三目並べ詳細仕様書

目次

[**1.** **概要** 1](#_Toc168409954)

[**2.** **用語の定義** 1](#_Toc168409955)

[**3.** **機能構成** 2](#_Toc168409956)

[3.1. 記号を入れたいマスを入力し記号を表示する 2](#_Toc168409957)

[3.2. ゲームを続けるか終了するかのキーを入力する 2](#_Toc168409958)

[3.3. マス目の状態を表示して終了判定を行う 2](#_Toc168409959)

[**4.** **画面構成** 4](#_Toc168409960)

[**4.1.** **マス目・〇×の表示** 4](#_Toc168409961)

[**4.2.** **手番の表示** 4](#_Toc168409962)

[**4.3.** **警告文の表示** 4](#_Toc168409963)

[**4.4.** **勝利数とリトライ・終了選択の表示** 5](#_Toc168409964)

[**5.** **ファイル構成** 6](#_Toc168409965)

[**5.1.** **sanmoku.c(main)** 7](#_Toc168409966)

[5.2. sanmoku.h 8](#_Toc168409967)

[**5.3.** **board.c** 9](#_Toc168409968)

[**5.3.1.** **ShowBoard** 9](#_Toc168409969)

[**5.3.2.** **CheckTurn** 10](#_Toc168409970)

[**5.3.3.** **ChangeState** 10](#_Toc168409971)

[5.4. board.h 10](#_Toc168409972)

[**5.5.** **input\_num.c** 11](#_Toc168409973)

[**5.5.1.** **InitBoard** 11](#_Toc168409974)

[**5.5.2.** **PushNum** 11](#_Toc168409975)

[**5.5.3.** **CountWin** 12](#_Toc168409976)

[**5.5.4.** **PlayEnd** 12](#_Toc168409977)

[5.6. input\_num.h 13](#_Toc168409978)

[**5.7.** **decision\_count.c** 13](#_Toc168409979)

[**5.7.1.** **RangeNum** 13](#_Toc168409980)

[**5.7.2.** **CheckNum** 14](#_Toc168409981)

[**5.7.3.** **CheckWin** 14](#_Toc168409982)

[**5.7.4.** **CheckDraw** 15](#_Toc168409983)

[5.8. decision\_count.h 15](#_Toc168409984)

# **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

# **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

## 記号を入れたいマスを入力し記号を表示する

* マスの番号は半角数字で「1」～「9」とする
* 4.2.の画像のように手番を表示する
* 入力を求める
* 半角英数字で１～９以外の値が入力された場合は「範囲内の値を入力してください。」と表示しもう１度入力を求める
* 既に入力されている半角英数字の１～９の値が再び入力された場合（重複した場合）「既に入力されています。」と表示しもう１度入力を求める
* 入力された番号のマスに記号を表示する
* 手番が変わりまた同じ動作を行う

## ゲームを続けるか終了するかのキーを入力する

* 終了は半角英数字の大文字で「Q」、リトライは半角英数字の大文字で「T」とする
* 4.4.の画像のように終了かリトライかを選ぶ画面を表示する
* 入力を求める
* 半角英数字の大文字で「Q」、「T」以外の文字が入力された場合は「TまたはQのみを入力してください。」と表示しもう１度入力を求める
* リトライが選択された場合、記号の勝利数がカウントされもう一度三目並べが始まる
* 終了が選択された場合、3.4.へ移行し三目並べが終了する

## マス目の状態を表示して終了判定を行う

* マスの状態を表示する
* 記号が表示されていないマスには番号を表示する
* マスの番号は半角の「１」～「９」、記号は半角英数字の大文字「O」、「X」を使用する
* 縦横斜めに同じ記号が３つ並んでいるかを確認する（勝利判定）
* 全てのマスに記号が表示されているかを確認する（引き分け判定）

## 結果を表示する

* 勝利した記号を参照し「O（X）の勝利」と表示し3.2.へ移行する
* 引き分けは表示せず3.2.へ移行する
* 3.2.で終了が選択された場合、カウントされた記号の勝利数がそれぞれ表示される

# **画面構成**

## **マス目・〇×の表示**



マスの数字の左右は半角スペース2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。

マスのスペースは

## **手番の表示**



4.1.の図の下に左の図にある手番の文章が表示される。



## **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**

フローチャート  
ダイアグラム

自動的に生成された説明

## **sanmoku.c(main)**

main関数を記述するファイル

ターン数をカウントする変数を宣言

重複しているか表す変数を宣言

入力した値の範囲を表す変数を宣言

〇×の手番を表す変数を宣言

勝ちを表す変数を宣言

引き分けを表す変数を宣言

入力された値を格納する変数を宣言

三目並べの１ゲームの終了と継続を示す変数の初期値をTRYで宣言

〇の勝利数を表す変数の初期値を0で宣言

×の勝利数を表す変数の初期値を0で宣言

数字と記号を格納するマスの配列の要素数9を宣言

説明：終了が選択されるまでのループの中に勝利か引き分け判定がつくまでのループを入れる。下記の順にプログラムを組む。

**終了が選択されるまでループ**

ターン数をカウントする変数を０で初期化

勝ちを表す変数をUNKNOWNで初期化

引き分けを表す変数UNKNOWNで初期化

InitBord関数、ShowBoard関数の順に呼び出す。

**勝利か引き分け判定がつくまでループ**

重複しているか表す変数をFALSEで初期化

入力した値の範囲を表す変数をFALSEで初期化

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。

RangeNum関数がFALSEを返した場合は、再度入力を促す。

同様に、CheckNum関数がFALSEを返した場合は、再度入力を促す。

ChangeState関数でマスの中身を数字と記号を置き換えた後、ShowBoard関数で記号を置き換えたマスを表示する。CheckWin関数、CheckDraw関数を呼び出し勝利・引き分け判定処理をする。

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。

**終了が選択されるまでループ**

どちらかが勝利した場合はCountWin関数で勝利数をカウントと4.4の文章を表示した後、PlayEnd関数を呼び出し、処理をする。

引き分けの場合は何も表示せずにPlayEnd関数を呼び出し、処理をする。

QUITの場合は終了し、TRYの場合は終了を選択されるまでループに戻りInitBord関数から再度始まる。

## sanmoku.h

ELEMENT：マスの配列の要素数9を定義する

WIDE\_GAP：縦の数字の差を示す３を定義

VERTICAL\_GAP：横の数字の差を示す３を定義

真偽を示す列挙型を定義（TRUE：真を示す、FALSE：偽を示す）

現在の手番を示す列挙型を定義（CIRCLE：〇の手番、CROSS：×の手番）

リトライと終了を示す列挙型を定義（TRY：リトライを示す、QUIT：終了を示す）

記号の勝利を示す列挙型を定義（CIRCLE\_WIN：〇の勝利、CROSS\_WIN：×の勝利、UNKNOWN：勝敗の結果が出ていない、DRAW：引き分け）

## **board.c**

|  |  |  |  |
| --- | --- | --- | --- |
| 関数 | ShowBoard | CheckTurn | ChangeState |
| 説明 | 現在のマスの中身に入っている数字や記号の表示 | 現在のターンを判定して表示する | 入力した値に当てはまるマスの中身の数字と記号を入れ替える |

### **ShowBoard**

|  |  |  |  |
| --- | --- | --- | --- |
| 引数 | 引数の型 | 戻り値 | 戻り値の型 |
| 数字と記号を格納するマスの配列 | char | なし | void |

マスの配列を受け取り、4.1の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**

|  |  |  |  |
| --- | --- | --- | --- |
| 引数 | 引数の型 | 戻り値 | 戻り値の型 |
| ターン数 | int | CIRCLEまたはCROSSが列挙型に格納された変数 | 現在の手番を示す列挙型を定義 |

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偶数、奇数を判別する。

偶数ならCIRCLE、奇数ならCROSSを返す。

### **ChangeState**

|  |  |  |  |
| --- | --- | --- | --- |
| 引数 | 引数の型 | 戻り値 | 戻り値の型 |
| 入力された値を格納する変数、  数字と記号を格納するマスの配列のポインタ、  CIRCLEまたはCROSSが列挙型に格納された変数 | char、  char、  現在の手番を示す列挙型 | なし | void |

入力された値を格納する変数に当てはまるマスの中身に入っている数字を現在のターンの記号に置き換える。

## board.h

## **input\_num.c**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 関数 | InitBord | PushNum | CountWin | PlayEnd |
| 説明 | 全てのマスの中身に数字を入れる | 値を入力する | 勝利数をカウントする | リトライか終了を入力する |

### **InitBoard**

|  |  |  |  |
| --- | --- | --- | --- |
| 引数 | 引数の型 | 戻り値 | 戻り値の型 |
| 数字と記号を格納するマスの配列のポインタ | char | なし | void |

全てのマスの中身に１～９の数字を１つずつ格納する。

### **PushNum**

|  |  |  |  |
| --- | --- | --- | --- |
| 引数 | 引数の型 | 戻り値 | 戻り値の型 |
| CIRCLEまたはCROSSが列挙型に格納された変数 | 現在の手番を示す列挙型 | 入力した値を格納した変数 | char |

CIRCLEが返された場合は、4.2の「〇の番です」を表示してCROSSが返された場合は、4.2の「×の番です」を表示して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**

|  |  |  |  |
| --- | --- | --- | --- |
| 引数 | 引数の型 | 戻り値 | 戻り値の型 |
| 〇の勝利数を表す変数と×の勝利数を表す変数のポインタ、  CIRCLE\_WINまたはCROSS\_WINが列挙型に格納された変数 | int、  記号の勝利を示す列挙型 | なし | void |

勝利した記号を受け取り、該当する記号を判別して4.4の勝者の文章を表示する

〇が勝利した場合、〇の勝利数を１加算して「〇の勝利」と表示する。

×が勝利した場合、×の勝利数を１加算して「×の勝利」と表示する。

### **PlayEnd**

|  |  |  |  |
| --- | --- | --- | --- |
| 引数 | 引数の型 | 戻り値 | 戻り値の型 |
| 〇の勝利数を表す変数、  ×の勝利数を表す変数 | int、  int | TRY またはQUITまたはERRORが列挙型に格納された変数 | リトライと終了を示す列挙型 |

〇と×の勝利数を表す変数を受け取り、4.4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、4.4の記号の合計勝利数を表示する。

指定されたキー以外が入力された場合は、4,3の警告文を表示して再度入力を促す。

## input\_num.h

## **decision\_count.c**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 関数 | RangeNum | CheckNum | CheckWin | CheckDraw |
| 説明 | 入力された値が範囲内かを判定する | 入力された値の中身に既に記号が入っているかを  判定する | 記号が縦・横・斜めに3つ揃っているか判定する | 引き分け判定をする(表示はしない) |

### **RangeNum**

|  |  |  |  |
| --- | --- | --- | --- |
| 引数 | 引数の型 | 戻り値 | 戻り値の型 |
| 入力された値を格納する変数 | char | TRUEまたはFALSE が列挙型に格納された変数 | 真偽を示す列挙型を定義 |

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、4.3の警告文を表示してFALSEを返す。

### **CheckNum**

|  |  |  |  |
| --- | --- | --- | --- |
| 引数 | 引数の型 | 戻り値 | 戻り値の型 |
| 入力された値を格納する変数、  数字と記号を格納するマスの配列のポインタ | char、  char | TRUEまたはFALSE が列挙型に格納された変数 | 真偽を示す列挙型を定義 |

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、4.3の警告文を表示してFALSEを返す。

### **CheckWin**

|  |  |  |  |
| --- | --- | --- | --- |
| 引数 | 引数の型 | 戻り値 | 戻り値の型 |
| 〇×の手番を表す変数、  数字と記号を格納するマスの配列 | 現在の手番を示す列挙型、  char | CIRCLE\_WINまたはCROSS\_WINまたはUNKNOWNが列挙型に格納された変数 | 記号の勝利を示す列挙型を定義 |

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。

縦横の数字の差はWIDE\_GAP,VERTICAL\_GAPを使用して埋める。

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が列挙型に格納された変数を返す。

### **CheckDraw**

|  |  |  |  |
| --- | --- | --- | --- |
| 引数 | 引数の型 | 戻り値 | 戻り値の型 |
| ターン数をカウントする変数 | int | DROWまたはUNKNOWNが列挙型に格納された変数 | 記号の勝利を示す列挙型 |

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。

## decision\_count.h