三目並べ詳細仕様書

Iチームメンバー

杉友泰宣　藤井健

目次

[1. 概要 4](#_Toc167190876)

[2. 画面遷移図 4](#_Toc167190877)

[3. システム機能フロー図 5](#_Toc167190878)

[3.1 フロー説明 5](#_Toc167190879)

[3.1.1 [プレイヤー名を入力] 5](#_Toc167190880)

[3.1.2 [プレイヤー名は10文字以内ですか] 5](#_Toc167190881)

[3.1.3 [盤面を初期化] 5](#_Toc167190882)

[3.1.4 [3×3のマスとプレイヤー名と先後を画面出力] 6](#_Toc167190883)

[3.1.5 [縦横のマス数を入力] 6](#_Toc167190884)

[3.1.6 [入力した値は規定値内か] 6](#_Toc167190885)

[3.1.7 [既に入力されていないか] 6](#_Toc167190886)

[3.1.8 [三目並べの画面のマスに「〇」「×」を出力] 7](#_Toc167190887)

[3.1.9 [一直線がすべて「〇」、または「×」ですか] 7](#_Toc167190888)

[3.1.10 [出力したのは9回目ですか] 7](#_Toc167190889)

[3.1.11 [結果を出力] 7](#_Toc167190890)

[3.1.12 [もう一度遊びますか] 8](#_Toc167190891)

[4. ファイル構成図 9](#_Toc167190892)

[5. 機能構成 9](#_Toc167190893)

[5.1 tic\_tac\_toe.c 9](#_Toc167190894)

[プレイヤー名入力についてのループ 10](#_Toc167190895)

[三目並べを行う前の処理 10](#_Toc167190896)

[縦横の入力についてのループ 10](#_Toc167190897)

[〇×出力に関するループ 10](#_Toc167190898)

[ゲームを終了するループ 11](#_Toc167190899)

[5.2 game\_ready.c 11](#_Toc167190900)

[NameRange()関数 11](#_Toc167190901)

[InitSquare()関数 11](#_Toc167190902)

[PreScreen()関数 11](#_Toc167190903)

[5.3 get\_input.c 12](#_Toc167190904)

[SquareNumber ()関数 12](#_Toc167190905)

[NumRange()関数 12](#_Toc167190906)

[MarkChecker()関数 12](#_Toc167190907)

[5.4 get\_output.c 13](#_Toc167190908)

[PrintMark()関数 13](#_Toc167190909)

[Bingo()関数 13](#_Toc167190910)

[SquareFull()関数 13](#_Toc167190911)

[5.5 get\_result.c 13](#_Toc167190912)

[GameResult()関数 13](#_Toc167190913)

[PlayAgain()関数 14](#_Toc167190914)

[5.6 define.h 14](#_Toc167190915)

# 概要

本詳細仕様書は、二人のプレイヤーが交互に縦横それぞれ1～3の数字を入力し、マスに「〇」または「×」を出力する三目並べの詳細を示すものである。

# 画面遷移図

ダイアグラム

自動的に生成された説明

# システム機能フロー図

グラフ, ダイアグラム, レーダー チャート

自動的に生成された説明

### フロー説明

#### [プレイヤー名を入力]

プレイヤー名をtic\_tac\_toe.cで入力する。

先に入力したプレイヤー名が1P、先攻、「〇」となり、後に入力したプレイヤー名が2P、後攻、「×」となる。

#### [プレイヤー名は10文字以内ですか]

入力したプレイヤー名が10文字以上であれば10文字以内で入力してくださいを出力し[3.1.1](#_[プレイヤー名を入力])に戻る。また10文字以内であれば[3.1.3](#_[盤面を初期化])に移行する。

#### [盤面を初期化]

3×3のマスをすべて空白にし、[3.1.4](#_[先攻後攻をランダムで決定])に移行する。

#### [3×3のマスとプレイヤー名と先後を画面出力]

「〇」「×」を出力するための3×3のマスと、[3.1.1](#_[プレイヤー名を入力])で入力したプレイヤー名、[3.1.4](#_[先攻後攻をランダムで決定])で決定された先攻後攻と「〇」「×」が出力し、[3.1.5](#_[縦横のマス数を入力])へ移行する。

ダイアグラム

自動的に生成された説明

#### [縦横のマス数を入力]

「〇」「×」を出力した回数をもとに、（1P or 2P）の番です。と出力し

縦の値を半角で1～3の整数値で入力してください。

横の値を半角で1～3の整数値で入力してください。を出力し、プレイヤーが数値を入力する。

#### [入力した値は規定値内か]

[3.1.5](#_[縦横のマス数を入力])で入力した数値が1～3の内の半角整数値であるかを判別し、1～3以内の半角整数値でなければ半角で1～3の以内の整数値を入力してください。を出力し、[3.1.5](#_[縦横のマス数を入力])に移行し、1～3以内であれば[3.1.7](#_[既に入力されていないか])に移行する。

#### [既に入力されていないか]

入力した数値の位置が既に埋まっていないかを判別し、入力されているなら入力済みのマスです。他のマスを選んでください。を出力し[3.1.5](#_[縦横のマス数を入力])に移行し、入力されてなければ[3.1.8](#_[「〇」「×」を交互に出力])に移行する。

#### [三目並べの画面のマスに「〇」「×」を出力]

三目並べの画面のマスに「〇」「×」を出力した回数をもとに、偶数回数は「〇」、奇数回数は「×」を出力し、[3.1.9](#_[一直線がすべて「〇」、または「×」ですか])に移行する。

ダイアグラム

自動的に生成された説明

#### [一直線がすべて「〇」、または「×」ですか]

縦、横、斜めのいずれかに三つの「〇」もしくは「×」が並んでいるかを判別し、並んでいなければ[3.1.10](#_[出力したのは9回目ですか])に移行し、並んでいれば[3.1.11](#_[勝ったプレイヤー名を出力])に移行する。

#### [出力したのは9回目ですか]

「〇」と「×」が3×3のマス全てが埋まってあるかを判別し埋まってなければ[3.1.5](#_[縦横のマス数を入力])に移行し、埋まっていれば[3.1.11](#_[結果を出力])に移行する。

#### [結果を出力]

どちらかが勝利した場合、「勝利したプレイヤー名」が勝ちました！を出力し、[3.1.13](#_[もう一度遊びますか])に移行する。引き分けだった場合、引き分けでした。を出力し、[3.1.13](#_[もう一度遊びますか])に移行する。

#### [もう一度遊びますか]

もう一度遊ぶ…0／やめる…0以外を出力する。

0を入力で「もう一度遊ぶ」、0以外を入力で「やめる」を選択する。

0を選択すると、[3.1.1](#_[プレイヤー名を入力])に移行する。

0以外を選択すると、ゲームを終わる。

# ファイル構成図

テーブル

自動的に生成された説明

# 機能構成

## tic\_tac\_toe.c

main関数を記述する

プレイヤー名を入力する配列を宣言。

先に入力したプレイヤー名が1P、先攻、「〇」となり、後に入力したプレイヤー名が2P、後攻、「×」となる。

縦横の値を入力する一次元配列を宣言。

出力した回数をカウントする変数の初期値を０で宣言。

「〇」「×」を出力する二次元配列の初期値をすべて空白で宣言。

###### プレイヤー名入力についてのループ

ループ内では全角半角スペース問わず１０文字以内での入力であるかNameRange()関数を用いて判定しTRUEの時終了する。

###### 三目並べを行う前の処理

InitSquare()関数、PreScreen()関数の順で関数を呼び出し、処理を行う。

###### 縦横の入力についてのループ

入力の判定を通るまで回すループ

ループ内ではSquareNumber ()関数、NumRange()関数、MarkChecker()関数の順で関数を呼び出し、処理を行う。

NumRange()関数、MarkChecker()どちらもTRUEの時ループを終了する。

###### 〇×出力に関するループ

全てのマスに「〇」「×」が出力されるもしくは縦横斜めで３つ「〇」「×」が出力されるまで回すループ

ループ回数をカウントし偶数回「〇」奇数回「×」を出力するように切り替える。

PrintMark()関数で「〇」「×」を含んだ三目並べ画面を出力する。

Output\_Count()関数がDROW、またはBingo()関数がP1\_WIN、またはP2\_WINのときループが終了する。

###### ゲームを終了するループ

GameResult()関数を用いて勝敗を表示し、PlayAgain()関数を用いてENDの時はゲームを終了しCONTINUEの時はプレイヤー名入力ループから同様に始まる。

## game\_ready.c

###### NameRange()関数

引数：プレイヤー名の情報をもつ配列 戻り値：TRUEまたはFALSE

説明：10文字以内であるかを判別し、10文字以下であれば戻り値はTRUEとなり、11文字以上であれば[3.1.2](#_[プレイヤー名は10文字以内ですか])の警告文を出力し、戻り値はFALSEとなる。

InitSquare()関数

引数：二次元配列のポインタ 戻り値：なし

説明：配列の要素をすべて空白で初期化する。

###### PreScreen()関数

引数：二次元配列、プレイヤー名 戻り値：なし

説明：「〇」「×」を出力するための3×3のマスとそのマスごとに空の二次元配列、入力したプレイヤー名、先攻後攻と「〇」「×」が出力する。

## get\_input.c

###### SquareNumber ()関数

引数：要素数２の一次元配列のポインタ、 戻り値：なし

説明：縦と横のマス数を配列に入力する。

###### NumRange()関数

引数：要素数２の一次元配列 戻り値：TRUEまたはFALSE

説明：配列に入力した数字が半角整数値1～3の範囲内かどうかを判別し、範囲内であればTRUE、範囲外であれば[3.1.6](#_[入力した値は規定値内か])の警告文を出力しFALSEを返す。

###### MarkChecker()関数

引数：要素数２の一次元配列、二次元配列 戻り値：TRUEまたはFALSE

説明：プレイヤーが指定した配列の要素番号に「〇」「×」が既に入力されてあるときは[3.1.7](#_[既に入力されていないか])の警告文を出力しFALSEを返す、またされてないときはTRUEを返す。

## get\_output.c

###### PrintMark()関数

引数：要素数２の一次元配列、二次元配列のポインタ 戻り値：なし

説明：プレイヤーが指定した配列の要素番号に「〇」「×」を入力し、三目並べ画面と入力したプレイヤー名、先攻後攻と「〇」「×」を出力する。

###### Bingo()関数

引数：「〇」「×」が入力された二次元配列 戻り値：P1\_WINまたはP2\_WINまたはFALSE

説明：「〇」が縦、横、斜めのどれかに三つ並んだか判定し並んでいるときはP1\_WIN、「×」のときはP2\_WIN、並んでいないときはFALSEを返す。

###### SquareFull()関数

引数：〇×を出力した回数を持つ変数 戻り値：DROWまたはFALSE

説明：〇×を出力した回数がCAPACITY\_FULLの時DROW、そうでないときはFALSEを返す。

## get\_result.c

###### GameResult()関数

引数：勝敗の情報をもつ変数 戻り値：なし

説明：勝敗の情報を持つ変数がP1\_WINまたはP2\_WINまたはDROWのときそれぞれ結果の出力をする。

###### PlayAgain()関数

引数：なし 戻り値：ENDまたはCONTINUE

説明：「やめる…０以外」を入力するとENDを出力、「もう一度遊ぶ…０」を入力するとCONTINUEを出力する。

## define.h

TRUE：真 FALSE：偽 END：三目並べを終了する

CONTINUE：三目並べをもう一度遊ぶ DROW：引き分け

1P\_WIN：1Pプレイヤーの勝利 2P\_WIN：2Pプレイヤーの勝利 CAPACITY\_FULL：三目並べのマスがすべて埋まっている。