2024

三目並べ 詳細仕様書

aチーム

岡野 有真、桑原 涼輔、枦山 晃平

目次

[1. 概要 3](#_Toc166846277)

[2. 画面遷移図 3](#_Toc166846278)

[3. ファイル構成 6](#_Toc166846279)

[4. 機能構成 7](#_Toc166846280)

[4-1. tic\_tac\_toe.cに含まれる関数 7](#_Toc166846281)

[4-1-1. main関数 7](#_Toc166846282)

[4-2. game\_start.cに含まれる機能 8](#_Toc166846283)

[4-2-1. 名前を入力する機能 8](#_Toc166846284)

[4-2-2. 先攻後攻を決める機能 8](#_Toc166846285)

[4-3. game\_playing.cに含まれる機能 10](#_Toc166846286)

[4-3-1. 〇か×を入力する機能 10](#_Toc166846287)

[4-4. game\_finish.cに含まれる機能 11](#_Toc166846288)

[4-4-1. 操作の例外処理機能 11](#_Toc166846289)

[4-4-2.判定機能 12](#_Toc166846290)

[4-4-3. リトライ機能 13](#_Toc166846291)

# 概要

詳細仕様書では、三目並べの画面遷移図、ファイル構成、機能構成について記載する。

# 画面遷移図

テキスト

自動的に生成された説明

テキスト

自動的に生成された説明

～～とーーには入力した名前が入る

テキスト

自動的に生成された説明テキスト

自動的に生成された説明テキスト

自動的に生成された説明テキスト

自動的に生成された説明

Aキーで左に入力

左上に〇を入力したい

テキスト

自動的に生成された説明

Wキーで上に入力

△はマスの真ん中に戻る

スペースキーで決定

次の番の入力

3マス揃わず9マス全部埋まる

縦か横か斜めで同じ記号が3マス揃う

グラフィカル ユーザー インターフェイス, アプリケーション

自動的に生成された説明グラフィカル ユーザー インターフェイス

自動的に生成された説明

(次ページへ続く)

(前ページの続き)

グラフィカル ユーザー インターフェイス

自動的に生成された説明グラフィカル ユーザー インターフェイス, アプリケーション

自動的に生成された説明

グラフィカル ユーザー インターフェイス, アプリケーション

自動的に生成された説明

**No**

**Yes**

三目並べ終了

1. から三目並べ再開

図1：三目並べの流れ(①から⑤は順番を表す)

コンピューターのスクリーンショット

自動的に生成された説明図1に三目並べ全体の画面遷移についてまとめた。また、すでに〇、×が入力されているマスに入力を試みた際の画面遷移を図2に示す。

このマスに×を入力

△を左上のマスに移動後、決定

コンピューターのスクリーンショット

自動的に生成された説明

図2：入力後のエラー処理

付番にはそれぞれ下の機能が必要である。

1. 名前を入力する機能
2. 先攻後攻を決める機能
3. ○か×を入力する機能
4. 判定機能
5. リトライ機能

また、図2では操作の例外処理機能が必要になる。

# ファイル構成

「Tic\_Tac\_Toe」というフォルダ及びソリューションをVisual Studio 2022で作成する。

同じソリューション内に、tic\_tac\_toe.c、game\_start.c 、game\_playing.c 、game\_finish.c のソースコードファイルを作成する。それぞれのソースコードファイルには以下の役割がある。

フォルダ「Tic\_Tac\_Toe」… 三目並べに必要なファイルをまとめたフォルダ

tic\_tac\_toe.sln … Visual Studio 2022の三目並べ用のソリューション

tic\_tac\_toe.c … main関数に三目並べの全体の処理の流れを記述したプログラムファイル

game\_start.c … 名前入力をする機能と先攻後攻を決める機能を記述したプログラムファイル

game\_playing.c … 〇×を入力する機能を記述したプログラムファイル

game\_finish.c … 判定機能とリトライ機能を記述したプログラムファイル

# 関数構成

## 4-1. tic\_tac\_toe.cに含まれる関数

### 4-1-1. main関数

　main関数には三目並べの全体の処理の流れを記述する。  
main関数のフローチャートは以下の通りである。



図 1:main関数のフローチャート

|  |  |  |  |
| --- | --- | --- | --- |
| 変数 | 内容 | 型 | 初期値 |
| 名前を格納する変数1 | プレイヤー１の名前を格納する変数 | char[] | “” |
| 名前を格納する変数2 | プレイヤー２の名前を格納する変数 | char[] | “” |
| プレイヤー１の先攻後攻変数 | プレイヤー１が先行か後攻か判断する変数 | int | 0 |
| プレイヤー2の先攻後攻変数 | プレイヤー２が先行か後攻か判断する変数 | int | 0 |

## 4-2. game\_start.cに含まれる関数

### 4-2-1. 名前を入力する関数

この関数は引数として三目並べ開始時に『プレイヤー名を入力してください』と表示した一行下に、二人の名前の間に改行をいれて入力する。この時入力した名前は、順番が回ってきたときと勝敗が決したときに表示される。全角１文字以上全角10文字以内として入力する。それ以外の入力の場合はターミナルの出力をクリアして再度入力表示をする。文字コードはUTF-8とする。

表 1:名前を入力する関数の変数

|  |  |  |  |
| --- | --- | --- | --- |
| 変数 | 内容 | 型 | 初期値 |
| カウント変数 | プレイ人数を数える変数 | int | 0 |

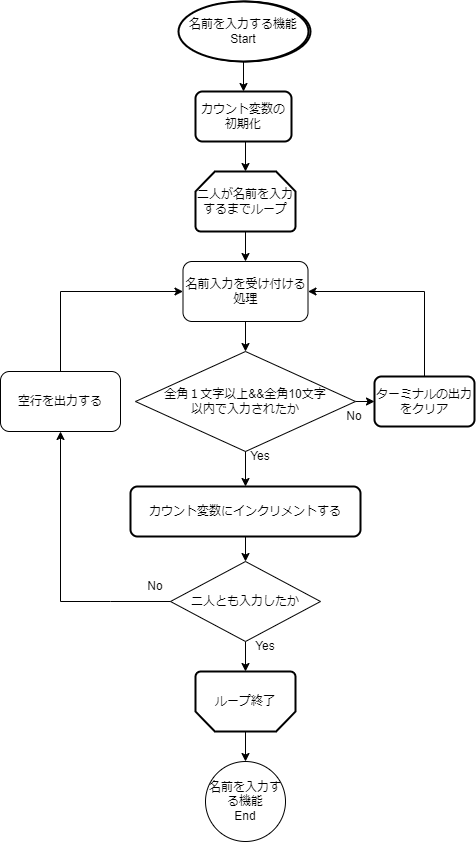


図 2：名前を入力する機能のフローチャート

### 4-2-2. 先攻後攻を決める関数

引数としてプレイヤー１の先攻後攻変数のポインタとプレイヤー２の先攻後攻変数のポインタを持つ。Rand関数を用いて奇数だったら最初に名前を入力したほうが先攻になりプレイヤー１の先攻後攻変数には1、偶数だったら最初に名前を入力したほうが後攻になり、プレイヤー１の先攻後攻変数には0を持つ。Rand関数を使うにあたってseed値はTime関数を用いて疑似的な乱数にする。

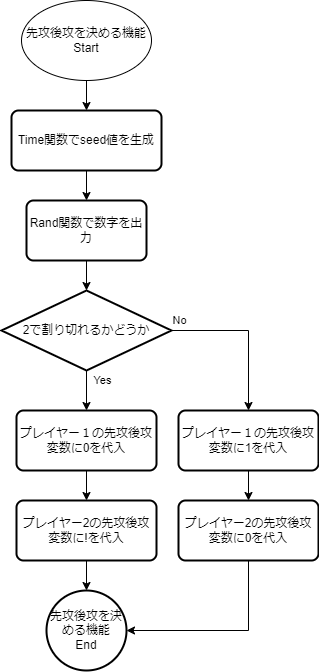


図 3：先攻後攻を決める機能のフローチャート

## 4-3. game\_playing.cに含まれる関数

### 4-3-1. 〇か×を入力する関数

二次元配列3×3の、〇×を入力するための「〇×入力用盤面」、また〇×の入力結果を表示するための「〇×表示用盤面」を用意する。△を操作子として使う。操作するための盤面では「A：左、W：上、S：下、D：右」のキー入力を行うことで盤面を△が移動するように表示する。スペースキーの入力を行うことで先攻の場合は〇、後攻の場合は×を△の位置で入力して盤面に表示する。また、盤面内の〇、×、―、△は全角文字で表示し、〇と×入力済みのマス目には入力できない。

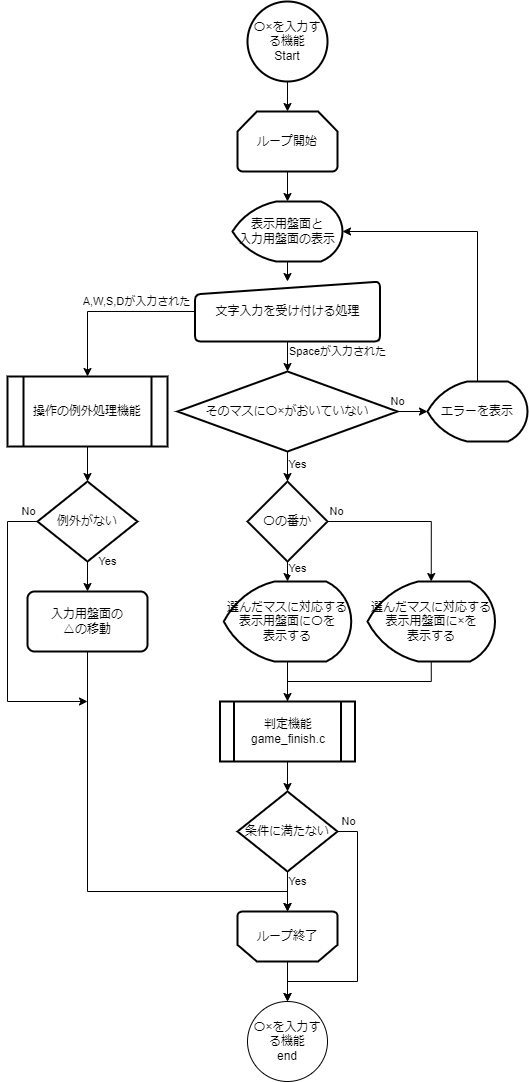


図 4：〇×を入力する機能のフローチャート

## 4-4. game\_finish.cに含まれる関数

### 4-3-2. OperatingException関数

表 1 ：OperatingException関数

|  |  |
| --- | --- |
| OperatingException関数 | |
| △の移動操作における例外処理を行う関数 | |
| 引数 | char型：〇×入力用盤面を表す配列 |
| 返り値 | int型：ゲームの進行を判定する文字 |

コンピューターの画面のスクリーンショット

自動的に生成された説明範囲外に出ようとしたら直前の位置に留まる。

範囲外へ移動しようとすると

コンピューターの画面のスクリーンショット

自動的に生成された説明

そのままの場所にとどまる

図5：範囲外に出ようとした時の処理

ダイアグラム

自動的に生成された説明

図 6：OperatingException関数のフローチャート

## 4-4. game\_finish.cに含まれる関数

### 4-4-1. GameDecision関数

表 1 ：GameDecision関数

|  |  |
| --- | --- |
| GameDecision関数 | |
| 三目並べの勝敗を決定する関数 | |
| 引数 | char 型：〇×表示用盤面を表す配列,  char型：先攻の人の名前  char型：後攻の人の名前 |
| 返り値 | char型：ゲームの進行を判定する文字 |

1マスずつ何の文字が格納されているか確認して、先に縦か横か斜めで同じ記号が3マス揃えば「～～さんの勝ちです。」(～～には入力した名前が入る)と表示する。3マス揃っていなかった場合は再び入力できる。お互いに3マス揃わずに9マスすべてが〇か×で埋まったら、「引き分けです。」と表示される。その後、勝敗が決していた場合はDecisionの頭文字である「半角の大文字『D』」を戻り値として、9マス埋まっておらず3マス揃っていない場合はFillの頭文字である「半角の大文字『F』」を戻り値とする。

ダイアグラム

自動的に生成された説明

図 7：GameDecision関数のフローチャート

以下に表１の条件を記載する。表1の盤面の例は、条件に対しての一例でありプログラムを指定するものではない。

グラフィカル ユーザー インターフェイス, アプリケーション, テーブル, Excel

自動的に生成された説明表 2 正解条件 （左：盤面の例、右：正解条件）

|  |  |  |
| --- | --- | --- |
| 条件 | 1, 4, 7が等しい | 3, 6, 9が等しい |
| 1, 2, 3が等しい | 3, 5, 7が等しい |
| 1, 5, 9が等しい | 4, 5, 6が等しい |
| 2, 5, 8が等しい | 7, 8, 9が等しい |

### 4-3-3. Retry\_Game関数

表 3：Retry\_Game関数

|  |  |
| --- | --- |
| Retry\_Game関数 | |
| 再び三目並べを遊ぶか確認する関数 | |
| 引数 | void |
| 戻り値 | int型：もう一度遊ぶか判断する変数 |

引き分けか勝敗が決した後、「もう一度遊びますか？Yes：0以外のキー or No：0」と表示する。Yesを選択した場合は三目並べが再開され、Noを選択した場合は、三目並べが終了する。

ダイアグラム

中程度の精度で自動的に生成された説明

図 8：Retry\_Game関数のフローチャート