三目並べ 詳細仕様書

チームG

2024

目次

[1 概要 2](#_Toc167201028)

[2 コーディング規約 2](#_Toc167201029)

[3 画面構成 3](#_Toc167201030)

[3.1 画面構成図 3](#_Toc167201031)

[3.2 画面遷移図 3](#_Toc167201032)

[3.3 画面詳細 4](#_Toc167201033)

[4 ファイル構成 5](#_Toc167201034)

[4.1 data.h 5](#_Toc167201035)

[4.1.1 define 5](#_Toc167201036)

[4.1.2 構造体 6](#_Toc167201037)

[4.2 game\_input.c（game\_input.h） 7](#_Toc167201038)

[4.3 game\_output.c（game\_output.h） 7](#_Toc167201039)

[4.4 game\_judgement.c（game\_judgement.h） 7](#_Toc167201040)

[4.5 marubatsu\_game.c 7](#_Toc167201041)

[5 機能構成 8](#_Toc167201042)

[5.1 入力・チェック機能（game\_input.c） 8](#_Toc167201043)

[5.1.1 InputByPlayer（プレイヤー入力受付）関数 8](#_Toc167201044)

[5.1.2 InputToValue（入力変換）関数 9](#_Toc167201045)

[5.1.3 InputProcess（入力取得）関数 10](#_Toc167201046)

[5.2 画面出力機能（game\_output.c） 11](#_Toc167201047)

[5.2.1 DisplayBoard（ボードエリア表示）関数 11](#_Toc167201048)

[5.2.2 DisplayStatus（ゲーム状況表示）関数 12](#_Toc167201049)

[5.2.3 DisplayResult（ゲーム結果表示）関数 13](#_Toc167201050)

[5.3 判定機能（game\_judgement.c） 14](#_Toc167201051)

[5.3.1 BoardSearch（ボード走査）関数 14](#_Toc167201052)

[5.3.2 BoardJudgement（ボード判定）関数 15](#_Toc167201053)

[5.3.3 isDuplicate（重複判定）関数 16](#_Toc167201054)

[5.3.4 WinnerJudgement（勝者判定）関数 17](#_Toc167201055)

[5.4 ゲーム進行機能（marubatsu\_game.c） 18](#_Toc167201056)

[5.4.1 SerialToCoordinates（要素番号変換）関数 18](#_Toc167201057)

[5.4.2 WaitingPlayerInput（入力待機）関数 19](#_Toc167201058)

[5.4.3 GameMainLoop（ゲームメインループ）関数 20](#_Toc167201059)

[5.4.4 GameInit（ゲーム初期化）関数 21](#_Toc167201060)

[5.4.5 main関数 22](#_Toc167201061)

# 概要

本仕様書は、三目並べの画面構成、ファイル構成、機能構成を記したものである。

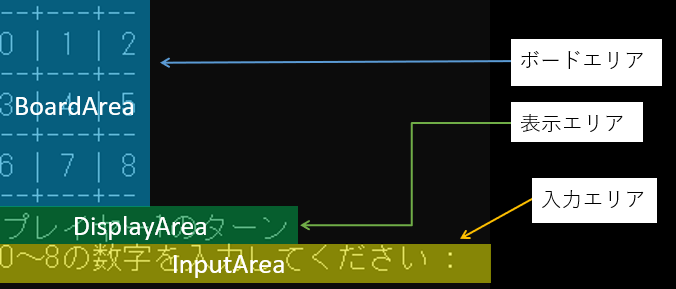
# コーディング規約

コーディング規約は、C言語コーディング規約（Y:\資料\C言語(基礎)\コーディング規約.pdf）を参照すること。

# 画面構成

## 画面構成図

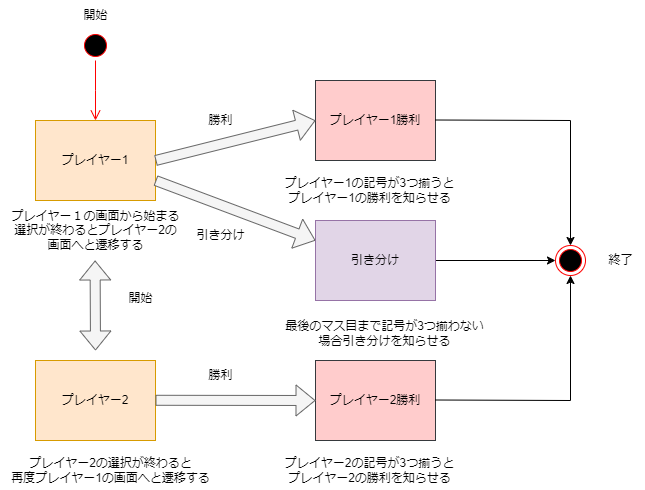
画面構成図を以下に示す。



画面構成図

## 画面遷移図

画面遷移図を以下に示す。



画面遷移図

## 画面詳細

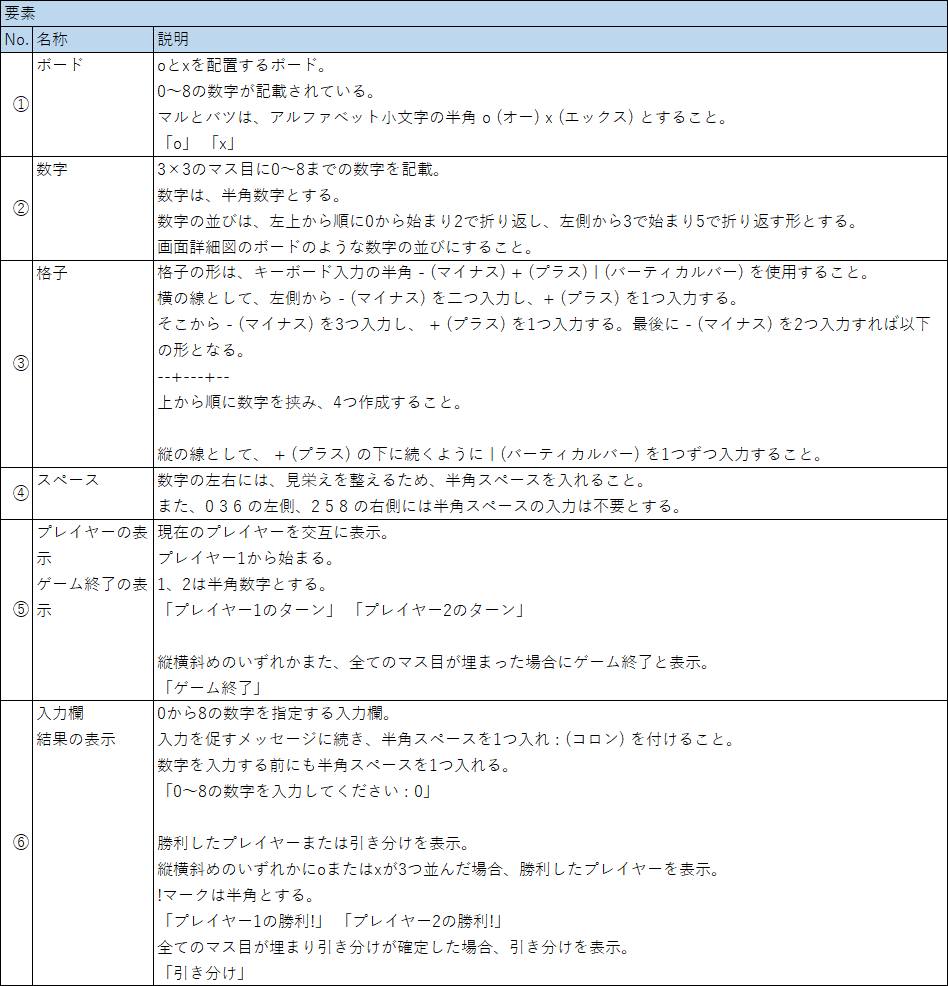
画面詳細図とその説明を以下に示す。

テキスト が含まれている画像

自動的に生成された説明

画面詳細図

画面の説明



# ファイル構成

ファイル構成を以下に示す。

## data.h

マクロ、構造体を宣言するファイル。

### define

PLAYER\_NUM=2 プレイヤーの人数

NAME\_LEN\_MAX=32 プレイヤー名の最大バイト数

BOARD\_HEIGHT=3 ボードの高さ

BOARD\_WIDTH=3 ボードの幅

INPUT\_LEN\_MAX=4 入力長の最大

INPUT\_VAL\_MIN=0 入力数値の最小値

INPUT\_VAL\_MAX

=BOARD\_HEIGHT \* BOARD\_WIDTH – 1 入力数値の最小値

INPUT\_ERR=INPUT\_MIN – 1 入力エラーを表す値

CONNECTED\_NUM\_FOR\_END=3 終了条件の同じ記号が並ぶ数

LINE=1 NO\_LINE=0 ラインが完成したか否か

PLAYER\_INDEX(turn)=turn % PLAYER\_NUM turnをplayer\_indexに変換する

EMPTY="" 空文字列

DUPLICATE=1 重複

NOT\_DUPLICATE=0 重複でない

### 構造体

|  |  |
| --- | --- |
| 型名 | 変数名 |
| COORDINATES | |
| int | x |
| int | y |
| GAME\_INFO | |
| char[BOARD\_HEIGHT][BOARD\_WIDTH] | game\_board |
| char[PLAYER\_NUM][NAME\_LEN\_MAX] | players |
| int | turn\_conter |
| int | player\_index |

## game\_input.c（game\_input.h）

入力チェック機能を実装するファイル。

## game\_output.c（game\_output.h）

画面出力機能を実装するファイル。

## game\_judgement.c（game\_judgement.h）

判定機能を実装するファイル。

## marubatsu\_game.c

三目並べプログラムのmain関数を実装するファイル。

data.h、game\_input.h、game\_output.h、game\_judgement.hをインクルードする。

# 機能構成

機能構成を以下に示す。

## 入力・チェック機能（game\_input.c）

プレイヤーの入力を受け付け、それがゲームの仕様に適合するかを確認し、数値またはエラーを取得する。文字列入力はscanf\_s、文字列出力はprintfを利用すること。

### InputByPlayer（プレイヤー入力受付）関数

戻り値、なし

表 1 InputByPlayer関数で使用する引数・変数

|  |  |  |
| --- | --- | --- |
| 型 | 変数名 | 説明・初期値 |
| 引数 | | |
| char[] | input | 入力文字列を格納する配列 |
| ローカル変数 | | |
| char | buffer | バッファを読み飛ばす用 |

scanf\_sの第一引数は、“%3[^\n]s”を指定する。

getchar()を呼び出し、戻り値でbufferを初期化（宣言）する。

条件文をbuffer != ‘\n’として、while文を使用する。

* + getchar()を呼び出し、戻り値をbufferに代入する。

### InputToValue（入力変換）関数

戻り値、serial\_numberまたはINPUT\_ERR

表 2 InputJudgement関数で使用する引数・変数

|  |  |  |
| --- | --- | --- |
| 型 | 変数名 | 説明・初期値 |
| 引数 | | |
| char[] | input | 入力文字列を格納している配列 |
| ローカル変数 | | |
| int | serial\_number | 文字列を数値に変換したものを格納する  0で初期化する |

* serial\_numberを初期化する。
* カウンタ変数をi、終了条件をinput[i]==’\0’としてfor文で配列を走査する。
  + wtype.hのiswdigit(input[i])関数が0を返した場合、”文字の入力は禁止です。”と表示し、戻り値としてINPUT\_ERRを返す。
  + serial\_numberに10を乗算する。
  + serial\_numberに(input[i]-‘0’)を加算する。
* serial\_numberがINPUT\_VAL\_MINより小さいかINPUT\_VAL\_MAXより大きい数値の場合、”0～8以外の数値は禁止です。” と表示し、戻り値としてINPUT\_ERRを返す。
* 戻り値としてserial\_numberを返す。

### InputProcess（入力取得）関数

戻り値、ボードの位置を表す数字またはINPUT\_ERR

表 3 InputProcess関数で使用する引数・変数

|  |  |  |
| --- | --- | --- |
| 型 | 変数名 | 説明・初期値 |
| ローカル変数 | | |
| char[] | input | 入力文字列を格納する配列  EMPTYで初期化する |
| int | serial\_number | ボードの位置を表す数字  InputJudgement関数で初期化する |

* inputを初期化する。
* 5.1.1項のInputByPlayer(input)関数を呼び出す。
* 5.1.2項のInputToValue (input)関数を呼び出し、戻り値でserial\_numberを初期化（宣言）する。
* 戻り値としてserial\_numberを返す。

## 画面出力機能（game\_output.c）

ボードエリア、表示エリア、入力エリアに分けて画面に出力する。画面構成は3章を参照すること。文字列出力はprintfを利用すること。

### DisplayBoard（ボードエリア表示）関数

戻り値、なし

|  |  |  |
| --- | --- | --- |
| 型 | 変数名 | 説明・初期値 |
| 引数 | | |
| char[][] | game\_board | ボードを表す２次元配列 |

* game\_board[row][column]として、二重for文で2次元配列を走査する。
* カウンタ変数は外側をrow、内側をcolumnとし、初期値はともに0、それぞれBOARD\_HEIGHT、BOARD\_WIDTHを終了値（処理を行わずにループを抜ける）とする。
* フォーマットは3.3節の①～④に準ずる。

### DisplayStatus（ゲーム状況表示）関数

戻り値、なし

表 4 DisplayStatus関数で使用する引数

|  |  |  |
| --- | --- | --- |
| 型 | 変数名 | 説明・初期値 |
| 引数 | | |
| char[] | player\_name | プレイヤーの名前を表す文字列 |
| char[][] | game\_board | ボードを表す２次元配列 |

* 5.2.1項のDisplayBoard(game\_board)関数を呼び出す。
* 3.3節の⑤に準じて、player\_nameを表示する。
* 3.3節の⑥に準じて、改行せずに入力欄を表示する。

### DisplayResult（ゲーム結果表示）関数

戻り値、なし

表 5 DisplayResult関数で使用する引数

|  |  |  |
| --- | --- | --- |
| 型 | 変数名 | 説明・初期値 |
| 引数 | | |
| char[] | winner\_name | 勝者の名前を表す文字列 |
| char[][] | game\_board | ボードを表す２次元配列 |

5.2.1項のDisplayBoard(game\_board)関数を呼び出す。

3.3節の⑤に準じてのゲーム終了の表示を行う。

3.3節の⑥に準じて、引き分けまたはwinner\_nameの表示を行う。

* + winner\_nameがEMPTYの場合に、引き分けの表示を行う。

## 判定機能（game\_judgement.c）

ゲームの判定に関する処理を行う。

### BoardSearch（ボード走査）関数

戻り値、カウント数

表 6 BoardSearch関数で使用する変数

|  |  |  |
| --- | --- | --- |
| 型 | 変数名 | 説明・初期値 |
| 引数 | | |
| char[][] | game\_board | ボードを表す２次元配列 |
| char | mark | 基準となる記号を表す |
| COORDINATES | current\_pos | 現在、捜査している2次元配列の位置を表す |
| COORDINATES | direction | 走査する方向のベクトルを表す |
| ローカル変数 | | |
| int | mark\_counter | 同じ記号の数をカウントする  0で初期化する |

* mark\_counterを初期化する。
* current\_pos.xにdirection.xを、current\_pos.yにdirection.yを加算する。
* 「current\_pos.xが0以上かつBOARD\_WIDTH未満」かつ  
  「current\_pos.yが0以上かつBOARD\_HEIGHT未満」かつ  
  「markがgame\_board[current\_pos.x][current\_pos.y]と等しい」場合、mark\_counterをインクリメントする。
* それ以外の場合、mark\_counterを返す。

### BoardJudgement（ボード判定）関数

戻り値、LINEまたはNO\_LINE

表 7 BoardJudgement関数で使用する引数・変数

|  |  |  |
| --- | --- | --- |
| 型 | 変数名 | 説明・初期値 |
| 引数 | | |
| char[][] | game\_board | ボードを表す２次元配列 |
| COORDINATES | mark\_pos | 基準となる2次元配列の位置を表す |
| ローカル変数 | | |
| COORDINATES [4] | directions | 2次元配列を走査する方向を定義する  {{1, 0}, {1, 1}, {0, 1}, {-1, 1}}で初期化する |
| char | mark | 基準となる記号を示す  game\_board[mark\_pos.y][mark\_pos.x]で初期化する |
| int | mark\_counter | 同じ記号が並んでいる回数をカウントする  1で初期化する |
| COORDINATES | current\_pos | 現在走査している位置を示す  mark\_posで初期化する |

* directions、symbolを初期化する。
* directions[i]として、二重for文で配列を走査する。
* カウンタ変数は外側をi, 内側をjとし、初期値はともに0、それぞれsizeof(directions) / sizeof(COORDINATES)、2を終了値（処理を行わずにループを抜ける）とする。
  + 外側for文で、mark\_counterを初期化する。
    - 内側for文で、5.3.1項のBoardSearch(game\_board, mark, mark\_pos, directions[i])関数を呼び出し、戻り値をmark\_counterに加算する。
    - directions[i]のxとyの符号を反転する。
  + 外側for文で、symbol\_counterが3（以上）の場合、戻り値としてLINEを返す。
* for文が終了した場合、戻り値としてNO\_LINEを返す。

### isDuplicate（重複判定）関数

戻り値、DUPLICATEまたはNOT\_DUPLICATE

|  |  |  |
| --- | --- | --- |
| 型 | 変数名 | 説明・初期値 |
| 引数 | | |
| char[][] | game\_board | ボードを表す２次元配列 |
| COORDINATES | mark\_pos | 2次元配列の位置を表す |
| ローカル変数 | | |
| int | is\_duplicate | 重複か否かを表す  iswdigit関数の否定で初期化する |

* !iswdigit(game\_board[mark\_pos.y][mark\_pos.x])を呼び出し、戻り値でis\_duplicateを初期化（宣言）する。
* 戻り値としてis\_duplicateを返す。

### WinnerJudgement（勝者判定）関数

戻り値、勝者の名前文字列の先頭アドレスまたはEMPTY: char\*

表 8 WinnerJudgement関数で使用する引数・変数

|  |  |  |
| --- | --- | --- |
| 型 | 変数名 | 説明・初期値 |
| 引数 | | |
| char[][] | player\_names | プレイヤー名を格納する2次元配列 |
| int | turn\_counter | ゲーム情報を格納する構造体変数 |
| ローカル変数 | | |
| char\* | winner | 勝者の名前文字列の先頭アドレスを格納する |

* turn\_counterがBOARD\_HEIGHT\*BOARD\_WIDTH以上の場合、winnerにEMPTYを格納する。
* それ以外の場合、winnerにplayer\_names[turn\_counter % PLAYER\_NUM]を格納する。
* 戻り値としてwinnerを返す。

## ゲーム進行機能（marubatsu\_game.c）

三目並べのゲームを進行する。

### SerialToCoordinates（要素番号変換）関数

戻り値、引数に対応する配列の位置情報(要素番号): COORDINATES

表 9 SerialToCoordinates関数で使用する引数・変数

|  |  |  |
| --- | --- | --- |
| 型 | 変数名 | 説明 |
| 引数 | | |
| int | serial\_number | 3.3節の②に準じたボードの位置を表す数字 |
| ローカル変数 | | |
| COORDINATES | indexes | 引数に応じた２次元配列の位置を表す |

* indexesを{serial\_number % BOARD\_WIDTH, serial\_number / BOARD\_WIDTH}で初期化する。
* 戻り値としてindexesを返す。

### WaitingPlayerInput（入力待機）関数

戻り値、入力に対応する配列の位置情報(要素番号): COORDINATES

表 10 WaitingPlayerInput関数で使用する引数・変数

|  |  |  |
| --- | --- | --- |
| 型 | 変数名 | 説明・初期値 |
| 引数 | | |
| GAME\_INFO\* | game\_info | ゲーム情報を格納する構造体変数 |
| ローカル変数 | | |
| int | serial\_number | ボードの位置を表す  INPUT\_ERRで初期化する |
| COORDINATES | indexes | ２次元配列の位置を表す  {INPUT\_ERR, 0}で初期化する |

* serial\_number、indexesを初期化する。
* 条件式をserial\_number==INPUT\_ERRとするwhile文を使用する。
  + 5.2.2項のDisplayStatus関数を呼び出す。
  + 5.1.3項のInputProcess関数を呼び出し、戻り値をserial\_numberに代入する。
  + serial\_numberがINPUT\_ERRの場合、continueする。
  + 5.4.1項のSerialToCoordinates(serial\_number)関数を呼び出し、戻り値をindexesに代入する。
  + 5.3.3項のisDuplicate(game\_info->game\_board, indexes)関数の条件式が正の場合、”既に埋まっています”とエラー表示してcontinueする。
* 戻り値としてindexesを返す。

### GameMainLoop（ゲームメインループ）関数

戻り値、なし

表 11 GameMainLoop関数で使用する引数・変数

|  |  |  |
| --- | --- | --- |
| 型 | 変数名 | 説明・初期値 |
| 引数 | | |
| GAME\_INFO\* | game\_info | ゲーム情報を格納する構造体変数 |
| ローカル変数 | | |
| int | is\_line | ラインができたか(LINEかNO\_LINE)を格納する  NO\_LINEで初期化する |
| int | serial\_number | 3.3節の②に準じたボードの位置を表す数字  関数で初期化する |
| COORDINATES | indexes | 記号を入れる位置を表す配列のインデックス  関数で初期化する |

* is\_lineを初期化する。
* game\_info->turn\_counterをカウンタ変数（初期値は0とする）、終了値（処理を行わずにループを抜ける）をBOARD\_HEIGHT \* BOARD\_WIDTHとしてfor文を使用する。
  + game\_info->player\_indexにマクロPLAYER\_INDEX(game\_info->turn\_counter)を代入する。
  + 5.2.1項のDisplayBoard(game\_info->game\_board)関数を呼び出す。
  + 5.4.2項のWaitingPlayerInput(&game\_info)関数を呼び出し、戻り値でindexesを初期化（宣言）する。
  + game\_info->game\_board[indexes.y][indexes.x]にgame\_info->marks[game\_info->players\_index]を代入する。
  + 5.3.2項のBoardJudgement(game\_info->game\_board, indexes)関数を呼び出し、戻り値をis\_lineに代入する。
  + is\_lineがLINEの場合、ループを抜ける。

### GameInit（ゲーム初期化）関数

戻り値、3.3節の②に準じた数字: int

表 12 GameInit関数で使用する引数

|  |  |  |
| --- | --- | --- |
| 型 | 変数名 | 説明・初期値 |
| GAME\_INFO\* | game\_info | ゲーム情報を格納する構造体変数  以下のように初期化する |
| game\_info | | |
| char[PLAYER\_NUM][NAME\_LEN\_MAX] | player\_names | プレイヤー名の文字列を格納する配列  {“プレイヤー1”,”プレイヤー2”}で初期化する |
| char[PLAYER\_NUM] | marks | 記号を格納する配列  {‘o’,’x’}で初期化する |
| char[BOARD\_HEIGHT][BOARD\_WIDTH] | game\_board | ボードの状態を格納する2次元配列  全要素を要素番号と同じ数値で初期化する |
| int | turn\_counter | ターン数をカウントする  0で初期化する |
| int | players\_index | 現在playersのどれを指しているかを表す  PLAYER\_INDEXマクロで初期化する |
| char[] | winner | 勝者の名前またはEMPTYを格納する配列  EMPTYで初期化する |

* game\_infoポインタ変数を表 12をもとに初期化する。

### main関数

表 13 main関数で利用する引数・変数

|  |  |  |
| --- | --- | --- |
| 型 | 変数名 | 説明・初期値 |
| GAME\_INFO | game\_info | ゲーム情報を格納する構造体変数  以下のように初期化する |
| game\_info | | |
| char[PLAYER\_NUM][NAME\_LEN\_MAX] | player\_names | プレイヤー名の文字列を格納する配列  {“プレイヤー1”,”プレイヤー2”}で初期化する |
| char[PLAYER\_NUM] | marks | 記号を格納する配列  {‘o’,’x’}で初期化する |
| char[BOARD\_HEIGHT][BOARD\_WIDTH] | game\_board | ボードの状態を格納する2次元配列  1次元配列を全てindexと同じで初期化する |
| int | turn\_counter | ターン数をカウントする  0で初期化する |
| int | players\_index | 現在playersのどれを指しているかを表す  関数で初期化する |
| char[] | winner | 勝者の名前またはEMPTYを格納する配列 |

* 5.4.4項のGameInit(&game\_info)関数を呼び出す。
* 5.4.3項のGameMainLoop(&game\_info)関数を呼び出す。
* 5.3.4項のWinnerJudgement(game\_info.turn\_counter)関数を呼び出し、戻り値でwinnerを初期化（宣言）する。
* 5.2.3項のDisplayResult(winner, game\_info.game\_board)関数を呼び出す。
* プログラムが終了する。