三目並べ　詳細仕様書

Mチーム

堀内敦遥・千葉孔鳳

2024

目次

[1 概要 1](#_Toc167884427)

[2 三目並べの全体の処理の流れ 2](#_Toc167884428)

[3 三目並べの画面構成 4](#_Toc167884429)

[4 三目並べの機能構成 5](#_Toc167884430)

[4.1 プレイヤーの名前を入力する 5](#_Toc167884431)

[4.2 先攻/後攻と使用するマークを表示する 6](#_Toc167884432)

[4.3 マス全体の状態を表示して終了判定を行う 7](#_Toc167884433)

[4.4 マークを書き込みたいマスを入力し、マークを書き込む 9](#_Toc167884434)

[4.5 結果を表示する 11](#_Toc167884435)

[5 三目並べのファイル構成とソースファイルの内容 13](#_Toc167884436)

[5.1 main.cの内容 14](#_Toc167884437)

[5.2 board\_func.cの内容 14](#_Toc167884438)

[5.3 board\_func.hの内容 14](#_Toc167884439)

[6 三目並べで使う列挙型 15](#_Toc167884440)

[6.1 マスの状態を表す型 15](#_Toc167884441)

[6.2 終了判定用の型 15](#_Toc167884442)

[7 三目並べで使う関数 16](#_Toc167884443)

[7.1 マスの番号を入力する関数 16](#_Toc167884444)

[7.2 マスの状態を表示する関数 16](#_Toc167884445)

[7.3 三目並べの終了判定を行う関数 17](#_Toc167884446)

[8 三目並べで使う変数 18](#_Toc167884447)

[8.1 マスの状態を格納する二次元配列 18](#_Toc167884448)

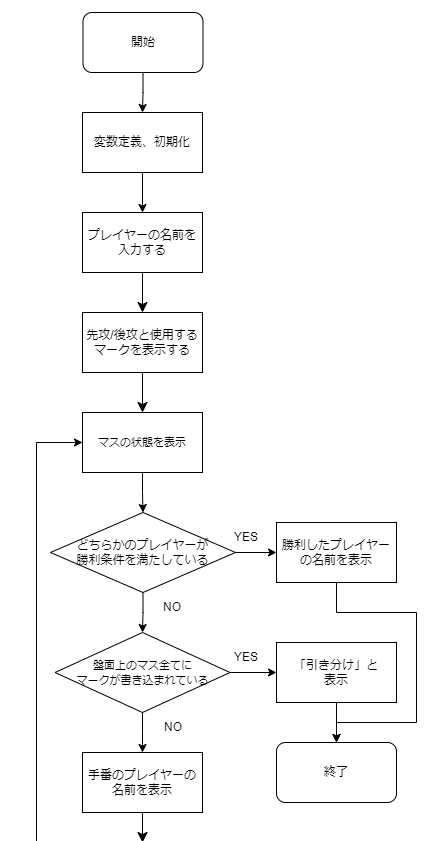
[9 main関数の流れ 19](#_Toc167884449)

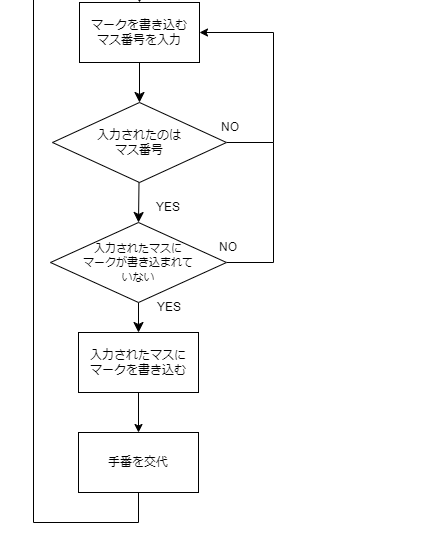
# 概要

本仕様書には、三目並べの画面・機能・ファイルそれぞれの構成とプログラム内で使用する列挙型・関数・変数の説明を記述している。なお以降の内容はC言語での開発を想定している。

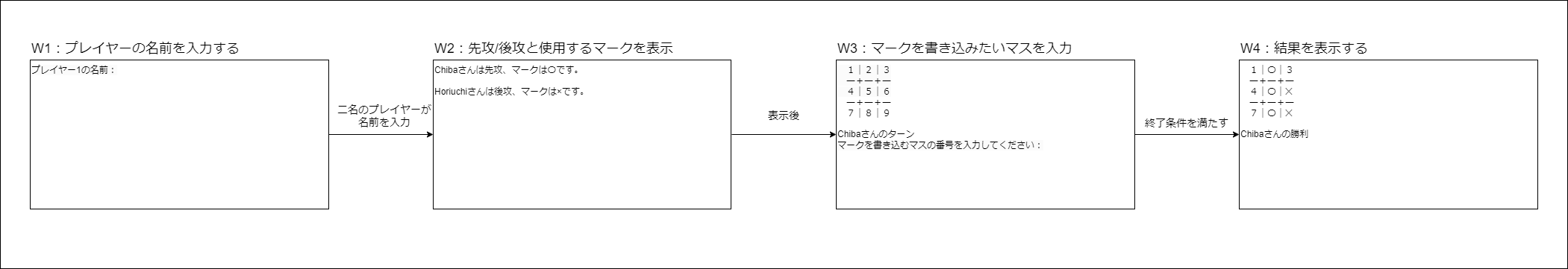
# 三目並べの全体の処理の流れ

三目並べの処理のフローチャートは以下のとおりである





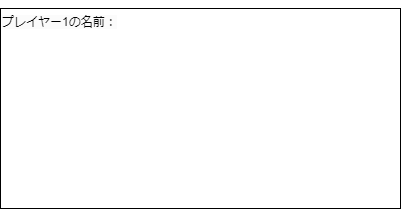
# 三目並べの画面構成

三目並べの画面遷移図は以下のとおりである

# 三目並べの機能構成

三目並べゲームは縦3マス×横3マスのものを使用する

## プレイヤーの名前を入力する



機能名：プレイヤーの名前を入力

処理内容：

・「プレイヤー１の名前：」と表示する

・プレイヤーの名前の入力を求める

・プレイヤーが入力した文字を配列に格納する

・その後、文言を「プレイヤー２の名前：」に変更して、もう一人のプレイヤーにも同じ動作を行う

・プレイヤーの名前を入力する際の**文字数の上限は半角英数字*で*15文字**とする

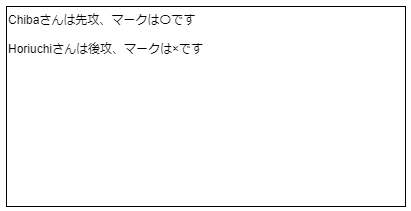
・文字数の上限を超えると「名前は15文字以内で入力してください」と表示しもう一度入力を求める

・全角文字が入力された場合は「名前は半角英数字で入力してください」と表示しもう一度入力を求める

・何も入力していない場合は「名前を入力してください」と表示しもう一度入力を求める

・以降最初に名前を入力したプレイヤーを「プレイヤー１」、次に入力したプレイヤーを「プレイヤー２」とする

## 先攻/後攻と使用するマークを表示する



機能名：先攻/後攻と使用するマークを表示する

処理内容：

・「(プレイヤー１の名前)さんは先攻、マークは〇です」と表示する

・「(プレイヤー２の名前)さんは後攻、マークは×です」と表示する

・プレイヤー1は先攻でマークは「〇」、プレイヤー２は後攻でマークは「×」とする

## マス全体の状態を表示して終了判定を行う

ダイアグラム

自動的に生成された説明

図形

自動的に生成された説明

機能名：マスの状態を表示して終了判定を行う

処理内容：

・マスの状態を参照し、表示する

・この時、**マークが書き込まれていないマスにはマス番号を表示する**

・マスの番号は全角の「１」～「９」、マスの区切り線は全角の「｜」「―」「＋」、マークは記号の「○」と「×」を使用する

勝利プレイヤーがいるかどうかを確認

・縦・横・斜めいずれかの列に同じマークが３つ並んでいるかを確認する

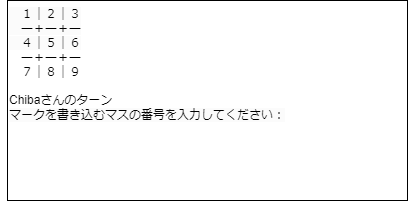
引き分けかどうかを確認

・すべてのマスにマークが書き込まれているかどうかを確認する

## マークを書き込みたいマスを入力し、マークを書き込む

ダイアグラム

自動的に生成された説明



機能名：マークを書き込みたいマスを入力し、マークを書き込む

処理内容：

・ゲームの開始時のプレイヤーは**プレイヤー１**とする

・「（プレイヤーの名前）さんのターン」と表示する

・「マークを書き込むマスの番号を入力してください：」と表示する

・マークを書き込みたいマスの番号を入力させる

・この時、**文字数の上限は半角英数字で1文字**とする

・マスの番号は半角数字で「1」～「9」とする

・マスの番号以外が入力された場合に「1~9のマス番号を入力してください：」というエラーを表示

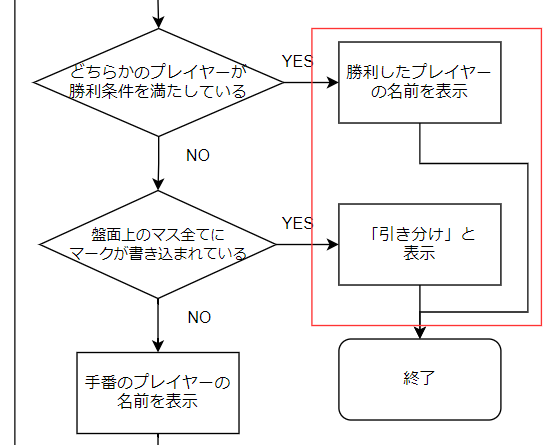
・すでに書き込まれていたマスを指定した場合の「マークが書き込まれていないマス番号を入力してください：」というエラーを表示

・エラーを表示したときに再度、書き込みたいマスの番号を入力

・プレイヤーが入力したマスにマークを書き込む

・手番のプレイヤーを交代する

## 結果を表示する

機能名：結果の表示

テキスト

低い精度で自動的に生成された説明

・勝利プレイヤーがいる場合

処理内容：

・勝利プレイヤーの名前を参照し、「（プレイヤーの名前）さんの勝利」と表示する

図形

自動的に生成された説明

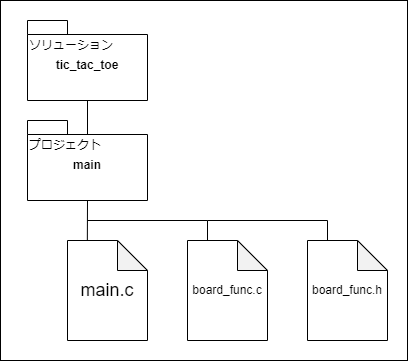
・引き分けの場合

処理内容：

・「引き分けです」と表示する

# 三目並べのファイル構成とソースファイルの内容

ファイル構成は以下のとおりである



## main.cの内容

main関数を記述する

## board\_func.cの内容

以下に示す関数の定義を記述する

ただし後述するboard\_func.hをインクルードすること

・関数ScanInput

・関数PrintBoard

・関数CheckBoard

## board\_func.hの内容

board\_func.cに記述した関数のプロトタイプ宣言および以下に示す列挙型の定義を記述する

・列挙型MARK

・列挙型RESULT

# 三目並べで使う列挙型

## マスの状態を表す型

型名：MARK

列挙子：｛BLANK, PLAYER1\_MARK, PLAYER2\_MARK｝

説明：

・マスにマークが書き込まれていない場合、BLANK。

・「〇」が書き込まれている場合、PLAYER1\_MARK

・「×」が書き込まれている場合、PLAYER2\_MARK

## 終了判定用の型

型名：RESULT

列挙子：｛NONE, PLAYER1\_WIN, PLAYER2\_WIN, DRAW｝

説明：

・終了条件を満たさない場合、NONE

・「〇」を書き込むプレイヤーが勝利した場合、PLAYER1\_WIN

・「×」を書き込むプレイヤーが勝利した場合、PLAYER2\_WIN

・引き分けの場合、DRAW

# 三目並べで使う関数

## マスの番号を入力する関数

関数名：ScanInput

引数：void

戻り値：int

処理内容：

マークを書き込みたいマスの番号を入力させる

この時、文字数の上限は半角英数字で1文字とし、入力内容をプレイヤーが確認できるようにする

マスの番号は半角数字で「1」～「9」とする

マスの番号以外が入力された場合は「1~9のマス番号を入力してください：」というエラーを表示し、再度入力させる

## マスの状態を表示する関数

関数名：PrintBoard

引数：マスの状態を持つMARK型3×3二次元配列

戻り値：void

処理内容：マス全体を4.3で示したように、マスの状態を持つ二次元配列を参照して表示する

マスの番号は全角の「１」～「９」、

マスの区切り線は全角の「｜」「―」「＋」、

マークはPLAYER1\_MARKのとき「○」、PLAYER2\_MARKのとき「×」を使用する。BLANKのときはマス番号を表示する。

空白は用いない

## 三目並べの終了判定を行う関数

関数名：CheckBoard

引数：マスの状態を持つMARK型3×3二次元配列

戻り値：RESULT型

処理内容：マスの状態を持つ二次元配列を参照して、

PLAYER1\_MARKが揃っているときはPLAYER1\_WIN、

PLAYER2\_MARKが揃っているときはPLAYER2\_WIN、

引き分けのときはDRAW、それ以外はNONEを返す

# 三目並べで使う変数

**三目並べで使う変数の名前は指定しない**

## マスの状態を格納する二次元配列

マスの状態を格納する配列は、MARK型3x3配列とし、すべての要素をBLANKで初期化する

# main関数の流れ

