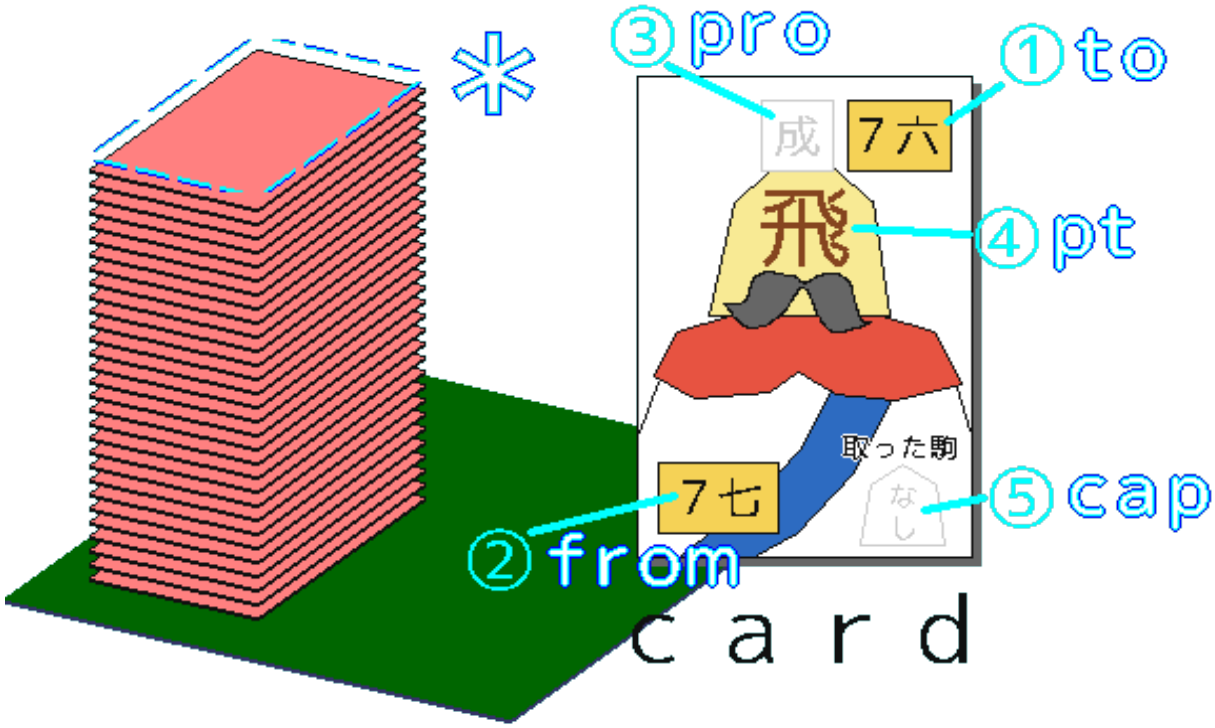
「 そうそう☆」

「 ちなみに、カードの図柄は　こうなっているぜ☆」



「 ぷっｗｗ」

「 各部名称を示すと　次の通りだぜ☆」



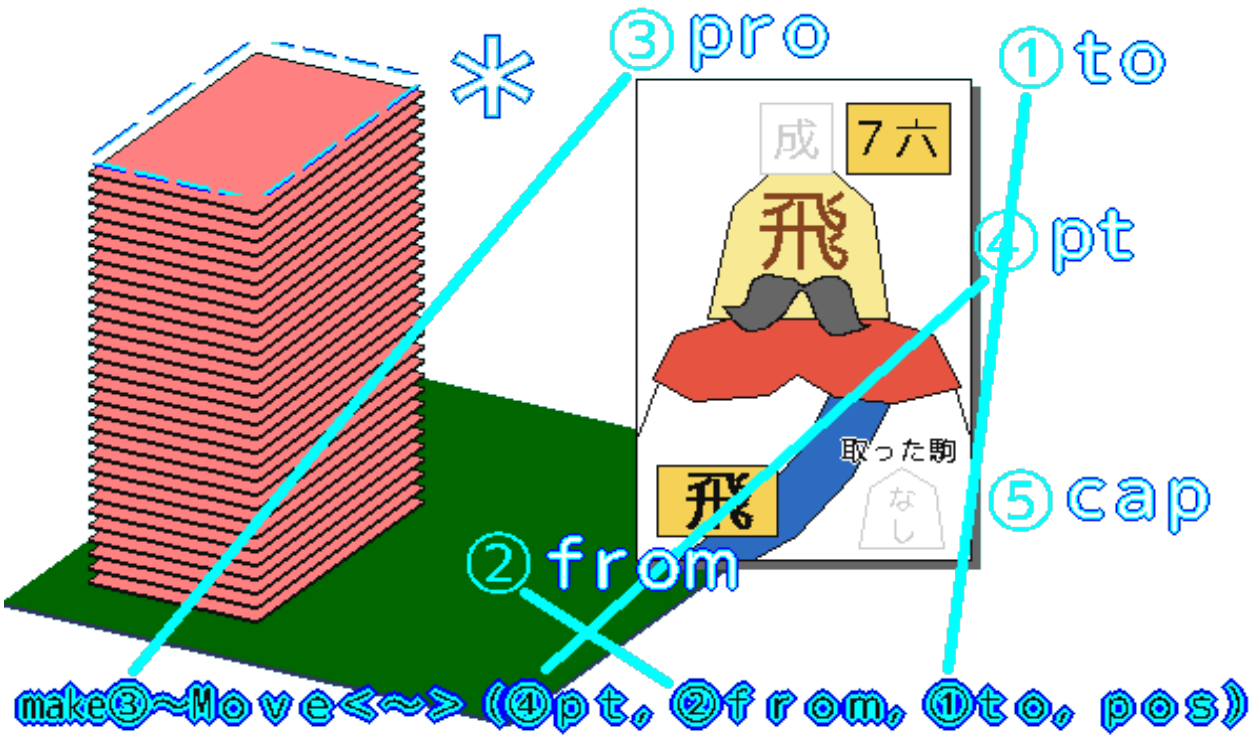
「 これは、さっきの　カンスーの　ヒキスー　にも合うぜ☆」

generateMoves.cpp

```
makePromoteMove<Capture>(pt, from, to, pos) :↓
makeNonPromoteMove<Capture>(pt, from, to, pos));↓
```

（※　丸かっこの中の　カンマ区切りのリストが、カンスーのヒキスーだぜ☆）

「 pt、　from、　to　☆
　カードの図柄に書くものを　入れているわけだぜ☆
　pos　は　現在の局面な☆」



「もう一回　見比べてみようぜ☆」

generateMoves.cpp

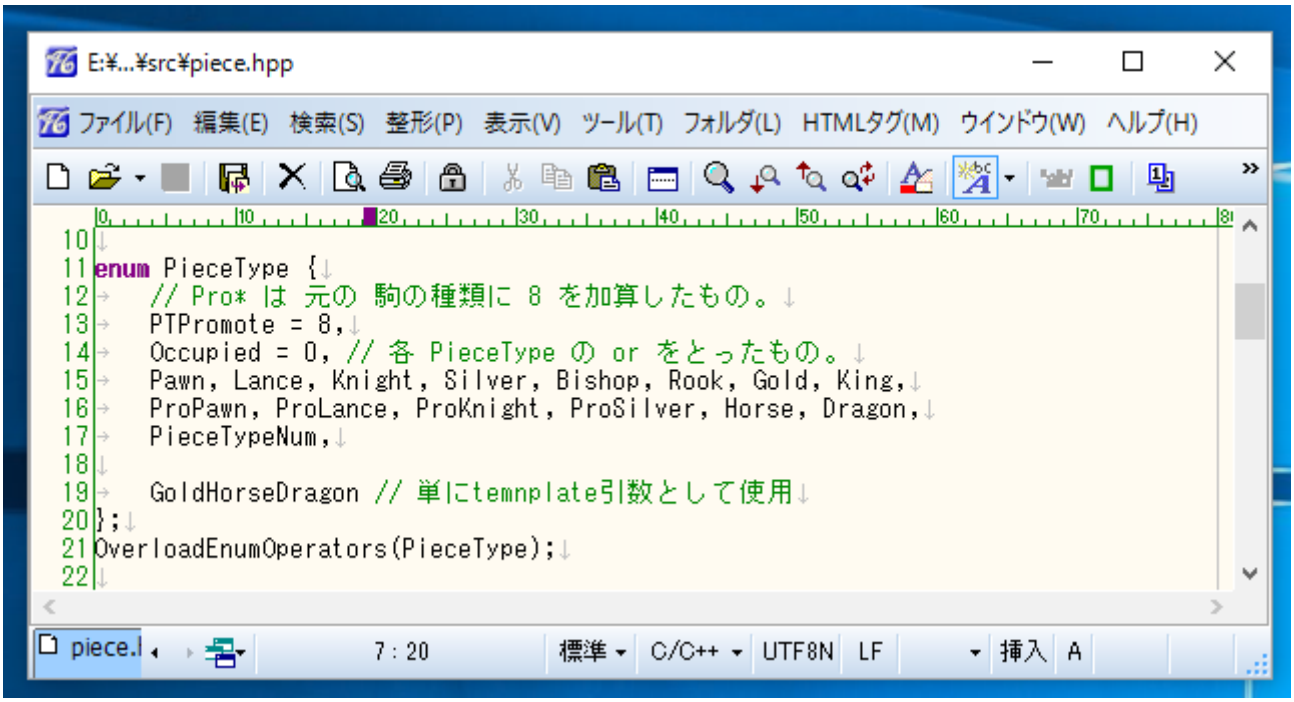
```
    switch (pt) {
    case Empty    : assert(false); break; // 最適化の為のダミー
    case Pawn     : case Lance    : case Knight   : case Silver   : case Bishop   : case Rook     :
        (*moveStackList++).move = ((canPromote(us, makeRank(to)) | canPromote(us, makeRank(from))) ?
                                   makePromoteMove<Capture>(pt, from, to, pos) :
                                   makeNonPromoteMove<Capture>(pt, from, to, pos));
        break;
    case Gold     : case King     : case ProPawn  : case ProLance : case ProKnight: case ProSilver: case
        (*moveStackList++).move = makeNonPromoteMove<Capture>(pt, from, to, pos);
    default       : UNREACHABLE;
```

「左が　ｍｏｖｅＳｔａｃｋＬｉｓｔ、　右が　カード　なのね」

「ｐｔ　＜＜　１６　も、説明しろだぜ☆」

「駒の種類は　数字で入っているんだが……☆」

pieceType.hpp



PieceType	漢字	10進数表記	2進数表記
Pawn	歩	1	00000001
Lance	香	2	00000010
Knight	桂	3	00000011
Silver	銀	4	00000100
Bishop	角	5	00000101
Rook	飛	6	00000110
Gold	金	7	00000111
King	玉	8	00001000
ProPawn	と	9	00001001
ProLance	杏	10	00001010
ProKnight	圭	11	00001011
ProSilver	全	12	00001100
Horse	馬	13	00001101
Dragon	竜	14	00001110

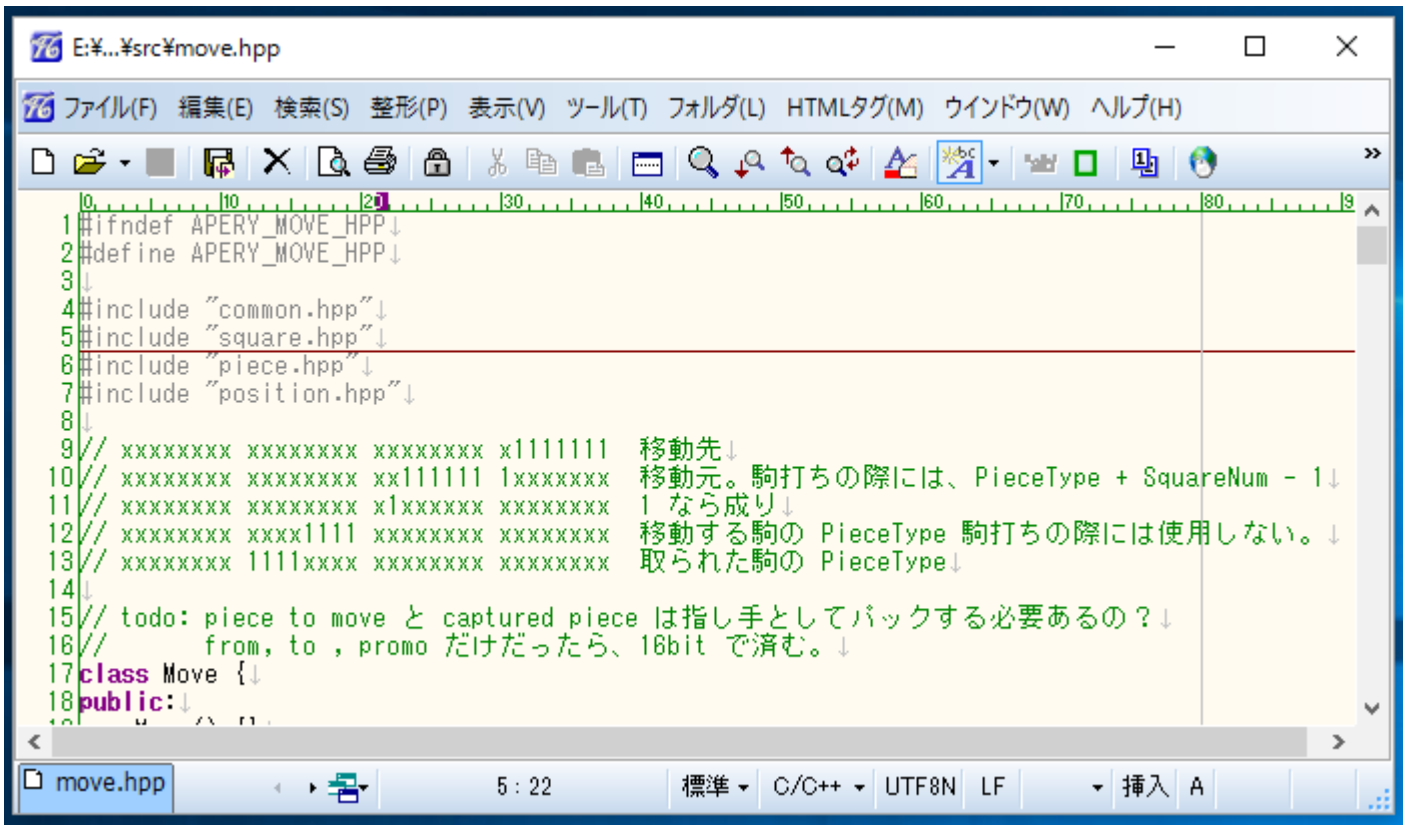
「飛　は、　１１１０　だろ☆
　これを　１６個　左にずらすと、次のようになるぜ☆」

１１１０　００００００００　００００００００

「ずらして　どうするの？」

「カードの図柄に　当てはめたんだぜ☆
　プログラムの説明には、次のようにあるぜ☆」

ｍｏｖｅ．ｈｐｐ



「１枚のカードは、２進数３２桁　なんだぜ☆
　この３２桁に　①ｔｏ、　②ｆｒｏｍ、　③ｐｒｏ、　④ｐｔ、　⑤ｃａｐ
　が　収まっているんだぜ☆」

「 ↓これ　が……☆」

①７六
②７七
③成らず
④飛車
⑤取った駒

「 ↓こう　なって……☆」

			①７六	
		②７七		←　７桁　ずれる
	③成らず			←　１４桁　ずれる
④飛車				←　１６桁　ずれる
⑤取った駒				←　２０桁　ずれる

「 ↓こう　だぜ☆！」

				↓
			↓	②７七
		↓	③成らず	
	↓	④飛車		
⑤取った駒				

「 漫画を描くのは　休み時間にしましょうよ。
　ちゆりは　もっと　勉強しましょう」

「 （（（（＾ｑ＾））））））」

「 そうだぜ☆？
　テストの解答欄は　教師を困らせるために　あるんじゃないんだぜ☆？」

「 言葉は通じているが、会話が通じないぜ☆！
　ここは　異次元なのかだぜ☆！　（＞＿＜）」

○秒☆

ZERO BYO

○秒を　高速化　しようぜ☆！

「エイプリー＿大樹の枝　は、
　ｐｔ　＜＜　１６　してるだろ☆」

「１６桁　左にずらしているようね」

「そこで　わたしは、次のように　考えたわけだぜ☆」

ＰｉｅｃｅＴｙｐｅ	漢字	１０進数表記	２進数表記
Ｐａｗｎ	歩	６５５３６	０００１　００００００００　００００００００
Ｌａｎｃｅ	香	１３１０７２	００１０　００００００００　００００００００
Ｋｎｉｇｈｔ	桂	１９６６０８	００１１　００００００００　００００００００
Ｓｉｌｖｅｒ	銀	２６２１４４	０１００　００００００００　００００００００
Ｂｉｓｈｏｐ	角	３２７６８０	０１０１　００００００００　００００００００
Ｒｏｏｋ	飛	３９３２１６	０１１０　００００００００　００００００００
Ｇｏｌｄ	金	４５８７５２	０１１１　００００００００　００００００００
Ｋｉｎｇ	玉	５２４２８８	１０００　００００００００　００００００００
ＰｒｏＰａｗｎ	と	５８９８２４	１００１　００００００００　００００００００
ＰｒｏＬａｎｃｅ	杏	６５５３６０	１０１０　００００００００　００００００００
ＰｒｏＫｎｉｇｈｔ	圭	７２０８９６	１０１１　００００００００　００００００００
ＰｒｏＳｉｌｖｅｒ	全	７８６４３２	１１００　００００００００　００００００００
Ｈｏｒｓｅ	馬	８５１９６８	１１０１　００００００００　００００００００
Ｄｒａｇｏｎ	竜	９１７５０４	１１１０　００００００００　００００００００

「 最初から　１６桁　ずらしておこうぜ☆」

「 それで　どれだけ　スピードアップ　したのよ？」

「 メモを残していないんだが、
　開発中は　右肩上がりに　ぐんぐん　強くなっていったぜ☆
　フラッドゲートに放流していても、
　改造すればするほど　バッタバッタ　相手を倒していくし、
　最終的に開発を終えたときには　ＷＣＳＣ２６　で　技巧　と　１手差だぜ☆」

「 数字は？」

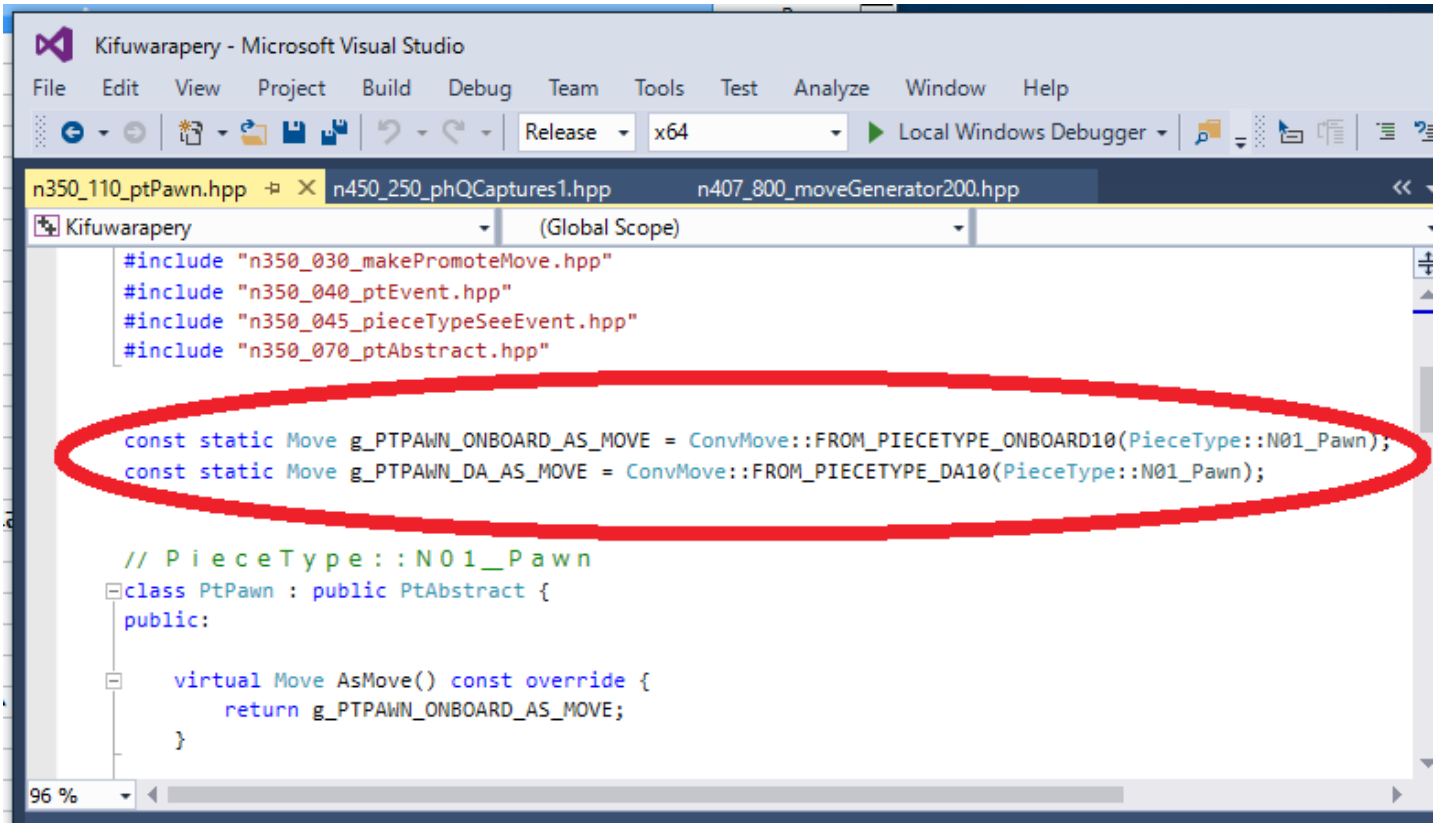
「 知らね☆」

「 お父んの目の前では　面白いことしか　起こっていないのか……☆」

「 面白かったことしか　記憶に残ってないんじゃないの？」

「 ひどい言われようだぜ☆」

n350_110_ptPawn.hpp



「盤上の移動と、　駒台からの打　に分けて
　駒　全種類分　カードを作っておいたのに……☆」

「値札を見てから　買い物しようぜ☆？」

「ダメよ。
千円札を　１００円玉　１０枚に両替すると
日本の景気が回復する世界に住んでいるのに」

「いっぱい　改造したぜ☆」

どっちが大きな分岐点☆？

IRO　　IRO

色で冷蔵庫を選ぶか、冷蔵庫を決めて色を選ぶか☆

generateMoves.cpp

```
    switch (pt) {
    case Empty   : assert(false); break; // 最適化の為のダミー
    case Pawn    : case Lance   : case Knight  : case Silver  : case Bishop  : case Rook    :
        (*moveStackList++).move = ((canPromote(us, makeRank(to)) | canPromote(us, makeRank(from))) ?
                                   makePromoteMove<Capture>(pt, from, to, pos) :
                                   makeNonPromoteMove<Capture>(pt, from, to, pos));
        break;
    case Gold    : case King    : case ProPawn  : case ProLance : case ProKnight: case ProSilver: case
        (*moveStackList++).move = makeNonPromoteMove<Capture>(pt, from, to, pos);
    default      : UNREACHABLE;
```

n407_800_moveGenerator200.hpp

```
// TODO: 配列のリミットチェックをしてないぜ☆（＾ｑ＾）
US == Color::Black
    ?
    PiecetypePrograms::m_PIECETYPE_PROGRAMS[pieceType]->Generate2RecaptureMoves_usBlack(moveStackList, pos, from, to)
    :
    PiecetypePrograms::m_PIECETYPE_PROGRAMS[pieceType]->Generate2RecaptureMoves_usWhite(moveStackList, pos, from, to)
    ;

eturn moveStackList;
```

「上が　エイプリー＿大樹の枝　のソースコードで、
下が　きふわらぷりー　だぜ☆
おんなじことを　やっているんだが、
書き方が違うんだぜ☆
わたしが書き直すと　レーティングが６００　下がるぜ☆」

「元に戻せだぜ☆」

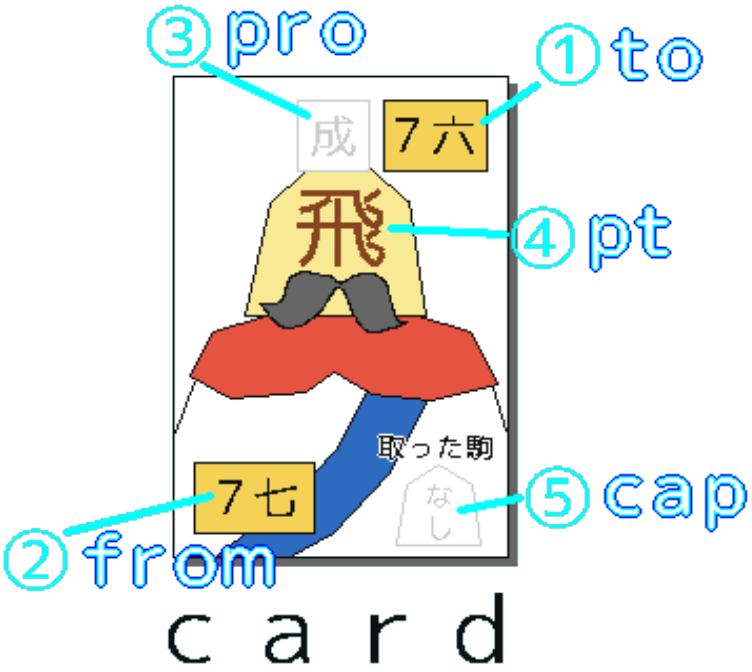
「無理☆」

「おんなじじゃないわよ。
ちゆりのは　なんで　ブラック（先手）と　ホワイト（後手）で　分かれてんの？」

「　白い冷蔵庫が　　１４台並んでいて　色が選べる　　ＮＹＯＪＩＭＡ　と、
白と黒の冷蔵庫が　２台並んでいて、メーカーを選べる　ＮＹＯＪＩＭＡ　、
好きな方を選べだぜ☆
どっちも同じだろ☆」

「違うと思うけどなぁ……☆」

「さっきのカードを　よく見るだぜ☆」



「カードには　カラー（先後）は無いな☆」

「あらまぁ、ほんとね」

「……☆（＾ｑ＾）；；；」

「 お父んが勝手に　先後の分岐を増やした☆」

「 先後の分岐はあるぜ☆」

n350_130_ptKnight.hpp

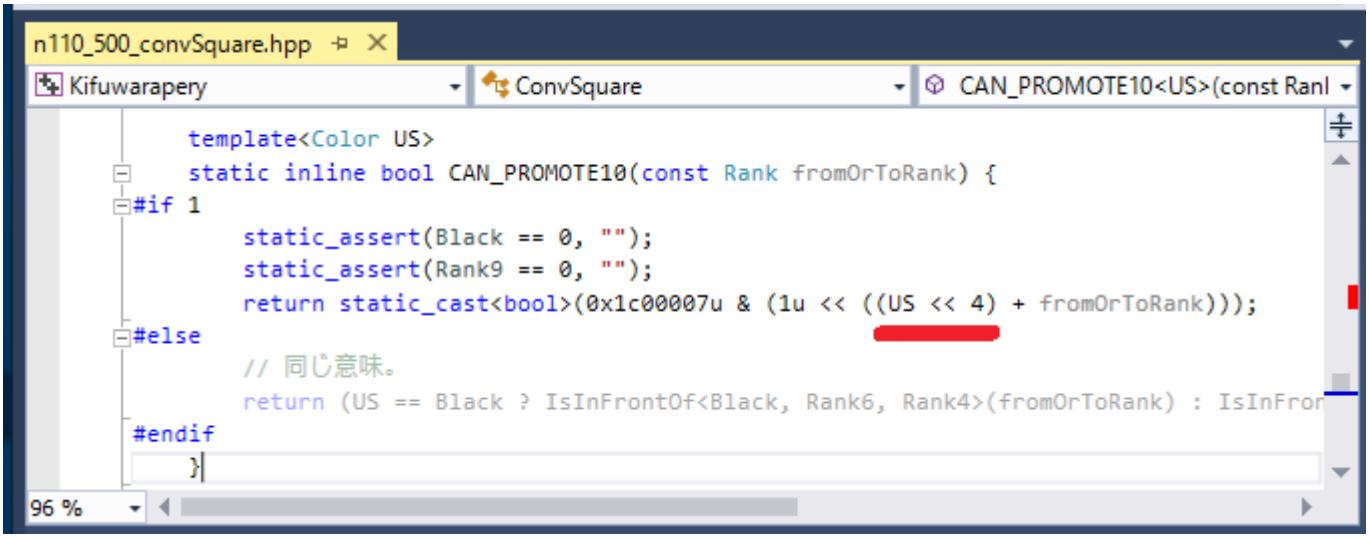
n350_130_ptKnight.hpp
Kifuwarapery
PtKnight
Generate2RecaptureMoves_usBlack(Mov

```
    // pin は省かない。//FORCE_INLINE
    void Generate2RecaptureMoves_usWhite(
        MoveStack* moveStackList,
        const Position& pos,
        const Square from,
        const Square to
        ) const override {
        moveStackList->m_move = g_makePromoteMove.GetSelectedMakeMove_ExceptPromote_CaptureCategory(
            this->AsMove(), from, to, pos);

        if (
            ConvSquare::CAN_PROMOTE10<Color::White>(ConvSquare::TO_RANK10(to))
            |
            ConvSquare::CAN_PROMOTE10<Color::White>(ConvSquare::TO_RANK10(from))
        ){
            MakePromoteMove::APPEND_PROMOTE_FLAG(moveStackList->m_move);
        }

        moveStackList++;
    }

    void Generate2RecaptureMoves_usBlack(
        MoveStack* moveStackList,
        const Position& pos,
        const Square from,
        const Square to
        ) const override {
        moveStackList->m_move = g_makePromoteMove.GetSelectedMakeMove_ExceptPromote_CaptureCategory(
            this->AsMove(), from, to, pos);

        if (
            ConvSquare::CAN_PROMOTE10<Color::Black>(ConvSquare::TO_RANK10(to))
            |
            ConvSquare::CAN_PROMOTE10<Color::Black>(ConvSquare::TO_RANK10(from))
            ) {
            MakePromoteMove::APPEND_PROMOTE_FLAG(moveStackList->m_move);
        }

        moveStackList++;
    }
```

96 %

「白（先手）は　４段より上、黒（後手）は　６段より下に　桂馬がいると
成り駒……、みたいな　処理があるんだぜ☆！
でも
分けただけで　意味ないな☆」

n110_500_convSquare.hpp



「ソースコードに埋め込んである
ビット演算は　先に済ませてしまおう……、
としていて　開発時間が足りなかった感じ……☆
やり残しがあるぜ☆」

「地道にレーティングを下げているようね」

「やめてくれだぜ☆」

「でも　こんな地道なやり方じゃ　６００もレーティング下がらないでしょ。
もっと　ガーッ　と下げてるところがあるでしょ」

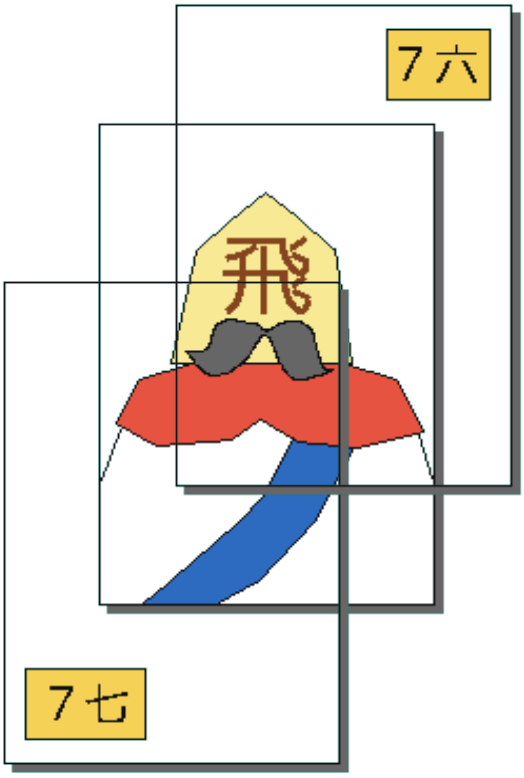
「地道を積み重ねて　６００　下げたんじゃないか☆？」

レイヤーを重ねてカードを作ろうぜ☆

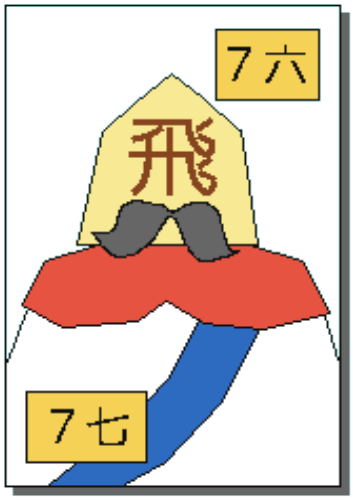
HARITUKE

透明なシールを張り付けるように　カード作成☆

「 指し手生成の理解は進んだんだぜ☆」



「 ｔｏ　　だけのカード、
　ｐｔ　　だけのカード、
　ｆｒｏｍ　だけのカード　を作っておいて……」



「 ガッチャンコ　するんだぜ☆
　縦棒　｜　を使えば「 ガッチャンコ　できるんだぜ☆」

ｎ１６５＿６００＿ｃｏｎｖＭｏｖｅ．ｈｐｐ

n165_600_convMove.hpp

Kifuwarapery　ConvMove

```
    static inline Move FROM_CAPTURED_PIECE_TYPE10(const PieceType captured) { return static

    // 移動する駒の種類、移動元、移動先　から指し手に変換
    static inline Move FROM_PT_SRC_DST20( //新型☆（＾ｑ＾）
        const Move pieceTypeAsMove,
        const Square from,
        const Square to
        ) {
        return pieceTypeAsMove
            |
            ConvMove::FROM_SRC10(from)
            |
            static_cast<Move>(to) // 移動先の Square→Move は、書式変換要らず☆（＾ｑ＾）
            ;
    }
    static inline Move FROM_PT_SRC_DST30( //旧型☆（＾ｑ＾）
        const PieceType pt, // 新型なら、ここをムーブにできるぜ☆（＾ｑ＾）
        const Square from,
        const Square to
    ) {
        return
            ConvMove::FROM_PIECETYPE_ONBOARD10(pt) // ピースタイプをムーブに変換するだけ☆
            |
            ConvMove::FROM_SRC10(from)
            |
            static_cast<Move>(to) // 移動先の Square→Move は、書式変換要らず☆（＾ｑ＾）
            ;
    }
```

```
    // 駒を打つ手の符号（Move）を作成するぜ☆（ q ）
    static inline Move Convert30_MakeDropMove_da(
        const Move pieceTypeAsDaMove,
        const Square to) {
        return
            pieceTypeAsDaMove//（打とうとしている駒の種類の）打のMove書式だぜ☆（＾ｑ＾）
            |
            static_cast<Move>(to) // 移動先の Square→Move は、書式変換要らず☆（＾ｑ＾）
            ;
    }

};
```

96 %

「わたしは　いろんなカードの作り方を　まとめなおしたんだぜ☆」

「ｇｅｎｅｒａｔｅＭｏｖｅｓ（ジェネレート・ムーブ）って、
データを　カードに印刷して　山札に置いてるだけなのね。
もっと、飛車は　駒にぶつかるまで進めるとか、
金は　上、斜め上、横、下　に進めるとか、
そういうことは　やらないの？」

「それは
ｇｅｎｅｒａｔｅＰｉｅｃｅＭｏｖｅｓ（ジェネレート・ピース・ムーブス）と、
ｇｅｎｅｒａｔｅＤｒｏｐＭｏｖｅｓ　（ジェネレート・ドロップ・ムーブス）
だぜ☆
もちろん　いじってある☆」

「……」

「駒の種類ごとに　クセ　があって　プログラムが異なるんだが、
歩　を例にして　ジェネレート・ピース・ムーブ　を紹介しておくぜ☆」

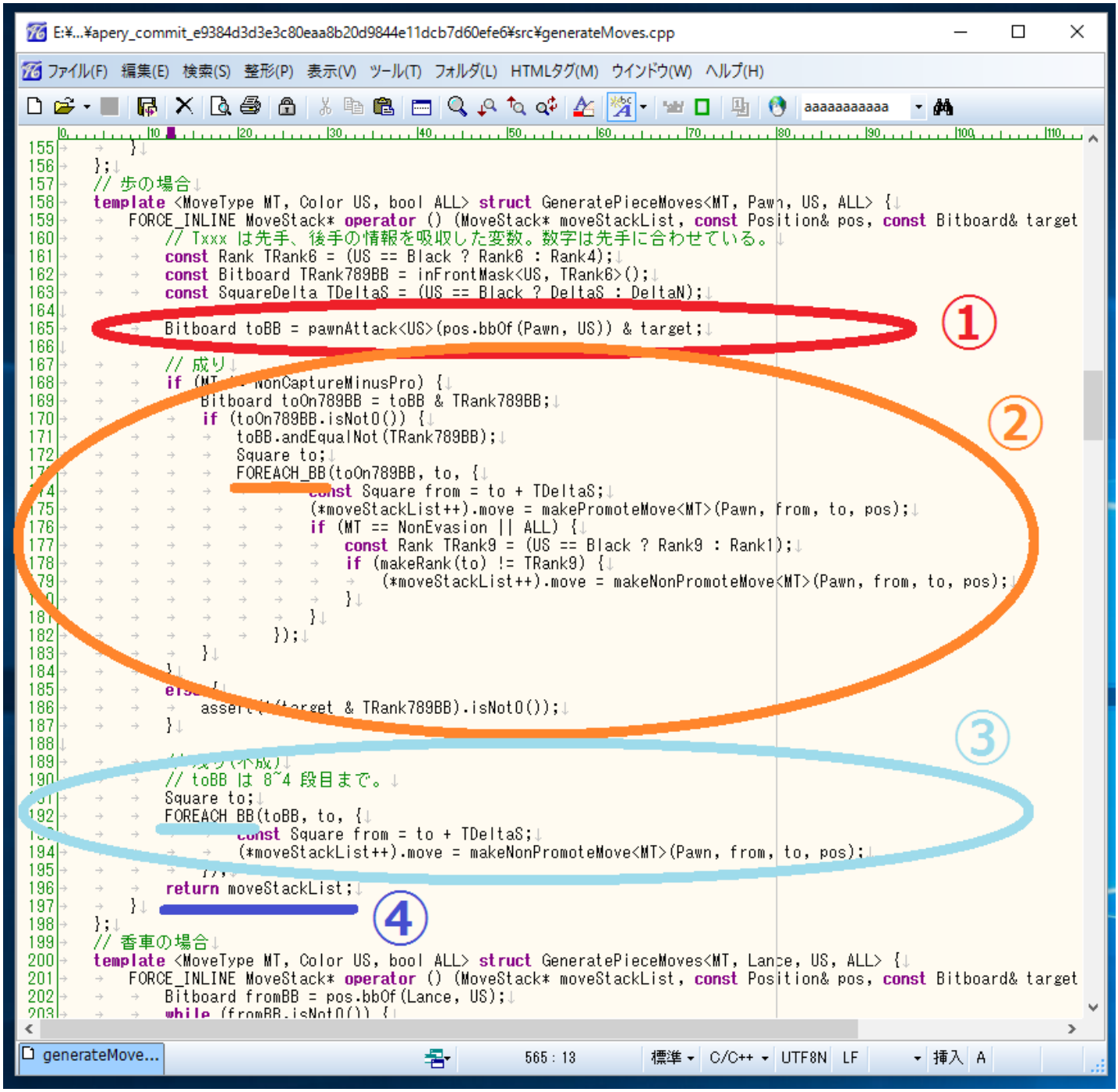
「３行で頼むぜ☆」

駒の利きの数だけ　カードを作ろうぜ☆（＾ｑ＾）！

MAA

まあ、歩の利きなんて　頭１つなんだが☆

generateMoves.cpp



「 **①　では、**
歩の利きマスが　将棋盤と同じ　9×9　の表　＋ｏＢＢ　に　印付けられるぜ☆

②　では、
先手なら　表の１～３段目だけ　引っこ抜いて　表＋ｏＯｎ７８９ＢＢ　が作られ、
この表に付いている印は全部　歩成り　として
カードを作って　山札に積むぜ☆

ついでに　表　＋ｏＢＢ　の１～３段目は　クリアーしているぜ☆

③　では、
先手なら　表の１～３段目がクリアーされた　＋ｏＢＢ　を使って、
残っている印の数だけ
成りではない　歩の移動　として　カードを作って　山札に積むぜ☆

④　では、
積み上げた　山札　を　関数呼び出し元に戻しているぜ☆」

「 歩の不成りとか　作ってないのね」

「 歩の利きなんか調べたって　８１マスの将棋盤の　１マスにしか
印が付かないと思うんだが、
１マス　１マス　ループカウンターを増やして　調べているのではなくて、
ビットが立っているマスを　調べて取ってくる　命令があるみたいなんだぜ☆
全部の　立っているビットを　調べ終わったら終わり　というループだぜ☆」

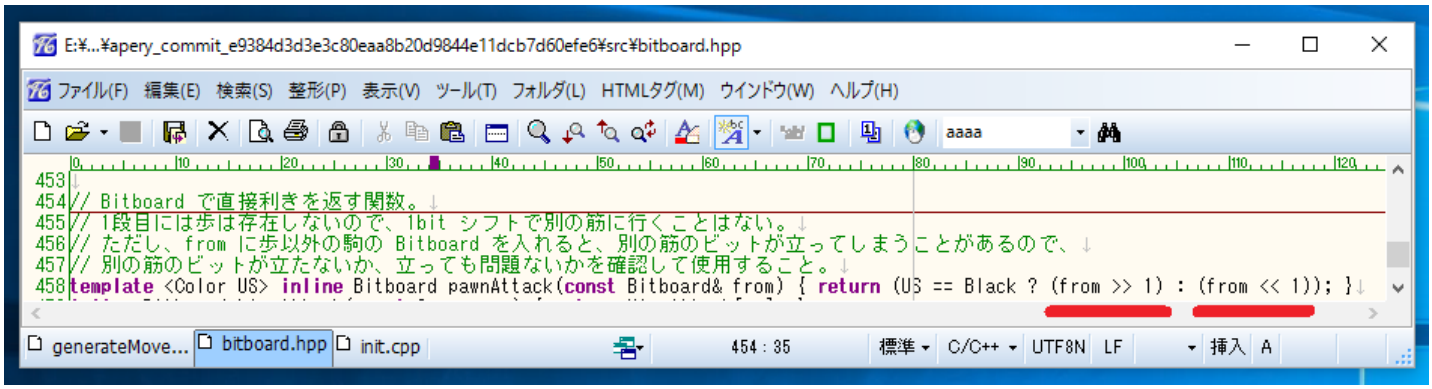
「 駒の利きは　どうやって　付けているんだぜ☆？
ｐａｗｎＡｔｔａｃｋ　について　ｋｗｓｋ（くわしく）☆」

８１個のビットボードを持とうぜ☆（＾ｑ＾）！

KOMA

駒の利きは　全てのマスだけ調べて持ってようぜ☆！

bitboard.hpp

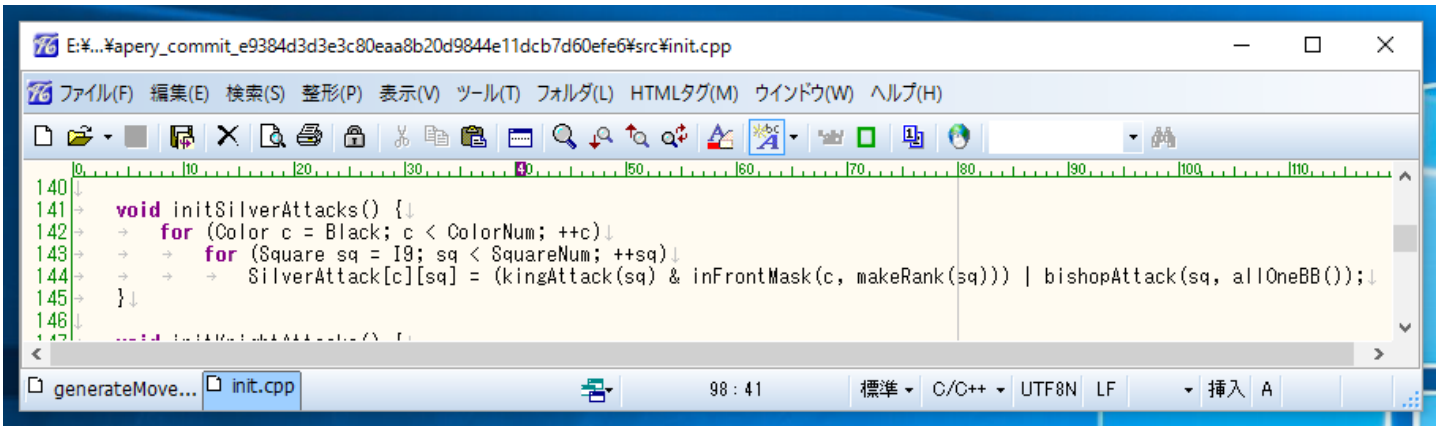


「歩は　先手番なら　１個前　の位置、
後手番なら　１個後ろの位置　が利きだぜ☆」

「じゃあ　銀は　どうやってるんだぜ☆？
ななめの　動きがあるぜ☆？」

「将棋エンジンを起動した直後に、
駒の種類別に、どのマスにいたら　どこに利くかの　一覧表　を
作っているんだぜ☆」

init.cpp



「ＳｉｌｖｅｒＡｔｔａｃｋ［ｃ］［ｓｑ］　という配列があって、
ｃ　は　カラー（先後）、ｓｑ　は　スクエア（升）　だぜ☆
配列の要素は　利きが立った　ビットボードだぜ☆

王様の動きの　頭横一列と、　角の動きの後ろ横一列を　重ねれば
銀の動きになると思うんだが、
マジック・ビットボードが　使われているのか、何をやっているのか分からないぜ☆」

「謎か☆？」

「謎だぜ☆！」

「分かっている途中なんじゃないの？」

「分かっている途中だぜ☆！」

ビットボードは　エイプリー＿大樹の枝　と同じ☆

FUSIGI

分かっている途中は、不思議なものだぜ☆

「 ビットボードは　いじってないぜ☆！」

「 じゃあ　ジェネレート・ピース・ムーブス　のどこを　いじったのよ？」

ジェネレート・ピース・ムーブスはいじったぜ☆

FUNCTOR

ファンクターって何だぜ☆？（＾ｑ＾）？

generateMoves.cpp

E:¥...¥apery_commit_e9384d3d3e3c80eaa8b20d9844e11dcb7d60efe6¥src¥generateMoves.cpp

ファイル(F) 編集(E) 検索(S) 整形(P) 表示(V) ツール(T) フォルダ(L) HTMLタグ(M) ウインドウ(W) ヘルプ(H)

```
253    // 銀の場合
254    template <MoveType MT, Color US, bool ALL> struct GeneratePieceMoves<MT, Silver, US, ALL> {
255        FORCE_INLINE MoveStack* operator () (MoveStack* moveStackList, const Position& pos, const Bitboard& target
256            Bitboard fromBB = pos.bbOf(Silver, US);
257            while (fromBB.isNot0()) {
258                const Square from = fromBB.firstOneFromI9();
259                const bool fromCanPromote = canPromote(US, makeRank(from));
260                Bitboard toBB = pos.attacksFrom<Silver>(US, from) & target;
261                while (toBB.isNot0()) {
262                    const Square to = toBB.firstOneFromI9();
263                    const bool toCanPromote = canPromote(US, makeRank(to));
264                    if (fromCanPromote | toCanPromote)
265                        (*moveStackList++).move = makePromoteMove<MT>(Silver, from, to, pos);
266                    (*moveStackList++).move = makeNonPromoteMove<MT>(Silver, from, to, pos);
267                }
268            }
269            return moveStackList;
270        }
271    };
```

generateMove... 253 : 16 標準 C/C++ UTF8N LF 挿入 A

「プログラムを見たとき、　ＵＳ　は　自分の色（先後）、
ＴＨＥＭ　は　相手の色（先後）、
ＭＴ　は　ムーブタイプで、　Ｓｉｌｖｅｒ　は　銀ピースタイプ、
ＡＬＬ　は　リーガル・オール、

と読めるようになってくるんだが、

色を　テンプレート化して、
ピースタイプは　先に１６ビットずらしておく、

といった改造を　進めている途中だったんだぜ☆」

「ＮＹＯＪＩＭＡ　ね」

「会話の通じない漫画だぜ☆」

n374_350_PieceMoveGenerator.hpp

n374_350_PieceMovesGenerator.hpp

Kifuwarapery | PieceMovesGenerator | GeneratePieceMoves_N01_Pawn<US>(MoveStack * moveStackList, cor

```
// 銀の場合
template <const Color US>
static FORCE_INLINE MoveStack* GeneratePieceMoves_N04_Silver(
    MoveStack* moveStackList,
    const PieceMoveEvent ptEvent,
    const Bitboard& target
) {
    Bitboard fromBB = ptEvent.m_pos.GetBbOf20<US>(N04_Silver);
    while (fromBB.Exists1Bit()) {
        const Square from = fromBB.PopFirstOneFromI9();
        const bool fromCanPromote = ConvSquare::CAN_PROMOTE10<US>(ConvSquare::TO_RANK10(from));
        const PieceTypeEvent ptEvent1(g_nullBitboard, US, from);
        Bitboard toBB = PiecetypePrograms::m_SILVER.GetAttacks2From(ptEvent1) & target;
        while (toBB.Exists1Bit()) {
            const Square to = toBB.PopFirstOneFromI9();
            const bool toCanPromote = ConvSquare::CAN_PROMOTE10<US>(ConvSquare::TO_RANK10(to));
            if (fromCanPromote | toCanPromote) {
                moveStackList->m_move = g_makePromoteMove.GetSelectedMakeMove_ExceptPromote_mt2(ptEvent.m_mt,g_PTSILVER_ONBOARD_AS_MOVE,    from, to, ptEvent.m_pos);
                MakePromoteMove::APPEND_PROMOTE_FLAG(moveStackList->m_move);
                moveStackList++;
            }

            moveStackList->m_move = g_makePromoteMove.GetSelectedMakeMove_ExceptPromote_mt2(ptEvent.m_mt,g_PTSILVER_ONBOARD_AS_MOVE,from, to, ptEvent.m_pos);
            moveStackList++;
        }
    }
    return moveStackList;
}
```

「ほかの変更点としては、ｔｅｍｐｌａｔｅ　から
ＭＴ、ＡＬＬ　を外したぜ☆

ｏｐｅｒａｔｏｒ（）　も外したし、ｓｔｒｕｃｔ　も止めたぜ☆」

「外したくなったのね」

「お父んが　大したことがないと思っている　改悪の１つは　これかだぜ☆」

「なんだこれ☆？　と思ったので　外したんだぜ☆
タイヘン　だったんだぜ☆」

「なんで　タイヘン　なことを　乗り越えるのよ！
これは　関数オブジェクト、ファンクター、ファンクショノイド　と呼ばれる手法で
関数を呼び出すよりも　関数オブジェクトを作成する方が　高速だから　わざわざ
ｓｔｒｕｃｔ型の（）演算子を使って　命令を実行するように　作ってあるの」

「そんなの知ってたら、　外さないぜ☆！」

ジェネレート・ドロップ・ムーブスも　いじったぜ☆

UNROLLER

アンローラーって何だぜ☆？（＾ｑ＾）？

「駒台の持ち駒を　打つ　指し手生成も　いじったぜ☆」

「今度は何を外したの？」

「持ち駒は　７種類あるわけだが……☆」

ピースタイプ

歩	桂	香	銀	金	角	飛

「エイプリー＿大樹の枝　では、持ち駒の指し手生成は
（１）　歩
（２）　歩以外
で　まず　大きく異なるんだぜ☆」

ピースタイプ

歩

桂	香	銀	金	角	飛

「歩は　先手のときは　１段目に歩を打ち込めないので
１段目に打つ手は省略だぜ☆

そして　二歩、打ち歩詰め　という特別ルールがあるのも　別口なところだぜ☆

ただし　先手の場合、敵玉に９段目に入玉されているとき、打ち歩詰めルールは
考えなくていいので　打ち歩詰めチェックは　省略されるぜ☆」

「二歩、打ち歩詰めは　打ち込む前に　チェックを　済ませているのね」

「 歩はこんなところだぜ☆
　残りを　細かく見ると、　桂、香　も　ちょっと特別扱いだぜ☆」

ピースタイプ

桂	香	銀	金	角	飛

「 どんな違いがあるの？」

「 先手の場合、桂は　１～２段目、　香は　１段目に　打ち込めないぜ☆

　銀、金、角、飛は　盤全体へ　打ち込めるぜ☆
　この４つは　ループで回しているぜ☆」

「 **（１）まず最初に**
　１段目に打ち込める　銀、金、角、飛　を、
　１段目に打ち込む　指し手　を生成するぜ☆

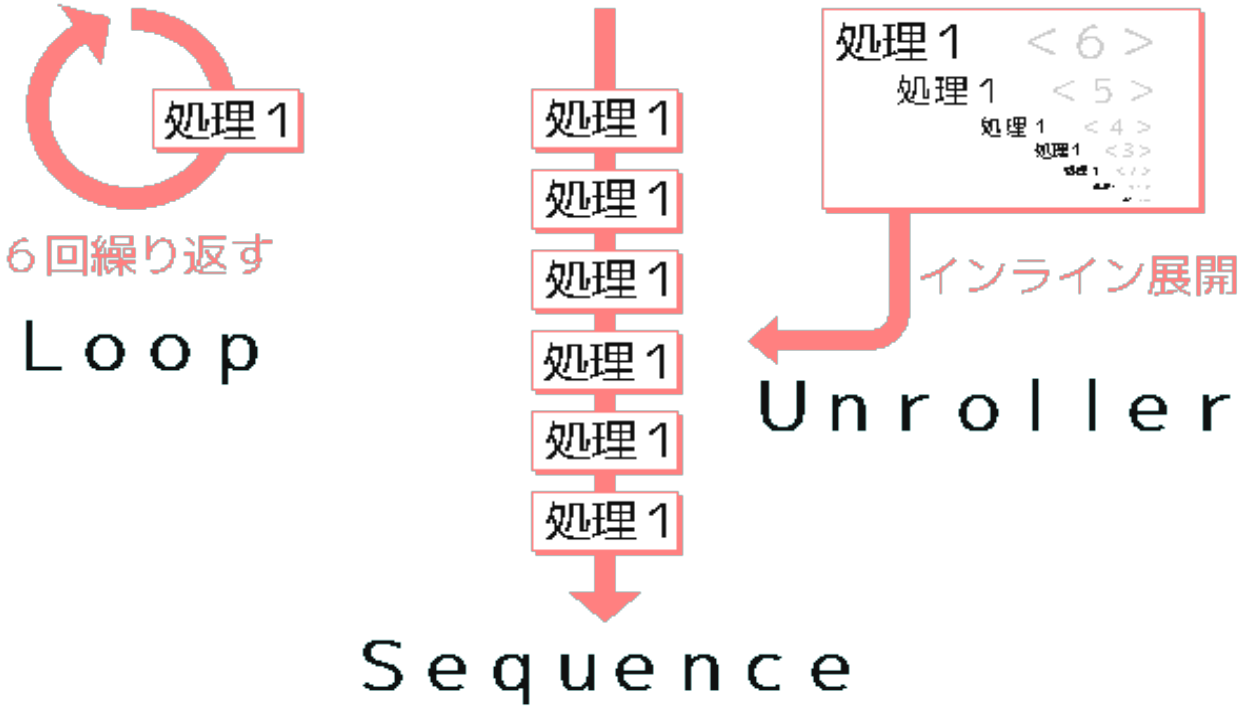
（２）次に
　２段目に打ち込める　香、銀、金、角、飛　を、
　２段目に打ち込む　指し手　を生成するぜ☆

（３）最後に
　３～９段目に打ち込める　桂、香、銀、金、角、飛　を、
　３～９段目に打ち込む　　指し手　を生成するぜ☆」

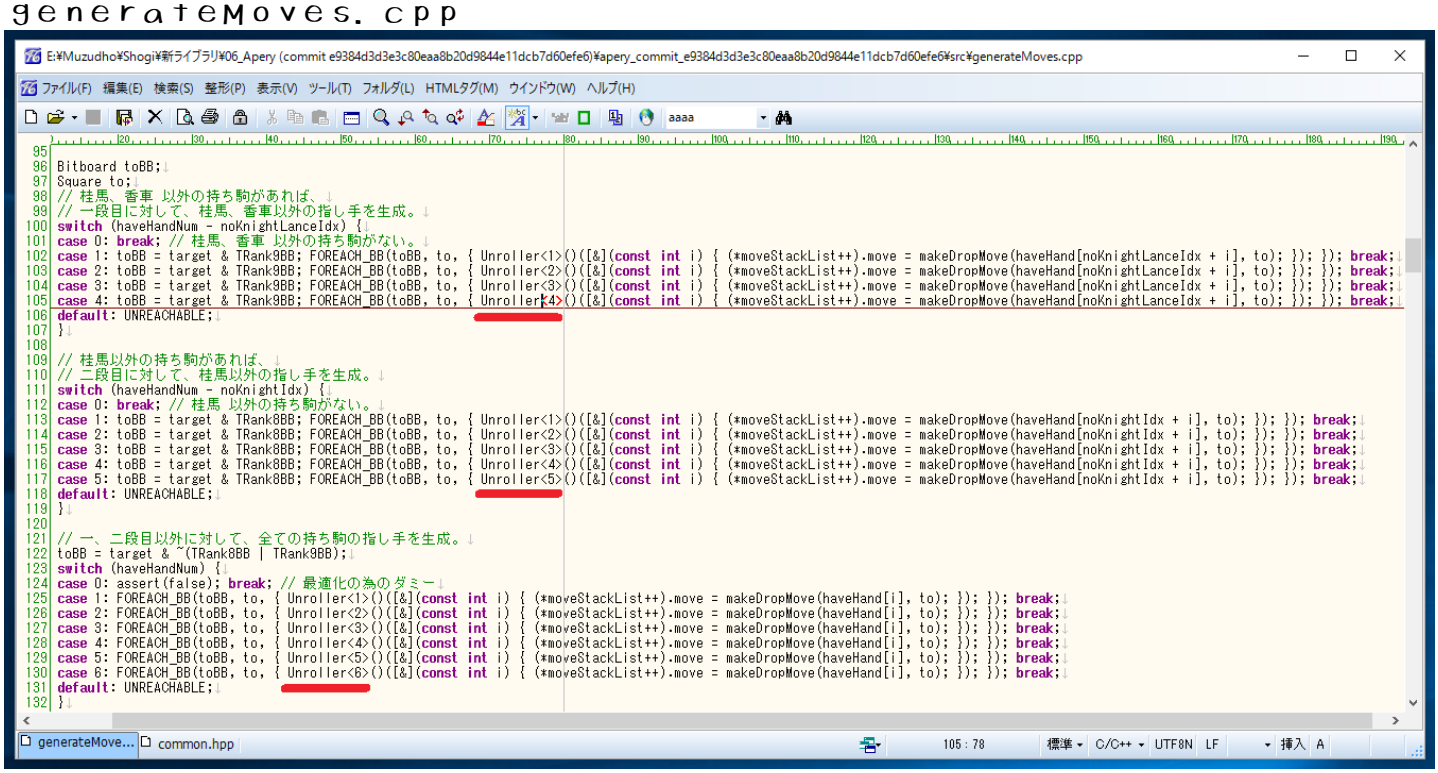
「 ビットボードの　マスク演算　を　うまく利用しているのね」

「 このとき、持ち駒の種類に応じて　ループの回数が変わるだろ☆
　（１）　のときは　０～４回☆
　（２）　のときは　０～５回☆
　（３）　のときは　０～６回☆

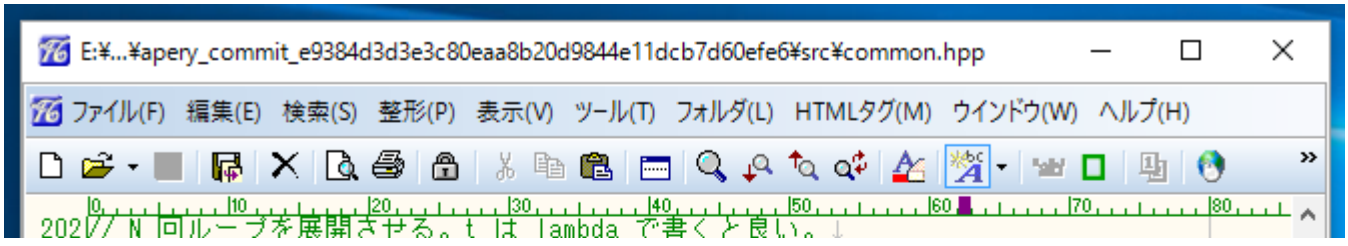
このループは、この場所のために
Ｕｎｒｏｌｌｅｒ　という技を使って
ループ文で回さずに、
カードを作って　山札に積む処理を　順番に　並べて実行しているぜ☆」



generateMoves.cpp



common.hpp



```
// こんな感じに書くと、lambda がテンプレート引数の数値の分だけ繰り返し生成される。
// Unroller<5>()([&](const int i){std::cout << i << std::endl;});
template <int N> struct Unroller {
    template <typename T> FORCE_INLINE void operator () (T t) {
        Unroller<N-1>()(t);
        t(N-1);
    }
};
template <> struct Unroller<0> {
    template <typename T> FORCE_INLINE void operator () (T) {}
};
```

common.hpp　212 : 63　標準　C/C++　UTF8N　LF　挿入　A

「 エイプリー＿大樹の枝　のソースコードが読めるようになってきただろ☆」

「 で、どこを　いじったんだぜ☆？」

n358_160_dropMakeHand6.hpp

n358_160_dropMakerHand6.hpp

Kifuwarapery　(Global Scope)

```
    MoveStack* MakeDropMovesToRank1234567(
        MoveStack* pMovestack,
        const DropMakerEvent& dmEvent,
        Move haveHandArr[6]
    ) const override {
        Bitboard toBB = dmEvent.m_target & ~(dmEvent.m_tRank8BB | dmEvent.m_tRank9BB);
        while (toBB.GetP(0)) {
            Square iTo = toBB.PopFirstOneRightFromI9();
            pMovestack->m_move = ConvMove::Convert30_MakeDropMove_da(haveHandArr[5], iTo);
            pMovestack++;
            pMovestack->m_move = ConvMove::Convert30_MakeDropMove_da(haveHandArr[4], iTo);
            pMovestack++;
            pMovestack->m_move = ConvMove::Convert30_MakeDropMove_da(haveHandArr[3], iTo);
            pMovestack++;
            pMovestack->m_move = ConvMove::Convert30_MakeDropMove_da(haveHandArr[2], iTo);
            pMovestack++;
            pMovestack->m_move = ConvMove::Convert30_MakeDropMove_da(haveHandArr[1], iTo);
            pMovestack++;
            pMovestack->m_move = ConvMove::Convert30_MakeDropMove_da(haveHandArr[0], iTo);
            pMovestack++;
        }
        while (toBB.GetP(1)) {
            Square iTo = toBB.PopFirstOneLeftFromB9();
            pMovestack->m_move = ConvMove::Convert30_MakeDropMove_da(haveHandArr[5], iTo);
            pMovestack++;
```

```
            pMovestack->m_move = ConvMove::Convert30_MakeDropMove_da(haveHandArr[4], iTo);
            pMovestack++;
            pMovestack->m_move = ConvMove::Convert30_MakeDropMove_da(haveHandArr[3], iTo);
            pMovestack++;
            pMovestack->m_move = ConvMove::Convert30_MakeDropMove_da(haveHandArr[2], iTo);
            pMovestack++;
            pMovestack->m_move = ConvMove::Convert30_MakeDropMove_da(haveHandArr[1], iTo);
            pMovestack++;
            pMovestack->m_move = ConvMove::Convert30_MakeDropMove_da(haveHandArr[0], iTo);
            pMovestack++;
        }
        return pMovestack;
    }
```

「 アンローラーも　もちろん　外しておいたぜ☆」

「 ほーう☆　ほーう☆　ほーう☆　おうー☆！」

「 ｔｏＢＢ．ＧｅｔＰ（０）　と
　ｔｏＢＢ．ＧｅｔＰ（１）　に分けているのは　何なの？」

エイプリー＿大樹の枝　の局面のマス番号の振り方☆

FUTATU

６４ｂｉｔ　２つで　１Ｂｉｔｂｏａｒｄ　だぜ☆

「 エイプリー＿大樹の枝は、将棋の盤を２つに割って持っているんだぜ☆」

p_[0]　（呼び名：　ｒｉｇｈｔ）

９	８	７	６	５	４	３	２	１	
		54	45	36	27	18	9	0	一
		55	46	37	28	19	10	1	二
		56	47	38	29	20	11	2	三
		57	48	39	30	21	12	3	四
		58	49	40	31	22	13	4	五
		59	50	41	32	23	14	5	六
		60	51	42	33	24	15	6	七
		61	52	43	34	25	16	7	八
		62	53	44	35	26	17	8	九

p_[1]　（呼び名：　ｌｅｆｔ）

９	８	７	６	５	４	３	２	１	
72	63								一
73	64								二
74	65								三
75	66								四
76	67								五
77	68								六

78	69								七
79	70								八
80	71								九

「 ついでに　持ち駒は　次のように番号が振られているぜ☆

△後手

歩	０枚目	118
香	０枚目	136
桂	０枚目	140
銀	０枚目	144
金	０枚目	148
角	０枚目	152
飛	０枚目	154

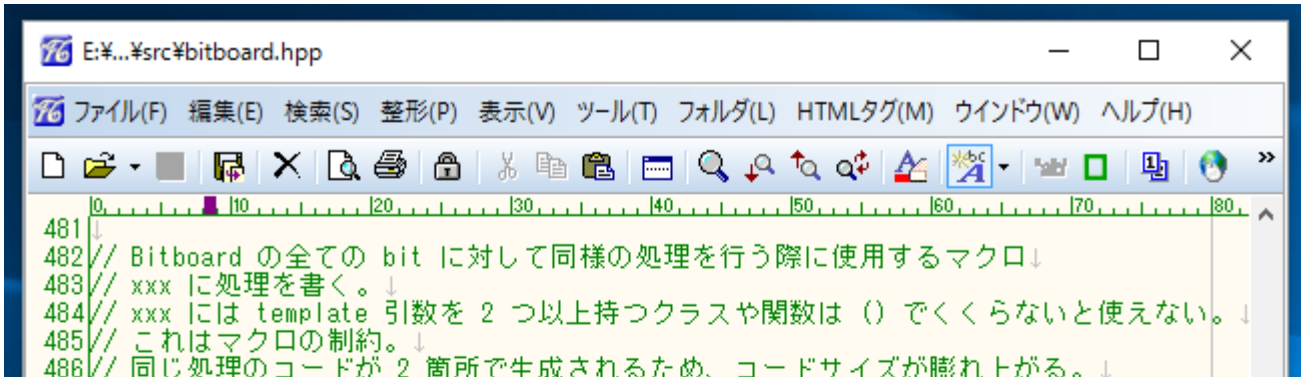
※飛２枚目の１５６まで
　合計１５７要素☆

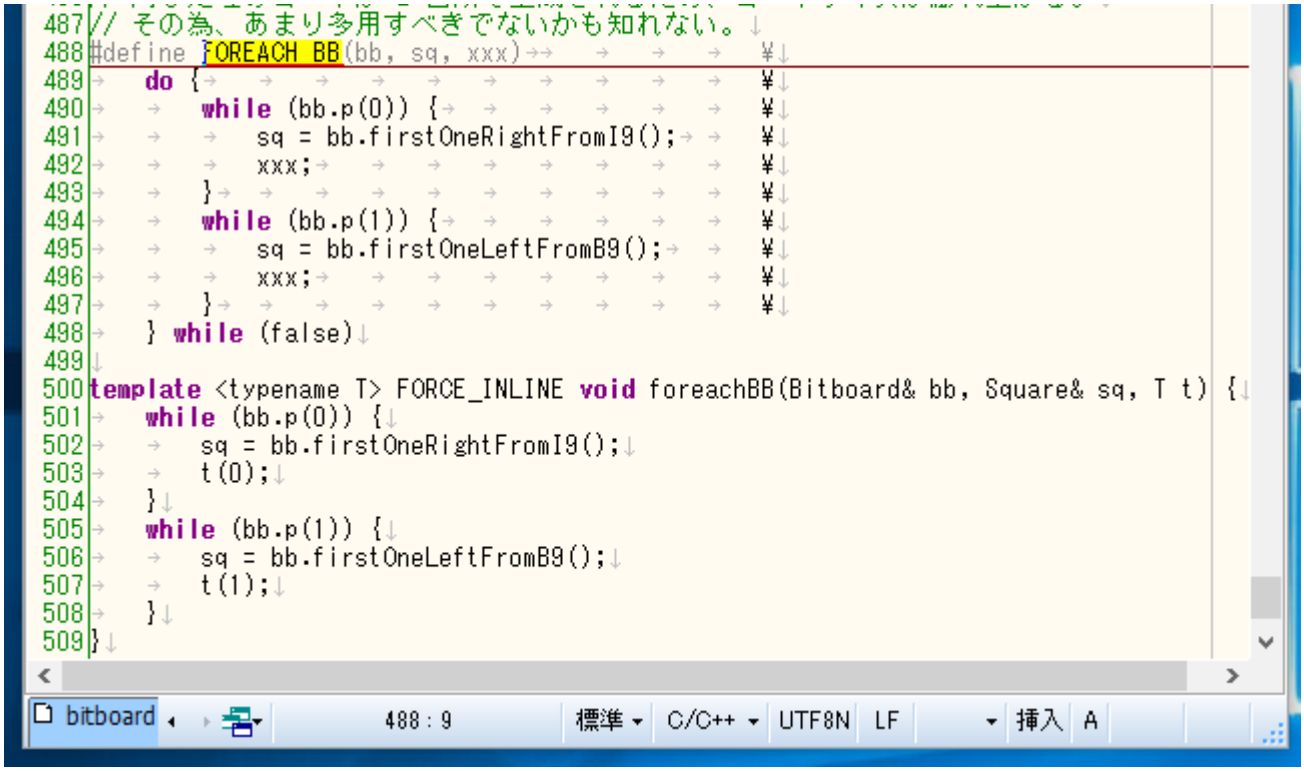
▲先手

飛	０枚目	116
角	０枚目	114
金	０枚目	110
銀	０枚目	106
桂	０枚目	102
香	０枚目	98
歩	０枚目	80

「 ＦＯＲＥＡＣＨ＿ＢＢ　マクロで　コード１つにしてくれていたのを
　分解しちゃったのね」

ｂｉｔｂｏａｒｄ．ｈｐｐ





```
487// その為、あまり多用すべきでないかも知れない。
488#define FOREACH_BB(bb, sq, xxx)                    ¥
489    do {                                            ¥
490        while (bb.p(0)) {                           ¥
491            sq = bb.firstOneRightFromI9();          ¥
492            xxx;                                    ¥
493        }                                           ¥
494        while (bb.p(1)) {                           ¥
495            sq = bb.firstOneLeftFromB9();           ¥
496            xxx;                                    ¥
497        }                                           ¥
498    } while (false)
499
500template <typename T> FORCE_INLINE void foreachBB(Bitboard& bb, Square& sq, T t) {
501    while (bb.p(0)) {
502        sq = bb.firstOneRightFromI9();
503        t(0);
504    }
505    while (bb.p(1)) {
506        sq = bb.firstOneLeftFromB9();
507        t(1);
508    }
509}
```

bitboard | 488 : 9 | 標準 | C/C++ | UTF8N | LF | 挿入 | A

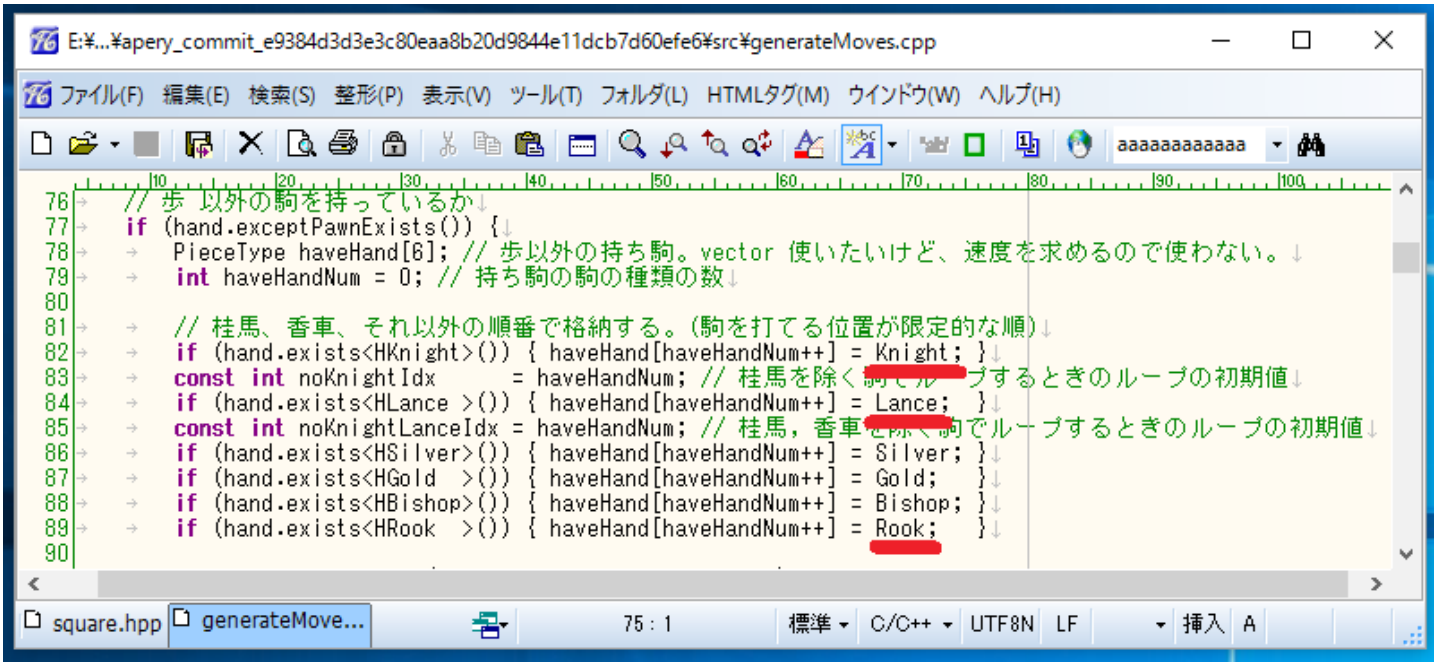
ジェネレート・ドロップ・ムーブスも　いじったぜ☆

POLYMORPHISM

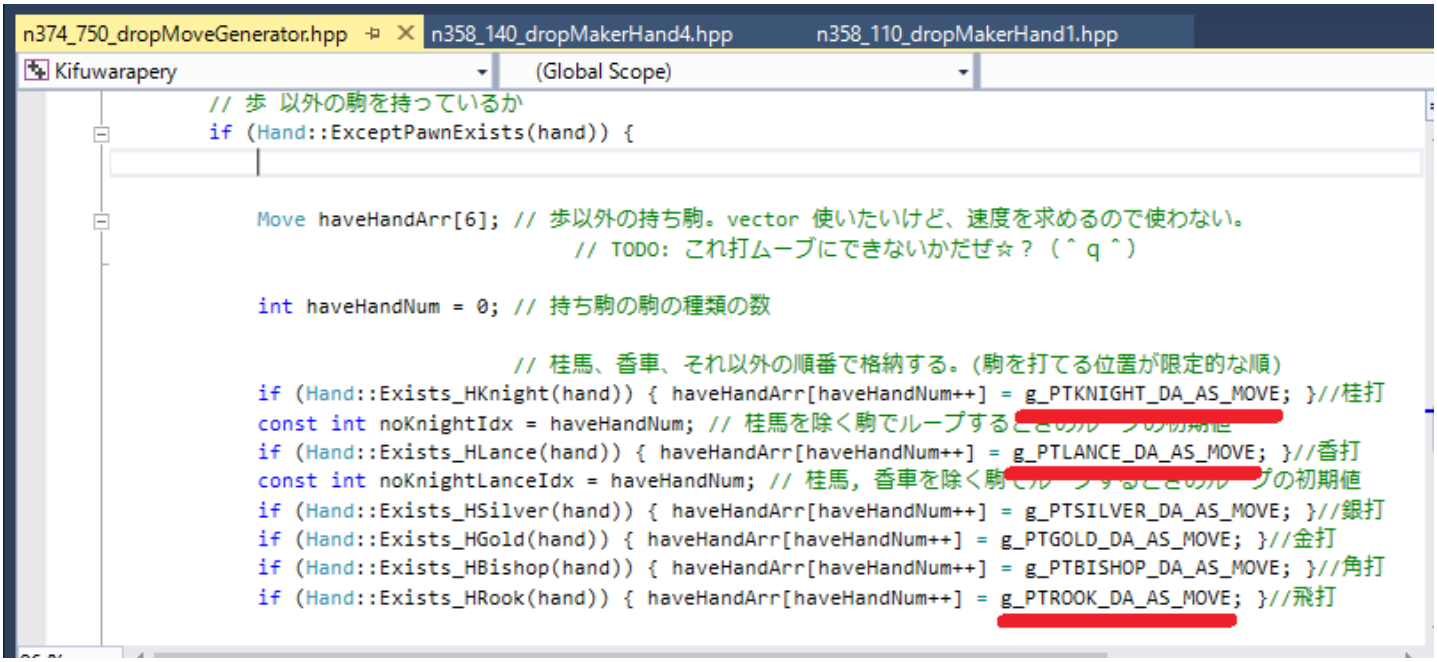
ポリモーフィズムは遅いのかなんだぜ☆（＾ｑ＾）？

「いじったのは　ここだぜ☆」

generateMoves.cpp



n374_750_dropMoveGenerator.hpp



「 駒の種類を　配列に入れるのではなく、
　持ち駒を　打つ　設定のカードを
　直接　入れておいたんだぜ☆」

「 打つ設定って、何だぜ～☆？」

「 移動元に　駒の種類を入れておくんだぜ☆
　絵にすると……」

「 7六飛打　のカードは、こういう感じになるぜ☆」

「 実行時間　〇　秒の　ビット演算を　〇　秒にするんでしょ？」

「 うむむ……☆
　他にも　いじっているぜ☆」

generateMoves.cpp

E:¥...¥src¥generateMoves.cpp

ファイル(F) 編集(E) 検索(S) 整形(P) 表示(V) ツール(T) フォルダ(L) HTMLタグ(M) ウインドウ(W) ヘルプ(H)

```
 96    Bitboard toBB;
 97    Square to;
 98    // 桂馬、香車 以外の持ち駒があれば、
 99    // 一段目に対して、桂馬、香車以外の指し手を生成。
100    switch (haveHandNum - noKnightLanceIdx) {
101    case 0: break; // 桂馬、香車 以外の持ち駒がない。
102    case 1: toBB = target & TRank9BB; FOREACH_BB(toBB, to, { Unroller<1>()([&](const int i) { (*mo
103    case 2: toBB = target & TRank9BB; FOREACH_BB(toBB, to, { Unroller<2>()([&](const int i) { (*mo
104    case 3: toBB = target & TRank9BB; FOREACH_BB(toBB, to, { Unroller<3>()([&](const int i) { (*mo
105    case 4: toBB = target & TRank9BB; FOREACH_BB(toBB, to, { Unroller<4>()([&](const int i) { (*mo
106    default: UNREACHABLE;
107    }
108
109    // 桂馬以外の持ち駒があれば、
110    // 二段目に対して、桂馬以外の指し手を生成。
111    switch (haveHandNum - noKnightIdx) {
112    case 0: break; // 桂馬 以外の持ち駒がない。
113    case 1: toBB = target & TRank8BB; FOREACH_BB(toBB, to, { Unroller<1>()([&](const int i) { (*mo
114    case 2: toBB = target & TRank8BB; FOREACH_BB(toBB, to, { Unroller<2>()([&](const int i) { (*mo
115    case 3: toBB = target & TRank8BB; FOREACH_BB(toBB, to, { Unroller<3>()([&](const int i) { (*mo
116    case 4: toBB = target & TRank8BB; FOREACH_BB(toBB, to, { Unroller<4>()([&](const int i) { (*mo
117    case 5: toBB = target & TRank8BB; FOREACH_BB(toBB, to, { Unroller<5>()([&](const int i) { (*mo
118    default: UNREACHABLE;
119    }
120
121    // 一、二段目以外に対して、全ての持ち駒の指し手を生成。
122    toBB = target & ~(TRank8BB | TRank9BB);
123    switch (haveHandNum) {
124    case 0: assert(false); break; // 最適化の為のダミー
125    case 1: FOREACH_BB(toBB, to, { Unroller<1>()([&](const int i) { (*moveStackList++).move = make
126    case 2: FOREACH_BB(toBB, to, { Unroller<2>()([&](const int i) { (*moveStackList++).move = make
127    case 3: FOREACH_BB(toBB, to, { Unroller<3>()([&](const int i) { (*moveStackList++).move = make
128    case 4: FOREACH_BB(toBB, to, { Unroller<4>()([&](const int i) { (*moveStackList++).move = make
129    case 5: FOREACH_BB(toBB, to, { Unroller<5>()([&](const int i) { (*moveStackList++).move = make
130    case 6: FOREACH_BB(toBB, to, { Unroller<6>()([&](const int i) { (*moveStackList++).move = make
131    default: UNREACHABLE;
132    }
```

generateMove... 82 : 43 標準 C/C++ UTF8N LF 挿入 A

「 持ち駒の　種類の数に応じて　ループの数を変えていた
　持ち駒の　指し手生成　の個所なんだが……☆」

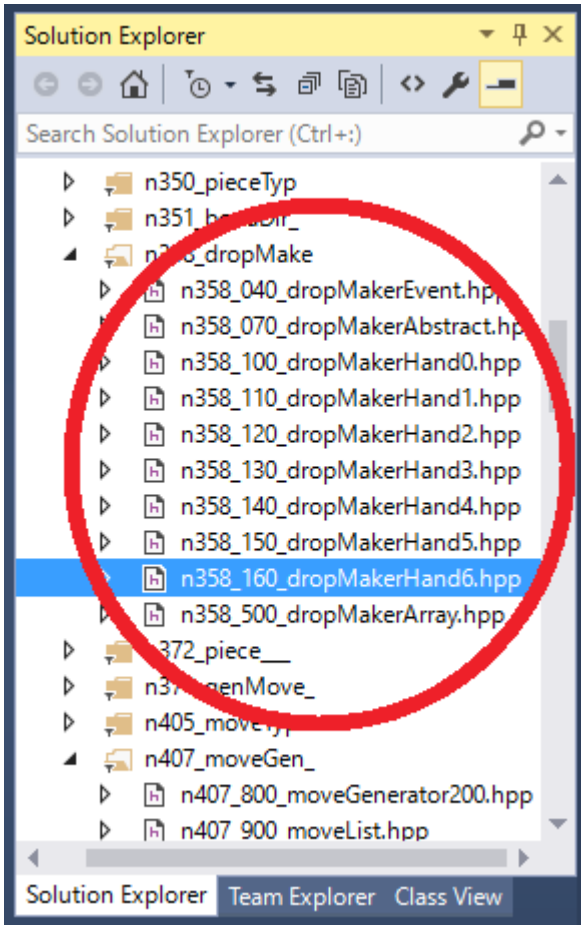
n374_750_dropMoveGenerator.hpp

n374_750_dropMoveGenerator.hpp

Kifuwarapery (Global Scope)

```
// 桂馬、香車 以外の持ち駒があれば、
// 一段目に対して、桂馬、香車以外の指し手を生成。
// FIXME: 配列の範囲チェックしてないぜ☆（＾ｑ＾）
pMovestack = this->m_pDropMakerArray[haveHandNum - noKnightLanceIdx]->MakeDropMovesToRank9ExceptNL(
    pMovestack,
    dmEvent,
    haveHandArr // TODO: これ打ムーブにできないかだぜ☆？（＾ｑ＾）
    );

// 桂馬以外の持ち駒があれば、
// 二段目に対して、桂馬以外の指し手を生成。
pMovestack = this->m_pDropMakerArray[haveHandNum - noKnightIdx]->MakeDropMovesToRank8ExceptN(
    pMovestack,
    dmEvent,
    haveHandArr // TODO: これ打ムーブにできないかだぜ☆？（＾ｑ＾）
    );

// 一、二段目以外に対して、全ての持ち駒の指し手を生成。
pMovestack = this->m_pDropMakerArray[haveHandNum]->MakeDropMovesToRank1234567(
    pMovestack,
    dmEvent,
    haveHandArr // TODO: これ打ムーブにできないかだぜ☆？（＾ｑ＾）
    );
```

「 配列に押し込んだぜ☆」



「 歩以外の　持ち駒の種類の数　0～6　に応じて
　実行するクラスが選ばれて……☆」

n374_750_dropMoveGenerator.hpp

n358_140_dropMakerHand4.hpp　n374_750_dropMoveGenerator.hpp

Kifuwarapery　(Global Scope)

```
// 桂馬、香車 以外の持ち駒があれば、
// 一段目に対して、桂馬、香車以外の指し手を生成。
// FIXME: 配列の範囲チェックしてないぜ☆（＾ｑ＾）
pMovestack = this->m_pDropMakerArray[haveHandNum - noKnightLanceIdx]->MakeDropMovesToRank9ExceptNL(
    pMovestack,
    dmEvent,
    haveHandArr // TODO: これ打ムーブにできないかだぜ☆？（＾ｑ＾）
    );

// 桂馬以外の持ち駒があれば、
// 二段目に対して、桂馬以外の指し手を生成。
pMovestack = this->m_pDropMakerArray[haveHandNum - noKnightIdx]->MakeDropMovesToRank8ExceptN(
    pMovestack,
    dmEvent,
    haveHandArr // TODO: これ打ムーブにできないかだぜ☆？（＾ｑ＾）
    );

// 一、二段目以外に対して、全ての持ち駒の指し手を生成。
pMovestack = this->m_pDropMakerArray[haveHandNum]->MakeDropMovesToRank1234567(
    pMovestack,
    dmEvent,
    haveHandArr // TODO: これ打ムーブにできないかだぜ☆？（＾ｑ＾）
    );
```

「 ポリモーフィズムを利用して、同じ呼び出し方で
　異なるクラスのメソッドが　呼び出されるんだぜ☆」

「 ポインターを使って　呼び出す関数を変えていたら
　インライン展開できる関数呼び出しより　遅くて、

　ポリモーフィズムは　グローバル関数呼び出しより　遅いんじゃないの？」

「 ……☆（＾ｑ＾）
　……☆（＾ｑ＾）！
　……☆（＾ｑ＾）！！」

ムーブ・フェーズは　エイプリー＿大樹の枝と同じ☆

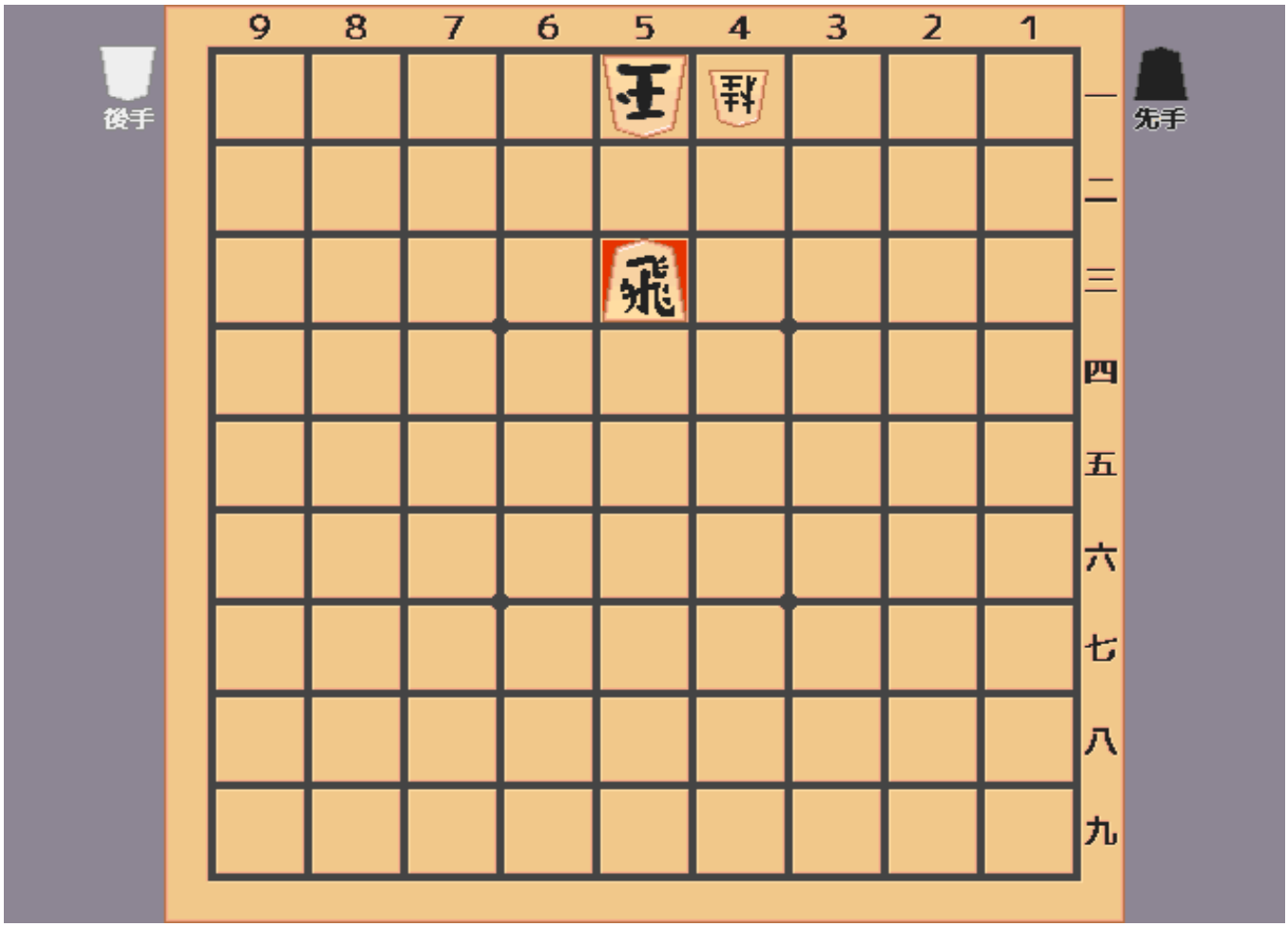
WAKE　WAKE

分からないところは、分け分け　しようぜ☆

「 ムーブ・フェーズも　分け分け　しただけで　いじってないぜ☆」

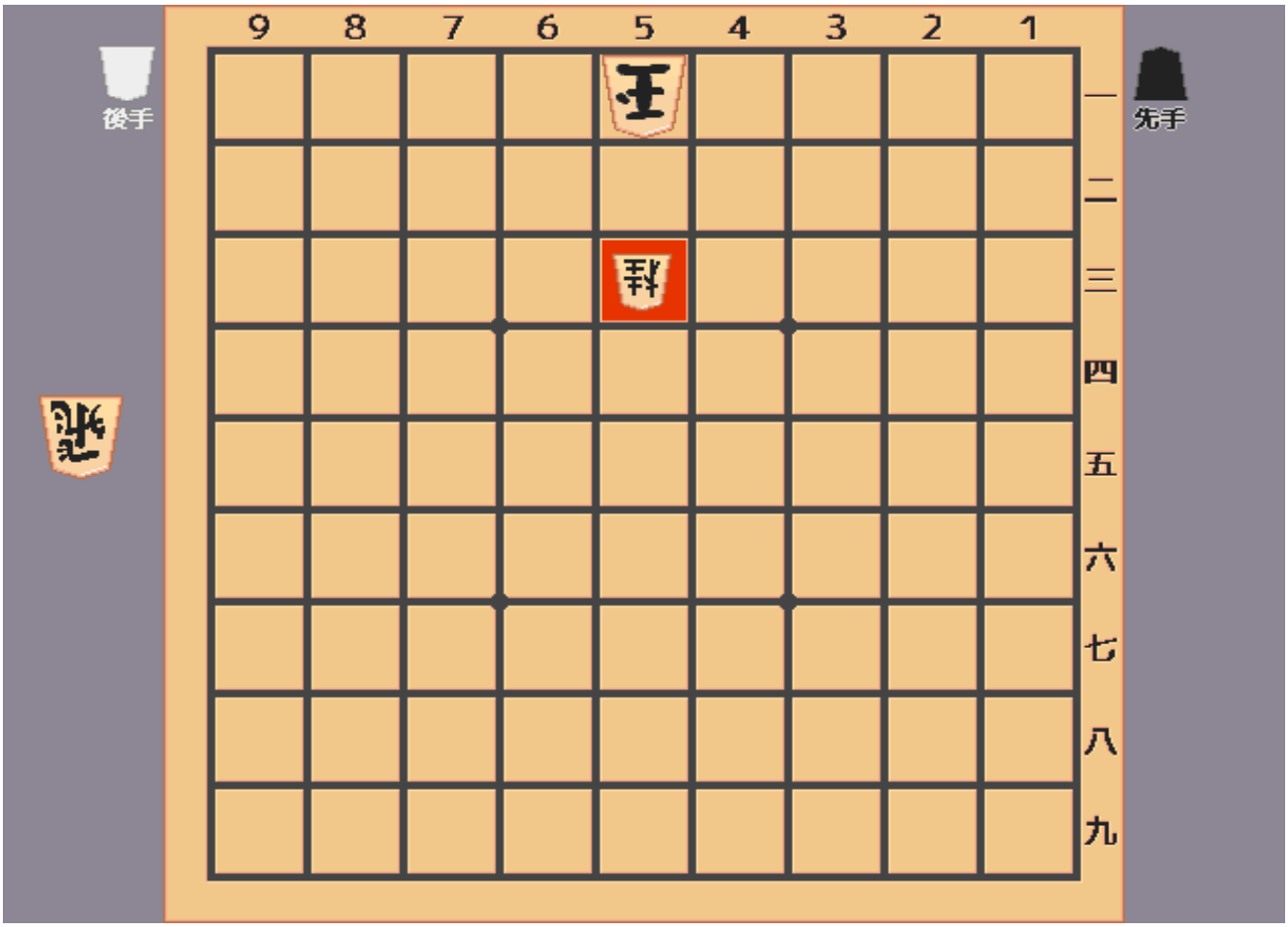
「 ムーブ・フェーズって何？」

「 理屈の前に、やってみようぜ☆」



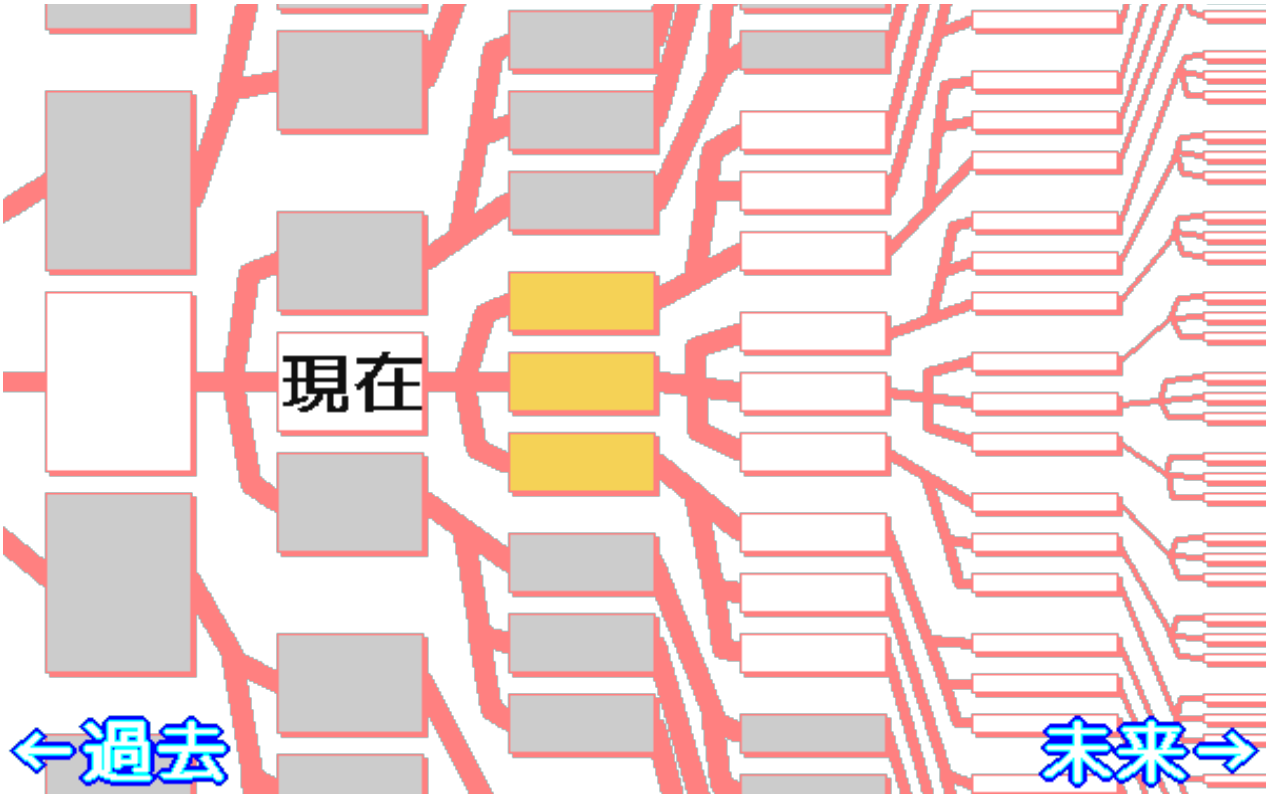
「 王手☆！」

「 ↓こうだろ☆」



「 そうそう……☆
王手されたからといって　玉が逃げることを　考えなくても、
駒を取る手から　優先して　考えていれば　王手は自然解消する……☆（＾ｑ＾）」

「 コンピューター将棋は　**数の暴力**　で
全部の手を　読んでるんじゃないの？」

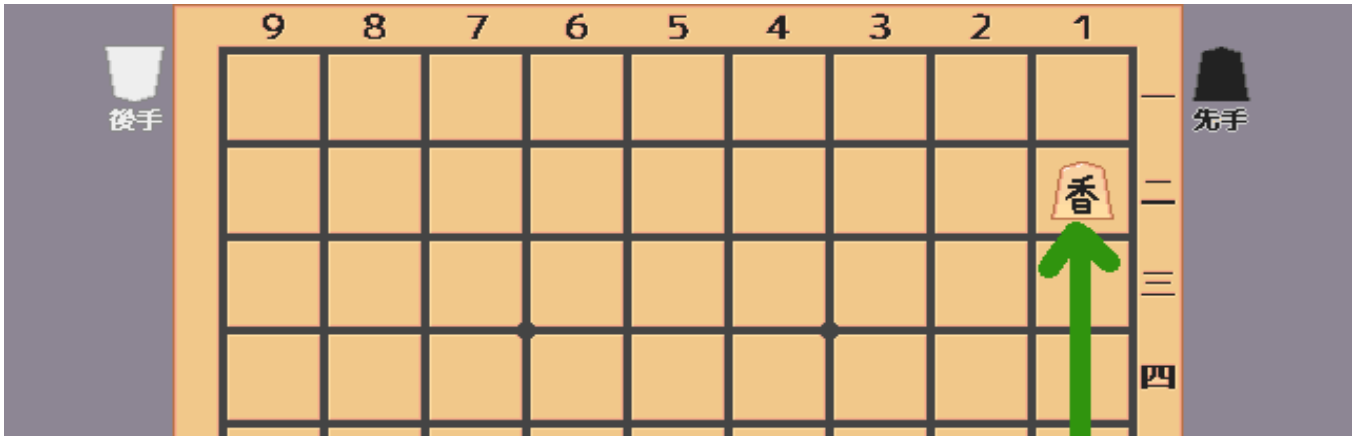


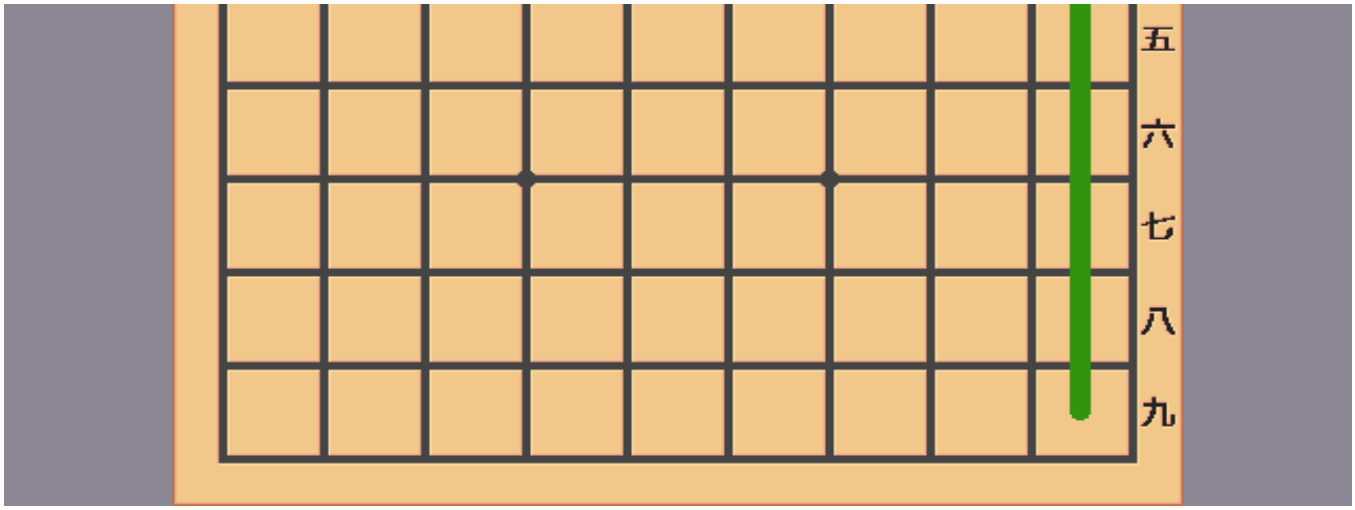
「 全部……、っていうのは　リーガル・ムーブと、スイード・リーガルを全部か、という話しだな☆」

ムーブタイプ

リーガル・オール	リーガル・ムーブ	スイード・リーガル

全部の合法手を読む場合（リーガル・オール）

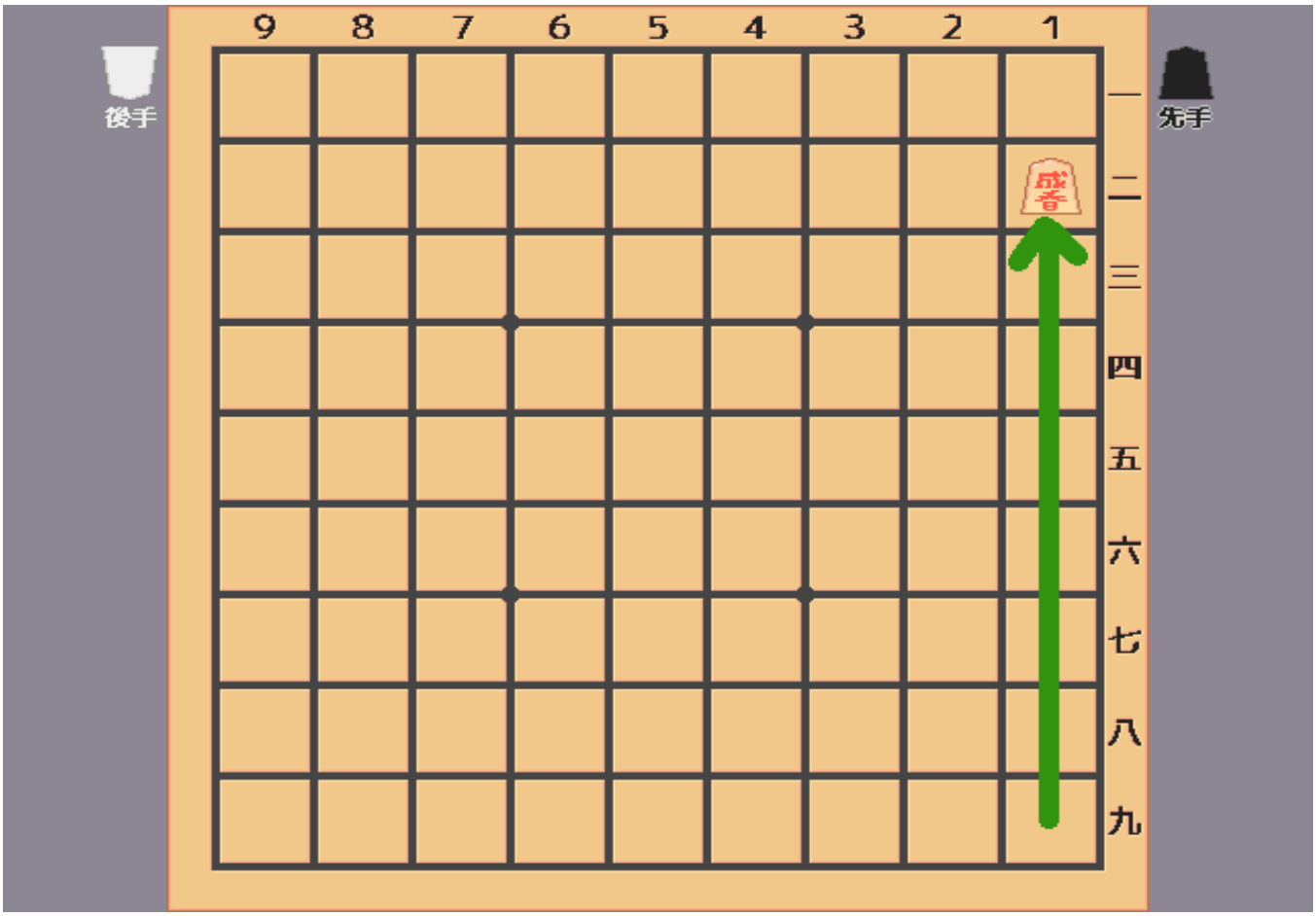




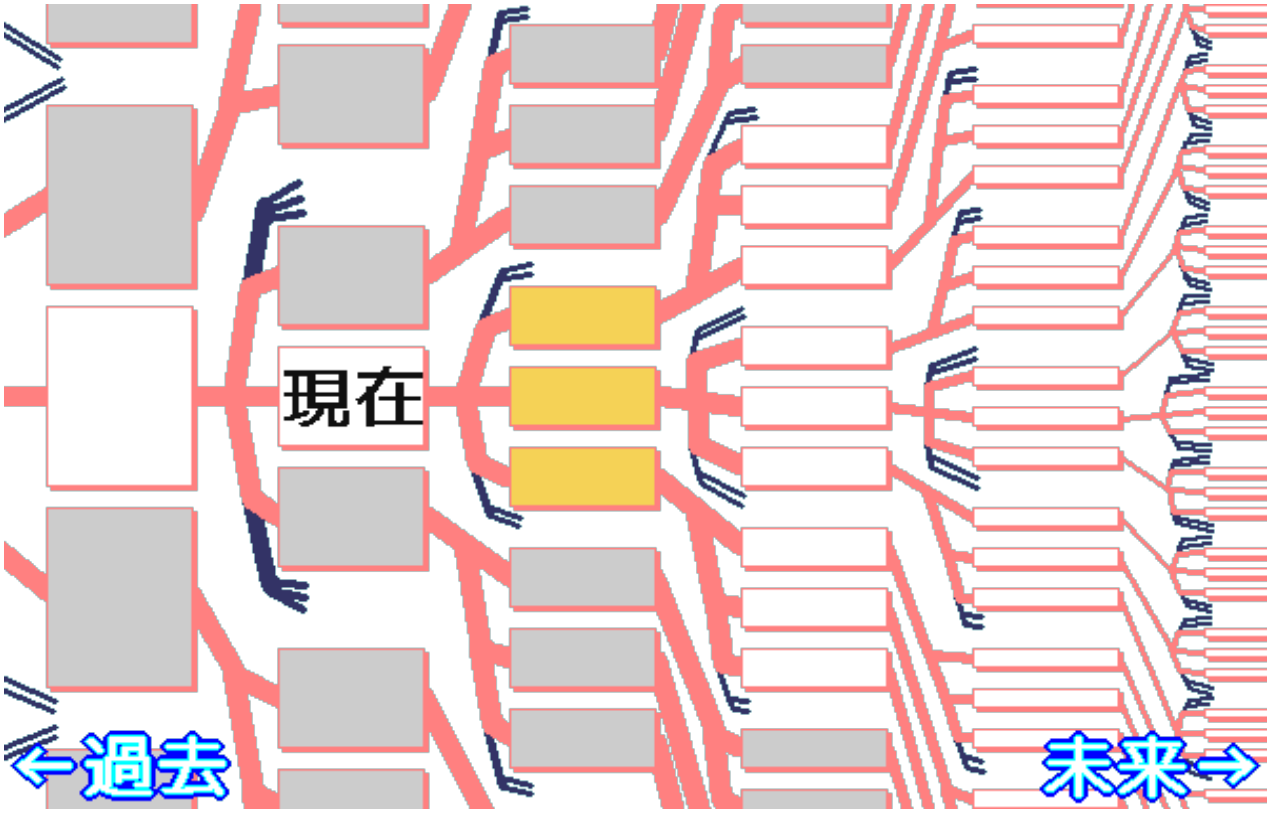
「リーガル・オールなんか　学習と　動作デバッグ　でしか見かけないぜ☆
学習用に拾ってきた棋譜には　▲１二香成らず　とか入ってるかもしれないからな☆

エイプリー＿大樹の枝　で　リーガル・ムーブ　と言えば
１二　に▲香が行ったら、成らず　は無い☆　必ず成るんだぜ☆」

いいとこ取りの合法手だけ読む場合（リーガル・ムーブ）



「リーガル・オールの　ニュアンスを
図に　**紺色の線**　で書き入れるぜ☆」



「リーガル・オールを　無かったことにすれば、
エイプリー＿大樹の枝も　数の暴力で　全部の手を
読もうとすることに変わりはないぜ☆　だが……☆」

借りてきた映画を見ようぜ☆

MOVE PHASE

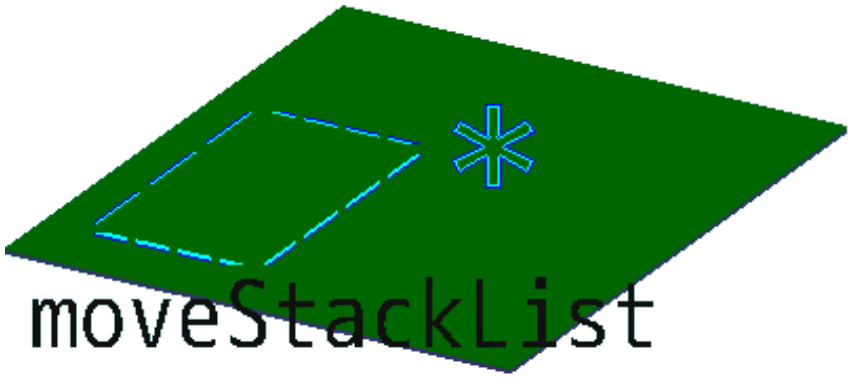
ムーブ・フェイズは　ビデオデッキに似ている☆

「この映画、２時間あるのか……☆

２時間のくせに
１７部構成　らしいんだぜ☆」

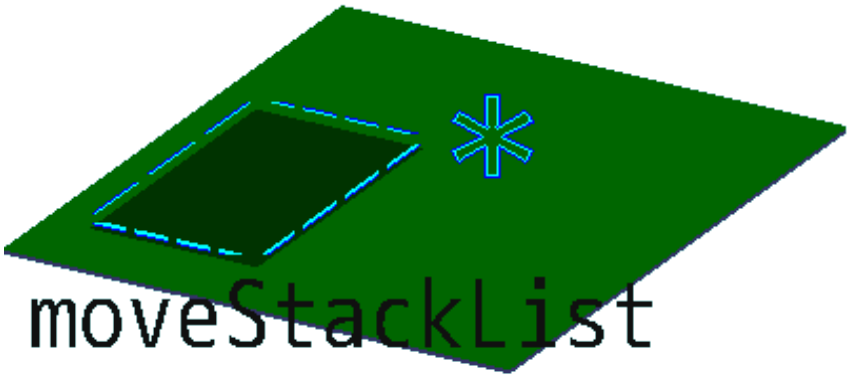
第〇部

電源を入れる☆



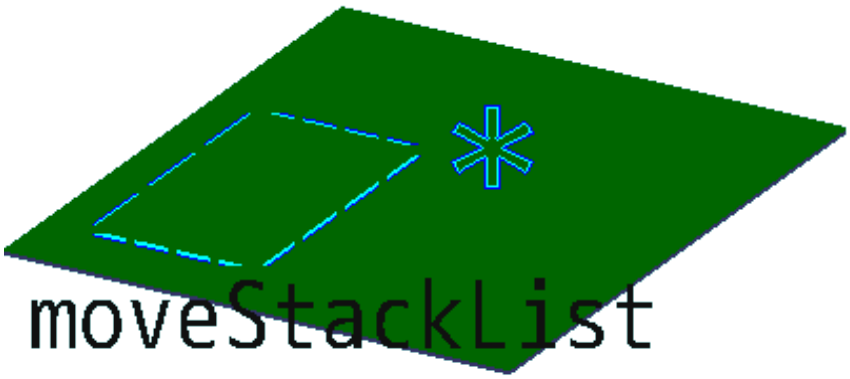
「データのダウンロードが遅いぜ☆」

ピョコッ☆



「 うん☆？」

ペタンッ☆



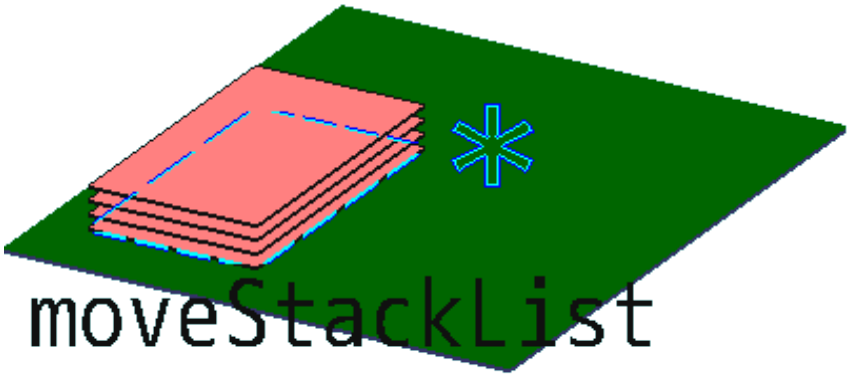
「 なんだぜ、今の☆？」

「 ビデオデッキに　過去に見た　一番面白かったシーンが残っていれば
出そうとしてくれたみたいだぜ☆」

「 要らない機能だぜ☆」

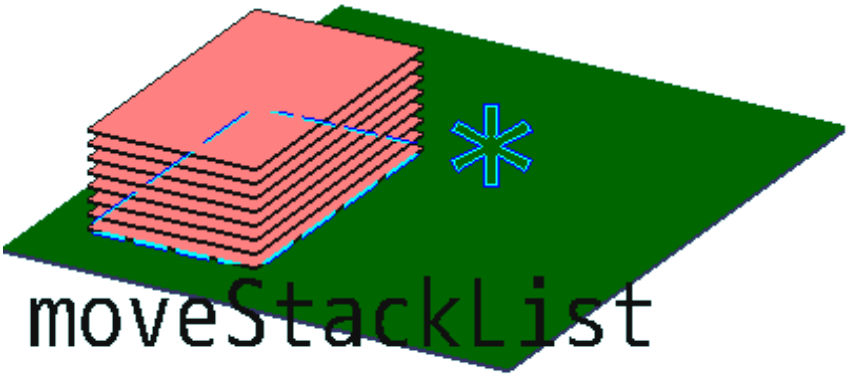
第１部

指し手生成中……☆



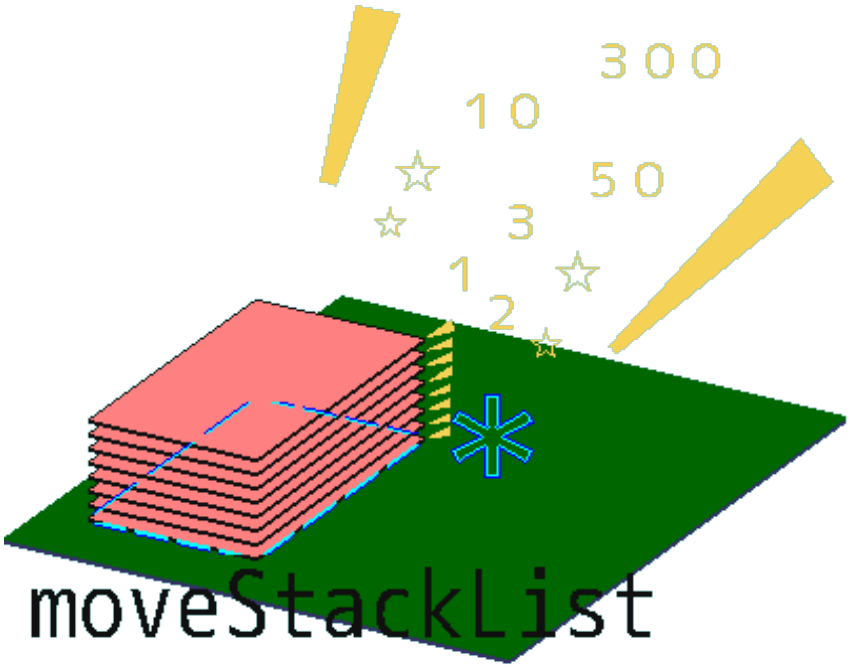
「 おっ　読み込みが始まったぜ☆」

指し手生成中……☆



「 まだか……☆
なんか　第１部　駒を取る手編　をダウンロードしているようだぜ☆
ファイル名がネタバレになっているな……☆」

ピコーン☆



「 何だぜ☆？」

「 この映画の冒頭１０分は
　駒を取る手の　人気シーンのダイジェストになっていて、

　飛車を取る香のシーンが　９８８点　の人気を獲得しているぜ☆」

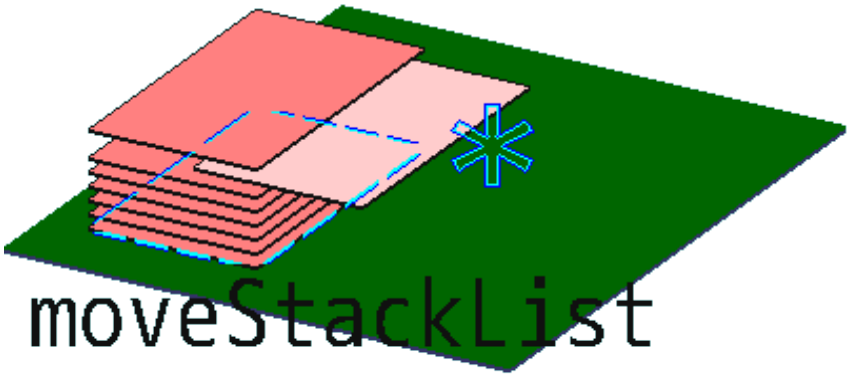
「 要らね☆」

「 ちなみに
　歩が　飛車を取ったシーンは　９８９点だぜ☆」

「 どれが　取るかで　点数が違うのかだぜ☆？」

「 １点ずつ違うぜ☆　高い駒で　取るほど１点低いぜ☆
　玉で取る手は　－１００００点だぜ☆」

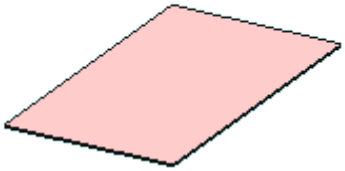
ウィィィン……

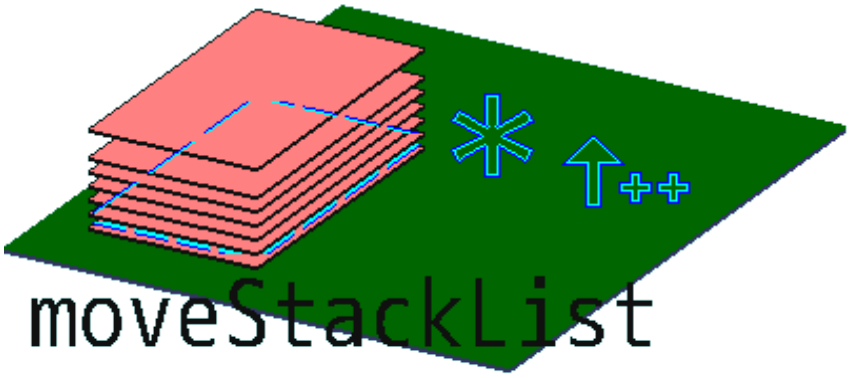


「 おっ☆　駒を取ったシーンが　抜かれているぜ☆？」

「 ベストシーンだぜ☆」

ウィィィン……



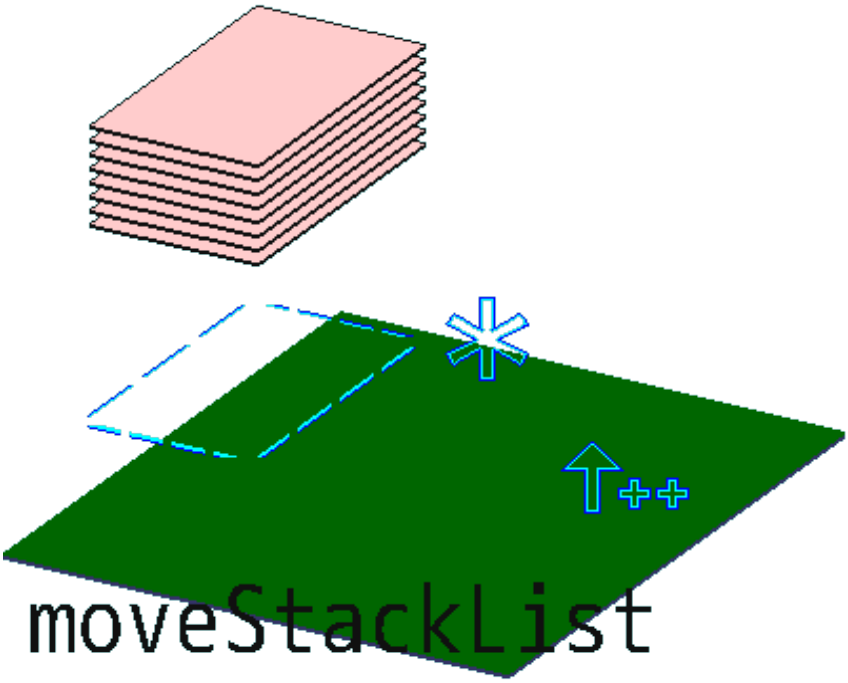


「 あれっ☆？　ベストシーンが　上に行っちゃったぜ☆？」

「 お見せするほどのシーンでは　無かったようだななんだぜ☆」

「 ベストシーンが　お見せするほどのシーンじゃ　無かったんなら……！」

ウィィィン……



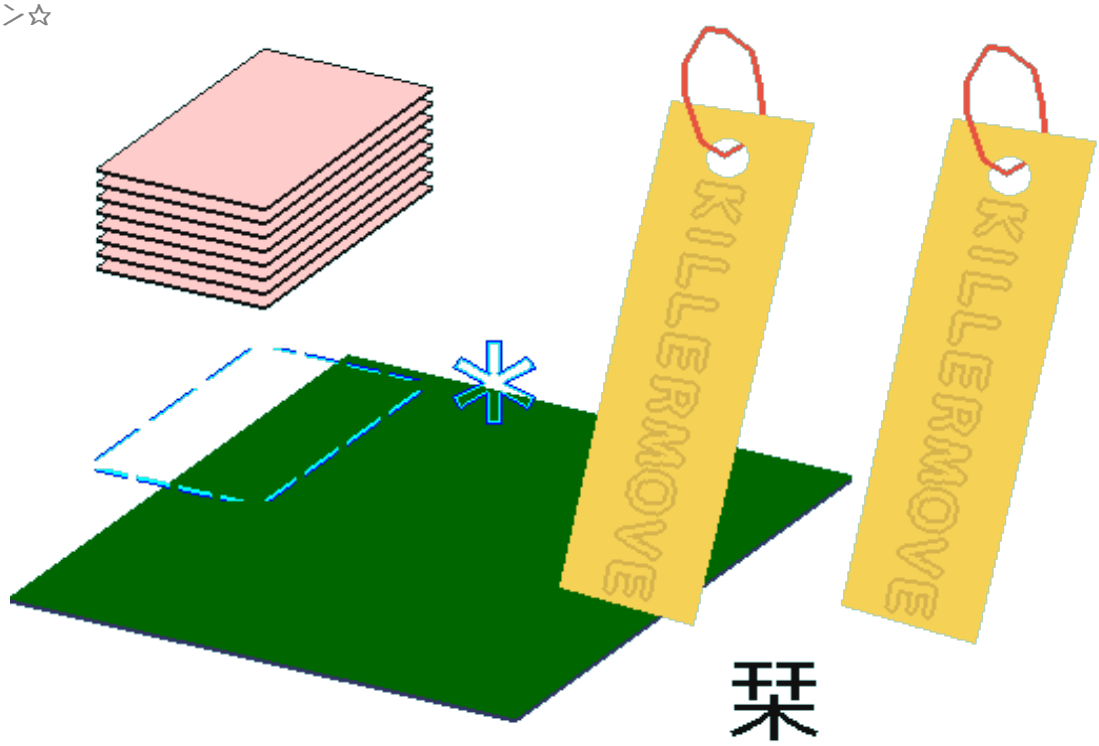
「 ベストシーンは全部　上　行っちゃった！」

「 わたしは　いつになったら　映画を見れるんだぜ☆？」

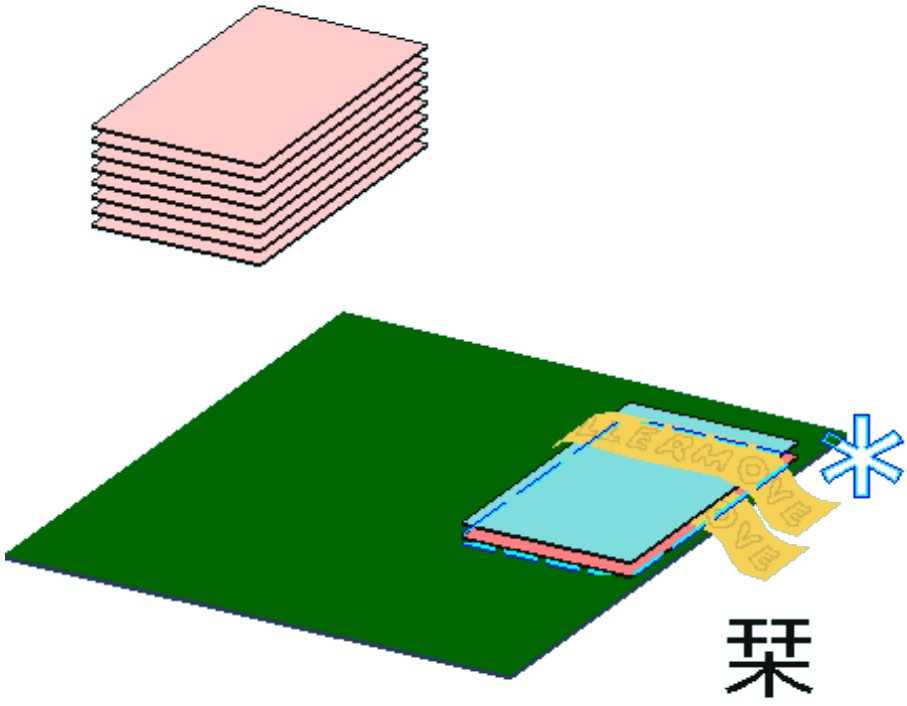
「 安心しろだぜ☆」

第２部

ピコーン☆



「 前に　この映画を見ていた人が
　栞（しおり）　を　挟んでいたんだぜ☆」



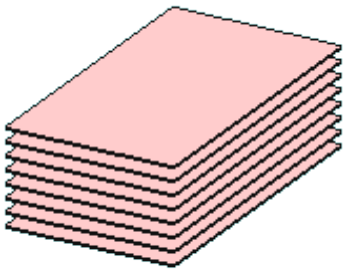
「 おススメのシーンから見ようぜ☆」

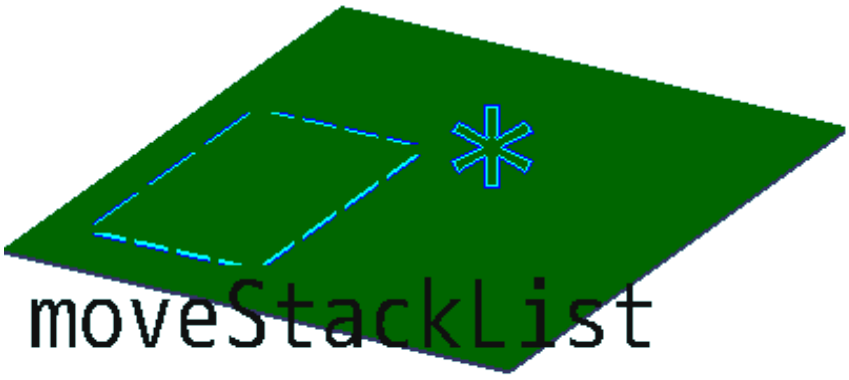
「 わたしは　先頭から見たいぜ☆」

「 ……☆！」

第３部

ピコーン☆

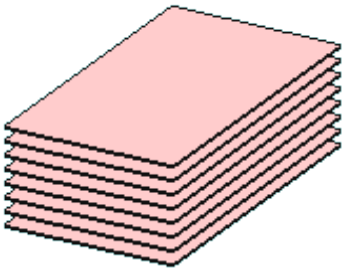


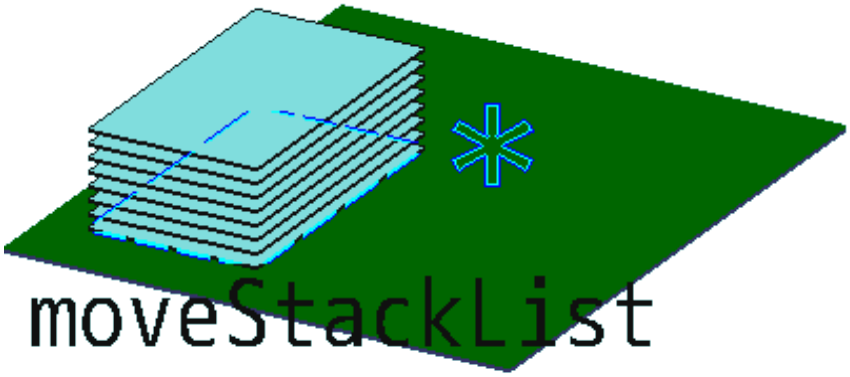


「 カーソルが　スタートに戻っちゃったんだけど！」

「 これが　ムーブ・フェーズ　だぜ☆
　続きから続けたり、初めに戻ったりを繰り返して
　少しずつ　山札を　積み足しながら　繰っていくぜ☆」

指し手生成中……☆



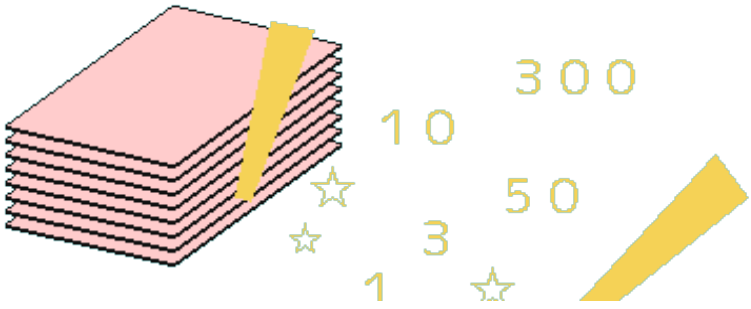


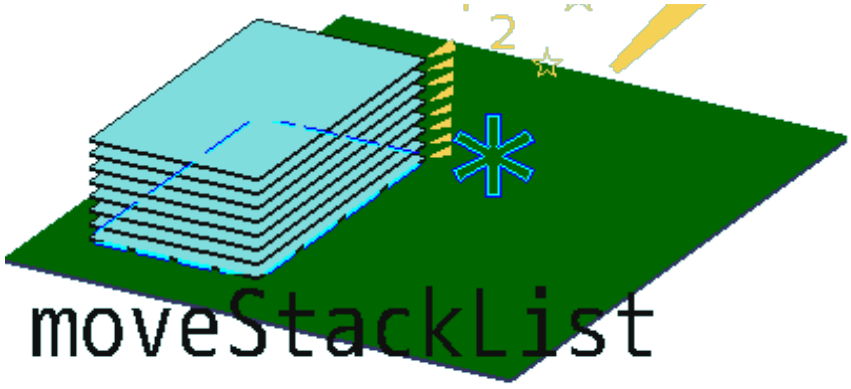
「 読み込みが始まったぜ☆

　なんか　第３部　駒を取らない手編（前編・盤上の手）

　をダウンロードしているようだぜ☆
　ファイル名がネタバレになっているな……☆」

ピコーン☆

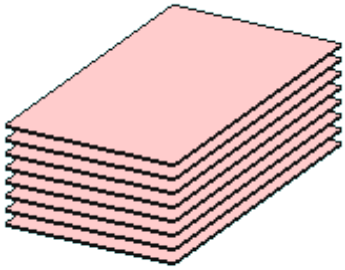


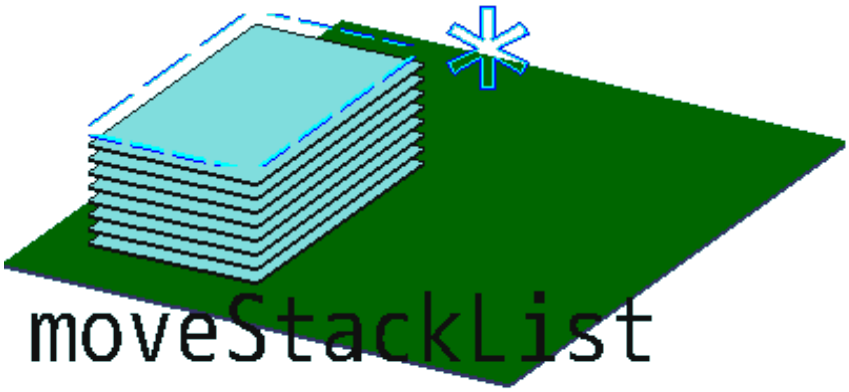


「 駒を取らない手（盤上の手） に 人気が付くのかだぜ☆？」

「 配列に累計値が 入っているようなんだが、詳細は調べてないぜ☆」

ピコーン☆

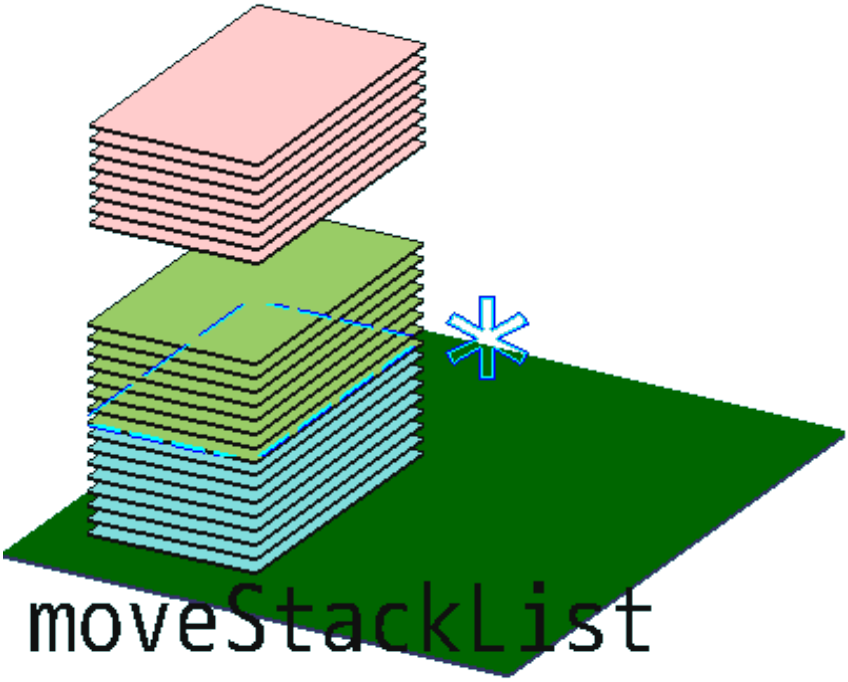




「 このビデオデッキ、まだ映画が始まらないぜ☆
シーク・バーが 読み込んだ山をすっとばして 先に進んでしまったぜ☆？」

「 まだ　読み込み中だぜ☆」

指し手生成中……☆

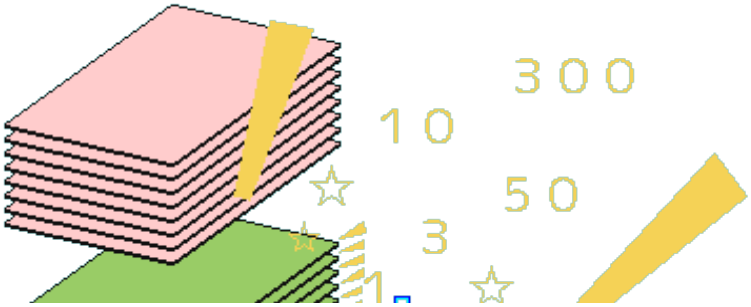


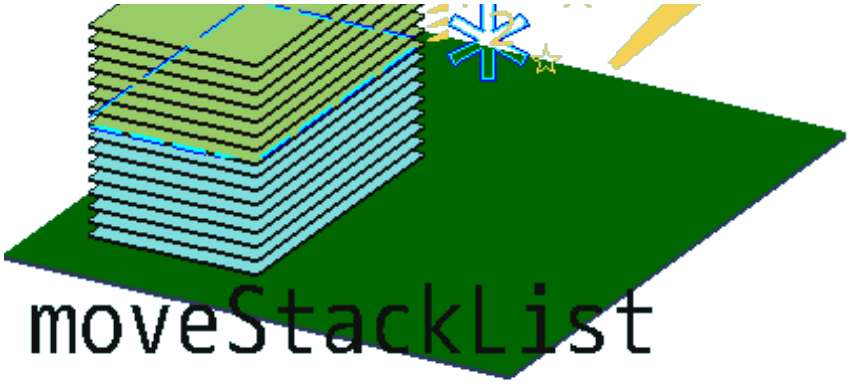
「 まだか……☆

なんか　第３部　駒を取らない手編（後編・持ち駒の手）

をダウンロードしているようだぜ☆
ファイル名がネタバレになっているな……☆」

ピコーン☆

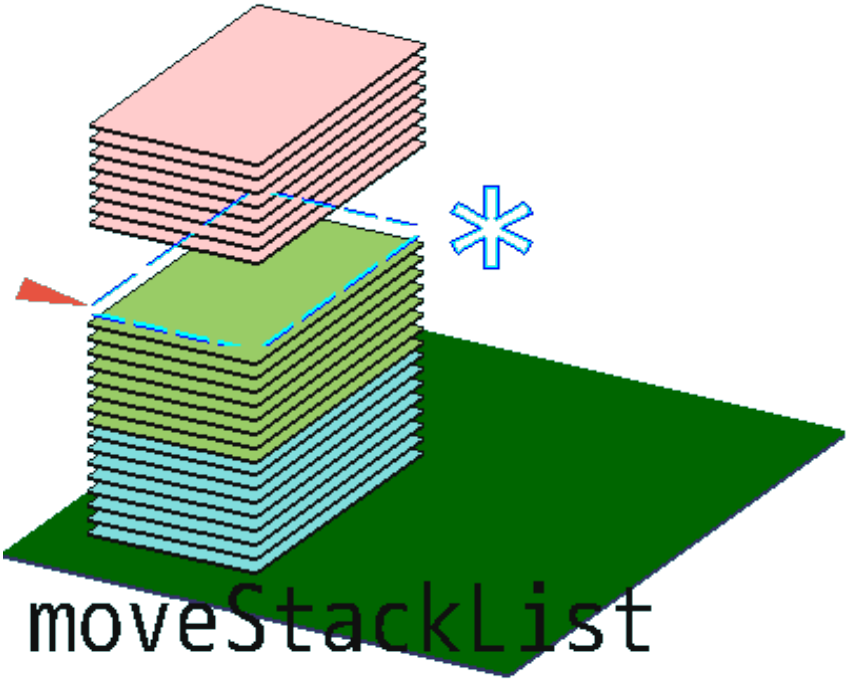




「 駒を取らない手（持ち駒を打つ手） に 人気が付くのかだぜ☆？」

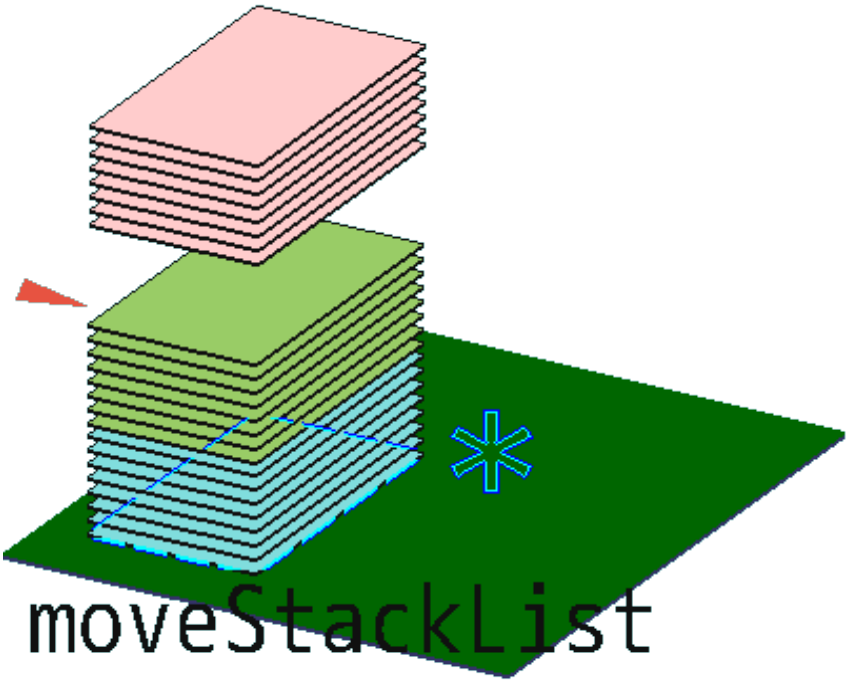
「 駒を取らない手（盤上の手）と同じルーチンで
配列に累計値が 入っているようなんだが、詳細は調べてないぜ☆」

ピコーン☆



「 また シーク が進んでしまったぜ☆
なんか マーク も付いているし☆」

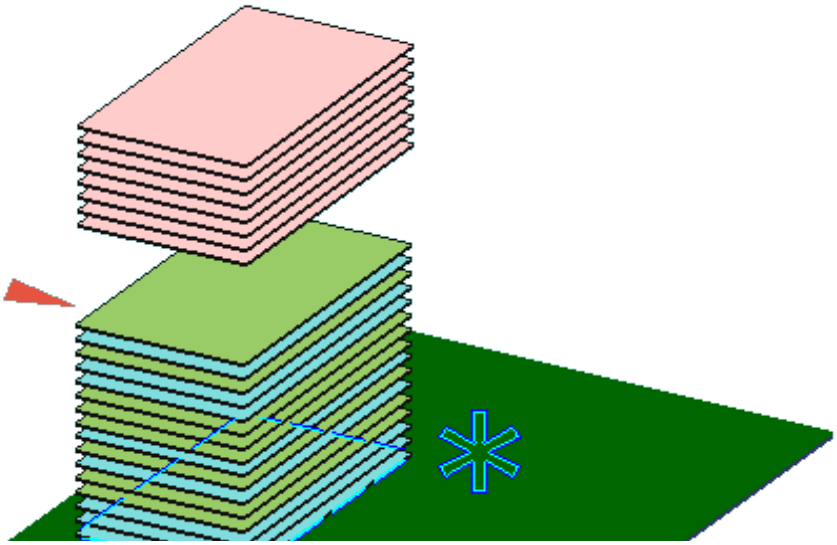
ピコーン☆



「と、思ったら　シーク　がまた　先頭に戻ったぜ☆？」

「進んだり、戻ったりして、だんだんと　繰っていくのが
ムーブ・フェーズなんだぜ☆」

ウィィィン……

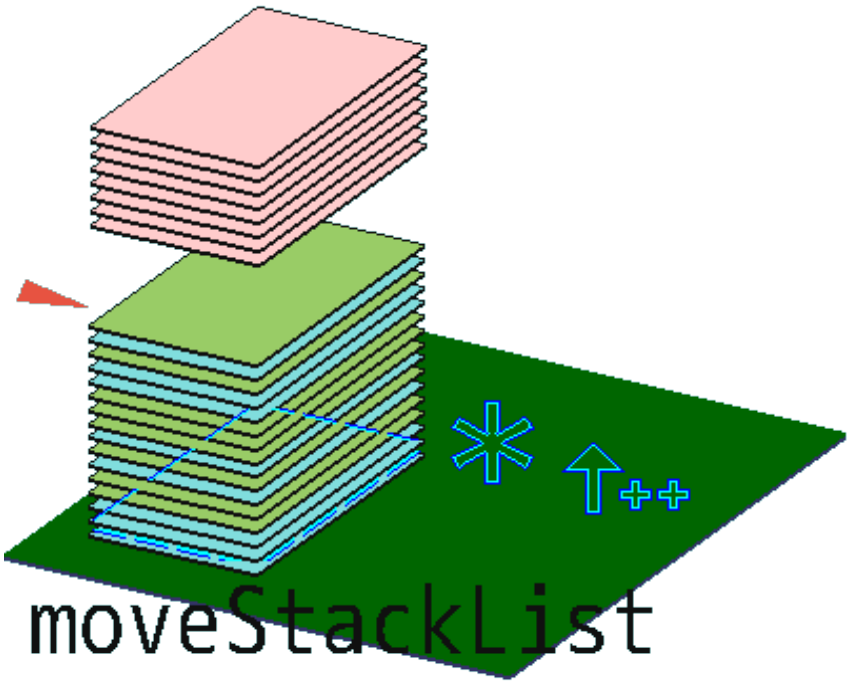


「 ☆！？　何が起こったんだぜ☆！？」

「 もちつけ☆
　シーク・バー　と、　マーカーの間で、人気順にソートされたんだぜ☆」

「 要らない機能だぜ……☆」
　わたしは　映画　を見たいんだぜ☆」

ピコーン☆



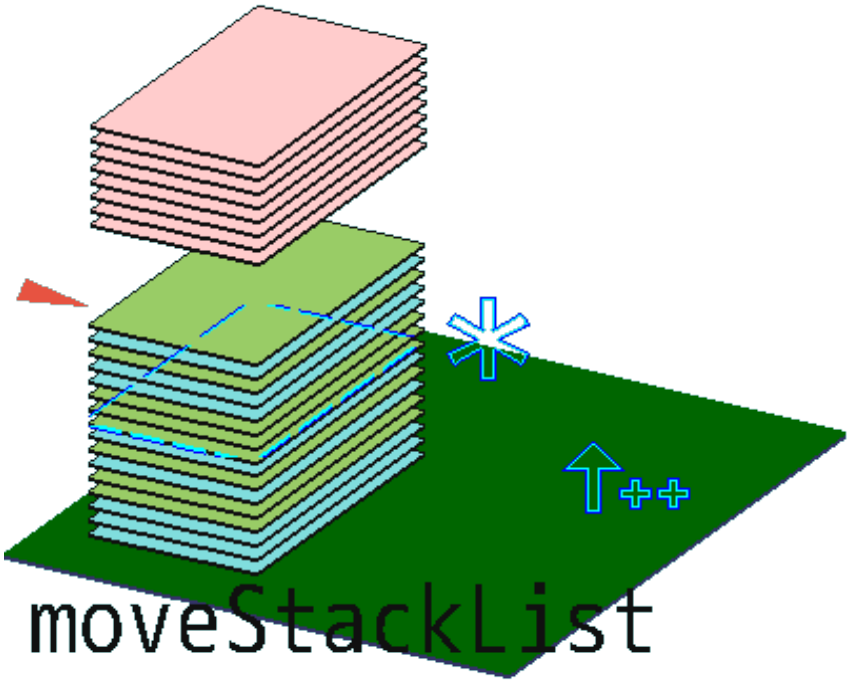
「 あっ☆！　当然　映画が始まったぜ☆！」

「 良かったな☆」

「 開始早々　ヒロインが爆発したわよ？」

「 世界も平和になっているしな☆」

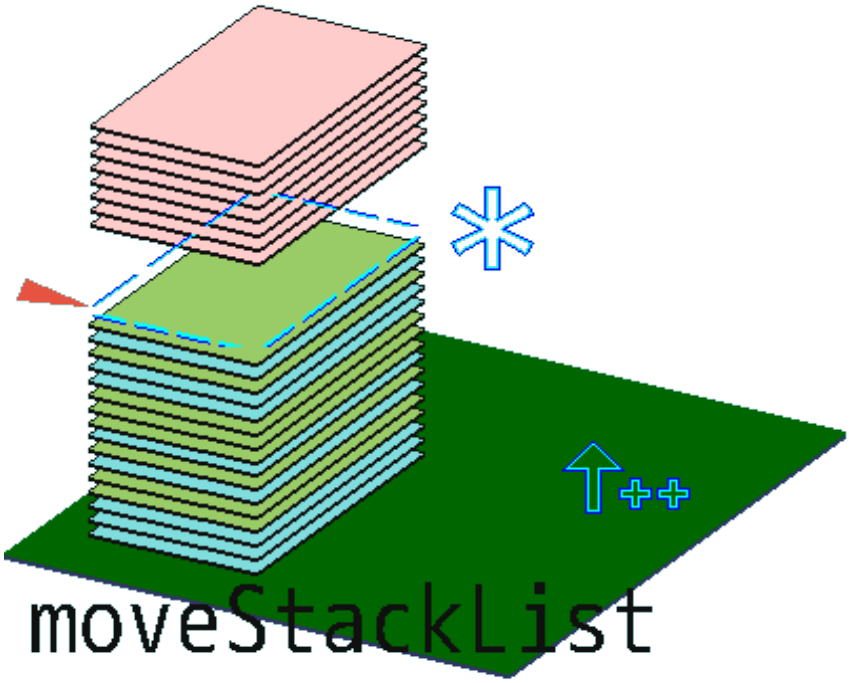
ピコーン☆



「 ジェットコースター的　展開だぜ☆！」

「 この映画は　ソートされているからな☆」

ピコーン☆

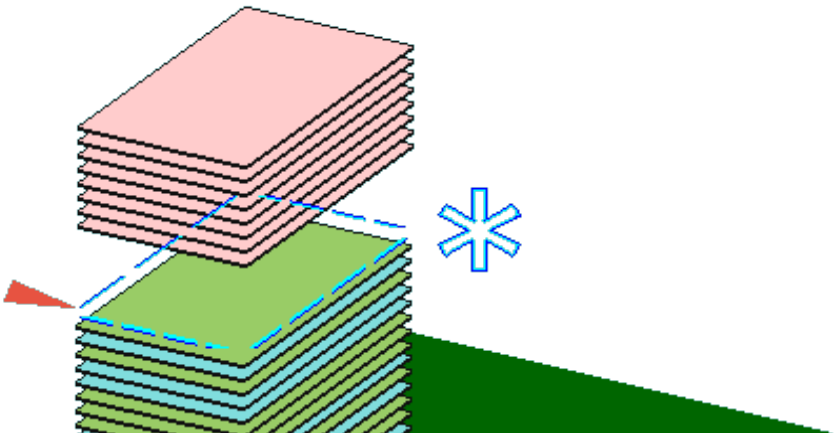


「 あ、終わったぜ☆」

「 すぐに　次の部が始まるぜ☆」

第４部

ピピッ☆！



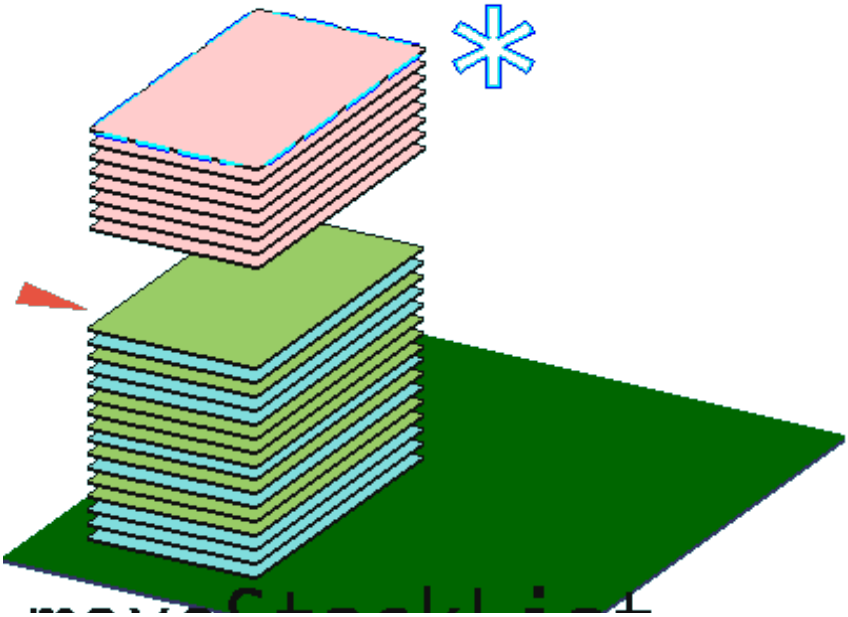
「……☆？」

第５部

「第４部が　何もなく終わってしまったぜ☆？」

「そういうこともあるのだろう☆」

ピコーン☆



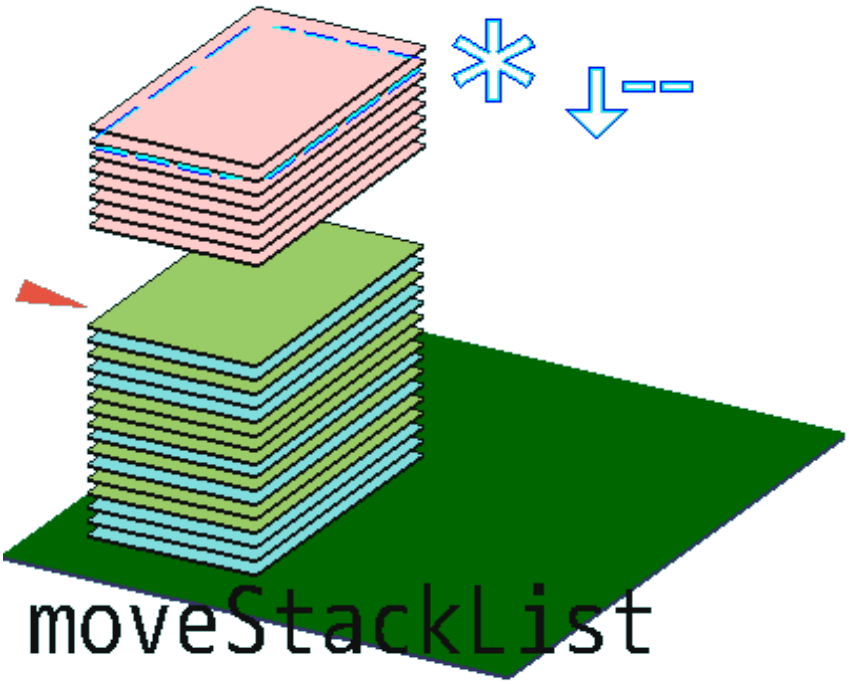
「 シークバー　が　おけつ　に行ってしまったぜ☆？」

「 しっぽの　頭だぜ☆」

「 貧乏削りの鉛筆みたいよね。
　これは　素敵じゃないわね」

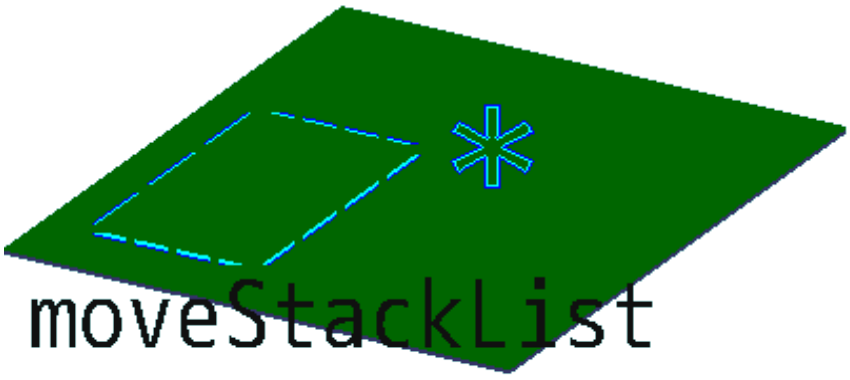
「 ひどいチャンバラが　始まったぜ☆
　武士が　頭突きをしているぜ☆」

ピコーン☆



「 こんどは　オケツ　を頭にして　始まったぜ☆」

「 これって　どんどん　ひどい　駒のたたき合いの順に
　並んでいるんじゃないの？」



「 あっ　オワタ☆」

「 この映画のタイトルは　武士の頭突き　でいいぐらいね」

「 じゃあ　次の映画は　『　武士の頭突き　』　をキーワードに
　借りてこようぜ☆？」

「 え～～～～っ☆！」

分け分けしただけ☆

SIKUMI

ＰＲ文書を書いていると、仕組みが分かってくるな☆

「 で、さっきも　言った通り
　ムーブ・フェーズは　変えてないぜ☆
　分け分け　しただけ☆」

「 エイプリー＿大樹の枝の　ソースコードでは、
　このビデオデッキは
　ムーブ・ピッカーという名前で　まとめられているぜ☆」

movePicker.cpp

E:¥...¥src¥movePicker.cpp

ファイル(F) 編集(E) 検索(S) 整形(P) 表示(V) ツール(T) フォルダ(L) HTMLタグ(M) ウインドウ(W) ヘルプ(H)

```
78 template <> Move MovePicker::nextMove<false>() {
79     MoveStack* ms;
80     Move move;
81     do {
82         // lastMove() に達したら次の phase に移る。
83         while (currMove() == lastMove()) {
84             goNextPhase();
85         }
86
87         switch (phase()) {
88
89         case MainSearch: case EvasionSearch: case QSearch: case QEvasionSearch: case ProbCut:
90             ++currMove_;
91             return ttMove_;
92
93         case PH_TacticalMoves0:
94             ms = pickBest(currMove_++, lastMove());
95             if (ms->move != ttMove_) {
96                 assert(captureThreshold_ <= 0);
97
98                 if (captureThreshold_ <= pos().see(ms->move)) {
99                     return ms->move;
100                }
101
102                // 後ろから SEE の点数が高い順に並ぶようにする。
103                (endBadCaptures_--)->move = ms->move;
104            }
105            break;
106
107        case PH_Killers:
108            move = (currMove_++)->move;
109            if (!move.isNone()
110                && move != ttMove_
111                && pos().moveIsPseudoLegal(move, true)
112                && pos().piece(move.to()) == Empty)
113                {
```

```
114→   →   →   →   return move;↓
115→   →   →   }↓
116→   →   →   break;↓
117↓
118→   →   case PH_NonTacticalMoves0:↓
119→   →   case PH_NonTacticalMoves1:↓
120→   →   →   move = (currMove_++)->move;↓
121→   →   →   if (move != ttMove_↓
122→   →   →   →   && move != killerMoves_[0].move↓
123→   →   →   →   && move != killerMoves_[1].move↓
124→   →   →   →   )↓
125→   →   →   {↓
126→   →   →   →   return move;↓
127→   →   →   }↓
128→   →   →   break;↓
129↓
130→   →   case PH_BadCaptures:↓
131→   →   →   return (currMove_--)->move;↓
132↓
133→   →   case PH_Evasions: case PH_QEvasions: case PH_QCaptures0:↓
134→   →   →   move = pickBest(currMove_++, lastMove())->move;↓
```

「ＰＨ＿ＴａｃｔｉｃａｌＭｏｖｅｓＯ　とか、
　ＰＨ＿Ｋｉｌｌｅｒｓ　って何なの？」

「すごろく　でいうと、
　なんちゃらサーチ　という名前が　スタート地点で、
　ＰＨなんとか　　という名前が　途中のマス　だぜ☆」

「６マス進んだり、３マス戻ったりするの？」

「しない☆
　進むオンリー☆」

「ＥｖａｓｉｏｎＳｅａｒｃｈ　と、
　ＱＥｖａｓｉｏｎＳｅａｒｃｈ　は　どう違うの？」

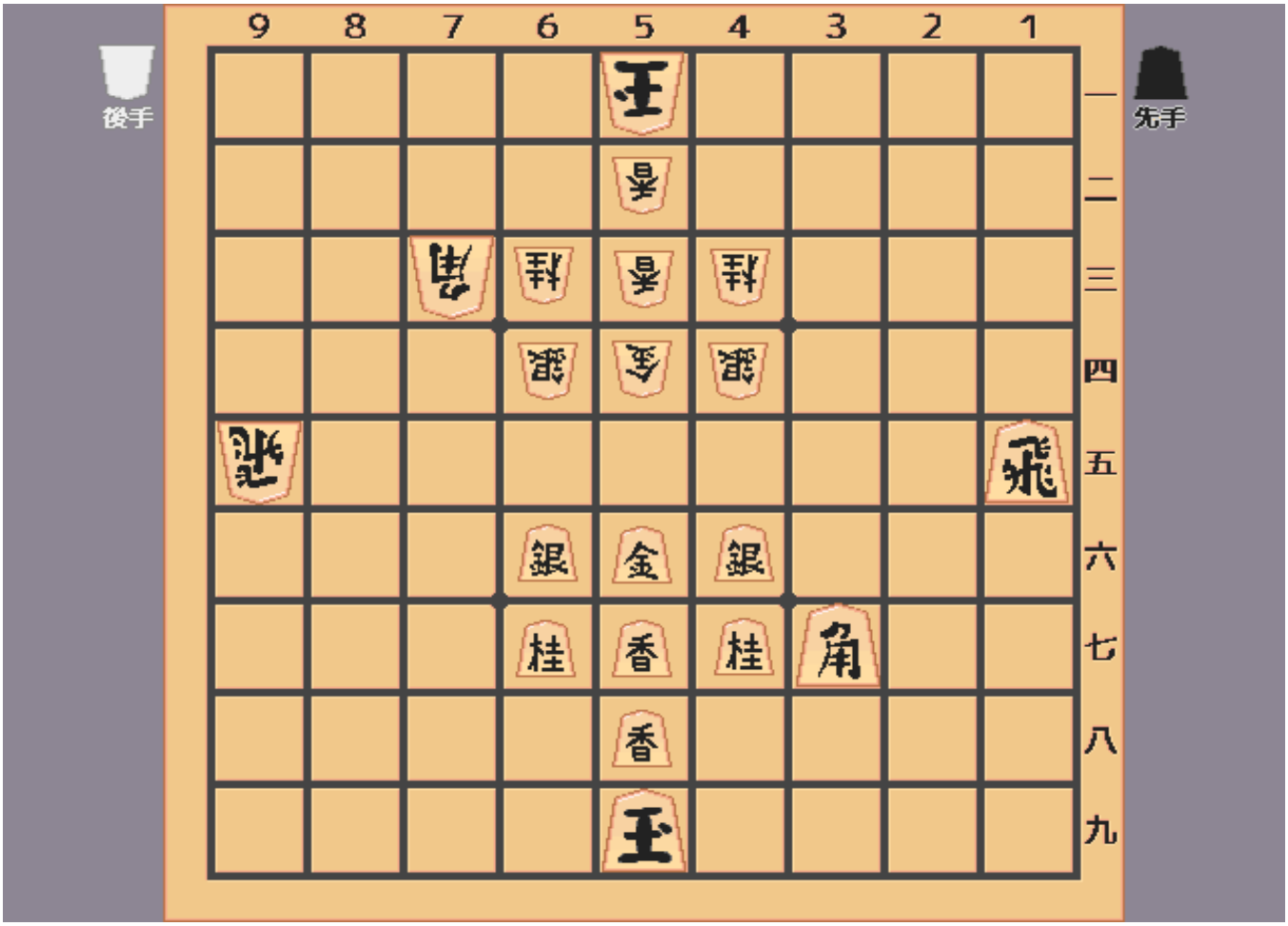
「何も違わない☆」

「 ……！」

「 静止探索で呼び出されたら、頭に　Ｑ　が付くぜ☆」

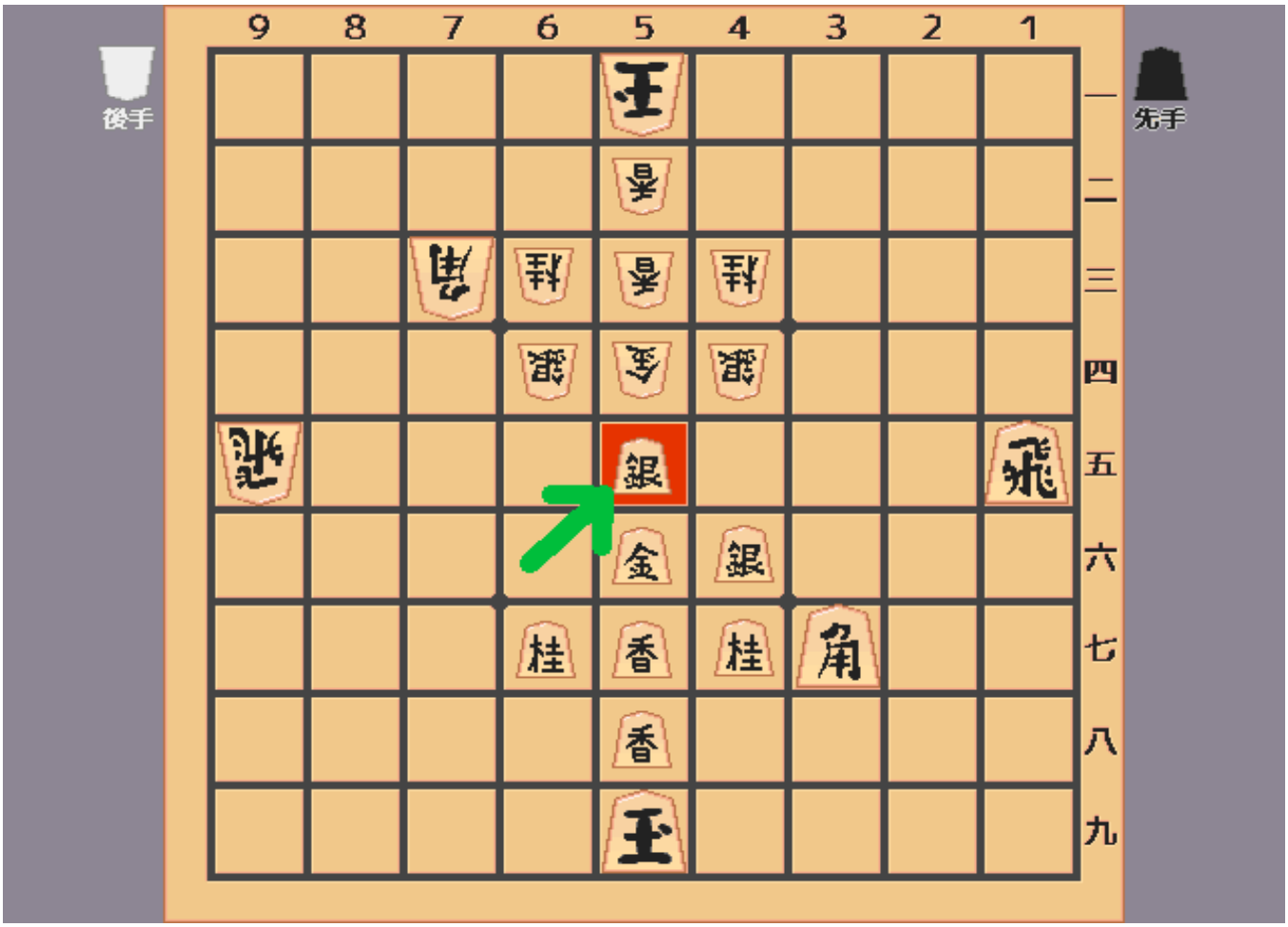
「 静止探索って何よ！」

「 例えば　コンピューターが　２０手先　を読んだら、
↓こういう局面になったとするだろ☆」



「 ……！」

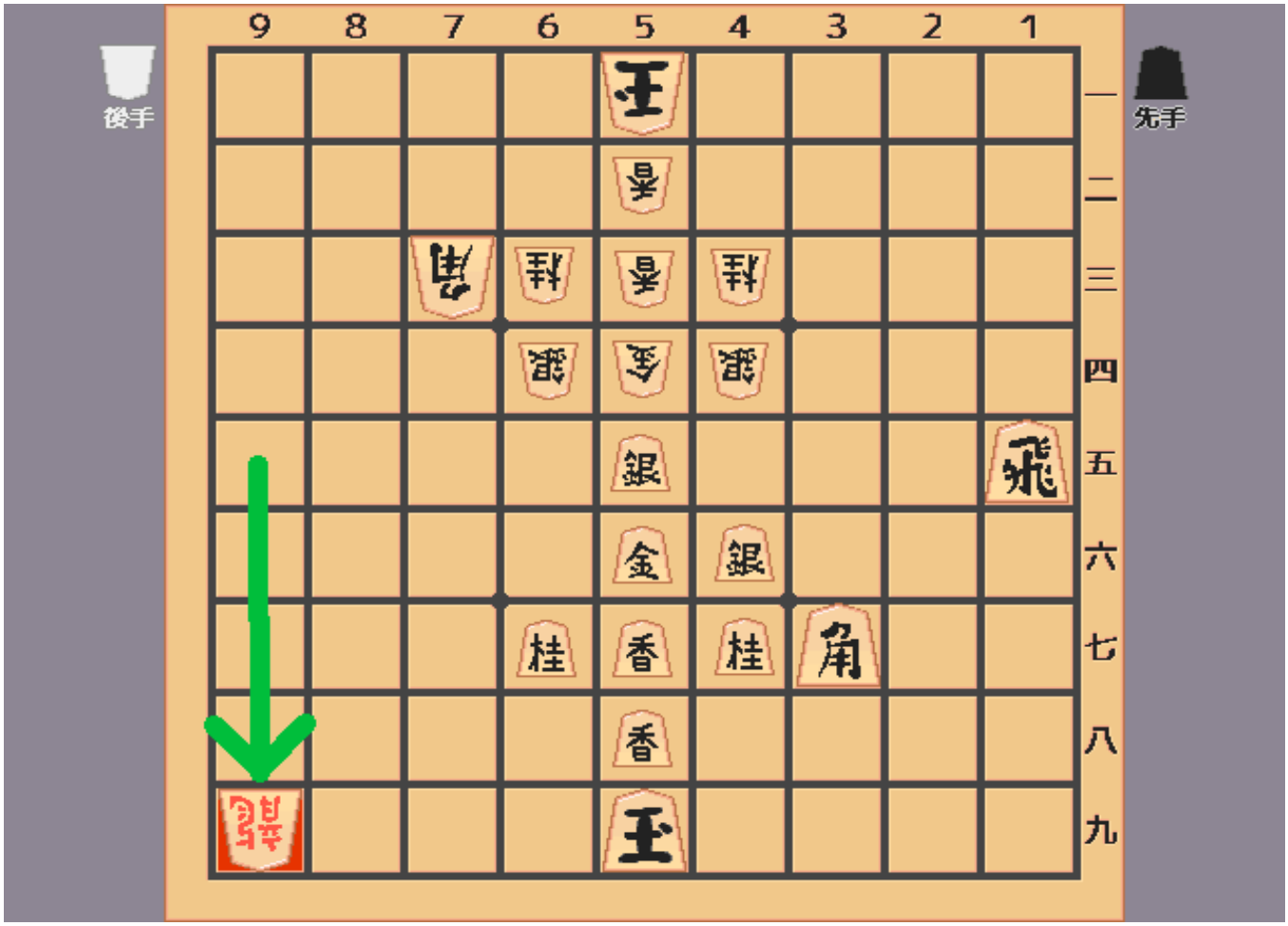
「 で、　２１手目　に　↓次の手
　を考えたところで　探索　を打ち切るとするぜ☆」



「 あいやーっ！」

「 こんな　アツアツ　のところで　探索を打ち切ったら
　危ないだろ☆」

「 まぁ　次の一手は　こうだがな☆」



「 ほんげーっ！」

「 そこで、
局面が　おとなしくなるまで　読みを延長しようぜ、
というのが　静止探索だぜ☆　Ｑ　☆」

「 Ｑ　が付くフェーズには、
ＱＣａｐｔｕｒｅｓ、　（駒を取る手）
ＱＥｖａｓｉｏｎｓ、　（王手を回避する手）
ＱＲｅｃａｐｔｕｒｅ、　（駒を取り返す手）
といったように、　アツアツ　な場面ばかりだな☆」

「 やることは　別に　普通の探索と　変わらないんだがな☆」

「 で、ムーブ・フェーズって、
　なんで　それを　細々　分けているわけ？

　１つのルーチンで　ちゃちゃーっと　やっちゃわないの？」

「 同じことするサブ・ルーチンが　１つのルーチンの中に散らばるぜ☆
　童謡　が　１番、２番、３番　とある、みたいなものだろ、
　ムーブ・フェーズは　番の多さの違う　童謡　が６曲ぐらい　ある感じ☆」

「 ビデオデッキだったり、童謡だったり、コロコロ変えるなんだぜ☆！」

「 で、これが　きふわらぶりー　の　ムーブ・フェーズ　だぜ☆」

n440_500_nextmoveEvent.cpp

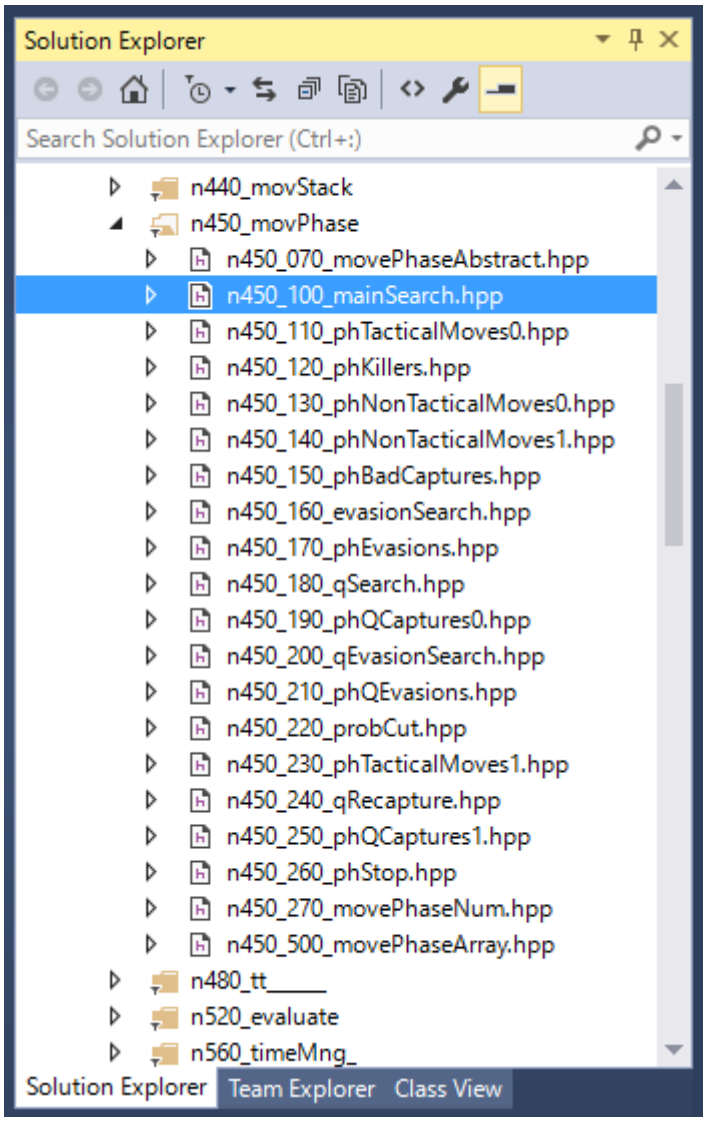
```
n440_500_nextmoveEvent.cpp
Kifuwarapery  →  NextmoveEvent  GetNextMove_NonSplitedNode()

Move NextmoveEvent::GetNextMove_NonSplitedNode() {
    do {
        // lastMove() に達したら次の phase に移る。
        while (GetCurrMove() == GetLastMove()) {
            GoNextPhase();
        }

        Move resultMove;
        bool isGotNext = g_movePhaseArray[GetPhase()]->GetNext2Move(resultMove, *this);

        if (isGotNext) {
            return resultMove;
        }

    } while (true);
}
```

「きふわらぷりー　では　ｓｗｉｔｃｈ文は　全て廃止され、
配列に押し込まれたぜ☆」

「ポリモーフィズムにしてしまったのね」

「そう☆　どのムーブ・フェーズも
ＧｏＮｅｘｔＰｈａｓｅ（）　と、
ＧｅｔＮｅｘｔ２Ｍｏｖｅ（～）　に押し込んだぜ☆

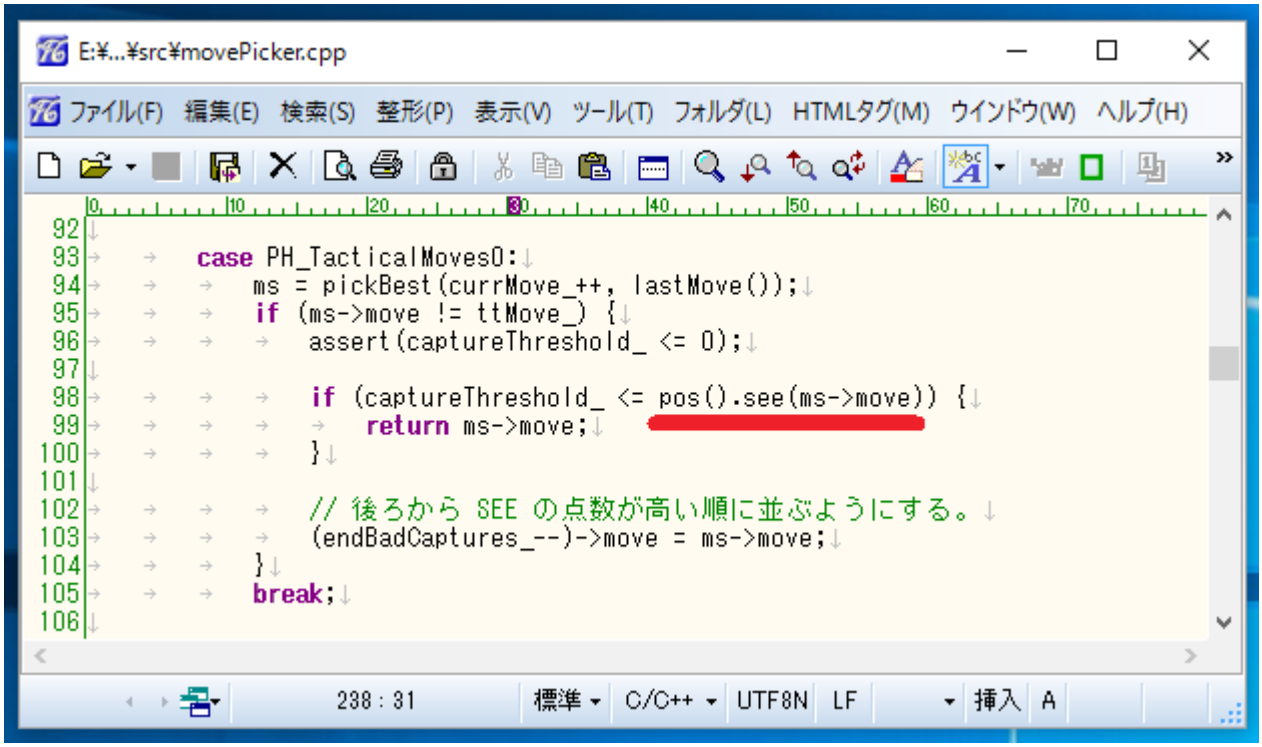
ムーブ・フェーズのそれぞれは、１つずつファイルに分けたぜ☆」



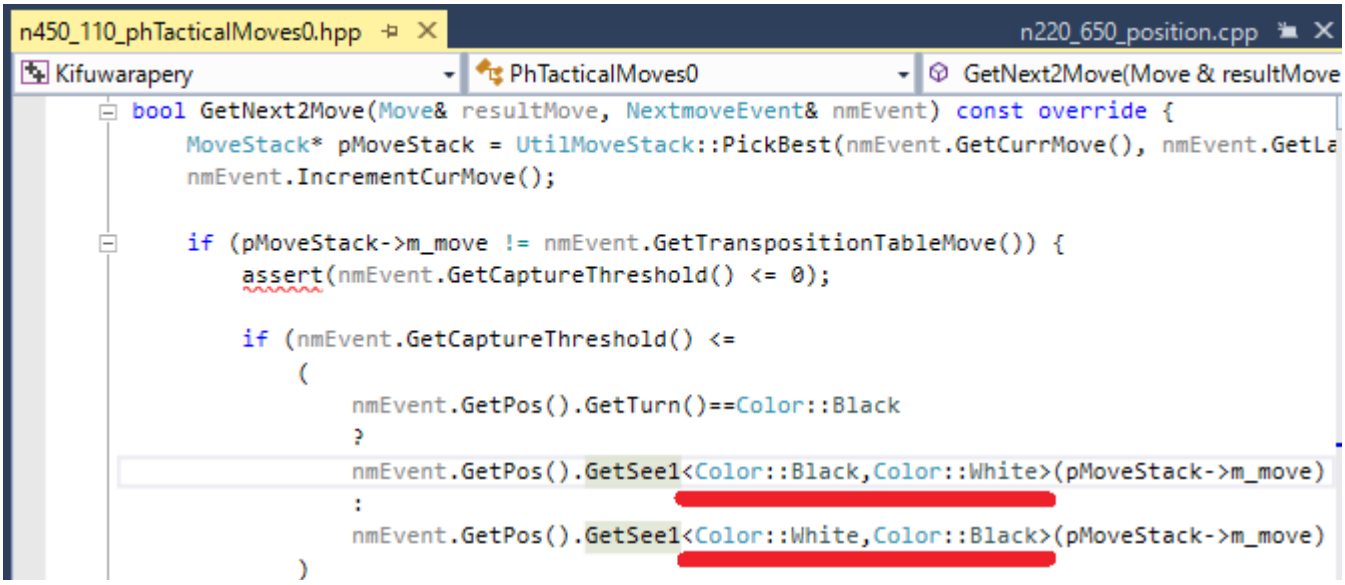
「 ファイルに分けただけで
やることは　変わってないんだななんだぜ☆」

「 先後の違いを　テンプレート　にして　定数として埋め込もうとしたんだが、
このあたりの改造は　大会数日前　に迫っていて　途中になってしまったぜ☆」

movePicker.cpp



n450_110_phTacticalMoves0.hpp



```
        ) {
            resultMove = pMoveStack->m_move;
            return true;
        }

        // 後ろから SEE の点数が高い順に並ぶようにする。
        nmEvent.GetEndBadCaptures()->m_move = pMoveStack->m_move;
        nmEvent.DecrementEndBadCaptures();
    }

    return false;
};
```

「 で、ムーブ・フェーズって　何だったのよ」

「 指し手を　１個　選ぶものだぜ☆

指し手生成は、　指し手を　一覧するものだろ☆

ムーブ・フェーズは、　ベストな　指し手　を
なるべく　早く　１個　探し出すための
段どりの　１つ１つ　だぜ☆」

まとめ　まとめ　しようぜ☆？

SE　　ASI

ベッドを背に合わせるかベッドに合わせて足を切るか☆

「ちゆりは　なんでもかんでも　ポリモーフィズム　にしたのね。
理由は　やりたいから」

「おっしゃる通りで☆」

「テンプレート　も　ばりばり　外していたぜ☆
理由は　やりたいから☆」

「その通りだぜ☆」

「他にも、
探索部　　も　分け分け　したし、
時間制御　も　いじったぜ☆

Ｃ＃プログラマーが　４か月で　Ｃ＋＋を覚えた方法とかも
書きたかったんだが、

ＰＲ文書を　差し替える締め切り　も迫ってきたので
ここまでだぜ☆」

「おつね☆」

「おつ～☆」

＞＞＞ Ｌｅｔ’ｓ　ｒｅｍｏｄｅｌｉｎｇ ａｎ ａｐｅｒｙ　ｔｗｉｇ！　Ｌｉｋｅ　ｄｅｖｉｌ！

☆

大会出場マシンの性能☆

CPU
　　Ｃｏｒｅ　ｉ７　　６５００Ｕ　　２．５ＧＨｚ

メモリ
　　４ＧＢ

ＯＳ
　　Ｗｉｎｄｏｗｓ　８．１　ｗｉｔｈ　Ｂｉｎｇ

開発言語
　　Ｃ＋＋

ライブラリ
　　Ａｐｅｒｙ、　なのはｍｉｎｉ（読み筋の表示のみ）

読みの深さ
　　１６～３２ぐらい

読みの速度（万手／秒）
　　６０

手法の特徴
　　Ｃ＃を読める自分に合わせて、メソッド名の先頭を大文字で始めるなど変えた。
　　ｓｔｒｕｃｔをｃｌａｓｓに変えたりした。
　　ｔｅｍｐｌａｔｅを外して　ポリモーフィズムを使ったりした。
　　時間制御は　１手２０秒、たまに長考をする配分を目指して改造した。
　　使用しているｓｗｉｔｃｈ文は０個にした。
　　代わりにポインタの配列でクラスメソッドを呼び出した。
　　学習はいじらず。

差し替え前のＰＲ文章

2015年 3月 27日
高橋 智史

第26回世界コンピュータ将棋選手権

きふわらべ　アピール文書

（＾～＾）

[プログラム名]

　　きふわらべ

[特徴]

　　Ｃ／Ｃ＋＋の言語を　将棋ライブラリを読んで　勉強中する予定だぜ☆（＾ｑ＾）

　　（１）Apery (commit: e9384d3d3e3c80eaa8b20d9844e11dcb7d60efe6)
　　（２）やねうら王 コンピューター将棋フレームワーク
　　（３）なのはmini
　　（４）れさぴょんｖ３
　　（５）bonanza6.0

　　これらのソースから　動かせたところを使う予定だぜ☆（＾＿＾）

　　（＾ｑ＾）
　　一番理解が簡単なソースコードとは　どれなのか……☆

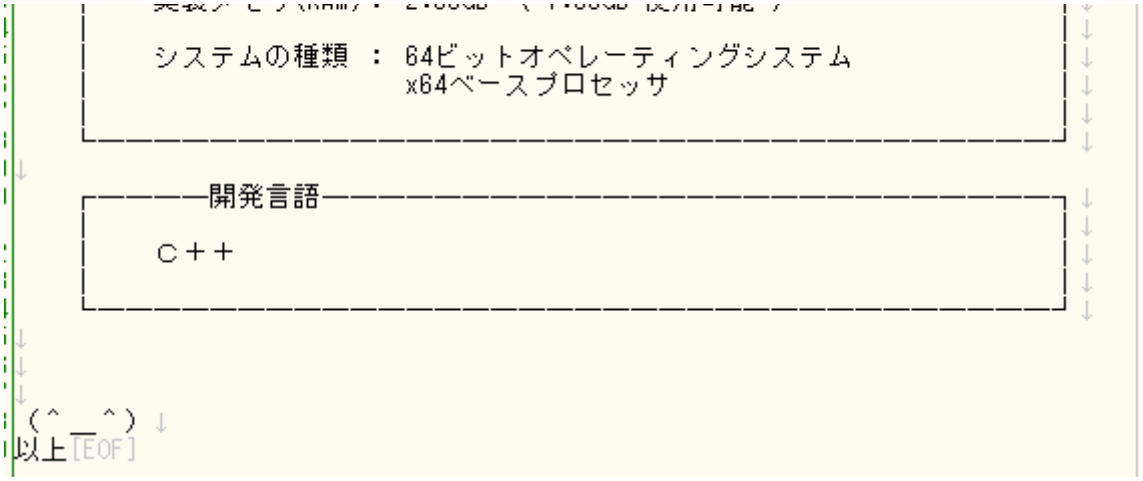
　　第25回の　きふわらべ　のソースとはまったく別物だぜ☆（＾ｑ＾）

　　（＾ｑ＾）独自性はどこかだって☆？
　　　　　　Microsoft Visual Studio 2015 C++ で作ってみるぜ☆

　　（＾～＾）↓低スペックノートＰＣ１台でがんばるソフトを作るぜ☆

（－＿－）... ...

```
┌─────ハードウェアスペック──────────────────┐
│                                              │
│   種　類        ：ノートパソコン　１台　(lenovo)
│                                              │
│   Ｏ　Ｓ        ：Windows 8.1 with Bing      │
│                                              │
│   ＣＰＵ        ：Intel(R) Celeron(R) CPU N2830
│                    @ 2.16GHz 2.16GHz         │
│                                              │
│   実装メモリ(RAM)：2.00GB （1.89GB 使用可能 ）
```



（＾＿＾）
以上[EOF]