三目並べ詳細仕様書

目次

[**1.** **概要** 1](#_Toc168661642)

[**2.** **用語の定義** 1](#_Toc168661643)

[**3.** **画面構成** 2](#_Toc168661644)

[3.1. マス目・〇×の表示 2](#_Toc168661645)

[3.2. 手番の表示 2](#_Toc168661646)

[3.3. 警告文の表示 2](#_Toc168661647)

[3.4. 勝利数とリトライ・終了選択の表示 3](#_Toc168661648)

[**4.** **ファイル構成** 4](#_Toc168661649)

[4.1. sanmoku.c(main) 4](#_Toc168661650)

[4.2. sanmoku.h 4](#_Toc168661651)

[4.3. board.c 4](#_Toc168661652)

[4.4. board.h 4](#_Toc168661653)

[4.5. input\_num.c 4](#_Toc168661654)

[4.6. input\_num.h 5](#_Toc168661655)

[4.7. decision\_count.c 5](#_Toc168661656)

[**5.** **機能構成** 6](#_Toc168661657)

[5.1. sanmoku.c(main)が持つ機能 7](#_Toc168661658)

[5.1.1. main関数 7](#_Toc168661659)

[5.2. board.cが持つ機能 8](#_Toc168661660)

[5.2.1. ShowBoard関数 8](#_Toc168661661)

[5.2.2. CheckTurn関数 8](#_Toc168661662)

[5.3. input\_num.cが持つ機能 9](#_Toc168661663)

[5.3.1. InitBord関数 9](#_Toc168661664)

[5.3.2. PushNum関数 9](#_Toc168661665)

[5.3.3. CountWin 9](#_Toc168661666)

[5.3.4. PlayEnd関数 9](#_Toc168661667)

[5.4. decision\_count.cが持つ機能 11](#_Toc168661668)

[5.4.1. RangeNum 11](#_Toc168661669)

[5.4.2. CheckNum 11](#_Toc168661670)

[5.4.3. CheckWin 11](#_Toc168661671)

[6. 変数構成 12](#_Toc168661672)

# **概要**

本書は作成する三目並べのファイル構成、機能構成、画面構成を示したものである。

# **用語の定義**

記号 → 半角小文字のoxのこと

手番 → 先攻・後攻の順のこと

マス 目 → 3×3 の格子のこと

キー → キーボード上の記号・文字・数字のこと

リトライ → 三目並べをもう1度行うこと

空 → 空というマスの状態を表す文字列が格納されていること

# **画面構成**

## マス目・〇×の表示

ダイアグラム が含まれている画像

自動的に生成された説明

マスの数字の左は半角スペース2個、右は半角スペース３個の幅

数字の上下はスペース１個分の高さ、マスの数字と記号は半角で表示する。

マス目を表示し、入力された値のマスにOXを表示する。

## 手番の表示



4.1.の図の下に左の図の手番の文章が表示され入力を求める。



## 警告文の表示



１～９の値以外が入力された場合に表示される。



記号の重複判定に当てはまった場合に表示される。



リトライ・終了の選択でTとQ以外が入力された場合に表示される。

## 勝利数とリトライ・終了選択の表示



勝ち判定された場合勝者を4.1の図の下に表示する。

その後勝利数の下にリトライ・終了選択文が表示される。

引き分けの場合は結果を表示せずにリトライ・終了選択文が表示される。







終了が選択された場合に記号ごとの合計勝利数を表示される。

# **ファイル構成**

## sanmoku.c(main)

main関数を記述するファイル

## sanmoku.h

以下の表にてマクロ、列挙型を定義している。

|  |  |  |
| --- | --- | --- |
| マクロ名 | 概要 | |
| ELEMENT | マスの配列の要素数：9 | |
| BOARD\_LEN | マスの縦の数の差の要素数：3 | |
| BOARD\_LINE | 記号OXが一致した要素数：2 | |
| 列挙型タグ名 | メンバ | 概要 |
| BOOL | TRUE, FALSE | 真偽を示す列挙型 |
| TURN | CIRCLE, CROSS | 手番を示す列挙型 |
| RETRY | TRY, QUIT | リトライ・終了を示す列挙型 |
| JUDGE | UNKNOWN, CIRCLE\_WIN, CROSS\_WIN, DROW | 記号の勝利・引き分けを示す  列挙型 |

## board.c

* ShowBoard

盤面を表示する関数

* CheckTurn

現在のターンを判定して表示する関数

* DisplayTurn

現在の手番を表示する関数

## input\_num.c

* PushNum

値を入力させる関数

* CountWin

勝利数をカウントする関数

* PlayEnd

リトライか終了かの入力を求め判別する関数

* ChangeMark

マスの中の数字と記号を入れ替える関数

## decision\_count.c

* RangeNum

値が範囲内かを判定する関数

* CheckMark

マスの中に既に記号が入っているかを判定する関数

* CheckWin

OX記号が縦、横、斜めに３つ揃っているかの勝利判定と引き分け判定(表示はしない)をする関数

# **機能構成**

下記の機能構成ではループで使用する変数の宣言については省略する。

ダイアグラム

自動的に生成された説明

## sanmoku.c(main)が持つ機能

### main関数

* **変数・配列**

|  |  |  |
| --- | --- | --- |
| 変数 | 変数の型 | 初期化 |
| ターン数をカウントする変数count\_turn | int型 | 0 |
| 記号OXが重複しているかの結果を格納する変数duplication | BOOL型 | FALSE |
| 入力が正しいかの判定結果を格納する変数input\_error | BOOL型 | FALSE |
| OXの手番を表す変数now\_turn | TURN型 | CIRCLE |
| 勝利判定結果を格納する変数result | JUDGE型 | UNKNOWN |
| 入力された値を格納する変数player\_input | char型 | 0 |
| 三目並べの終了と継続の判定結果を格納する変数retry | RETRY型 | TRY |
| 〇の勝利数を表す変数count\_circle\_win | int型 | 0 |
| ×の勝利数を表す変数count\_cross\_win | int型 | 0 |
| 配列 | 配列の型 | 初期化 |
| 数字と記号oxを格納する配列  board [ELEMENT] | char型 | ⅿain関数で宣言をして  InitBoard関数で初期化 |

* **終了が選択されるまでループ**
* InitBord関数、ShowBoard関数の順に呼び出す。
* QUITの場合は終了し、TRYの場合は最初に戻りInitBord関数から再度始まる。
* **勝利か引き分け判定がつくまでループ**
* 勝利か引き分け判定がつくまで回すループ。
* CheckTurn関数を処理しPushNum関数、RangeNum関数、CheckNum関数の順で呼び出し、処理をする。
* CheckWin関数でCIRCLE\_WINまたはCROSS\_WIN、CheckDraw関数でDRAWが返された場合はループを抜け出す。
* UNKNOWNが返された場合は、ターン数を１加算してCheckTurn関数から再度ループする（ターン数のカウントを行う）。
* 「終了が選択されるまでループ」内で行う。

## board.cが持つ機能

### ShowBoard関数

* 引数：char型board[ELEMENT]
* 戻り値：なし
* 変数：なし
* 処理内容：マスの配列boardを受け取り、マス目を表示するとともにマスの中身の数字または記号を表示させる。

### CheckTurn関数

* 引数：turn
* 戻り値：TURN型 CIRCLE or CROSS
* 変数：なし
* 処理内容：turnが偶数ならCIRCLE、奇数ならCROSSを返す。

### DisplayNum関数

* 引数：now\_turn
* 戻り値：なし
* 変数：なし
* 処理内容：CIRCLEを受け取った場合「〇の番です：」と表示して、CROSSを受け取った場合は、「×の番です：」と表示する。改行はしない。

## input\_num.cが持つ機能

### InitBord関数

* 引数：char\*board
* 戻り値：なし
* 変数：なし
* 処理内容：マスの配列の中にboard要素順に１～９の数字をそれぞれ１つずつ格納する。

### PushNum関数

* 引数：なし
* 戻り値：入力した値を格納した変数input
* 変数：char型 入力した値を格納する変数input 初期値0
* 処理内容：getch()を用いプレイヤーに入力を求め、入力された値をinputに格納し、返す。

### CountWin関数

* 引数：TURN型now\_turn

int\* count\_circle\_win

int\* count\_cross\_win

* 戻り値：なし
* 変数：なし
* 処理内容：TURN型変数now\_turnを受け取り、now\_turnが「CIRCLE」のときcount\_circle\_winを+1し、「〇の勝利」と文章を表示する。now\_turnが「CROSS」のときcount\_cross\_winを+1し、「×の勝利」と文章を表示する。

### PlayEnd関数

* 引数：int count\_circle\_win

int count\_cross\_win

* 戻り値：TRY またはQUITまたはERROR
* 変数：char 入力した文字を格納する変数 初期値0
* 処理内容：〇と×の勝利数を表す変数count\_circle\_win、count\_cross\_winを受け取り、「リトライ(T)or終了(Q)：」の文章を表示させプレイヤーに入力を求め、リトライ(T)か終了(Q)かのどちらかが入力されたか判別する。

リトライの場合は、TRYを返す。

終了の場合は、QUITを返して、記号の合計勝利数をそれぞれ表示する。

指定されたキー以外が入力された場合は、警告文を表示して再度入力を促す。

### ChangeMark関数

* 引数：char board[ELEMENT]

char player\_input

TURN型 now\_turn

* 戻り値：なし

処理内容：board配列のplayer\_input番目の要素をnow\_turnが「CIRCLE」のとき半角oで、now\_turnが「CROSS」のとき半角xで置き換える。

## decision\_count.cが持つ機能

### RangeNum関数

* 引数： char player\_input
* 戻り値：BOOL型 TRUE or FALSE
* 変数：なし
* 処理内容：player\_inputを受け取り、半角の１以上９以下の値か判別する。

範囲内の場合は、TRUEを返す。

範囲外の場合は、FALSEを返す。

### CheckNum関数

* 引数：char player\_input、

char board[ELEMENT]

* 戻り値：BOOL型 TRUE or FALSE
* 変数：int player\_inputからchar型をint型に変換し格納する変数
* 処理内容：player\_inputを受け取り、player\_inputからchar型をint型に変換し、board[player\_inputからchar型をint型に変換し格納する変数]に記号が格納されていない場合はTRUEを返す。

格納されている場合は、FALSEを返す。

### CheckWin関数

* 引数：char board[ELEMENT]、

TURN型 now\_turn、

int count\_turn

* 戻り値：JUDGE型 変数
* 変数：int ターン数

int 条件に合致した回数をカウントする変数same\_mark\_count

* 処理内容：盤面の記号が縦横斜めいずれかに同じ記号が揃っているか判定。

**横の判定**：[0]から順に一つ隣の要素と比較するループを回す。

ループはBOARD\_LENまで行い値が同じだったらsame\_mark\_countを＋１

ループ終了後、探索基準となる要素にBOARD\_LENを加算し、same\_mark\_countを０で初期化して同様に探索を繰り返す。

same\_mark\_countが2以上になったとき、now\_turnが「CIRCLE」なら戻り値CIRCLE\_WINを、now\_turnが「CROSS」なら戻り値CROSS\_WIN返す。

**縦の判定**：[0]から順に三つ隣の要素と比較するループを回す。

ループはBOARD\_LENまで行い値が同じだったらsame\_mark\_count＋１

ループ終了後、探索基準となる要素にBOARD\_LENを加算し、same\_mark\_countを０で初期化して同様に探索を繰り返す。

same\_mark\_countが2以上になったとき、now\_turnが「CIRCLE」なら戻り値CIRCLE\_WINを、now\_turnが「CROSS」なら戻り値CROSS\_WIN返す。

**左上から右下への斜め判定**：[0]から順に四つ隣の要素と比較するループを回す。

ループはBOARD\_LENまで行い値が同じだったらsame\_mark\_count＋１

ループ終了後、探索基準となる要素にBOARD\_LENを加算し、same\_mark\_countを０で初期化して同様に探索を繰り返す。

same\_mark\_countが2以上になったとき、now\_turnが「CIRCLE」なら戻り値CIRCLE\_WINを、now\_turnが「CROSS」なら戻り値CROSS\_WIN返す。

**右上から左下への斜め判定**：[2]から順に二つ隣の要素と比較するループを回す。

ループはBOARD\_LENまで行い値が同じだったらsame\_mark\_count＋１

ループ終了後、探索基準となる要素にBOARD\_LENを加算し、same\_mark\_countを０で初期化して同様に探索を繰り返す。

same\_mark\_countが2以上になったとき、now\_turnが「CIRCLE」なら戻り値CIRCLE\_WINを、now\_turnが「CROSS」なら戻り値CROSS\_WIN返す。

最後に、count\_turnを受け取り、ターン数８(ELEMENT-1)であるかを判別する。

count\_turnが８の場合、DROWを返す。

count\_turnが８ではない場合、UNKNOWNを返す。