三目並べ 詳細仕様書

チームG

2024

目次

[1 概要 3](#_Toc167351681)

[2 コーディング規約 3](#_Toc167351682)

[3 画面構成 4](#_Toc167351683)

[3.1 画面構成図 4](#_Toc167351684)

[3.2 画面遷移図 4](#_Toc167351685)

[3.3 画面詳細 5](#_Toc167351686)

[4 ファイル構成 6](#_Toc167351687)

[4.1 data.h 6](#_Toc167351688)

[4.1.1 マクロ 6](#_Toc167351689)

[4.1.2 構造体 7](#_Toc167351690)

[4.2 game\_input.c（game\_input.h） 8](#_Toc167351691)

[4.3 game\_output.c（game\_output.h） 8](#_Toc167351692)

[4.4 game\_judgement.c（game\_judgement.h） 8](#_Toc167351693)

[4.5 marubatsu\_game.c 8](#_Toc167351694)

[5 機能構成 9](#_Toc167351695)

[5.1 入力・チェック機能（game\_input.c） 9](#_Toc167351696)

[5.1.1 InputProcess（入力取得）関数 9](#_Toc167351697)

[5.2 画面出力機能（game\_output.c） 10](#_Toc167351698)

[5.2.1 DisplayBoard（ボードエリア表示）関数 10](#_Toc167351699)

[5.2.1 DisplayStatus（ゲーム状況表示）関数 11](#_Toc167351700)

[5.2.1 DisplayResult（ゲーム結果表示）関数 12](#_Toc167351701)

[5.3 判定機能（game\_judgement.c） 13](#_Toc167351702)

[5.3.1 BoardJudgement（ボード判定）関数 13](#_Toc167351703)

[5.4 ゲーム進行（marubatsu\_game.c） 14](#_Toc167351704)

[5.4.1 ゲーム初期化部分 15](#_Toc167351705)

[5.4.1 ゲームメインループ部分 16](#_Toc167351706)

[5.4.2 入力待機部分 17](#_Toc167351707)

# 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。

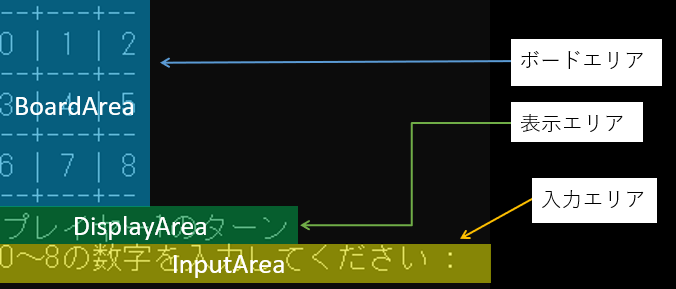
# コーディング規約

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

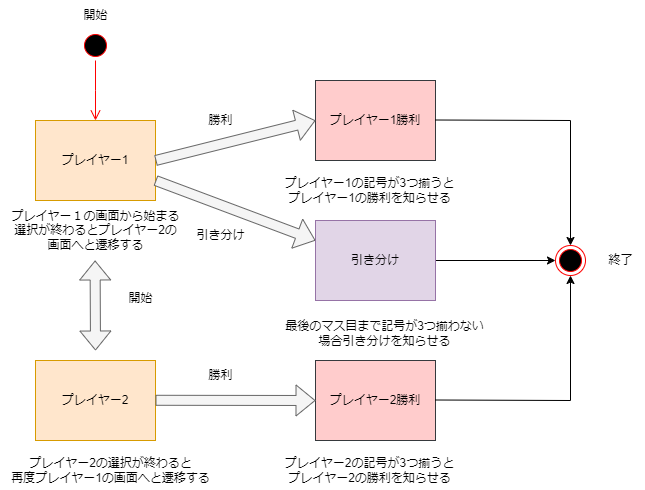
画面構成図を以下に示す。



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

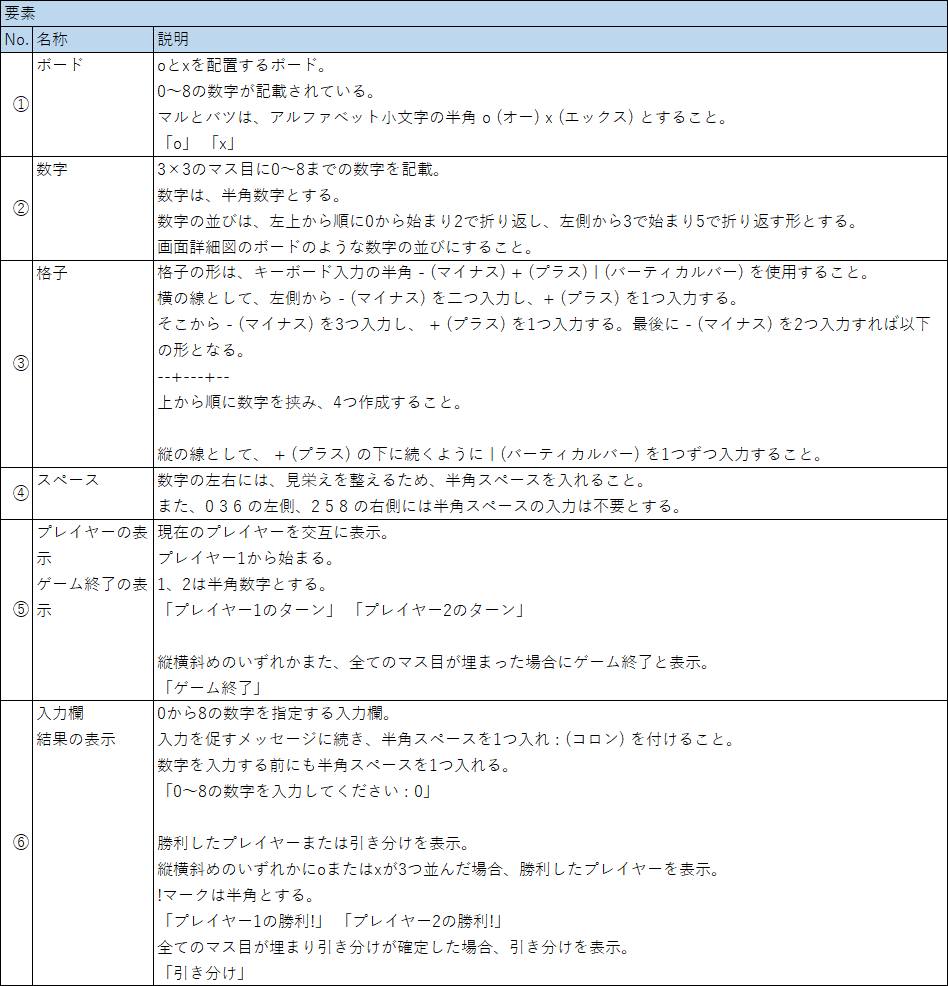
画面詳細図とその説明を以下に示す。

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。

### マクロ

PLAYER\_NUM=2 プレイヤーの人数

NAME\_LEN\_MAX=17 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=2 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

WIN=1 勝者がいる

DRAW=0 引き分け

### 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 |
| char[PLAYER\_NUM] | marks |
| 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ゲーム進行部分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文字列として取得し、それがゲームの仕様に適するか確認し、数値またはエラーに変換する。

### InputProcess（入力取得）関数

標準入力から1文字を取得し、それが0から8の数字であるかを確認し、数値かエラーマクロに変換する。

戻り値、ボードの位置を表す数字またはINPUT\_ERR

表 1 InputProcess関数で使用するローカル変数

|  |  |  |
| --- | --- | --- |
| 型 | 初期値 | 説明 |
| char[] | {‘\0’} | 入力文字列を格納する配列 |
| int | INPUT\_ERR | ボードの位置を表す数字 |

* プレイヤーの入力を1文字だけ受け付け、ローカル変数の配列に格納する。
* 意図しない入力を受け付けないために、バッファの読み飛ばしを行う。
* 入力文字列が数値以外、または9の場合、”0～8を入力してください。”と表示して改行、入力エラーマクロを返す。
* 入力文字列をボードの位置を表すint型の数値に変換し、変数に格納する。
* 戻り値を返す。

## 画面出力機能（game\_output.c）

ゲームの情報を画面に出力する。  
画面構成は3章に記載されている。

### DisplayBoard（ボードエリア表示）関数

3.3節の画面詳細の通りにボードを画面に表示する。

戻り値、なし

表 2 DisplayBoard関数で使用する引数

|  |  |  |
| --- | --- | --- |
| 型 | 初期値 | 説明 |
| char[][] | - | ボードを表す２次元配列 |
| int | 0 | ループのカウントと要素アクセスに利用する |

* ボードを表す2次元配列を、ループを利用して画面に出力する。
* 終了条件には、ボードの高さと幅のマクロを利用する。
* 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

3.3節の画面詳細の通りにプレイヤー名と入力欄を画面に表示する。

戻り値、なし

表 3 DisplayStatus関数で使用する引数

|  |  |  |
| --- | --- | --- |
| 型 | 初期値 | 説明 |
| char[] | - | プレイヤーの名前を表す文字列 |

* 3.3節の⑤に準じて、プレイヤー名を表示する。
* 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

3.3節の画面詳細の通りに勝者と入力欄を画面に表示する。

戻り値、なし

表 4 DisplayResult関数で使用する引数

|  |  |  |
| --- | --- | --- |
| 型 | 初期値 | 説明・初期値 |
| char[] | - | 勝者の名前を表す文字列 |
| int | - | WINかDRAWのマクロが入る |

3.3節の⑤に準じて、ゲーム終了の表示を行う。

3.3節の⑥に準じて、引き分けまたは勝者の名前の表示を行う。

* + 引数がDRAW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。

### BoardJudgement（ボード判定）関数

ボードの縦３列、横３行、斜め２本の計８ラインに対して、同じ記号が３つ並んでいるかを判定する。

戻り値、LINEまたはNO\_LINE

表 5 BoardJudgement関数で使用する引数・ローカル変数

|  |  |  |
| --- | --- | --- |
| 型 | 初期値 | 説明 |
| char[][] | - | ボードを表す２次元配列 |
| int | 0 | 同じ記号が並んでいる回数をカウントする |

* ループを利用して、計8ラインを走査する。
* ボードの1ラインに対して、同じ記号が並んでいるかをカウントする。
* カウントが3以上の場合、戻り値としてライン完成マクロを返す。
* 最後に、戻り値としてライン未完成マクロを返す。

## ゲーム進行（marubatsu\_game.c）

三目並べのゲームを進行するmain関数を実装する。

main関数は本章の3つの部分と関数呼び出しから構成される。

* 5.4.1項のゲーム初期化部分を配置する。
* 5.4.1項のゲームメインループ部分を配置する。
  + ゲームメインループ内に5.4.2項の入力待機部分を配置する。
* DisplayBoard関数を呼び出す。
* メインループのカウンタ変数をもとに勝敗の情報を取得し、それに応じてDisplayResult関数を呼び出す。
* プログラムが終了する。

以下にmain関数のフローチャートを示す。

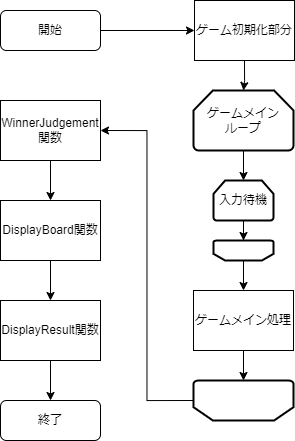
* 

図 1 main関数のフローチャート

### ゲーム初期化部分

ゲームの初期化を行う。

表 6 ゲーム初期化部分で初期化するGAME\_INFO型 構造体変数

|  |  |  |
| --- | --- | --- |
| 型 | メンバ変数名 | 説明・初期値 |
| 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 |

* game\_infoポインタ変数を表 6をもとに初期化する。

### ゲームメインループ部分

ゲームの進行に関する関数や処理を行う。

表 7 GameMainLoop関数で使用する変数

|  |  |  |
| --- | --- | --- |
| 型 | 初期値 | 説明 |
| int | NO\_LINE | ラインができたか(LINEかNO\_LINE)を格納する |
| int | INPUT\_ERR | 3.3節の②に準じたボードの位置を表す数字 |
| COORDINATES | {INPUT\_ERR,0} | ２次元配列の位置を表す |

* ターン数をカウンタ変数としてループを行う。
  + PLAYER\_INDEXマクロに現在のターン数を入れて、どちらのプレイヤーの番かを取得する。
  + DisplayBoard関数を呼び出す。
  + DisplayStatus関数を呼び出す。
  + 5.4.2項の入力待機部分を配置する。
  + 入力待機部分で取得した位置情報をもとに、ボードの該当要素に現在のプレイヤーの記号を代入する。
  + BoardJudgement関数を呼び出し、戻り値がLINEの場合、ループを抜ける。

### 入力待機部分

プレイヤーの入力から2次元配列の位置を表す構造体に変換する。

表 8 入力待機部分で使用する変数

|  |  |  |
| --- | --- | --- |
| 型 | 初期値 | 説明 |
| int | INPUT\_ERR | ボードの位置を表す |
| COORDINATES | {INPUT\_ERR,0} | ２次元配列の位置を表す |

* ローカル変数を初期化する。
* ボードの位置を表す変数がINPUT\_ERRと等しい限り、ループを続行する。
* DisplayBoard関数を呼び出す。
* DisplayStatus関数を呼び出す。
* InputProcess関数を呼び出し、ボードの位置を表す変数を更新する。
* INPUT\_ERRと等しい場合、その後の処理をスキップしてループを続行する。
* ボードの位置を表す変数の剰余と徐算の結果で、それぞれCOORDINATES構造体のメンバ変数x、yを更新する。
* 位置の構造体変数を使用して2次元配列にアクセスし、これが数字であるかを条件文とする条件分岐を行う。
* 条件式が負の場合、”既に埋まっています”とエラー表示して改行、ボードの位置を表す変数をINPUT\_ERRにする。