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](#_Toc167261287)

[2. 画面遷移図 3](#_Toc167261288)

[3. ファイル構成 6](#_Toc167261289)

[4. マクロ定義、構造体、列挙型の宣言 7](#_Toc167261290)

[マクロ定義 7](#_Toc167261291)

[列挙型 7](#_Toc167261292)

[FIRSTMOVE列挙型(game\_start.h) 7](#_Toc167261293)

[BOOL列挙型(game\_finish.h) 7](#_Toc167261294)

[構造体 9](#_Toc167261295)

[PLAYER構造体(game\_start.h) 9](#_Toc167261296)

[5. 関数構成 10](#_Toc167261297)

[5-1. tic\_tac\_toe.cに含まれる関数 10](#_Toc167261298)

[5-1-1. main関数 10](#_Toc167261299)

[5-2. game\_start.cに含まれる関数 13](#_Toc167261300)

[5-2-1. InputName関数 13](#_Toc167261301)

[5-2-2.ChoiceFirstAttack関数 15](#_Toc167261302)

[5-2-3. ChangePlay関数 16](#_Toc167261303)

[5-3. game\_playing.cに含まれる関数 17](#_Toc167261304)

[5-3-1. ShowPlayer関数 17](#_Toc167261305)

[5-3-2. ShowBoard関数 18](#_Toc167261306)

[5-3-3. Input関数 19](#_Toc167261307)

[5-3-4.ChoicePosition関数 20](#_Toc167261308)

[5-3-5. PlayerMove関数 22](#_Toc167261309)

[5-3-6. OperatingException関数 24](#_Toc167261310)

[5-4. game\_finish.cに含まれる関数 26](#_Toc167261311)

[5-4-1. GameDecision関数 26](#_Toc167261312)

[5-4-2. ShowResult関数 28](#_Toc167261313)

[5-4-3. Retry\_Game関数 29](#_Toc167261314)

# 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、宣言・定義、関数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

左上に〇を入力したい

Aキーで左に入力

テキスト

自動的に生成された説明

C

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

B

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

付番の画面はそれぞれ下の関数で表示される。

InputName関数→①

ChoiceFirstAttack関数→②

ShowPlayer関数→③-A、③-B

ShowBoard関数→③-C

ShowResult関数→④

Retry\_Game関数→⑤

また、図2はChoicePosition関数の「エラーを表示」のところで表示される。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。また、game\_start.h、game\_playing.h、game\_finish.hのヘッダーファイルも作成する。それぞれの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.h…game\_start.cのヘッダーファイル,ここでPLAYER構造体とFIRSTMOVE列挙型、PLAYWAIT列挙型を宣言する。

game\_start.c …InputName関数とChoiceFirstAttack関数、ChangePlay関数を記述したプログラムファイル

game\_playing.h…game\_playing.cのヘッダーファイル

game\_playing.c …ShowPlayer関数とShowBoard関数、Input関数とChoicePosition関数とPlayerMove関数とOperatingException関数を記述したプログラムファイル

game\_finish.h…game\_finish.cのヘッダーファイル、ここでBOOL列挙型を宣言する。

game\_finish.c … GameDecision関数とShowResult関数,Retry\_Game関数を記述したプログラムファイル

# マクロ定義、構造体、列挙型の宣言

## マクロ定義

|  |  |  |  |
| --- | --- | --- | --- |
| 内容 | 定義名 | 値 | ファイル名 |
| プレイヤーの数 | PLAYER\_MEM | 2 | tic\_tac\_toe.c |
| 名前のサイズ | NAME\_SIZE | 21 | tic\_tac\_toe.c |
| 盤面のサイズ:x | BOARD\_SIZE\_X | 3 | tic\_tac\_toe.c |
| 盤面のサイズ:y | BOARD\_SIZE\_Y | 3 | tic\_tac\_toe.c |
| プレイヤー１ | PLAYER\_ONE | 0 | game\_start.h |
| プレイヤー２ | PLAYER\_TWO | 1 | game\_start.h |

名前のサイズは21としているが、内訳としては全角１０文字(20)+ヌル文字(1)とする。

また盤面のサイズは二次元配列としているためXとYに分け定義している。

## 列挙型

### FIRSTMOVE列挙型(game\_start.h)

|  |  |
| --- | --- |
| FIRSTMOVE列挙型 | |
| 先攻か後攻か判断する列挙型 | |
| 列挙定数 | FIRST |
| LAST |

### BOOL列挙型(game\_finish.h)

|  |  |
| --- | --- |
| BOOL列挙型 | |
| TRUE、FALSEを返す列挙型 | |
| 列挙定数 | FALSE |
| TRUE |

## 構造体

### PLAYER構造体(game\_start.h)

|  |  |  |  |
| --- | --- | --- | --- |
| PLAYER構造体 | | | |
| プレイヤーの情報を格納する構造体 | | | |
| メンバ | 変数 | 型 | 初期値 |
| 名前を格納する変数 | char[NAME\_SIZE] | “” |
| 先攻か後攻か判断する列挙型変数 | FIRSTMOVE型 | FIRST |

# 関数構成

## 5-1. tic\_tac\_toe.cに含まれる関数

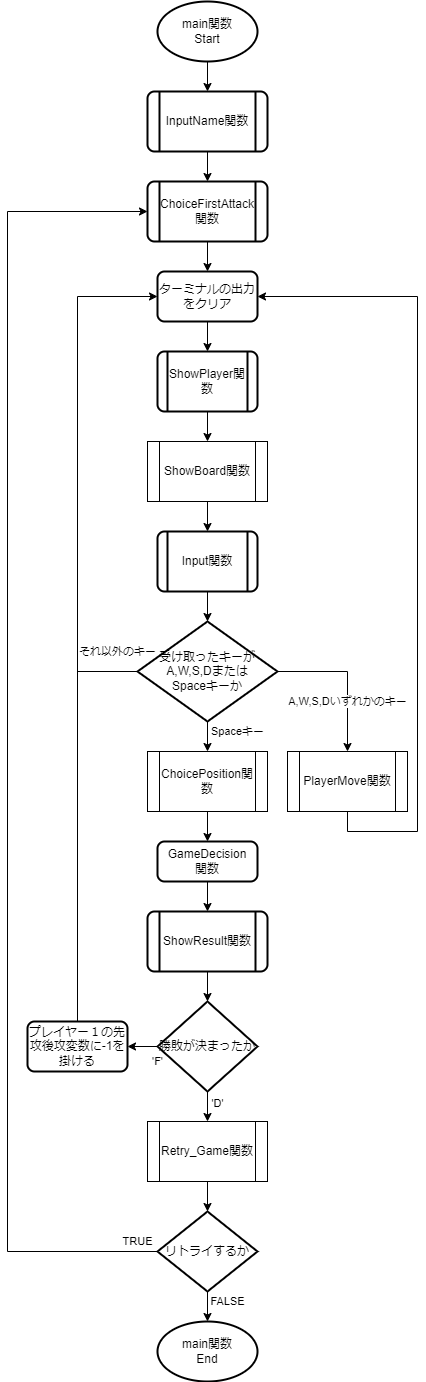
### 5-1-1. main関数

表 1：main関数

|  |  |
| --- | --- |
| main関数 | |
| 三目並べの全体の処理の流れを表す関数 | |
| 引数 | void |
| 返り値 | void |

　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。

|  |  |  |  |
| --- | --- | --- | --- |
| 変数 | 内容 | 型 | 初期値 |
| 手番変数 | 現在の手番を格納する変数 | int | PLAYER\_ONE |
| PLAYER構造体配列 | プレイヤー情報を格納するための構造体 | PLAYER[PLAYER\_MEM] | 構造体に記載 |
| 〇×入力用盤面配列 | 操作子を動かすための配列 | char[BOARD\_SIZE\_X][ BOARD\_SIZE\_Y] | ‘-’ |
| 〇×表示用盤面配列 | 〇×表示や判定に使うための配列 | char[BOARD\_SIZE\_X][ BOARD\_SIZE\_Y] | 123456789 |



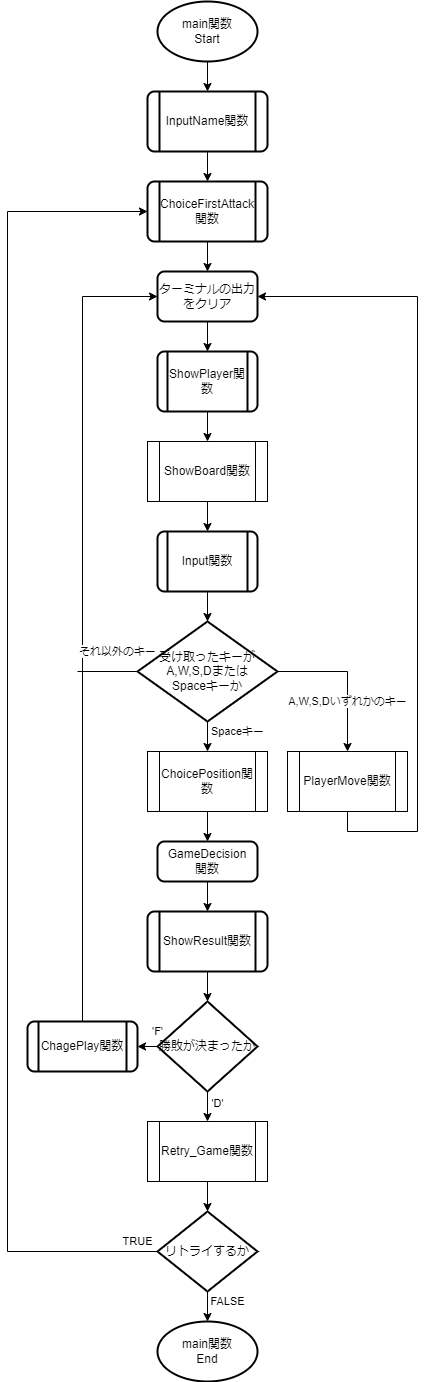


図 1:main関数のフローチャート

## 5-2. game\_start.cに含まれる関数

### 5-2-1. InputName関数

表 3：InputName関数

|  |  |
| --- | --- |
| InputName関数 | |
| プレイヤー名を入力するための関数 | |
| 引数 | PLAYER型:PLAYER構造体配列の先頭ポインタ |
| 返り値 | void |

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。入力した文字中に1Byte文字が入っていれば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Shift-JIS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 |

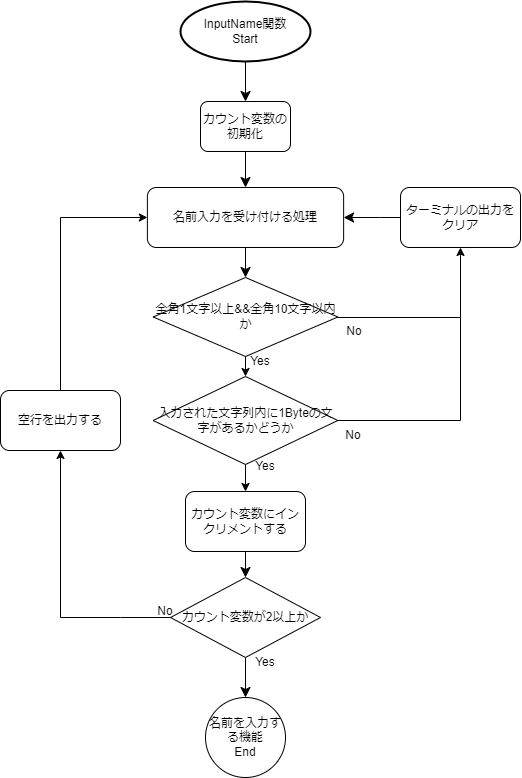


図 2:InputName関数のフローチャート

### 5-2-2.ChoiceFirstAttack関数

表 5：ChoiceFirstAttack関数

|  |  |
| --- | --- |
| ChoiceFirstAttack関数 | |
| 先攻後攻を決定するための関数 | |
| 引数 | int型:手番変数ポインタ  PLAYER型:PLAYER構造体配列の先頭ポインタ |
| 返り値 | void |

引数として手番変数のポインタとPLAYER構造体配列の先頭ポインタを持つ。rand関数を用いて奇数だったら一人目が先攻になりPLAYER構造体のメンバのFIRSTMOVE型変数にFIRSTを代入する、偶数だったらPLAYER構造体のメンバのFIRSTMOVE型変数にLASTを代入する。二人目にはその逆の値を格納する。rand関数を使うにあたってsrand関数のseed値はTime関数を用いて乱数にする。

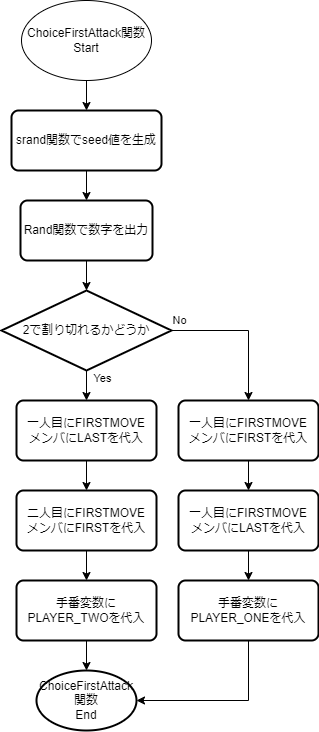


図 3：ChoiceFirstAttack関数のフローチャート

### 5-2-3. ChangePlay関数

|  |  |
| --- | --- |
| ChangePlay関数 | |
| 操作する人を交代する関数 | |
| 引数 | int型:手番変数 |
| 返り値 | int型 |



図 3：ChangePlay関数のフローチャート

## 5-3. game\_playing.cに含まれる関数

### 5-3-1. ShowPlayer関数

|  |  |
| --- | --- |
| ShowPlayer関数 | |
| プレイヤーの手番を表示する関数 | |
| 引数 | PLAYER型:PLAYER構造体 |
| 返り値 | void型 |

　ShowPlayer関数は手番であるPLAYER構造体を引数として持つ。



図 4：ShowPlayer関数のフローチャート

### 5-3-2. ShowBoard関数

|  |  |
| --- | --- |
| ShowBoard関数 | |
| 盤面の表示をする関数 | |
| 引数 | char型 : 〇×表示用盤面配列  char型 : 〇×入力用盤面配列 |
| 返り値 | void型 |

ダイアグラム

自動的に生成された説明

図 5：ShowBoard関数のフローチャート

### 5-3-3. Input関数

|  |  |
| --- | --- |
| Input関数 | |
| 〇または×の位置を入力して盤面に表示させる関数 | |
| 引数 | void型 |
| 返り値 | int型:文字コード を返す |

表 6：Input関数

グラフィカル ユーザー インターフェイス, テキスト

自動的に生成された説明

図 6:Input関数のフローチャート

### 5-3-4.ChoicePosition関数

表 7：ChoicePosition関数

|  |  |
| --- | --- |
| ChoicePosition関数 | |
| 〇または×を選んだマスに表示させる関数 | |
| 引数 | PLAYER型:PLAYER構造体  char型：〇×入力用盤面配列  char型：〇×表示用盤面配列の先頭ポインタ |
| 返り値 | void |

ChoicePosition関数は手番であるPLAYER構造体と入力と出力用の盤面の配列を引数として持つ。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。

・〇×表示用盤面について

〇×表示用盤面は、三目並べの途中経過を表示することで、〇×を置いてある場所や勝負の過程を一目で認識できるようにしている。また、本盤面で使用される配列は勝敗の判定にも使用される。

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本盤面で使用する配列は、char型の配列であり、半角数字の1, 2, 3, 4, 5, 6, 7, 8, 9で構成されている。配列の並べ方の概念図を図2に示す。

図2：配列の並べ方の概念図

〇×入力用盤面で操作子△の移動が完了し、スペースキーが押下されると、先攻の場合、本配列の対応する場所に半角小文字「o」が代入され、後攻の場合、半角小文字「x」が代入される。半角小文字oは本盤面で表示される全角の丸「〇」に対応し、半角小文字xは全角のバツ「×」に対応する。1～9の半角数字は全角ハイフン「―」に対応する。

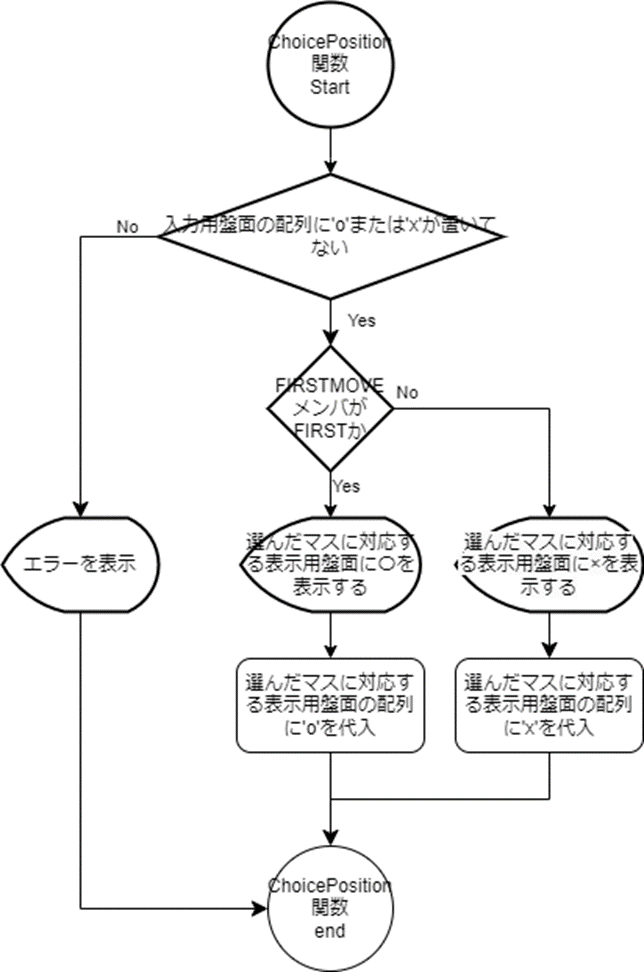


図 7:ChoicePosition関数のフローチャート

### 5-3-5. PlayerMove関数

表 8：PlayerMove関数

|  |  |
| --- | --- |
| PlayerMove関数 | |
| 例外操作がないか判断し操作子△を移動させる関数 | |
| 引数 | char型：〇×入力用盤面配列の先頭ポインタ |
| 返り値 | void |

△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。

　・〇×入力用盤面について

〇×入力用盤面は操作子△を移動することで、指定の場所に〇または×を入力することを目的としている。

本盤面で使用する配列は、char型の配列であり、半角数字の「1」と「0」で構成されている。半角数字の0は実際に表示される〇×入力用盤面の全角ハイフン「―」に対応し、半角数字の1は全角三角「△」に対応している。

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

自動的に生成された説明本配列は、AWSDキーの入力に対応して0と1を入れ替えることにより操作子△の移動また〇×の入力操作を可能にする。概念図として図１を示す。

0と1を入れ替え

Aキーで左に移動

〇を入力したい

図1：操作子△の移動方法の概念図

1の場所は初め3×3マスの真ん中のマスに置かれており、スペースキーを押下した後は再び真ん中のマスに1は戻る。また、スペースキーを押下した後の操作子△の場所に、〇または×が代入されていた場合は、1を真ん中のマスに戻し再び操作子△の移動を要求する。

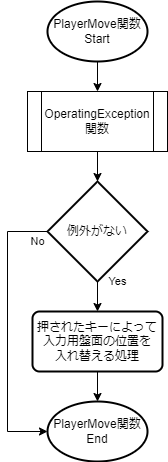


図 8：PlayerMove関数のフローチャート

### 5-3-6. 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

OperatingException関数はゲームの進行を判定する変数として、Yesの時はBOOL型TRUEを返し、Noの時はBOOL型FALSEを返す。

ダイアグラム

自動的に生成された説明

図 9：OperatingException関数のフローチャート

## 5-4. game\_finish.cに含まれる関数

### 5-4-1. 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,  PLAYER型:PLAYER構造体配列 |
| 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。GameDecision関数は、引き分けの場合は「半角の大文字『D』」、〇が勝利した場合は「半角の大文字『O』」、×が勝利した場合は「半角の大文字『X』」、9マス埋まっておらず3マス揃っていない場合は「半角の大文字『F』」を戻り値とする。

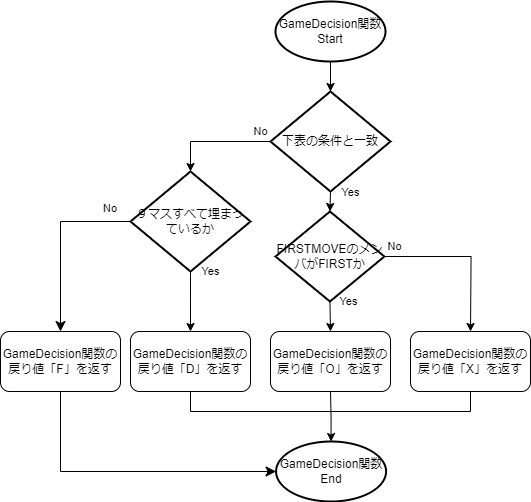


図 10：GameDecision関数のフローチャート

以下に表 2の条件を記載する。表 2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 |

### 5-4-2. ShowResult関数

|  |  |
| --- | --- |
| ShowResult関数 | |
| 結果を判定し表示する関数 | |
| 引数 | char型: ゲームの進行を判定する文字  PLAYER型:PLAYER構造体配列 |
| 返り値 | char型: ゲームの進行を判定する文字 |

グラフ, ダイアグラム

自動的に生成された説明

図 11：ShowResult関数のフローチャート

### 5-4-3. 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。Noの条件は半角の0とし、Yesの条件はNoの条件以外とする。Retry\_Game関数は、もう一度遊ぶか判断する変数として、Yesの時はBOOL型TRUEを返し、Noの時はBOOL型FALSEを返す。

図 12：Retry\_Game関数のフローチャート

