**三目並べ  
詳細仕様書**

Fチーム  
小島隆聖、柴田鈴花

目次

[目次 2](#_Toc168306098)

[1. 概要 3](#_Toc168306099)

[2. ファイル機能構成 3](#_Toc168306100)

[2.1 ソースファイル 3](#_Toc168306101)

[2.2 ヘッダーファイル 5](#_Toc168306102)

[2.3 ファイル構成図 7](#_Toc168306103)

[3. 関数説明 8](#_Toc168306104)

[3.1 ソースファイル内容 8](#_Toc168306105)

[3.2 ヘッダーファイル内容 15](#_Toc168306106)

[4. 画面構成 17](#_Toc168306107)

[4.1 盤面の表示方法 17](#_Toc168306108)

[4.2 画面遷移一覧 17](#_Toc168306109)

[4.3 画面遷移 18](#_Toc168306110)

# 概要

本仕様書は、C言語を用いた三目並べシステムのファイル機能構成と関数説明、画面構成を示したものである。

# ファイル機能構成

* 1. ソースファイル

1. 全体進行

|  |  |
| --- | --- |
| ソースファイル名 | main.c |
| 関数 | main、PlayerChange |
| 機能 | * メイン * プレイヤーを交代する |

1. 盤面入力機能

|  |  |
| --- | --- |
| ソースファイル名 | input.c |
| 関数 | BoardInitialize、PlayerInput、BoardUpdate |
| 機能 | * 3×３の9マス盤面の初期化 * プレイヤーの入力 * 盤面に記号を入力 |

1. リトライ機能

|  |  |
| --- | --- |
| ソースファイル名 | retry.c |
| 関数 | Retry |
| 機能 | * リトライor終了のキーボード入力 * リトライの場合 三目並べをリスタート 終了の場合 三目並べ終了 |

1. エラー機能

|  |  |
| --- | --- |
| ソースファイル名 | error.c |
| 関数 | InputErrorBoard、InputErrorRetry |
| 機能 | * 入力エラーがあるか判定 |

1. 盤面出力機能

|  |  |
| --- | --- |
| ソースファイル名 | board\_output.c |
| 関数 | BoardOutput |
| 機能 | * 現在の盤面情報を表示 |

1. 勝敗表示機能

|  |  |
| --- | --- |
| ソースファイル名 | result.c |
| 関数 | ResultJudge、ResultOutput |
| 機能 | * 三目並べの勝敗の判定 * 勝敗の表示 |

* 1. ヘッダーファイル

1. input.h

|  |  |
| --- | --- |
| ヘッダーファイル名 | input.h |
| プロトタイプ宣言  （関数名のみ記載） | BoardInitialize、PlayerInput、BoardUpdate |

1. retry.h

|  |  |
| --- | --- |
| ヘッダーファイル名 | retry.h |
| プロトタイプ宣言  （関数名のみ記載） | Retry |

1. error.h

|  |  |
| --- | --- |
| ヘッダーファイル名 | error.h |
| プロトタイプ宣言  （関数名のみ記載） | InputErrorBoard、InputErrorRetry |

1. board\_output.h

|  |  |
| --- | --- |
| ヘッダーファイル名 | board\_output.h |
| プロトタイプ宣言  （関数名のみ記載） | BoardOutput |

1. result.h

|  |  |
| --- | --- |
| ヘッダーファイル名 | result.h |
| プロトタイプ宣言  （関数名のみ記載） | ResultJudge、ResultOutput |

1. enum.h

|  |  |
| --- | --- |
| ヘッダーファイル名 | enum.h |
| プロトタイプ宣言  （関数名のみ記載） | (型名)  GAME\_JUDGE  (列挙子)  PLAYER１＿WIN、PLAYER2＿WIN、DRAW  (型名)  RETRY\_JUDGE  (列挙子)  RETRY、FINISH  (型名)  ERROR\_JUDGE  (列挙子)  ERROR、NONE\_ERROR  (型名)  PLAYER  (列挙子)  PLAYER1、PLAYER2  (マクロ定義)  #define INDEX 3  →配列の行と列を表す  #define MAX\_TURN 9  →最大ターン数を表す |

* 1. ファイル構成図

**ヘッダーファイル**

**①input.h  
（プロトタイプ宣言**（関数名のみ記載）**）**

├ BoardInitialize

├ PlayerInput

└ BoardUpdate

**②retry.h**

└ Retry

**③error.h**

├ InputErrorBoard

└ InputErrorRetry

**④board\_output.h**

└ BoardOutput

**⑤result.h**

├ ResultJudge

└ ResultOutput

**⑥enum.h（列挙体）（マクロ定義）**

・ GAME\_JUDGE

├── PLAYER1\_WIN

├── PLAYER2\_WIN

├── DRAW

└── CONTINUE

・ RETRY\_JUDGE

├── RETRY

└── FINISH

・ ERROR\_JUDGE

├── ERROR

└── NONE\_ERROR

・ PLAYER

├── PLAYER1

└── PLAYER2

・#define INDEX 3

・#define MAX\_TURN 9

**ソースファイル**

**(関数 )**

**①main.c**

├ main

└ PlayerChange

**②input.c**

├ BoardInitialize

├ PlayerInput

└ BoardUpdate

**③retry.c**

└ Retry

**④error.c**

├ InputErrorBoard

└ InputErrorRetry

**⑤board\_output.c**

└ BoardOutput

**⑥result.c**

├ ResultJudge

└ ResultOutput

# 関数説明

## ソースファイル内容

1. main.c（全体進行）
2. **BoardInitialize関数の呼び出し**→初期盤面マスの作成
3. **BoardOutput関数の呼び出し**→受け取った盤面を表示
4. **PlayerInput関数の呼び出し**→入力値を受け取る
5. **BoardUpdate関数の呼び出し**→char型配列[INDEX][INDEX]に  
   記号を格納
6. **ResultJudge関数の呼び出し**→読み取った盤面から勝敗判定
7. **勝利/引き分け判定がない場合**→PlayerChange関数を呼び出し後  
   ②に戻る
8. **勝利/引き分け判定がある場合**→BoardOutput関数を呼び出す
9. **ResultOutput関数の呼び出し**→勝敗結果を表示
10. **Retry関数の呼び出し**→リトライor終了を判定
11. **リトライの場合**→①に戻る
12. **終了の場合**→終了画面を表示してプログラム終了

③

⑥

⑦

②

⑧

①

④

⑤

⑨

⑪

⑩

ダイアグラム

自動的に生成された説明

CONTINUE

CONTINUE以外

FINISH

RETRY

* main関数

必要変数（変数名は任意）

* char型配列[INDEX][INDEX]（三目並べの盤面情報）
* int型変数（ターン数）
* PLAYER型変数（手番）

他に必要な変数がある場合は任意で宣言してよい。

（処理）

* 1. char型配列[INDEX][INDEX]で盤面を作成する。
  2. ターン数を示すint型変数に1を代入する。先手はPLAYER1とする。
  3. BoardInitialize関数を呼び出す。
  4. “～ 三目並べ ～”、“― ゲーム開始 ―”を画面表示する。

テキスト

中程度の精度で自動的に生成された説明

1. BoardOutput関数を呼び出す。
2. PlayerInput関数を呼び出す。
3. BoardUpdate関数を呼び出す。
4. ResultJudge関数を呼び出す。
5. ResultJudge関数の戻り値がCONTINUEであれば、PlayerChange関数を呼び出し後、2．に戻る。  
   ※２．～６．の処理を１ターンとする。２．に戻る度にターン数が増加する。
6. ResultJudge関数の戻り値がCONTINUE以外であればBoardOutput関数呼び出し後、ResultOutput関数を呼び出す。
7. Retry関数を呼び出す。
8. 戻り値がRETRYの場合、1.2．に戻る。
9. 戻り値がFINISHの場合、“― 終了 ―”を画面表示してプログラムを終了する。

テキスト が含まれている画像

自動的に生成された説明

* PlayerChange関数

（処理）

1. 引数としてPLAYER型定数を受け取る。
2. 定数を判定する。

* 引数のPLAYER型がPLAYER1の場合 PLAYER型PLAYER2を返す
* 引数のPLAYER型がPLAYER2の場合 PLAYER型PLAYER1を返す

1. input.c（盤面入力機能）

* BoardInitialize関数

（処理）

1. 引数(char型配列[INDEX][INDEX])を受け取る。
2. 盤面に下図の様に入力位置を示す数字(半角文字)を代入する。

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

* PlayerInput関数

（処理）

1. 引数としてPLAYER型定数、char型配列[INDEX][INDEX]を受け取る。
2. ”1～9の数字を入力してください”の文字を改行し、下図のように  
   画面表示する。

* PLAYER型定数がPLAYER1の場合 “プレイヤー１(o)　入力：”
* PLAYER型定数がPLAYER2の場合 “プレイヤー２(x)　入力：”

カレンダー

自動的に生成された説明

1. キーボードからの入力値（char型）を受け取る。
2. 入力値（char型）をInputErrorBoard関数に渡す。
3. InputErrorBoard関数の戻り値が

* ERRORが返ってきた場合  
  “エラー：適切な数字を入力してください”を画面表示して、  
  改行し空行を入れる。その後、2．に戻る。

テキスト が含まれている画像

自動的に生成された説明

* NONE\_ERRORが返ってきた場合char型配列[INDEX][INDEX]に  
  入力値（char型）と一致するものがあるか調べる。

一致する場合 入力値（char型）を返す。

一致しない場合 “エラー：適切な数字を入力してください”を画面表示して、改行し空行を入れる。  
その後2．に戻る。

テキスト が含まれている画像

自動的に生成された説明

* BoardUpdate関数

（処理）

1. 引数としてPLAYER型定数、入力済みの値（char型）、  
   char型配列[INDEX][INDEX]のポインタを受け取る。（戻り値なし）
2. char型配列[INDEX][INDEX]のポインタの各要素を参照し、  
   入力済みの値（char型）と一致した場合、その要素に  
   以下の記号を代入する。

* PLAYER型定数がPLAYER1の場合 ‘o’を代入する
* PLAYER型定数がPLAYER2の場合 ‘x’を代入する

※入力値（char型）が半角数字‘1～9’以外の場合  
→assert関数でプログラムの実行を終了する。

1. retry.c（リトライ機能）

* Retry関数

（処理）

1. 下図のように”リトライ…１,　終了…2”を改行後、“入力：”の文字を  
   画面表示する。

時計 が含まれている画像

自動的に生成された説明

1. キーボードからの入力値（char型）を受け取る。
2. 入力値（char型）をInputErrorRetry関数に渡す。
3. InputErrorRetry関数の戻り値が

* ERRORの場合  
  “エラー：適切な数字を入力してください”を画面表示して、改行し空行を入れる。その後、1．に戻る。

テキスト が含まれている画像

自動的に生成された説明

* NONE\_ERRORの場合、次の判定を行う。

入力値（char型）が半角数字‘1’の場合 戻り値RETRYを返す

入力値（char型）が半角数字‘2’の場合 戻り値FINISHを返す

1. error.c（エラー機能）

* InputErrorBoard関数

（処理）

1. 引数として入力値（char型）を受け取る。
2. 引数(入力値（char型）)を判定する。

* 入力値（char型）が半角数字‘1~9’以外の場合 ERRORを返す
* 入力値（char型）が半角数字‘1~9’である場合 NONE\_ERRORを返す
* InputErrorRetry関数

（処理）

1. 引数として入力値（char型）を受け取る。
2. 引数（入力値（char型））を判定する

* 入力値（char型）が半角数字‘1’、‘2’以外の場合 ERRORを返す
* 入力値（char型）が半角数字‘1’、‘2’である場合 NONE\_ERRORを返す

1. board\_output.c (盤面出力機能)

* BoardOutput関数

（処理）

1. 引数としてchar型配列[INDEX][INDEX]を受け取る。
2. 受け取った盤面を読み取り下図の様に画面表示する。

（盤面構成については4.1 盤面の表示方法にて記載）

グラフィカル ユーザー インターフェイス, アプリケーション

自動的に生成された説明

1. result.c (勝敗表示機能)

* ResultJudge関数

（処理）

1. 引数（char型配列[INDEX][INDEX]・PLAYER型定数・現在のターン数（int型））を受け取る。
2. 読み取ったchar型配列[INDEX][INDEX]を走査し、以下の順序で勝敗を判定する。(2.1→2.2→2.3)
   1. 縦・横・斜めが同一記号で3つ並んでいる場

* PLAYER1の場合 PLAYER1\_WINを返す
* PLAYER2の場合 PLAYER2\_WINを返す
  1. 現在のターン数（int型）が最大ターン数（MAX\_TURN）である場合

→DRAWを返す。

* 1. 2.1、2.2の条件を満たさなかった場合

→CONTINUEを返す。

* ResultOutput関数

（処理）

1. 引数としてGAME\_JUDGE型定数を受け取る。

* 引数がPLAYER1\_WINの場合  
  →“プレイヤー１の勝利！”を画面に表示する。

図形, 矢印

自動的に生成された説明

* 引数がPLAYER2\_WINの場合  
  →“プレイヤー２の勝利！”を画面に表示する。

図形, 矢印

自動的に生成された説明

* 引数がDRAWの場合  
  →“引き分け”を画面に表示する。

図形, 矢印

自動的に生成された説明

* 引数がCONTINUEの場合  
  →assert関数でプログラムの実行を終了する。

## ヘッダーファイル内容

①input.h

* BoardInitialize  
  （引数）char型配列[INDEX][INDEX]  
  （戻り値）void
* PlayerInput  
  （引数）PLAYER型、char型配列[INDEX][INDEX]  
  （戻り値）char型（入力値）
* BoardUpdate  
  （引数）PLAYER型、char型（入力値）、char型配列[INDEX][INDEX]  
  （戻り値）void

②retry.h

* Retry  
  （引数）void  
  （戻り値）RETRY\_JUDGE型

③error.h

* InputErrorBoard  
  （引数）char型（入力値）、  
  （戻り値）ERROR\_JUDGE型
* InputErrorRetry  
  （引数）char型（入力値）  
  （戻り値）ERROR\_JUDGE型

④board\_output.h

* BoardOutput  
  （引数）char型配列[INDEX][INDEX]  
  （戻り値）void

⑤result.h

* ResultJudge  
  （引数）char型配列[INDEX][INDEX]、PLAYER型、int型（ターン数）  
  （戻り値）GAME\_JUDGE型
* ResultOutput  
  （引数）GAME\_JUDGE型  
  （戻り値）void

⑥enum.h

（型名）

GAME\_JUDGE

（列挙子）

* PLAYER1\_WIN
* PLAYER2\_WIN
* DRAW
* CONTINUE

（型名）

RETEY\_JUDGE

（列挙子）

* RETRY
* FINISH

（型名）

ERROR\_JUDGE

（列挙子）

* ERROR
* NONE\_ERROR

（型名）

PLAYER

（列挙子）

* PLAYER1
* PLAYER2

（マクロ定義）

#define INDEX 3、#define MAX\_TURN 9

# 画面構成

### 盤面の表示方法

* 数字と記号は全て半角を使用する。
* 記号は小文字の‘o(オー)’と‘x(エックス)’を使用する。
* 盤面の構成として‘＋’、‘-’、‘|’を半角で作成する。
* 盤面は縦13行、横７行で構成されたものとする。
* マス内の数字と記号の両隣には半角スペースを使用する。

グラフィカル ユーザー インターフェイス, アプリケーション

自動的に生成された説明

**␣　␣**

### 画面遷移一覧

1. ゲーム開始＆プレイヤー１入力画面
2. プレイヤー１入力後＆プレイヤー２入力画面
3. プレイヤー２入力後画面
4. プレイヤー１勝利画面
5. プレイヤー２勝利画面
6. 引き分け画面
7. リトライ画面
8. エラー画面

### 画面遷移

1. テキスト

   中程度の精度で自動的に生成された説明ゲーム開始＆プレイヤー１入力画面

main関数（1.1.4）

↓

BoardOutput関数（2．）

↓

PlayerInput関数（2．）

カレンダー

自動的に生成された説明

1. プレイヤー１入力後＆プレイヤー２入力前画面

PlayerInput関数（2．）

↓

BoardOutput関数（2．）

↓

PlayerInput関数（2．）

カレンダー

中程度の精度で自動的に生成された説明

1. プレイヤー２入力後画面

テキスト

中程度の精度で自動的に生成された説明

PlayerInput関数（2．）

↓

BoardOutput関数（2．）

↓

PlayerInput関数（2．）

1. プレイヤー１勝利画面

ダイアグラム, 図形, 矢印

自動的に生成された説明

BoardOutput関数（2．）

↓

ResultOutput関数

1. プレイヤー２勝利画面

BoardOutput関数（2．）

↓

ResultOutput関数

図形, 矢印

自動的に生成された説明

1. 引き分け画面

BoardOutput関数（2．）

↓

ResultOutput関数

図形, 矢印

自動的に生成された説明

1. リトライ画面

Retry関数（１．）

↓

main関数（10．）

テキスト が含まれている画像

自動的に生成された説明

1. エラー画面

テキスト

自動的に生成された説明

Retry関数（4．）

PlayerInput関数（5．）

テキスト

自動的に生成された説明

エラー表示後**改行**し、**空行**をいれる