三目並べ 詳細仕様書

Cチーム  
メンバー  
村上 海斗　川島 冬嗣

目次

[1. 概要 1](#_Toc166838186)

[2. 語句定義 1](#_Toc166838187)

[3. ファイル説明 1](#_Toc166838188)

[3.1 ファイル構成 1](#_Toc166838189)

[3.2 ヘッダファイル 3](#_Toc166838190)

[4. グローバル変数 5](#_Toc166838191)

[5. 関数説明 5](#_Toc166838192)

[5.1 関数一覧 5](#_Toc166838193)

[5.2 game\_progress関数 8](#_Toc166838194)

[5.3 main関数 9](#_Toc166838195)

[6. 機能構成 10](#_Toc166838196)

[6.1 機能一覧 10](#_Toc166838197)

[6.2 フローチャート 12](#_Toc166838198)

[7. 画面構成 14](#_Toc166838199)

# 概要

本書は、「三目並べ」のファイル構成、関数説明、機能構成、画面構成についてまとめた詳細仕様書です。

# 語句定義

本書で記載される語句は、表 1の通り定義します。

表 1語句定義

|  |  |  |
| --- | --- | --- |
| No | 語句 | 定義 |
| 1 | 勝利条件 | 駒が縦、横、斜めどれか一列揃った場合 |
| 2 | 引き分け条件 | 勝利条件を満たさず、盤面が駒で埋まった場合 |
| 3 | 終了条件 | 勝利条件、引き分け条件を満たした場合 |
| 4 | 再プレイ | もう一度、三目並べで遊ぶこと |

# ファイル説明

## ファイル構成

ファイル構成は図 1の通りです。各ファイルの役割を表 2に示しています。

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

図 1ファイル構成

表 2各ファイルの役割

|  |  |
| --- | --- |
| ファイル名 | 役割 |
| main.c | main関数、三目並べのゲーム進行を制御する。 |
| tutorial.h / tutorial.c | チュートリアルを表示する。 |
| board.h / board.c | 三目並べの盤面を制御する。 |
| basic.h | 基本の定義・宣言を行う。 |

## ヘッダファイル

各ヘッダファイルで使用する定義を表 3～表 5に示しています。

basic.hはmain.c、board.cで、board.h、tutoial.hはmain.cで使用します。

表 3basic.h

|  |  |  |
| --- | --- | --- |
| 定義名・宣言名 | 定義・宣言 | 説明 |
| マクロ | NAME\_LEN (10) | プレイヤー名を格納する変数のサイズ |
| FALSE (0) | “FALSE”を0とする |
| TRUE (1) | “TRUE”を1とする |
| enum型 | TURN | FIRST\_TURN（先手のターン）、SECOND\_TURN（後手のターン）を定義 |
| 構造体 | typedef struct PLAYER | 構造体の中身は、プレイヤー名を表す name (char\*型)、 駒の種類を表す piece(char型)とする。 |

表 4board.h

|  |  |  |
| --- | --- | --- |
| 定義名・宣言名 | 定義・宣言 | 説明 |
| マクロ | BOARD\_SIZE (3) | マスのサイズを3×3の設定 |
| enum型 | RESULT | WIN（勝利）、DRAW（引き分け）、NONE（未確定） の3つを定義 |
| プロトタイプ | void print\_now\_board(void) | プロトタイプ宣言 |
| int put\_piece(int row, int column, PLAYER player) |
| RESULT judge\_game(void) |
| void clean\_board(void) |

表 5tutorial.h

|  |  |  |
| --- | --- | --- |
| 定義名・宣言名 | 定義・宣言 | 説明 |
| プロトタイプ | void view\_tutorial(void) | プロトタイプ宣言 |

# グローバル変数

board.cでは表 6のグローバル変数を定義する。

表 6グローバル変数

|  |  |
| --- | --- |
| 変数名 | 説明 |
| char[][] board | 盤面を保存するための変数。配列の要素数はそれぞれBOARD\_SIZE(3)とする。 |

# 関数説明

## 関数一覧

使用する関数は、以下の表 7の通りです。出力表示については、画面構成を基に作成してください。

表 7関数の説明

|  |  |  |
| --- | --- | --- |
| ファイル名 | 関数名 | 説明 |
| tutorial.c | void view\_tutorial(void) | main関数で呼び出される。チュートリアルを表示する。 |
| board.c | void print\_now\_board(void) | game\_progress関数内で呼び出される。現在の盤面を表示する。表示する盤面のデザインは以下の通りである。  左の図は初期の盤面を表示している。  右の図は1行2列に○、2行2列に×を配置した状態の盤面を表している。 |
| int put\_piece(int row, int column, PLAYER player) | game\_prgress関数内で呼び出される。row、columnはそれぞれ、ユーザ入力から得た行・列(1~3)を意味し、boardのrow-1(行)column-1(列)の位置に、playerのpieceを配置する。駒を配置出来た場合にはTRUE(1)、配置出来なかった場合（座標が不正または駒が既に配置されている）にはFALSE(0)を戻り値として返す。 |
| RESULT judge\_game(void) | game\_progress関数内で呼び出される。終了条件に当てはまるか判断する。引き分けならDRAW、どちらかが勝ちならWIN、未確定ならNONEを、戻り値として返す。 |
| void clean\_board(void) | game\_progress関数内で呼び出される。盤面(char型配列board)の要素を全て半角スペースに初期化する。 |
| main.c | TURN change\_turn(TURN now) | game\_progress関数内で呼び出される。nowがFIRST\_TURNならSECOND\_TURNを、nowがSECOND\_TURNならFIRST\_TURNを戻り値として返す。また、交代した旨を表示する。 |
| void game\_progress(char\* name1, char\* name2) | main関数内で呼び出される。ゲームの進行を行う。5.2で詳しく説明。 |
| void print\_result(RESULT result, PLAYER player) | game\_progress関数内で呼び出される。引数のresultに応じてゲーム結果を表示する。resultがDRAWのとき引き分け表示、WINのときプレイヤー名と勝利を表示する。(本来起こり得ないが、) resultがNONEのときassert関数を用いてプログラムを強制終了させる。 |
| int retry\_game(void) | main関数内で呼び出される。ユーザ入力を受け再プレイするか決める。ユーザ入力が1の場合、TRUEを戻り値として返し、1以外の場合FALSEを返す。 |
| int main(void) | メイン処理を行う。5.3で詳しく説明。 |

## game\_progress関数

game\_progress関数での処理の手順について説明します。

1. 先手、後手の情報を設定するために、「name1、’O’」（先手）と「name2、’X’」（後手）の組み合わせで、それぞれ任意のPLAYER型変数に格納する。また、現在のターンのプレイヤーを表す任意のPLAYER型変数を作成し、初期値を先手にする。
2. 現在のターンを表す任意のTURN型変数を作成し、初期値をFIRST\_TURNにする。
3. 結果を表す任意のRESUL型変数を作成し、初期値をNONEとする。
4. clean\_board関数を呼び出す（盤面初期化）。
5. print\_now\_board関数を呼び出す（初期盤面表示）。
6. resultがNONEであれば、⑦～⑩をループ。
7. 現在のターンのプレイヤー名を表示し、座標入力を受ける。（不正な場合、警告文を表示し、再度入力要求を行う。）
8. put\_piece関数を呼び出し、戻り値がTRUEなら⑨へ、FALSE（駒の配置が出来なかった）なら⑦に戻る。
9. print\_now\_board関数を呼び出す（配置後盤面表示）。
10. judge\_game関数を呼び出し、戻り値がWINまたはDRAW（NONE以外）ならループを抜け⑫へ移る。
11. change\_turnを呼び出す。(⑦に戻る)
12. print\_result関数を呼び出す。

## main関数

main関数での処理の手順について説明します。

1. ユーザからの入力を受け、入力が半角数字1であれば（判定にはTRUEを用いる）、view\_tutorial関数を呼び出す。1以外の入力であれば、そのまま②へ移る。
2. ゲーム開始！” を表示する。
3. 任意のchar[]型変数(要素数はNAME\_LEN)を2つ作成し、先手・後手のプレイヤー名の入力を受け、それぞれ格納する。プレイヤー名が不正な（サイズオーバー、半角でない）場合、警告文を表示し、再度ユーザからの入力を要求する。
4. game\_progress関数を呼び出す。引数には、③でプレイヤー名を格納した変数を入れる。
5. retry\_game関数を呼び出し、戻り値がTRUEなら④に戻る。FALSEであれば⑥へ移る。
6. “ゲーム終了！” を表示し、終了する。

# 機能構成

## 機能一覧

入力機能、出力機能の一覧と説明を表 8、表 9に示しています。

表 8入力機能一覧

|  |  |  |
| --- | --- | --- |
| No. | 入力機能一覧 | 説明 |
| 01 | チュートリアル表示確認 | main関数内で、ユーザ入力を受け、半角数字1が入力されるとview\_tutorial関数を呼び出す。1以外の入力の場合、処理は行わない。 |
| 02 | プレイヤー名入力 | main関数内で、ユーザ入力を受け、プレイヤー名を決定する。プレイヤー名は半角英数字9字以下としており、それを満たさない入力の場合、警告文を表示し、再度入力要求する。 |
| 03 | 座標入力 | game\_progress関数内でユーザ入力を受け、put\_piece関数を呼び出し、入力された座標に駒を配置する。入力方法は、〇行×列を「〇␣×」(〇と×の間は半角スペース)の形で入力する。座標が不正な場合（既に駒が配置済み、マスのサイズよりも大きい）、警告文を表示し、再度入力要求する。 |
| 04 | 再プレイの有無確認 | main関数内retry\_game関数を呼び出し、半角数字1が入力されると、再度三目並べを行う。1以外の入力の場合、処理は行わない。 |

表 9出力機能一覧

|  |  |  |
| --- | --- | --- |
| No. | 出力機能一覧 | 説明 |
| 01 | チュートリアル表示 | main関数内でview\_tutorial関数を呼び出し、チュートリアルを表示する。 |
| 02 | 盤面表示 | game\_progress関数内でprint\_now\_board関数を呼び出し、現在の盤面を表示する。行と列を見やすくし、先手が配置した駒を’O’、後手が配置した駒を’X’と表示する。 |
| 03 | 入力要求表示 | game\_progress関数内で、現在のターンのプレイヤーと入力を要求する文章を表示する。 |
| 04 | 結果表示 | game\_progress関数内でprint\_result関数を呼び出し、結果を表示する。勝利条件を満たした場合、勝者を表示、引き分け条件を満たした場合、引き分けを表示する。 |

## フローチャート

フローチャートを図 2に示しています。

ダイアグラム

自動的に生成された説明

図 2フローチャート

図 3機能8

# 画面構成

画面構成を図 5～図 6に示します。これらの図はチュートリアルに表示する内容に、緑字で説明を加えたものです。以下に文字の色分けについて説明します。

* 緑字… 本仕様書のための説明であり、チュートリアル、実際のゲーム画面どちらにも表示しない。
* 青字… チュートリアルの説明であり、チュートリアルのみに表示し、実際のゲーム画面には表示しない。
* 赤字… 警告を表す文字であり、チュートリアル、実際のゲーム画面どちらにも表示する。
* 白字… ゲームの進行を表す文字であり、チュートリアル、実際のゲーム画面どちらにも表示する。

テキスト

自動的に生成された説明

図 4画面構成1

テキスト

自動的に生成された説明

図 5画面構成2

テキスト

中程度の精度で自動的に生成された説明

図 6画面構成3