**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](#_Toc168572040)

[1. 概要 3](#_Toc168572041)

[2. ファイル機能構成 3](#_Toc168572042)

[2.1 ソースファイル 3](#_Toc168572043)

[2.2 ヘッダーファイル 5](#_Toc168572044)

[2.3 ファイル構成図 7](#_Toc168572045)

[3. 関数説明 8](#_Toc168572046)

[3.1 ソースファイル内容 8](#_Toc168572047)

[3.2 ヘッダーファイル内容 19](#_Toc168572048)

[4. 画面構成 22](#_Toc168572049)

[4.1 盤面の表示方法 22](#_Toc168572050)

[4.2 画面遷移図 23](#_Toc168572051)

# 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 |
| 機能 | * メイン |

1. 入力機能

|  |  |
| --- | --- |
| ソースファイル名 | input.c |
| 関数 | Input、InputRetry |
| 機能 | * プレイヤー１、プレイヤー２の入力 * リトライ入力 |

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終了の判定 * 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 | ErrorInput、ErrorInputRetry、ErrorBoardUpdate |
| 機能 | * 入力エラーがあるか判定 |

1. リザルト機能

|  |  |
| --- | --- |
| ソースファイル名 | result.c |
| 関数 | ResultJudge、ResultOutput |
| 機能 | * 勝敗判定 * 勝敗出力 |

1. 盤面更新機能

|  |  |
| --- | --- |
| ソースファイル名 | board.c |
| 関数 | BoardInitialize、BoardUpdate、BoardOutput |
| 機能 | * 盤面の初期化 * 盤面の更新 * 盤面の出力 |

1. 手番交代機能

|  |  |
| --- | --- |
| ソースファイル名 | player\_change.c |
| 関数 | PlayerChange |
| 機能 | * プレイヤー1とプレイヤー2の手番交代 |

* 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） | Input、InputRetry |

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） | ErrorInput、ErrorInputRetry、ErrorBoardUpdate |

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. board.h

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

1. player\_change.h

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

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、CONTINUE  (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 |

* 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）**）**

├ Input

└ InputRetry

**②retry.h**

└ Retry

**③error.h**

├ ErrorInput

├ ErrorInputRetry

└ ErrorBoardUpdate

**④result.h**

├ ResultJudge

└ ResultOutput

**⑤board.h**

├ BoardInitialize

├ BoardUpdate

└ BoardOutput

**⑥player\_change.h**

└ PlayerChange

**⑦enum.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

**ソースファイル**

**(関数 )**

**①main.c**

└ main

**②input.c**

├ Input

└ InputRetry

**③retry.c**

└ Retry

**④error.c**

├ ErrorInput

├ ErrorInputRetry

└ ErrorBoardUpdate

**⑤result.c**

├ ResultJudge

└ ResultOutput

**⑥board.c**

├ BoardInitialize

├ BoardUpdate

└ BoardOutput

**⑦player\_change.c**

└ PlayerChange

# 関数説明

### ソースファイル内容

1. main.c（全体進行）

（フローチャート）

ダイアグラム

自動的に生成された説明

* 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）
* char型変数（プレイヤーの入力値）
* 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（int型）を０、手番をPLAYER1で初期化する。
  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. ターン数（int型）を1増やす。
2. BoardOutput関数を呼び出す。
3. Input関数を呼び出す。
4. ErrorInput関数を呼び出す。
5. ErrorInput関数の戻り値がERRORであれば、4．に戻る。  
   また、NONE\_ERROEであれば7．に進む。
6. ErrorBoardUpdate関数を呼び出す。
7. ErrorBoardUpdate関数の戻り値がERRORであれば、4．に戻る。  
   また、NONE\_ERROEであれば9．に進む。
8. BoardUpdate関数を呼び出す。
9. ResultJudge関数を呼び出す。
10. 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。また、CONTINUE以外であれば12．に進む。
11. BoardOutput関数呼び出し後、ResultOutput関数を呼び出す。
12. InputRetry関数を呼び出す。
13. ErrorInputRetry関数を呼び出す。
14. ErrorInputRetry関数の戻り値がERRORであれば、13．に戻る。  
    また、NONE\_ERROEであれば16．に進む。
15. Retry関数を呼び出す。
16. Retry関数の戻り値がRETRYの場合、1.2．に戻る。  
    また、FINISHであれば18．に進む。
17. “― 終了 ―”を画面表示してプログラムの実行を終了する。

テキスト が含まれている画像

自動的に生成された説明

1. input.c（入力機能）

* Input関数

1. 引数としてPLAYER型変数を受け取る。
2. 入力値を受け取るchar型変数を作成し、‘\0’を代入する。
3. “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型変数がPLAYER1の場合 “プレイヤー１(o)　入力：”
* PLAYER型変数がPLAYER2の場合 “プレイヤー２(x)　入力：”
* PLAYER型変数が‘0’、‘1’以外の場合、exit関数でプログラムの実行を終了する

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の先頭1文字を受け取り、char型変数に代入する。  
   例）‘abc’が入力された場合 ‘a’を受け取る
2. 入力バッファの読み飛ばしを行う。バッファに入力値が残っていた場合  
   char型変数に‘\0’を代入する。
3. 戻り値として入力値（char型）を返す。

* InputRetry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. 入力値を受け取るchar型変数を作成し、‘\0’を代入する。
2. キーボードからの入力値の先頭一文字を受け取り、char型変数に代入する。  
   例）‘abc’が入力された場合 ‘a’を受け取る
3. 入力バッファの読み飛ばしを行う。バッファに入力値が残っていた場合  
   char型変数に‘\0’を代入する。
4. 入力値（char型）を返す。
5. retry.c（リトライ機能）

* Retry関数

（処理）

1. 引数として入力値（char型）を受け取る。  
   ※入力の先頭一文字だけ受け取る。  
   例）「abc」が入力された場合 「a」を受け取る
2. 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

※引数の入力値（char型）が半角数字‘1’、‘2’以外の場合、assert関数で  
プログラムの実行を終了する。

1. error.c（エラー機能）

* ErrorInput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  
  “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ERRORを返す

テキスト が含まれている画像

自動的に生成された説明

* 入力値（char型）が半角数字‘1～9’である場合NONE\_ERRORを返す
* ErrorInputRetry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  
  “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ERRORを返す

テキスト が含まれている画像

自動的に生成された説明

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

（処理）

1. 引数として入力値（char型）とchar型配列[INDEX][INDEX]を受け取る。
2. char型配列[INDEX][INDEX]の各要素を走査し、入力値（char型）と一致するものがあるか判定する。

* 一致するものがない場合

“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ERRORを返す。

テキスト が含まれている画像

自動的に生成された説明

* 一致するものがある場合

NONE\_ERRORを返す

※引数の入力値（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. result.c (リザルト機能)

* ResultJudge関数

（処理）

1. 引数（char型配列[INDEX][INDEX]・PLAYER型変数・  
   ターン数（int型））を受け取る。

※ターン数（int型）＜１の場合、

ターン数（int型）＞最大ターン数（INDEX\*INDEX）の場合

→assert関数でプログラムの実行を終了する

1. 読み取ったchar型配列[INDEX][INDEX]を走査し、以下の順序(2.1→2.2→2.3)で勝敗を判定する。
   1. 縦・横・斜めが同一記号('o'、'x'のみ)で3つ並んでいる場合

* PLAYER1の場合 PLAYER1\_WINを返す
* PLAYER2の場合 PLAYER2\_WINを返す
  1. ターン数（int型）が最大ターン数（INDEX \* INDEX）  
     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。

* 1. GAME\_JUDGE型変数が‘0～3’以外の場合  
     →exit関数でプログラムの実行を終了する
* ResultOutput関数

（処理）

1. 引数としてGAME\_JUDGE型変数を受け取る。

* 引数がPLAYER1\_WINの場合  
  →“プレイヤー１の勝利！”を画面表示後、改行する

（‘１’と‘！’は全角を使用）

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！”を画面表示後、改行する

（‘２’と‘！’は全角を使用）

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”を画面表示後、改行する

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
* GAME\_JUDGE型変数が‘0～3’以外の場合  
  →exit関数でプログラムの実行を終了する

1. board.c (盤面更新機能)

* BoardInitialize関数

（処理）

1. 引数（char型配列[INDEX][INDEX]ポインタ）を受け取る。
2. char型配列[INDEX][INDEX]ポインタの各要素が下図のようになるよう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 |

* BoardUpdate関数

（処理）

1. 引数としてPLAYER型変数、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型変数がPLAYER1の場合 ‘o’を代入する
* PLAYER型変数がPLAYER2の場合 ‘x’を代入する

※PLAYER型変数が‘0’、‘1’以外の場合  
→exit関数でプログラムの実行を終了する

※入力済みの値（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。

* 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ように画面表示する。

（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. player\_change.c（手番交代機能）

* PlayerChange関数

（処理）

1. 引数としてPLAYER型変数を受け取る。
2. PLAYER型変数を判定する。

* 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

※PLAYER型変数が‘0’、‘1’以外の場合  
→exit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

* Input  
  （引数）PLAYER型  
  （戻り値）char型（入力値）
* InputRetry  
  （引数）なし  
  （戻り値）char型（入力値）

②retry.h

* Retry  
  （引数）char型（入力値）  
  （戻り値）RETRY\_JUDGE型

③error.h

* ErrorInput  
  （引数）char型（入力値）  
  （戻り値）ERROR\_JUDGE型
* ErrorInputRetry  
  （引数）char型（入力値）  
  （戻り値）ERROR\_JUDGE型
* ErrorBoardUpdate

（引数）char型（入力値）  
（戻り値）ERROR\_JUDGE型

④result.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

⑤board.h

* BoardInitialize  
  （引数）char型配列[INDEX][INDEX]ポインタ  
  （戻り値）void
* BoardUpdate  
  （引数）PLAYER型、char型（入力済みの値）  
  （戻り値）void
* BoardOutput  
  （引数）char型配列[INDEX][INDEX]  
  （戻り値）void

⑥player\_change.h

* PlayerChange  
  （引数）PLAYER型  
  （戻り値）PLAYER型

⑦enum.h

（型名）

GAME\_JUDGE

（列挙子）

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

（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

# 画面構成

### 盤面の表示方法

* 表示する盤面は下図のような画面構成とする。
* 使用する数字と記号、スペースは全て半角とする。
* 記号は小文字の‘o(オー)’、‘x(エックス)’と、‘＋’、‘-’、‘|’を使用する。
* 三目並べ盤面の枠線は"+---+---+---+"の合計13文字と、'|'をスペース3文字分空けて合計4文字置くことで作成する。
* 盤面のマスの内部はスペース3文字のうち左から2文字目のスペース位置に数字または小文字のプレイヤー記号('o'、'x')を置いて表示する。
* マス内の数字またはプレイヤー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

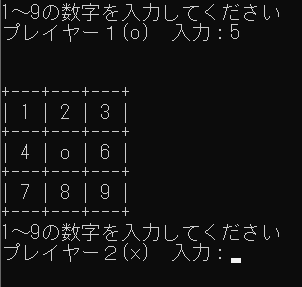
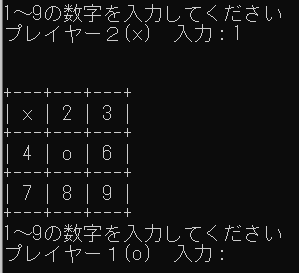
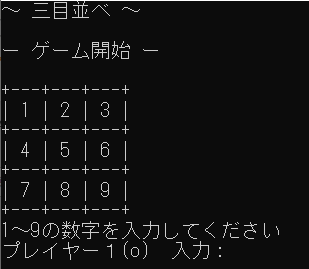
‘-’３つの両隣に‘+’１つ表示

半角スペースの隣に‘|’１つ表示

**␣ ␣**

### 画面遷移図

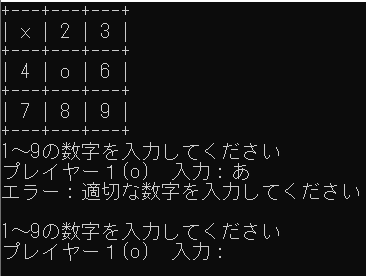
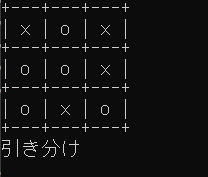
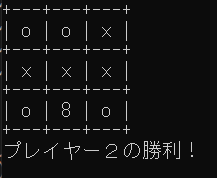
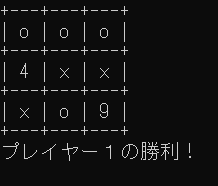
**＿**…入力後部分、**＿**…エラー文字部分



* ゲーム開始から勝敗表示までの流れ

プレイヤー１が  
**入力**したとき

プレイヤー２が  
**入力**したとき

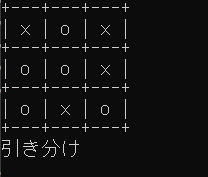
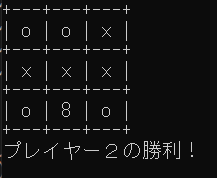
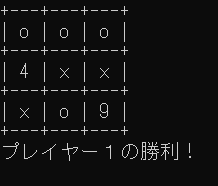


引き分けのとき

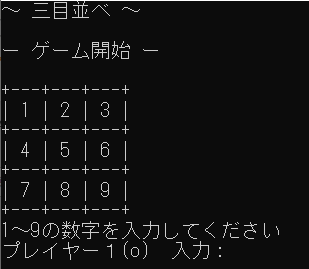
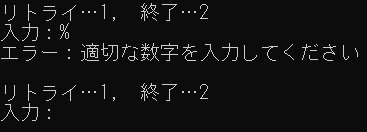
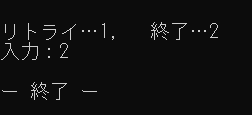
プレイヤー２が  
勝利したとき

プレイヤー１が  
勝利したとき

入力エラーのとき



* 勝敗表示からリトライ入力までの流れ



**2を入力**したとき

入力エラーのとき

**1を入力**したとき

ゲーム開始に戻る