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 → 半角大文字の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 | 合致した行と列の要素数：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, ERROR | 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

現在のマスの中に入っている「１～９」の数字や記号を表示する関数

* CheckTurn

現在のターンを判定して表示する関数

* DisplayNum

現在の手番を表示する関数

## board.h

## input\_num.c

* InitBoard

全てのマスに１～９の数字をそれぞれ格納する関数

* PushNum

値を入力させる関数

* CountWin

勝利数をカウントする関数

* PlayEnd

リトライか終了かを入力を求め判別する関数

* ChangeState

当てはまるマスの中の数字と記号を入れ替える関数

## input\_num.h

## decision\_count.c

* RangeNum

値が範囲内かを判定する関数

* CheckNum

マスの中に既に記号が入っているかを判定する関数

* CheckWin

記号が縦横斜めに３つ揃っているかの勝利判定とマス目が記号で全て埋まっているかの引き分け判定(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。



## 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 | int型 | 0 |
| 重複しているか表す変数 | BOOL型 | FALSE |
| 入力した値の範囲を表す変数 | BOOL型 | FALSE |
| 〇×の手番を表す変数 | TURN型 | CIRCLE |
| 結果を表す変数 | JUDGE型 | UNKNOWN |
| 入力された値を格納する変数 | char型 | 0 |
| 三目並べの１ゲームの終了と継続を示す変数 | RETRY型 | TRY |
| 〇の勝利数を表す変数 | int型 | 0 |
| ×の勝利数を表す変数 | int型 | 0 |
| 配列 | 配列の型 | 初期化 |
| 数字と記号を格納するマスの配列  [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**
* 終了が選択されるまで回すループ。
* 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（マスの配列）
* 戻り値：なし
* 変数：なし
* 処理内容：マスの配列を受け取り、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。

記号の場合は、半角の記号を表示する。

数字の場合は、半角の数字を表示する。

### CheckTurn関数

* 引数：int（ターン数）
* 戻り値：TURN型 CIRCLE or CROSS
* 変数：なし
* 処理内容：mainのループの中でカウントさせたターン数で現在のターン数を受け取り偶数、奇数を判別する。

偶数ならCIRCLE、奇数ならCROSSを返す。

### DisplayNum関数

* 引数：CheckTurnの戻り値を格納した変数
* 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。

## input\_num.cが持つ機能

### InitBord関数

* 引数：char\*（数字と記号を格納するマスの配列のポインタ）
* 戻り値：なし
* 変数：なし
* 処理内容：マスの配列の中に１～９の数字を順にそれぞれ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
* 変数：char 入力した値を格納する変数 初期値0
* 処理内容：プレイヤーに入力を求め、入力された値を配列に格納する。

### CountWin関数

* 引数：CIRCLE\_WINまたはCROSS\_WINが格納された変数、

int\*（勝利数をカウントするポインタ）

* 戻り値：なし
* 変数：char 入力した値を格納する変数 初期値0
* 処理内容：勝利した記号を受け取り、該当する記号を判別して勝利数を1加算し、「〇の勝利」、「×の勝利」と文章を表示する。

### PlayEnd関数

* 引数：int（〇の勝利数を表す変数）

int（×の勝利数を表す変数）

* 戻り値：TRY またはQUITまたはERRORが格納された変数
* 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。

## decision\_count.cが持つ機能

### RangeNum関数

* 引数： char（入力された値を格納する変数）
* 戻り値：BOOL型 TRUE or FALSE
* 変数：なし
* 処理内容：入力された値を受け取り、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、「範囲内の値を入力してください。」と警告文を表示してFALSEを返す。

### CheckNum関数

* 引数：char（入力された値を格納する変数）、

char\*（数字と記号を格納するマスの配列のポインタ）

* 戻り値：BOOL型 TRUE or FALSE
* 変数：int 入力された数値から文字列としての数値を数値としての数値に変換し格納する変数
* 処理内容：入力された値を格納する変数を受け取り、値に該当するマスの中身に記号が格納されているか判別する。

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、「既に入力されています。」と警告文を表示してFALSEを返す。

### CheckWin関数

* 引数：char（マスの配列）、

CIRCLEまたはCROSSが格納された変数、

int（ターン数）

* 戻り値：JUDGE型 CIRCLE\_WIN or CROSS\_WIN or DROW or UNKNOWN
* 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

* 処理内容：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カウント＋１

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

カウントが終了条件を満たしたなら戻り値を返す。

**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カウント＋１

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

カウントが終了条件を満たしたなら戻り値を返す。

**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カウント＋１

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

カウントが終了条件を満たしたなら戻り値を返す。

**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カウント＋１

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

カウントが終了条件を満たしたなら戻り値を返す。

縦横の数字の差はGAPを使用する。

〇が揃った場合は、CIRCLE\_WINを返す。

×が揃った場合は、CROSS\_WINを返す。

揃っていない場合はUNKNOWNが格納された変数を返す。

ループの中でターン数をカウントさせ現在のターン数を受け取り、ターン数８(ELEMENT-1)であるかを判別する。

カウントしたターン数が８の場合、DROWを返す。

カウントしたターン数が８ではない場合、UNKNOWNを返す。