Eチーム

川崎　聖矢　谷口　亮太郎

詳細仕様書

目次

[1. 概要 2](#_Toc167871986)

[2. 定義 2](#_Toc167871987)

[3. ファイル構成 2](#_Toc167871988)

[3.1 全体進行 2](#_Toc167871989)

[3.2 入力機能 2](#_Toc167871990)

[3.3 処理機能 3](#_Toc167871991)

[3.4 出力機能 3](#_Toc167871992)

[3.5 ヘッダファイル 3](#_Toc167871993)

[4. 画面構成 4](#_Toc167871994)

[5. 全体のフローチャート図 6](#_Toc167871995)

[6. 機能構成 7](#_Toc167871996)

[6.1 入力機能 7](#_Toc167871997)

[6.2 処理機能 9](#_Toc167871998)

[6.3 出力機能 10](#_Toc167871999)

## 概要

本仕様書はファイル構成、機能構成、フローチャート図、画面構成の説明を記載する。

## 定義

記号 「o」と「x」を指す。

プレイヤー 三目並べを操作する人。

## ファイル構成

### 全体進行

|  |  |
| --- | --- |
| ファイル名 | main.c |
| 関数 | main |
| 機能 | * ゲーム進行 |

### 入力機能

|  |  |
| --- | --- |
| ファイル名 | input.c  input.h |
| 関数 | BoardInit  PlayerInput  BoardUpdate |
| 機能 | * 盤面の初期化 * 入力を受け取る * 盤面の更新 |

### 処理機能

|  |  |
| --- | --- |
| ファイル名 | judge.c  judge.h |
| 関数名 | ResultJudge  NextTurn |
| 機能 | * 勝敗判定 * 手番進行 |

### 出力機能

|  |  |
| --- | --- |
| ファイル名 | output.c  output.h |
| 関数 | PrintBoard  PrintResult |
| 機能 | * 盤面を表示 * 勝敗の結果を表示 |

### ヘッダファイル

|  |  |
| --- | --- |
| ファイル名 | enum.h |
| プロトタイプ宣言 | (型名) RESULT  (列挙子)PLAYER1\_WIN, PLAYER2\_WIN, DRAW, RESULT\_NONE  (型名)TURN  (列挙子)TURN\_PLAYER1, TURN\_PLAYER2  ※define TRUE,FALSEの使用は自由。  マクロ  **NUM**   * **配列の行数** * **配列の列数**   #define NUM 3  #define TRUE 1  #define FALSE 0  グローバル変数  char g\_board[NUM][NUM]; |

## 画面構成

初期画面は以下の通りである。

テキスト

自動的に生成された説明

3×3の盤面を表示する。

記号が入力されていないマスの内部に1～9の数字を表示する。

画面の構成は以下のように設計する。

テキスト

自動的に生成された説明

先攻の順番

1～9の番号をキーボード入力。

(例)

5番を入力。

「5」が「o」に置き換わる。

テキスト

自動的に生成された説明

後攻の順番

1～9の番号をキーボード入力。

(例)

9番を入力。

「9」が「x」に置き換わる。

テキスト

自動的に生成された説明

３つ並ばずに盤面が全て埋まった場合

「引き分けです」と表示される。

３つ並んだら「○○の勝ち」と表示される。

（例）

先攻が３つ並んだ場合

「先攻の勝ち」と表示される。

テキスト

自動的に生成された説明

## 全体のフローチャート図

**盤面の初期化**

**BoardInit**

**開始**

**入力を受け取る**

**PlayerInput**

RESULT\_NONE以外

RESULT\_NONE

**手番進行**

**NextTurn**

**終了**

**勝敗の結果を表示**

**PrintResult**

**勝敗判定**

**ResultJudge**

**盤面表示**

**PrintBoard**

**盤面の更新**

**BoardUpdate**

**盤面表示**

**PrintBoard**

## 機能構成

### 入力機能

* + 1. **盤面の初期化**

(関数名)BoardInit

(引数)なし

(返り値)なし

グローバル変数のchar型配列（char g\_board[NUM][NUM]）の各要素に空白（半角）を入れる。

* + 1. **入力を受け取る**

(関数名)PlayerInput

(引数)int\*(行番号), int\*(列番号),TURN(手番)

(返り値)なし

1. **メッセージを表示する。**

* 引数のTURNがTURN\_PLAYER1の場合

“先攻の番です。”

“番号を入力してください。”

“番号 : ”

と表示する。

テキスト

自動的に生成された説明

* 引数のTURNがTURN\_PLAYER2の場合

“後攻の番です。”

“番号を入力してください。”

“番号 : ”

と表示する。

テキスト

自動的に生成された説明

1. **入力を受け取る**
2. **入力が正しいか確認する。**
   * 文字（半角、全角）や記号が入力された場合。
   * 数字が範囲外。
   * スペース（半角、全角）を入れた場合。
   * "入力が正しくないです。”

と表示して1. に戻る。

1. **入力から対応する配列の行と列を確認し、int型ポインタにいれる。**

それぞれ割り算とその余りで求められる。

（行番号）(入力 - 1) / NUM （列番号）(入力 - 1) % NUM

(例)真ん中(5)の場合、5 – 1 = 4、4を3で割ったら1余り1で1行1列と確認できる。

1. **配列の中身を確認する。**

* 正しくない場合
  + 配列の中身が空白（半角）” “でない。

"入力が正しくないです。”

と表示して1. に戻る。

* + 1. **盤面の更新**

(関数名)BoardUpdate

(引数)int(行番号),int(列番号),TURN(手番)

(返り値)なし

* + 引数のTURNがTURN\_PLAYER1の場合

グローバル変数の配列（char g\_board[NUM][NUM]）に「o」を入れる。

* + 引数のTURNがTURN\_PLAYER2の場合

グローバル変数の配列（char g\_board[NUM][NUM]）に「x」を入れる。

* + int型の引数が0～2以外の場合、Assert関数で強制終了する。

### 処理機能

* + 1. **勝敗判定**

(関数名)ResultJudge

(引数)TURN(手番)

(戻り値)RESULT(結果)

1. **どちらの手番かを受け取る**
2. **手番のプレイヤーの記号が3つ連続で並んでいるか確認する。**
   1. 配列の各行に3つ連続で同じ記号があるか確認。
      * 3つ連続で同じ記号がある場合
        + 引数がTURN\_PLAYER1…PLAYER1\_WINを返す
        + 引数がTURN\_PLAYER2…PLAYER2\_WINを返す
   2. 配列の各列に3つ連続で同じ記号があるか確認。
      * 3つ連続で同じ記号がある場合
        + 引数がTURN\_PLAYER1…PLAYER1\_WINを返す
        + 引数がTURN\_PLAYER2…PLAYER2\_WINを返す
   3. 配列の斜めに3つ連続で同じ記号があるか確認。
      * 3つ連続で同じ記号がある場合
        + 引数がTURN\_PLAYER1…PLAYER1\_WINを返す
        + 引数がTURN\_PLAYER2…PLAYER2\_WINを返す
3. **全ての配列の要素が埋まっていないことを確認する。**
   * + 埋まっていた場合DRAWを返す
4. **ここまでで返り値がない場合**
   * + RESULT\_NONEを返す
     1. **手番進行**

(関数)NextTurn

(引数)TURN(現在の手番)

(戻り値)TURN(次の手番)

1. 引数でTURN\_PLAYER1かTURN\_PLAYER2を受け取る。
2. 返り値は受け取ったものとは逆を返す。

### 出力機能

* + 1. **盤面表示**

(関数名)PrintBoard

(引数)なし

(返り値)なし

1. 配列の中身によって表示を変える

* グローバル変数のchar型配列（char g\_board[NUM][NUM]）に記号が入っている場合  
  %2cで2文字分の大きさで記号を表示する。
* グローバル変数のchar型配列（char g\_board[NUM][NUM]）に記号が入っていない場合  
  対応する数字を%2dの大きさで表示する。  
  対応する数字は((行番号) \* NUM) + (列番号 + 1)で求められる。  
  (例)  
  1行2列に対応する数字は(1 \* 3) + (2 + 1) = 6

1. 下の図のように表示する。

数字や文字は半角で表示する。

文字が書かれている

中程度の精度で自動的に生成された説明

* + 1. **勝敗の結果を表示**

(関数名)PrintResult

(引数)RESULT (勝敗結果)

(返り値)なし

受け取った引数によって表示を変える。

* PLAYER1\_WINの場合  
  “先攻の勝ちです。” と表示する。



* PLAYER2\_WINの場合



* DRAWの場合  
  “引き分けです。” と表示する。



* RESULT\_NONEの場合

Assert関数を呼び出して強制終了する。