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](#_Toc167375114)

[2. 画面遷移図 4](#_Toc167375115)

[3. システム機能フロー図 5](#_Toc167375116)

[3.1 フロー説明 5](#_Toc167375117)

[3.1.1 [盤面を初期化] 5](#_Toc167375118)

[3.1.2 [3×3のマスと先後を画面出力] 5](#_Toc167375119)

[3.1.3 [縦横のマス数を入力] 6](#_Toc167375120)

[3.1.4 [入力した値は規定値内か] 6](#_Toc167375121)

[3.1.5 [既に入力されていないか] 6](#_Toc167375122)

[3.1.6 [三目並べの画面のマスに「〇」「×」を出力] 6](#_Toc167375123)

[3.1.7 [一直線がすべて「〇」、または「×」ですか] 7](#_Toc167375124)

[3.1.8 [出力したのは9回目ですか] 7](#_Toc167375125)

[3.1.9 [結果を出力] 7](#_Toc167375126)

[3.1.10 [もう一度遊びますか] 7](#_Toc167375127)

[4. ファイル構成図 9](#_Toc167375128)

[5. 機能構成 9](#_Toc167375129)

[5.1 tic\_tac\_toe.c 9](#_Toc167375130)

[三目並べを行う前の処理 10](#_Toc167375131)

[縦横の入力についてのループ 10](#_Toc167375132)

[「〇」「×」出力に関するループ 10](#_Toc167375133)

[ゲームを終了するループ 11](#_Toc167375134)

[5.2 game\_ready.c 11](#_Toc167375135)

[InitSquare()関数 11](#_Toc167375136)

[PreScreen()関数 11](#_Toc167375137)

[5.3 get\_input.c 11](#_Toc167375138)

[SquareNumber ()関数 11](#_Toc167375139)

[NumRange()関数 12](#_Toc167375140)

[MarkChecker()関数 12](#_Toc167375141)

[5.4 get\_output.c 12](#_Toc167375142)

[PrintMark()関数 12](#_Toc167375143)

[Bingo()関数 12](#_Toc167375144)

[SquareFull()関数 13](#_Toc167375145)

[5.5 get\_result.c 13](#_Toc167375146)

[GameResult()関数 13](#_Toc167375147)

[PlayAgain()関数 13](#_Toc167375148)

[5.6 define.h 14](#_Toc167375149)

# 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, レーダー チャート

自動的に生成された説明

### フロー説明

#### [盤面を初期化]

3×3のマスをすべて空白にし、[3.1.2](#_[先攻後攻をランダムで決定])に移行する。

#### [3×3のマスと先後を画面出力]

「〇」「×」を出力するための3×3のマスと、先攻後攻と「〇」「×」が出力し、[3.1.3](#_[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、（〇or×）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]

[3.1.3](#_[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、[3.1.3](#_[縦横のマス数を入力])に移行し、1～3以内であれば[3.1.5](#_[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[3.1.3](#_[縦横のマス数を入力])に移行し、入力されてなければ[3.1.6](#_[「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、[3.1.7](#_[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[3.1.8](#_[出力したのは9回目ですか])に移行し、並んでいれば[3.1.9](#_[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[3.1.3](#_[縦横のマス数を入力])に移行し、埋まっていれば[3.1.9](#_[結果を出力])に移行する。

#### [結果を出力]

「〇」が勝利した場合、「〇」が勝ちました！

「×」が勝利した場合、「×」が勝ちました！を出力し、[3.1.10](#_[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、[3.1.10](#_[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

tic\_tac\_toe.cはmain関数を記述するファイルである。

縦横の値を入力する要素数２の一次元配列を宣言。

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」「×」を出力する各要素数３の二次元配列の初期値をすべて空白で宣言。

勝敗の結果を格納する変数の初期値をUNKNOWNで宣言。

###### もう一度遊ぶ際のループ

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。

Bingo()関数、SquareFull()関数の戻り値が、UNKNOWNのとき縦横の入力のループから再度処理を行う。O\_WIN、またはX\_WIN、DROW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。三目並べを行う前の処理から始める。

## game\_ready.c

InitSquare()関数

関数の型：void型

引数：二次元配列のポインタ 戻り値：void

説明：配列の要素をすべて空白で初期化する。

###### PreScreen()関数

関数の型：void型 引数：二次元配列 戻り値：void

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、先攻後攻と「〇」「×」が出力する。 表示画面…[3.1.2](#_[3×3のマスと先後を画面出力])

## get\_input.c

###### SquareNumber ()関数

関数の型：void型

引数：一次元配列のポインタ、「〇」「×」を出力した回数を持つ変数

戻り値：void

説明：手番を出力し、縦と横のマス数を配列のポインタに入力する。

表示画面…[3.1.3](#_[縦横のマス数を入力])

###### NumRange()関数

関数の型：列挙型

引数：一次元配列 戻り値：TRUEまたはFALSEを代入した変数

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[3.1.4](#_[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

関数の型：列挙型

引数：一次元配列、二次元配列 戻り値：TRUEまたはFALSEを代入した変数

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[3.1.5](#_[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

関数の型：void型

引数：一次元配列、二次元配列のポインタ 戻り値：void

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、三目並べ画面と先攻後攻と「〇」「×」を出力する。 表示画面…[3.1.6](#_[三目並べの画面のマスに「〇」「×」を出力])

###### Bingo()関数

関数の型：列挙型

引数：二次元配列 戻り値：O\_WINまたはX\_WINまたはUNKNOWNを格納した変数

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O\_WIN、「×」のときはX\_WIN、並んでいないときはFALSEを格納した変数を返す。

###### SquareFull()関数

関数の型：列挙型

引数：「〇」「×」を出力した回数を持つ変数 戻り値：DROWまたはUNKNOWNを格納した変数

説明：「〇」「×」を出力した回数がCAPACITY\_FULLの時DROW、そうでないときはUNKNOWNを格納した変数を返す。

## get\_result.c

###### GameResult()関数

関数の型：void型

引数：勝敗の情報をもつ変数 戻り値：void

説明：勝敗の情報を持つ変数がO\_WINまたはX\_WINまたはDROWのときそれぞれ結果の出力をする。 表示画面… [3.1.9](#_[勝ったプレイヤー名を出力])

###### PlayAgain()関数

関数の型：列挙型

引数：なし 戻り値：ENDまたはCONTINUEを格納した変数

説明：「もう一度遊ぶ…０/やめる…０以外」で0以外を入力するとENDを出力、０を入力するとCONTINUEを出力する。 表示画面…[3.1.10](#_[もう一度遊びますか])

## define.h

真か偽を表す列挙型

|  |  |
| --- | --- |
| TRUE：真 | FALSE：偽 |

三目並べの続行を表す列挙型

|  |  |
| --- | --- |
| CONTINUE：三目並べをもう一度遊ぶ | END：三目並べを終了する |

勝利を表す列挙型

|  |  |
| --- | --- |
| UNKNOWN：勝敗の結果が出ていない。 | DROW：引き分け |
| O\_WIN：「〇」の勝利 | X\_WIN：「×」の勝利 |

マスが埋まっているかを表すための定義

|  |
| --- |
| #define CAPACITY\_FULL 9 |