池田 伊織
2015年8月4日

Procon26　競技部門　モジュール仕様書

Ver 1.0

1. 概要

モジュールは、問題の入出力、石の回転・反転等などを行う関数・構造体群である。各ソルバは、基本的に本モジュールを用いて作成する。

本モジュールの構成については表1に示す。

表 1. モージュールの構成

名称	概要
procon26_module.h	石・ボードの構造体の定義や各定数等の定義部
procon26_modio.h	問題の入出力に関係するモジュールの定義部
procon26_modlib.h	石・ボードに対する処理に関係するモジュールの定義部
procon26_modio.cpp	問題の入出力に関係するモジュールの実装部
procon26_modlib.cpp	石・ボードに対する処理に関係するモジュールの実装部

2. インターフェース

2.1　procon26_module

procon26_module の構成については以下に示す。

STONE_SIZE　：石の大きさ
BOARD_SIZE　：ボードの大きさ

struct Stone　：石を表す構造体
struct Board　：ボードを表す構造体
struct Problem：問題を表す構造体
class Answer　：解答を表すクラス(各石)
　　std::string toString()：解答を文字列に変換する関数

struct Answers：解答を表す構造体(全体)

2.2　procon26_modio(procon26_ModuleInputOutput)

procon26_modio の構成については以下に示す。

Stone *getStoneByString(std::string stone)
文字列から石に変換する関数。
　　stone　：石の情報を持つ文字列
　　return　：引数で与えられた文字列を Stone 構造体に格納し返す

Board *getBoardByString(std::string board)
文字列からボードに変換する関数。
　　board　：ボードの情報を持つ文字列
　　return　：引数で与えられた文字列を Board 構造体に格納し返す

void inputStone(Stone *stone)
標準入力から石を読み込み構造体に変換する関数。
　　stone　：変換した石を保存する構造体配列のポインタ

void inputBoard(Board *board)
標準入力からボードを読み込み構造体に変換する関数。
　　board　：変換したボードを保存する構造体ポインタ

void showStone(const Stone *stone)
標準出力に石を出力する関数。
　　stone　：出力する石の構造体ポインタ
　　(定数 block_0, block_1 の文字を変更することで、
　　出力される文字変更可能)

void showBoard(const Board *board)
標準出力にボードを出力する関数。
　　board　：出力するボードの構造体ポインタ
　　(定数 block_0, block_1 の文字を変更することで、
　　出力される文字変更可能)

2.3　procon26_modlib(procon26_ModuleLibrary)

procon26_modlib の構成については以下に示す。

int countBit(unsigned char)　　：ビットを数える関数
int countBitOfStone(const Stone*)　　：石のビットを数える関数
Stone *quarryStone(const Board *board, int x, int y)
ボードから指定した座標からの 8 * 8 を切り取る関数。

- board　：切り取りたいボードの構造体ポインタ
- x　：切り取る座標の左上の x 座標
- y　：切り取る座標の左上の y 座標
- return　：切り取った範囲を石として Stone 構造体に格納しポインタを返す

各種 shift 関数：
Stone *shift[Dir](const Stone *stone, int times, int filler = 0)
石を各方向にシフトする関数。

- stone　：シフトしたい石の構造体ポインタ
- times　：シフト回数
- filler　：シフトしたときに詰める値(省略可：デフォルトは 0)
- return　：シフトした石を Stone 構造体に格納しポインタを返す

各種 turn 関数
Stone *turn(const Stone *stone, int n)
石を右に 90 ° * n 回転させる関数。

- stone　：回転したい石の構造体ポインタ
- n　：回転する角度'(90° * n)(0 ≦ n ≦ 3)
- return　：回転した石を Stone 構造体に格納しポインタを返す

Stone *flip(const Stone *stone)
石を反転させる関数。

- stone　：反転させたい石の構造体ポインタ
- return　：反転した石を Stone 構造体に格納しポインタを返す

bool isEmptyStone(const Stone *stone)
石が空か調べる関数。

- stone　：空か調べたい石の構造体ポインタ
- return　：空だったら true, そうでなければ false を返す

池田 伊織
2015 年 8 月 4 日

Stone *getTouchingStone(const Board *board, const Stone *stone, int x, int y)
ボードの指定した座標に石を置いたとき、触れている石を返す関数。

board　：石を置くボードの構造体ポインタ
stone　：置く石の構造体ポインタ
x　　：石を置く x 座標
y　　：石を置く y 座標
return　：触れている石を格納した構造体ポインタ

Board *placeStone(const Board *board, const Stone *stone, int x, int y)
ボードに石を置く関数。

board　：石を置きたいボードの構造体ポインタ
stone　：置く石の構造体ポインタ
x　　：石を置く x 座標
y　　：石を置く y 座標
return　：石を置いた後のボードの構造体ポインタ

bool canPlace(const Board *board, const Stone *stone, int x, int y)
ボードに石が置けるか調べる関数。(障害物の有無のみを調べ、触れているかわ調べない)

board　：石を置きたいボードの構造体ポインタ
stone　：置く石の構造体ポインタ
x　　：石を置く x 座標
y　　：石を置く y 座標
return　：石が置けるならば true を返し、そうでなければ false を返す

bool isEqualStone(const Stone *stone1, const Stone *stone2)
stone1 と stone2 が等しいか調べる関数。

stone1　：石の構造体ポインタ
stone2　：石の構造体ポインタ
return　：石が等しければ true を返し、そうでなければ false を返す

Stone *cloneStone(const Stone *stone)
stone のコピーを行う関数。

stone　：コピーしたい石の構造体ポインタ
return　：石のコピーを格納した Stone 構造体のポインタを返す

その他の関数については省略し、新たに関数が追加された場合は、本仕様書は更新される場合がある。