

# C言語プログラミング能力認定試験

## 1 級実技試験

### テーマプログラム

[第5版]

1. 要求仕様書	・・・	1
2. システム仕様書	・・・	3
3. ソースプログラムリスト		
common.h	・・・	19
main.h	・・・	19
nyuukai.h	・・・	20
keisoku.h	・・・	20
sakujyo.h	・・・	20
main.c	・・・	21
nyuukai.c	・・・	28
keisoku.c	・・・	32
sakujyo.c	・・・	41

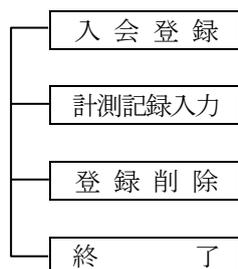
試験問題に記載されている会社名又は製品名は、それぞれ各社の商標又は登録商標です。  
なお、試験問題では、® 及び ™ を明記していません。

# アスレチッククラブ会員管理プログラム 要求仕様書

## 1. 目的

本システムは、アスレチッククラブにおける、メンバーの管理を目的としている。対象となるアスレチッククラブは、現状では設備の関係で、200人までの会員しか登録できないが、将来は数千人のメンバーを登録することも考えられる。

## 2. メニュー



## 3. 処理説明

- (1) 入会登録 入会希望者があったとき、メンバーの空きがあれば、空いている会員コードを与えて、会員登録する。空いている番号の中では、古いものを優先する。メンバーの空きがなければ、「残念ながらも今メンバーの空きがありません」と表示して終了する。
- (2) 計測記録入力
- ① 毎回、運動後に計測記録を、会員コード、運動日とともに入力する。
  - ② 入力されたデータをもとに、ある算式により、運動指数を計算する。
  - ③ 記録は、今までの運動回数、入会時データ、最高記録データ、最新 10 回分までの運動指数である。
  - ④ 結果の出力は、個人の履歴データと、全メンバー中の最高記録の 1 位から 10 位までの結果データを画面に表示する。
- (3) 登録削除 退会者があったとき、その計測記録を削除し、会員コードを解放する。  
(退会者の履歴データは保存しない。)

## 4. システム詳細

### (1) ファイルイメージ

#### ① 空きコード表：

会員コードの空きを記録した表

1レコード目は、空きコードの件数が格納され、2レコード目以降に、空きコードが古い順に格納される。

10	← 空きコード件数
5	
12	
65	
3	
126	
32	
1	
60	
30	
83	← 最新の空きコード

#### ② コード・データ対照表：

会員コードから計測データ表のデータのある行を示す表

1	32
2	51
3	68
4	24
.	.
.	.
.	.
.	.
199	120
200	7

↑  
会員コードは連番のため省略

↑  
データ表の位置を表す  
空きコードには0が入る

#### ③ 計測データ表：計測記録データファイル

会員コード	A	B	C	D	E	...	N	O	P

A：計測回数  
D：最高記録日付  
G：最新データ  
.....

B：初回日付  
E：最高記録データ  
H：1回前データ  
O：8回前データ

C：初回データ  
F：最新計測日付  
I：2回前データ  
P：9回前データ

### (2) 入力データ

#### ① 会員コード

#### ② 運動実績

	負荷	セット	回数
運動1	W1	S1	N1
運動2	W2	S2	N2
運動3	W3	S3	N3
運動4	W4	S4	N4
運動5	W5	S5	N5

#### ③ 負荷係数

運動1の係数 F1=0.24  
運動2の係数 F2=0.36  
運動3の係数 F3=0.52  
運動4の係数 F4=1.05  
運動5の係数 F5=2.13

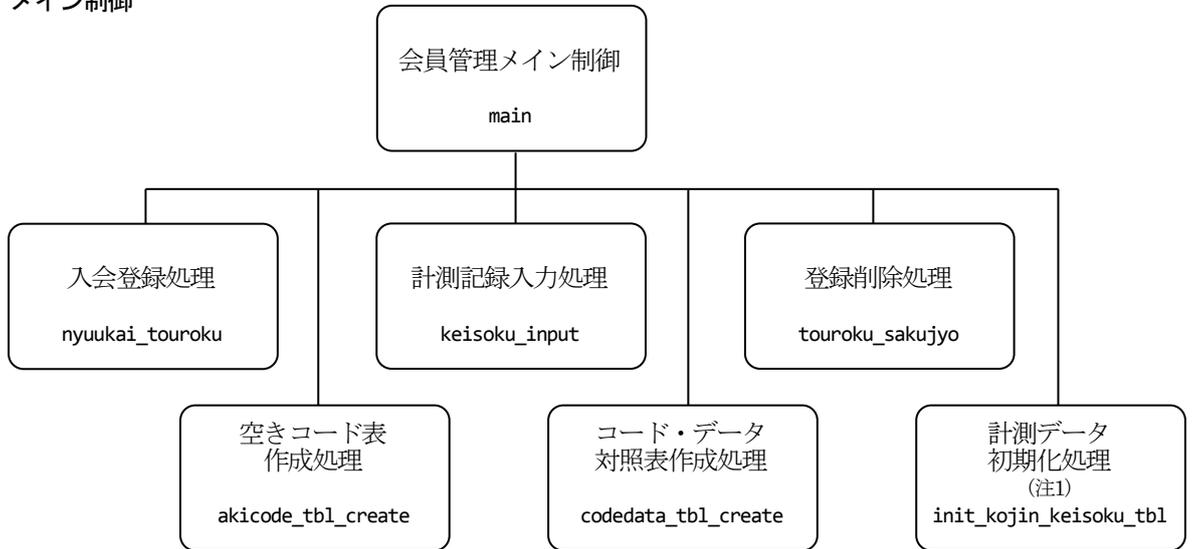
#### ④ 運動指数の計算式

$$\sqrt{\sum_{i=1}^5 F_i * W_i * S_i * (N_i^2 / (N_i - 1))}$$

# アスレチッククラブ会員管理プログラム システム仕様書

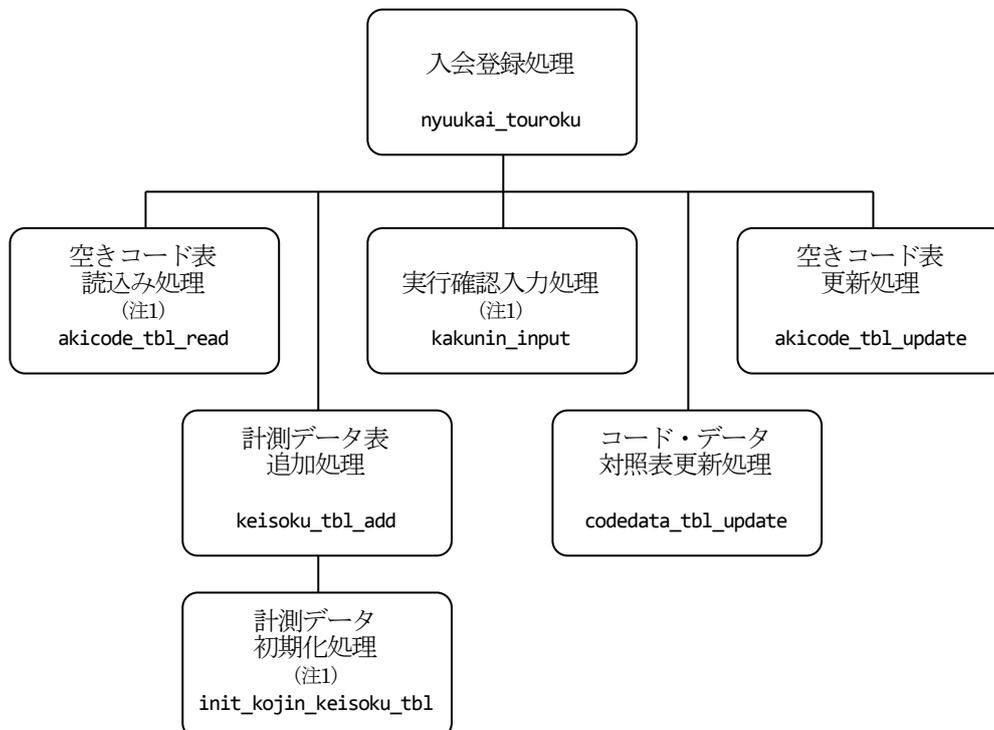
## 1. 関数構成図

### 1. 1. メイン制御



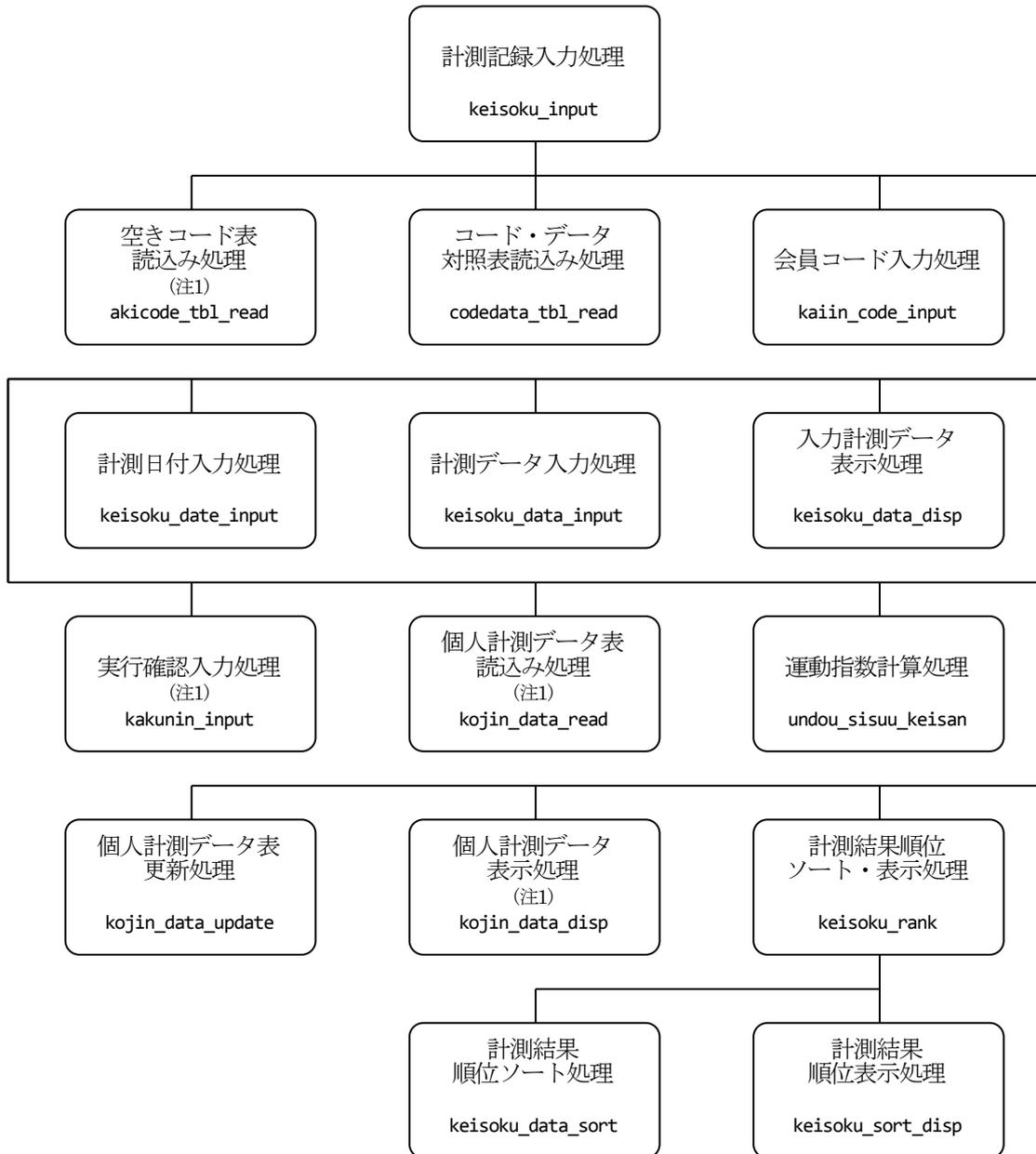
注1：共通ルーチン

### 1. 2. 入会登録処理



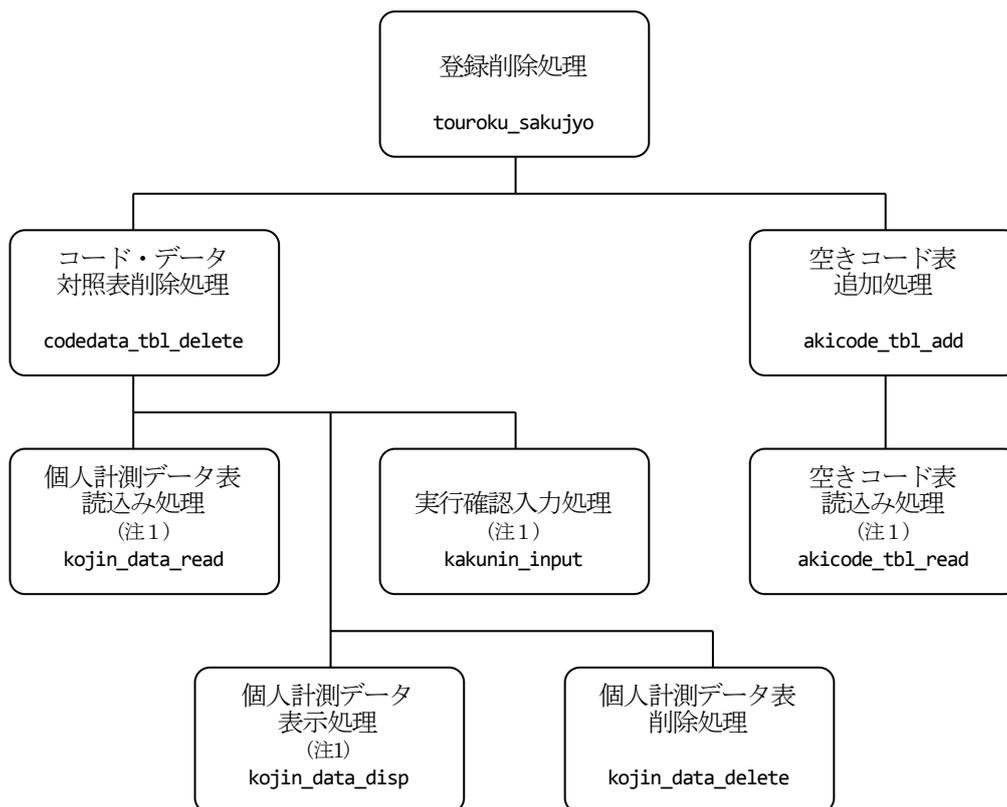
注1：共通ルーチン

1. 3. 計測記録入力処理



注1：共通ルーチン

1. 4. 登録削除処理



注1：共通ルーチン

## 2. ファイル設計

### 2. 1. ファイル一覧

No.	ファイル名	識別子	REC数	備考
1	空きコード表	akicode.tbl	可変	1レコード目：件数
2	コード・データ対照表	codedata.tbl	200	
3	計測データ表	keisoku.tbl	可変	
4	計測データ表ワーク	keisoku.tmp	可変	削除処理時使用

#### 2. 1. 1. 空きコード表

- ・現時点における空き会員コードを格納する。  
(1レコード目は、空きコードの件数とする。)

※ 初期状態

空きコード数 : 200

空きコード : 1から200が順にセットされる。

No.	項目名	フィールド名	属性	備考
1	空きコード数	akicode_tbl	int	1レコード目
2	空きコード	akicode_tbl	int	MAXレコード数 200

#### 2. 1. 2. コード・データ対照表

- ① 会員毎の計測データ表ファイルのレコード位置を格納する。
- ② 対応する会員コードは、本ファイルのレコード番号とする。  
(例えば、1レコード目には、会員コード=1のデータ位置がセットされる。)
- ③ 未登録の会員には、0がセットされる。

※ 初期状態はすべて0がセットされる。

No.	項目名	フィールド名	属性	備考
1	計測データ表位置	codedata_tbl	int	200レコード固定

### 2. 1. 3. 計測データ表

- ① 会員毎の履歴データを格納する。
- ② 本ファイルは、会員の登録処理時に作成される。

No.	項目名	フィールド名	属性	備考
1	会員コード	kaiin_code	int	
2	計測回数	count	int	
3	初回日付	first_date	char[9]	YYYYMMDD
4	初回データ	first_data	int	
5	最高記録日付	max_date	char[9]	YYYYMMDD
6	最高記録データ	max_data	int	
7	最新計測日付	soku_date	char[9]	YYYYMMDD
8	計測データ	soku_data	int[10]	

※ 計測データ

- [0] … 最新データ
- [1] … 1回前データ
- [2] … 2回前データ
- [3] … 3回前データ
- [4] … 4回前データ
- [5] … 5回前データ
- [6] … 6回前データ
- [7] … 7回前データ
- [8] … 8回前データ
- [9] … 9回前データ

### 2. 1. 4. 計測データ表ワーク

- ① 内容は、計測データ表と同様である。
- ② 本ファイルは、会員登録削除時にテンポラリファイルとして作成され、削除終了時に、計測データ表ファイルとして、リネームされる。

### 3. ファイル構成

#### 3. 1. ヘッダファイル

No.	ファイル名	概要	備考
1	common.h	共通定義ヘッダファイル	共通定数定義, 構造体定義等
2	main.h	メイン制御および共通処理ヘッダファイル	
3	nyuukai.h	入会登録処理ヘッダファイル	
4	keisoku.h	計測記録入力処理ヘッダファイル	
5	sakujyo.h	登録削除処理ヘッダファイル	

#### 3. 2. プログラムファイル

No.	ファイル名	概要	備考
1	main.c	メイン制御関連および共通処理プログラム	共通データ領域も含む
2	nyuukai.c	入会登録処理プログラム	
3	keisoku.c	計測記録入力処理プログラム	
4	sakujyo.c	登録削除処理プログラム	

#### 3. 3. プログラムファイルと関数の対応

##### 3. 3. 1. メイン制御関連および共通処理プログラム (main.c)

・ main	会員管理メイン制御
・ akicode_tbl_create	空きコード表作成
・ codedata_tbl_create	コード・データ対照表作成
・ akicode_tbl_read	空きコード表読み込み (共通プログラム)
・ kakunin_input	実行確認入力 (共通プログラム)
・ kojim_data_read	個人計測データ表読み込み (共通プログラム)
・ kojim_data_disp	個人計測データ表示 (共通プログラム)
・ init_kojim_keisoku_tbl	計測データ初期化 (共通プログラム)

##### 3. 3. 2. 入会登録処理プログラム (nyuukai.c)

・ nyuukai_touroku	入会登録処理
・ akicode_tbl_update	空きコード表更新
・ keisoku_tbl_add	計測データ表追加
・ codedata_tbl_update	コード・データ対照表更新

### 3. 3. 3. 計測記録入力処理プログラム (keisoku.c)

• keisoku_input	計測記録入力処理
• codedata_tbl_read	コード・データ対照表読み込み
• kaiin_code_input	会員コード入力
• keisoku_date_input	計測日付入力
• keisoku_data_input	計測データ入力
• keisoku_data_disp	入力計測データ表示
• undou_sisuu_keisan	運動指数計算
• kojim_data_update	個人計測データ表更新
• keisoku_rank	計測結果順位ソート・表示
• keisoku_data_sort	計測結果順位ソート
• keisoku_sort_disp	計測結果順位表示

### 3. 3. 4. 登録削除処理プログラム (sakujyo.c)

• touroku_sakujyo	登録削除処理
• codedata_tbl_delete	コード・データ対照表削除
• kojim_data_delete	個人計測データ表削除
• akicode_tbl_add	空きコード表追加

## 4. 関数定義書

### 4. 1. アスレチッククラブ会員管理メイン制御

#### 4. 1. 1. 会員管理メイン制御

書式	int main( void )
パラメータ	なし
戻り値	なし
処理概要	<ul style="list-style-type: none"><li>・アスレチッククラブの会員管理処理のメイン制御を行う。</li><li>・初期処理として空きコード表、コード・データ対照表を読み込みモードでOPENした後に、エラーが発生した場合、各ファイルが存在しないとみなして新規に作成する。また、このとき、作成中メッセージを表示する。</li></ul>

#### 4. 1. 2. 空きコード表作成

書式	int akicode_tbl_create( void )
パラメータ	なし
戻り値	OK( 0 ) : 正常終了 NG(-1) : 空きコード表ファイルI/Oエラー
処理概要	<ul style="list-style-type: none"><li>・空きコード表ファイルを書込みモードでOPENする。</li><li>・空きコードテーブルに以下の値をセットする。 空きコードテーブル[0] = 200(空きコード数) 空きコードテーブル[1] = 1 空きコードテーブル[2] = 2 : 空きコードテーブル[200] = 200</li><li>・空きコード表ファイルに空きコードテーブルを書き込む。</li></ul>

#### 4. 1. 3. コード・データ対照表作成

書式	int codedata_tbl_create( void )
パラメータ	なし
戻り値	OK( 0 ) : 正常終了 NG(-1) : コード・コード対照表ファイルI/Oエラー
処理概要	<ul style="list-style-type: none"><li>・コード・データ対照表ファイルを書込みモードでOPENする。</li><li>・コード・データ対照表テーブルをNULLクリアする。</li><li>・コード・データ対照表ファイルに書き込む。</li></ul>

#### 4. 1. 4. 計測データ初期化 (共通)

書式	struct KEISOKU_TBL init_kojin_keisoku_tbl( void )
パラメータ	なし
戻り値	初期化された状態の計測データ
処理概要	初期化された状態の計測データを返す。

#### 4. 2. 入会登録処理

##### 4. 2. 1. 入会登録

書式	int nyuukai_touroku( void )
パラメータ	なし
戻り値	OK( 0 ) : 正常終了 NG(-1) : 入会登録処理異常終了
処理概要	<ul style="list-style-type: none"><li>・ 空きコード表ファイルを参照し、会員の空きコードが存在する場合、入会登録処理を行う。空きコードがない場合、残念メッセージを表示してリターンする。</li><li>・ 入会登録処理<ol style="list-style-type: none"><li>1. 空きコード表から登録した会員コードを削除する。</li><li>2. 計測データ表にレコードを追加する。</li><li>3. コード・データ対照表に、追加した計測データ表のレコード位置をセットし、更新する。</li></ol></li></ul>

##### 4. 2. 2. 空きコード表読み込み (共通)

書式	int akicode_tbl_read( void )
パラメータ	なし
戻り値	OK( 0 ) : 正常終了 NG(-1) : 空きコード表ファイルI/Oエラー
処理概要	<ul style="list-style-type: none"><li>・ 空きコード表ファイルを読み込みモードでOPENする。</li><li>・ 空きコード表ファイルをEOFになるまで空きコード表に読み込む。</li></ul>

##### 4. 2. 3. 実行確認入力 (共通)

書式	int kakunin_input( char *msg )
パラメータ	char *msg : 確認メッセージ内容
戻り値	OK( 0 ) : Yes 入力 ... 処理実行 NG(-1) : No 入力 ... 処理中止
処理概要	<ul style="list-style-type: none"><li>・ 渡された確認メッセージを表示し、Y/Nの入力を行う。</li></ul>

##### 4. 2. 4. 空きコード表更新

書式	int akicode_tbl_update( void )
パラメータ	なし
戻り値	OK( 0 ) : 正常終了 NG(-1) : 空きコード表ファイルI/Oエラー
処理概要	<ul style="list-style-type: none"><li>・ 空きコード表ファイルを書込みモードでOPENする。 (ファイルが存在する場合、内容を破棄する。)</li><li>・ 空きコード表から登録会員コードを削除(編集)し、空きコード表ファイルに書き込む。</li><li>・ 空きコード表編集<ol style="list-style-type: none"><li>1. 2レコード目のデータを削除し、データを詰める。</li><li>2. 1レコード目の空きコード数をデクリメントする。</li></ol></li></ul>

#### 4. 2. 5. 計測データ表追加

書式	int    keisoku_tbl_add( long *fptr, int kaiin_code )
パラメータ	long   *fptr         : 計測データ表の追加書き込みを行ったファイル位置(出力) int    kaiin_code   : 登録会員コード
戻り値	OK( 0)   : 正常終了 NG(-1)   : 計測データ表ファイルI/Oエラー
処理概要	<ul style="list-style-type: none"> <li>・計測データ表ファイルを追加モードでOPENする。</li> <li>・計測データ表ファイルの最後に1レコードを追加する。</li> <li>・追加したファイルポインタを求め、fptrにセットする。</li> </ul>

#### 4. 2. 6. コード・データ対照表更新

書式	int    codedata_tbl_update( int kaiin_code, long fptr )
パラメータ	int    kaiin_code   : 登録会員コード long   fptr         : 計測データ表の追加書き込みを行ったファイル位置
戻り値	OK( 0)   : 正常終了 NG(-1)   : コード・データ対照表ファイルI/Oエラー
処理概要	<ul style="list-style-type: none"> <li>・コード・データ対照表ファイルを読み書きモードでOPENする。</li> <li>・コード・データ対照表テーブルに読み込む。</li> <li>・登録する会員コードをインデックスとして、fptrより計測データのレコード番号を求め、対応するコード・データ対照表にセットする。 レコード番号 = (fptr / 計測データ表レコード長) + 1 fptrが0から始まるため1を加算する。</li> <li>・セット終了後、ファイルポインタを先頭に戻し、ファイルに書き込む。</li> </ul>

#### 4. 3. 計測記録入力処理

##### 4. 3. 1. 計測記録入力

書式	int keisoku_input( void )
パラメータ	なし
戻り値	OK( 0 ) : 正常終了 NG(-1) : 計測記録入力処理異常終了
処理概要	<ul style="list-style-type: none"><li>・計測記録入力処理のメイン制御を行う。</li><li>・空きコード表の読み込みを行い、登録会員の存在チェックを行う。</li><li>・コード・データ対照表を読み込む。</li><li>・計測情報データを入力し運動指数を求め、個人情報及び計測結果の上位10人までのデータを表示する。</li></ul>

##### 4. 3. 2. コード・データ対照表読み込み

書式	int codedata_tbl_read( void )
パラメータ	なし
戻り値	OK( 0 ) : 正常終了 NG(-1) : コード・データ対照表ファイルI/Oエラー
処理概要	<ul style="list-style-type: none"><li>・コード・データ対照表ファイルを読み込みモードでOPENする。</li><li>・コード・データ対照表の読み込みを行う。</li></ul>

##### 4. 3. 3. 会員コード入力

書式	void kaiin_code_input( int *kaiin_code )
パラメータ	int *kaiin_code : 入力会員コード(出力)
戻り値	なし
処理概要	<ul style="list-style-type: none"><li>・会員コードの入力を行い、以下のチェックを行う。<ol style="list-style-type: none"><li>1. ニューメリック・チェック</li><li>2. 範囲チェック (0&lt;会員コード≤200)</li><li>3. 会員コード登録チェック</li></ol></li></ul>

##### 4. 3. 4. 計測日付入力

書式	void keisoku_date_input( char *keisoku_date )
パラメータ	char *keisoku_date : 入力計測日付(出力)
戻り値	なし
処理概要	<ul style="list-style-type: none"><li>・計測日付の入力を行い、以下のチェックを行う。<ol style="list-style-type: none"><li>1. 入力桁数チェック (YYYYMMDD : 8桁)</li><li>2. ニューメリック・チェック</li><li>3. 月範囲チェック (1≤月≤12)</li><li>4. 日範囲チェック (1≤日≤31)</li></ol></li></ul>

#### 4. 3. 5. 計測データ入力

書式	void keisoku_data_input( int idx )
パラメータ	int idx : 運動種別インデックス
戻り値	なし
処理概要	<ul style="list-style-type: none"> <li>・ 負荷, セット数, 回数を入力を行い, 以下のチェックを行う。             <ol style="list-style-type: none"> <li>1. 入力桁数チェック (3桁より大きい場合, エラーとする。)</li> <li>2. ニューメリック・チェック</li> <li>3. 上限値範囲チェック (1 ≤ 入力値 ≤ 100) (ただし, セット数・回数のときのみ)</li> </ol> </li> <li>・ 入力データをテーブルにセットする。</li> </ul>

#### 4. 3. 6. 入力計測データ表示

書式	void keisoku_data_disp( int kaiin_code, char *keisoku_date )
パラメータ	int kaiin_code : 会員コード char *keisoku_date : 計測日付
戻り値	なし
処理概要	<ul style="list-style-type: none"> <li>・ 入力された計測データの表示を行う。</li> </ul>

#### 4. 3. 7. 運動指数計算

書式	void undou_sisuu_keisan( int *undou_sisuu )
パラメータ	int *undou_sisuu : 運動指数算出結果
戻り値	なし
処理概要	<ul style="list-style-type: none"> <li>・ 入力された計測データをもとに運動指数の計算を行う。</li> <li>・ 計算式を以下に示す。( <math>N_i \leq 1</math> の場合, 計算対象外とする。)</li> </ul> $\sqrt{\sum_{i=1}^5 F_i * W_i * S_i * (N_i^2 / (N_i - 1))}$ <p> <math>F_i</math> : 負荷係数 ( 0.24, 0.36, 0.52, 1.05, 2.13 )  <math>W_i</math> : 入力負荷 ( MAX 999 )  <math>S_i</math> : セット数 ( MAX 100 )  <math>N_i</math> : 運動回数 ( MAX 100 )         </p>

#### 4. 3. 8. 個人計測データ表更新

書式	int kojinn_data_update( int kaiin_code, char *keisoku_date, int undou_sisuu )
パラメータ	int kaiin_code : 会員コード char *keisoku_date : 計測日付 int undou_sisuu : 運動指数
戻り値	OK( 0 ) : 正常終了 NG(-1) : 計測データ表ファイルI/Oエラー
処理概要	<ul style="list-style-type: none"> <li>・計測データ表に該当データをセットする。</li> <li>・1回目 : 初回データ, 最高記録データ, 最新データにセットする。</li> <li>・2回目以降 : 最高記録の場合, 最高記録データにセットする。 履歴データを移動し, 最新データをセットする。</li> <li>・計測データ表ファイルを読書きモードでOPENする。</li> <li>・ファイルポインタを該当レコード位置にシークする。</li> <li>・計測データ表ファイルの書き込みを行う。</li> </ul>

#### 4. 3. 9. 計測結果順位ソート・表示

書式	int keisoku_rank( void )
パラメータ	なし
戻り値	OK( 0 ) : 正常終了 NG(-1) : 計測データ表ファイルI/Oエラー
処理概要	<ul style="list-style-type: none"> <li>・計測データ表を読込みモードでOPENする。</li> <li>・計測データ表のすべてのレコードを読み込み, ソート用テーブルにセットする。</li> <li>・計測データが存在する会員の件数を求める。(計測回数が0以上)</li> <li>・セットしたテーブルを最高記録データでソートし, 上位10人までの結果を表示する。</li> </ul>

#### 4. 3. 10. 計測結果順位ソート

書式	void keisoku_data_sort( int cnt )
パラメータ	int cnt : 計測データ件数
戻り値	なし
処理概要	<ul style="list-style-type: none"> <li>・計測データのソート用テーブルを計測データ件数分, 最高記録データでソートする。</li> </ul>

#### 4. 3. 11. 計測結果順位表示

書式	void keisoku_sort_disp( int cnt )
パラメータ	int cnt : 計測データ件数
戻り値	なし
処理概要	<ul style="list-style-type: none"> <li>・ソートされた計測データのソート用テーブルの上位10人までの会員コード, 最高記録データを表示する。</li> <li>・10人に満たない場合は, 計測データ件数分表示する。</li> </ul>

#### 4. 3. 1 2. 個人計測データ表読み込み (共通)

書式	int kojinn_data_read( int kaiin_code )
パラメータ	int kaiin_code : 会員コード
戻り値	OK( 0 ) : 正常終了 NG(-1) : 計測データ表ファイルI/Oエラー
処理概要	<ul style="list-style-type: none"> <li>・計測データ表ファイルを読み込みモードでOPENする。</li> <li>・会員のデータ位置までシークする。 データ位置 = (コード・データ対照表[会員コード - 1] - 1) × 計測データ表レコード長</li> <li>・計測データ表の読み込みを行う。</li> </ul>

#### 4. 3. 1 3. 個人計測データ表示 (共通)

書式	void kojinn_data_disp( int kaiin_code, char *msg )
パラメータ	int kaiin_code : 表示会員コード char *msg : 表示メッセージ
戻り値	なし
処理概要	<ul style="list-style-type: none"> <li>・計測データ表の個人データを表示する。</li> <li>・表示項目 <ol style="list-style-type: none"> <li>1. 会員コード</li> <li>2. 計測回数</li> <li>3. 初回日付</li> <li>4. 初回データ</li> <li>5. 最高記録日付</li> <li>6. 最高記録データ</li> <li>7. 最新計測日付</li> <li>8. 最新計測データ</li> <li>9. 計測データ(1回前~9回前)</li> </ol> </li> </ul>

#### 4. 4. 登録削除処理

##### 4. 4. 1. 登録削除

書式	int touroku_sakujyo( void )
パラメータ	なし
戻り値	OK( 0) : 正常終了 NG(-1) : 登録削除処理 異常終了
処理概要	<ul style="list-style-type: none"> <li>・退会する会員コードを入力し、該当する会員コードのデータを更新する。</li> <li>・登録削除処理             <ol style="list-style-type: none"> <li>1. コード・データ対照表に、削除する計測データ表のレコード位置を0にクリアする。</li> <li>2. 計測データ表から該当レコードを削除する。</li> <li>3. 空きコード表に退会する会員コードを追加する。</li> </ol> </li> </ul>

##### 4. 4. 2. コード・データ対照表削除

書式	int codedata_tbl_delete( int kaiin_code )
パラメータ	int kaiin_code : 退会会員コード
戻り値	OK( 0) : 正常終了 CANCEL( 1) : 処理中止 NG(-1) : コード・データ対照表ファイルI/Oエラー 計測データ表ファイルI/Oエラー
処理概要	<ul style="list-style-type: none"> <li>・コード・データ対照表ファイルを読書きモードでOPENする。</li> <li>・コード・データ対照表ファイルを読み込み、退会する会員の計測データ表レコード位置を取得する。</li> <li>・取得したレコード位置で計測データ表を読み込み、退会者データを表示する。(kojin_data_read, kojिन_data_disp を呼び出す。)</li> <li>・表示したデータを削除してよいかの確認後、計測データ表から該当レコードを削除する。(kakunin_input, kojिन_data_delete を呼び出す。)</li> <li>・削除が正常に終了したら、コード・データ対照表の該当位置に0をセットして、更新する。</li> </ul>

##### 4. 4. 3. 空きコード表追加

書式	int akicode_tbl_add( int kaiin_code )
パラメータ	int kaiin_code : 退会会員コード
戻り値	OK( 0) : 正常終了 NG(-1) : 空きコード表ファイルI/Oエラー
処理概要	<ul style="list-style-type: none"> <li>・空きコード表の読み込みを行う。(akicode_tbl_read を呼び出す。)</li> <li>・空きコード表ファイルを書込みモードでOPENする。(ファイルが存在する場合、内容を破棄する。)</li> <li>・空きコード表に退会会員コードを追加(編集)し、空きコード表ファイルに書き込む。</li> <li>・空きコード表編集             <ol style="list-style-type: none"> <li>1. ファイルの最後に退会会員コードを追加する。</li> <li>2. 1レコード目の空きコード数をインクリメントする。</li> </ol> </li> </ul>

#### 4. 4. 4. 個人計測データ表削除

書式	int kojindata_delete ( int kaiin_code )
パラメータ	int kaiin_code : 退会会員コード
戻り値	OK( 0) : 正常終了 NG(-1) : 計測データ表ファイルI/Oエラー
処理概要	<ul style="list-style-type: none"> <li>・計測データ表ファイルを読込みモードでOPENする。</li> <li>・ワークファイルを書込みモードでOPENする。</li> <li>・計測データ表をEOFになるまで読み込み, 退会会員のデータ以外をワークファイルに書き込む。</li> <li>・各ファイルをCLOSEし, 計測データ表ファイルを削除する。</li> <li>・削除後に, ワークファイルを計測データ表ファイルにリネームする。</li> </ul>

## アスレチッククラブ会員管理プログラム ソースプログラムリスト

```

/*****/
/* common.h */
/* 共通ヘッダファイル */
/*****/

#define MEMBER_MAX 200 /* メンバー数 MAX */
#define AKICODE_TBL_NAME "akicode.tbl" /* 空きコード表ファイル名 */
#define CODEDATA_TBL_NAME "codedata.tbl" /* コード・データ対照表ファイル名 */
#define KEISOKU_TBL_NAME "keisoku.tbl" /* 計測データ表ファイル名 */

#define TRUE 1 /* 真 */
#define FALSE 0 /* 偽 */
#define OK 0 /* 正常 */
#define CANCEL 1 /* 処理中止 */
#define NG -1 /* 異常 */

/* 計測データテーブル */
struct KEISOKU_TBL {
    int kaiin_code; /* 会員コード */
    int count; /* 計測回数 */
    char first_date[ 9 ]; /* 初回日付 */
    int first_data; /* 初回データ */
    char max_date[ 9 ]; /* 最高記録日付 */
    int max_data; /* 最高記録データ */
    char soku_date[ 9 ]; /* 最新計測日付 */
    int soku_data[ 10 ]; /* 計測データ */
};

/* 入力計測データ */
struct KEISOKU_INPUT {
    int huka; /* 負荷 */
    int set; /* セット */
    int kaisuu; /* 回数 */
};

.....

/*****/
/* main.h */
/* アスレチッククラブ会員管理ヘッダファイル */
/*****/

static int codedata_tbl_create( void );
static int akicode_tbl_create( void );

int akicode_tbl_read( void );
int kakunin_input( char *msg );
int kojim_data_read( int kaiin_code );
void kojim_data_disp( int kaiin_code, char *msg );
struct KEISOKU_TBL init_kojim_keisoku_tbl( void );

```

```
/*
 * nyuukai.h
 * 入会登録処理ヘッダファイル
 */

int nyuukai_touroku( void );

static int akicode_tbl_update( void );
static int keisoku_tbl_add( long *fptr, int kaiin_code );
static int codedata_tbl_update( int kaiin_code, long fptr );
```

---

```
/*
 * keisoku.h
 * 計測記録入力処理ヘッダファイル
 */

int keisoku_input( void );

static int codedata_tbl_read( void );
static void kaiin_code_input( int *kaiin_code );
static void keisoku_date_input( char *keisoku_date );
static void keisoku_data_input( int idx );
static void keisoku_data_disp( int kaiin_code, char *keisoku_date );
static void undou_sisuu_keisan( int *undou_sisuu );
static int kojim_data_update( int kaiin_code, char *keisoku_date, int undou_sisuu );
static int keisoku_rank( void );
static void keisoku_data_sort( int cnt );
static void keisoku_sort_disp( int cnt );
```

---

```
/*
 * sakujoyo.h
 * 登録削除処理ヘッダファイル
 */

int touroku_sakujoyo( void );

static int codedata_tbl_delete( int kaiin_code );
static int kojim_data_delete( int kaiin_code );
static int akicode_tbl_add( int kaiin_code );
```

```

/*****
/* main.c */
/* アスレチッククラブ会員管理プログラム */
/*****
#include <stdio.h>
#include <string.h>

#include "common.h"
#include "main.h"
#include "nyuukai.h"
#include "keisoku.h"
#include "sakujoyo.h"

/*****
/* 共通データ */
/*****

/* 空きコード表 */
int akicode_tbl[ MEMBER_MAX + 1 ];

/* コード・データ対照表 */
int codedata_tbl[ MEMBER_MAX ];

/* 個人別計測データ表 */
struct KEISOKU_TBL kojिन_keisoku_tbl;

/*ソート用計測データ表 */
struct KEISOKU_TBL sort_keisoku_tbl[ MEMBER_MAX ];

/*****
/* 会員管理メイン制御 */
/* メインルーチン */
/* */
/* パラメータ : なし */
/* リターン : なし */
/*****
int main( void )
{
    int loop = TRUE; /* ループフラグ */
    char work[ 128 ]; /* 入力ワーク */
    FILE *fp; /* ファイルポインタ */
    char *fname1 = AKICODE_TBL_NAME; /* 空きコード表ファイル */
    char *fname2 = CODEDATA_TBL_NAME; /* コード・データ対照表ファイル */
    int i; /* インデックス */

    /* 空きコード表ファイル OPEN -> NULL ? */
    if( (fp = fopen( fname1, "rb" )) == NULL ) {
        printf( "%n 空きコード表ファイルを作成しています" );

        /* 空きコード表ファイル作成 */
        akicode_tbl_create( );
    }
    else {
        /* 空きコード表ファイル CLOSE */
        fclose( fp );
    }

    /* コード・データ対照表ファイル OPEN -> NULL ? */
    if( (fp = fopen( fname2, "rb" )) == NULL ) {
        printf( "%n コード・データ対照表ファイルを作成しています" );

        /* コード・データ対照表ファイル作成 */
        codedata_tbl_create( );
    }
    else {
        /* コード・データ対照表ファイル CLOSE */
        fclose( fp );
    }
}

```

```

while( loop ) {
    /* テーブル初期クリア */
    akicode_tbl[ 0 ] = 0;
    for( i = 0; i < MEMBER_MAX; i++) {
        akicode_tbl[ i + 1 ] = 0;
        codedata_tbl[ i ] = 0;
        sort_keisoku_tbl[ i ] = init_kojin_keisoku_tbl();
    }
    kojin_keisoku_tbl = init_kojin_keisoku_tbl();

    printf( "\n" );
    printf( "\n *****" );
    printf( "\n アスレチッククラブ メンバー管理プログラム" );
    printf( "\n *****" );
    printf( "\n 処理を選択してください" );
    printf( "\n 1:入会登録" );
    printf( "\n 2:計測記録入力" );
    printf( "\n 3:登録削除" );
    printf( "\n E:終了" );
    printf( "\n ? " );

    /* 処理区分入力 */
    work[ 0 ] = '\0';
    scanf( "%s", work );

    /* 入力桁数チェック -> 1以外 ? */
    if( strlen( work ) != 1 ) {
        printf( "\n 入力ミスです" );
        continue;
    }

    switch( work[ 0 ] ) {
        case '1': /* 入会登録 */
            nyuukai_touroku( );
            break;

        case '2': /* 計測記録入力 */
            keisoku_input( );
            break;

        case '3': /* 登録削除 */
            touroku_sakujyo( );
            break;

        case 'e': /* 終了 */
        case 'E':
            loop = FALSE;
            break;

        default:
            printf( "\n 入力ミスです" );
            break;
    }
}
return OK;
}

```

```

/*****/
/* 会員管理メイン制御 */
/* 空きコード表 作成処理 */
/* */
/* パラメータ : なし */
/* リターン : 0:OK */
/* -1:NG */
/*****/
static int akicode_tbl_create( void )
{
    int ret; /* リターンコード */
    int i; /* インデックス */
}

```

```

FILE    *fp;                               /* ファイルポインタ    */
char    *fname = AKICODE_TBL_NAME;        /* 空きコード表ファイル */

/* 空きコード編集 */
akicode_tbl[ 0 ] = MEMBER_MAX;
for( i = 1; i < MEMBER_MAX + 1; i++ ) {
    akicode_tbl[ i ] = i;
}

/* 空きコード表ファイル OPEN -> NULL ? */
if( (fp = fopen( fname, "wb" )) == NULL ) {
    printf( "%n 空きコード表ファイル OPEN エラー" );
    return NG;
}

/* 空きコード表ファイル WRITE -> 1以外 ? */
if( (ret = fwrite( (char *)akicode_tbl, sizeof( akicode_tbl ), 1, fp ) )
    != 1 ) {
    printf( "%n 空きコード表ファイル WRITE エラー" );
    ret = NG;
}
else {
    ret = OK;
}

/* 空きコード表ファイル CLOSE */
fclose( fp );

return ret;
}

/*****
/* 会員管理メイン制御          */
/*   コード・データ対照表   作成処理      */
/*                               */
/*   パラメータ   : なし          */
/*   リターン     : 0:OK          */
/*               : -1:NG         */
*****/
static int codedata_tbl_create( void )
{
    int    ret;                               /* リターンコード      */
    FILE   *fp;                               /* ファイルポインタ    */
    char   *fname = CODEDATA_TBL_NAME;        /* コード・データ対照表ファイル */
    int    i;                                 /* インデックス        */

    /* コード・データ対照表ファイル OPEN -> NULL ? */
    if( (fp = fopen( fname, "wb" )) == NULL ) {
        printf( "%n コード・データ対照表ファイル OPEN エラー" );
        return NG;
    }

    for( i = 0; i < MEMBER_MAX; i++ )
        codedata_tbl[ i ] = 0;

    /* コード・データ対照表ファイル WRITE -> 1以外 ? */
    if( (ret = fwrite( (char *)codedata_tbl, sizeof( codedata_tbl ), 1, fp ) )
        != 1 ) {
        /* WRITE エラー */
        printf( "%n コード・データ対照表ファイル WRITE エラー" );
        ret = NG;
    }
    else {
        ret = OK;
    }

    /* コード・データ対照表ファイル CLOSE */
    fclose( fp );
}

```

```

    return ret;
}

/*****
/* 共通ルーチン */
*****/

/*****
/* 共通ルーチン */
/* 空きコード表 読み込み処理 */
/* */
/* パラメータ : なし */
/* リターン : 0:OK */
/* -1:NG */
*****/
int akicode_tbl_read( void )
{
    int ret; /* リターンコード */
    int i; /* インデックス */
    FILE *fp; /* ファイルポインタ */
    char *fname = AKICODE_TBL_NAME; /* 空きコード表ファイル */

    /* 空きコード表ファイル OPEN -> NULL ? */
    if( (fp = fopen( fname, "rb" )) == NULL ) {
        printf( "\n 空きコード表ファイル OPEN エラー" );
        return NG;
    }

    for( i = 0; i < MEMBER_MAX + 1; i++ ) {
        /* 空きコード表ファイル READ -> 1以外 ? */
        if( (ret = fread( (char *)&akicode_tbl[ i ], sizeof( int ), 1, fp ))
            != 1 ) {
            /* READ エラーあり ? */
            if( ferror( fp ) != 0 ) {
                printf( "\n 空きコード表ファイル READ エラー" );
                ret = NG;
            }
            else {
                /* ファイル EOF でない? */
                if( feof( fp ) == 0 ) {
                    printf( "\n 空きコード表ファイル READ エラー" );
                    ret = NG;
                }
                else {
                    ret = OK;
                }
            }
        }
        break;
    }
}

/* 空きコード表ファイル CLOSE */
fclose( fp );

return ret;
}

/*****
/* 共通ルーチン */
/* 実行確認入力処理 */
/* */
/* パラメータ : 確認メッセージ */
/* リターン : 0:OK */
/* -1:NG */
*****/
int kakunin_input( char *msg )

```

```

{
  int    ret;                /* リターンコード */
  int    loop = TRUE;       /* ループフラグ */
  char   work[ 128 ];       /* 入力ワーク */

  while( loop ) {
    /* 確認表示 */
    printf( msg );
    printf( "¥n ? " );

    /* Y/N入力 */
    work[ 0 ] = '¥0';
    scanf( "%s", work );

    /* 入力桁数チェック -> 1以外 ? */
    if( strlen( work ) != 1 ) {
      printf( "¥n 入力ミスです" );
      continue;
    }

    switch( work[ 0 ] ) {
      case 'Y': /* Yes */
      case 'y':
        ret = OK;
        loop = FALSE;
        break;

      case 'N': /* No */
      case 'n':
        ret = NG;
        loop = FALSE;
        break;

      default:
        printf( "¥n 入力ミスです" );
        break;
    }
  }

  return ret;
}

/*****
/* 共通ルーチン */
/* 個人計測データ表 読み込み処理 */
/* */
/* パラメータ : 会員コード */
/* リターン : 0:OK */
/* -1:NG */
*****/
int kojिन_data_read( int kaiin_code )
{
  int    ret;                /* リターンコード */
  FILE   *fp;               /* 計測データ表ファイルポインタ */
  long   fpptr;             /* 計測データポインタ */
  char   *fname = KEISOKU_TBL_NAME; /* 計測データ表ファイル */

  /* 計測データ表ファイル OPEN -> NULL ? */
  if( (fp = fopen( fname, "rb" )) == NULL ) {
    printf( "¥n 計測データ表ファイル OPEN エラー" );
    return NG;
  }

  /* 該当データポインタセット */
  fpptr = ( codedata_tbl[ kaiin_code - 1 ] - 1 ) *
          sizeof( struct KEISOKU_TBL );

  /* 計測データ表ファイルを対象の位置まで SEEK -> OK ? */
  if( (ret = fseek( fp, fpptr, SEEK_SET )) != OK ) {

```

```

printf( "¥n 計測データ表ファイル SEEK エラー" );

/* 計測データ表ファイル CLOSE */
fclose( fp );
return NG;
}

/* 計測データ表ファイル READ -> 1以外 ? */
if( (ret = fread( (char *)&kojin_keisoku_tbl, sizeof( koin_keisoku_tbl ),
                1, fp )) != 1 ) {
    printf( "¥n 計測データ表 READ エラー" );
    ret = NG;
}
else {
    ret = OK;
}

/* 計測データ表ファイル CLOSE */
fclose( fp );

return ret;
}

```

```

/*****
/* 共通ルーチン */
/* 個人計測データ 表示処理 */
/* */
/* パラメータ : 会員コード */
/* 表示メッセージ */
/* リターン : なし */
*****/
void koin_data_disp( int kaiin_code, char *msg )
{
    printf( msg );
    printf( "¥n 会員コード %3d", kaiin_code );

    if( koin_keisoku_tbl.count != 0 ) {
        printf( "¥n¥n 計測回数 初回日付 データ " );
        printf( " 最高日付 データ 最新日付 データ" );

        printf( "¥n %3d", koin_keisoku_tbl.count );

        printf( " %4.4s-%2.2s-%2.2s",
            &koin_keisoku_tbl.first_date[ 0 ],
            &koin_keisoku_tbl.first_date[ 4 ],
            &koin_keisoku_tbl.first_date[ 6 ] );

        printf( " %4d", koin_keisoku_tbl.first_data );

        printf( " %4.4s-%2.2s-%2.2s",
            &koin_keisoku_tbl.max_date[ 0 ],
            &koin_keisoku_tbl.max_date[ 4 ],
            &koin_keisoku_tbl.max_date[ 6 ] );

        printf( " %4d", koin_keisoku_tbl.max_data );

        printf( " %4.4s-%2.2s-%2.2s",
            &koin_keisoku_tbl.soku_date[ 0 ],
            &koin_keisoku_tbl.soku_date[ 4 ],
            &koin_keisoku_tbl.soku_date[ 6 ] );

        printf( " %4d", koin_keisoku_tbl.soku_data[ 0 ] );

        printf( "¥n¥n 1回前 2回前 3回前 4回前 " );
        printf( "5回前 6回前 7回前 8回前 9回前" );

        printf( "¥n " );
        printf( " %4d", koin_keisoku_tbl.soku_data[ 1 ] );
        printf( " %4d", koin_keisoku_tbl.soku_data[ 2 ] );
    }
}

```

```

        printf( "    %4d", kojim_keisoku_tbl.soku_data[ 3 ] );
        printf( "    %4d", kojim_keisoku_tbl.soku_data[ 4 ] );
        printf( "    %4d", kojim_keisoku_tbl.soku_data[ 5 ] );
        printf( "    %4d", kojim_keisoku_tbl.soku_data[ 6 ] );
        printf( "    %4d", kojim_keisoku_tbl.soku_data[ 7 ] );
        printf( "    %4d", kojim_keisoku_tbl.soku_data[ 8 ] );
        printf( "    %4d", kojim_keisoku_tbl.soku_data[ 9 ] );
    }
    else {
        printf( "    計測データがありません" );
        return;
    }
    return;
}

```

```

/*****
/* 共通ルーチン
/* 計測データ 初期化処理
/*
/* パラメータ : なし
/* リターン : 計測データ
*****/
struct KEISOKU_TBL init_kojim_keisoku_tbl( void )
{
    static struct KEISOKU_TBL tbl = {
        0, 0, " ", 0, " ", 0, " ", 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
    };
    return tbl;
}

```

```

/*****
/* nyuukai.c */
/* 入会登録処理プログラム */
/*****
#include <stdio.h>
#include <string.h>

#include "common.h"
#include "main.h"
#include "nyuukai.h"

extern int akicode_tbl[ MEMBER_MAX + 1 ]; /* 空きコード表 */
extern int codedata_tbl[ MEMBER_MAX ]; /* コード・データ対照表 */
extern struct KEISOKU_TBL kojcin_keisoku_tbl; /* 個人別計測データ表 */

/*****
/* 入会登録処理 */
/* メインルーチン */
/* パラメータ : なし */
/* リターン : 0:OK */
/* -1:NG */
/*****
int nyuukai_touroku( void )
{
    int ret; /* リターンコード */
    int kaiin_code; /* 登録会員コード */
    long fptr; /* ファイルポインタ */
    char msg[ 64 ]; /* メッセージエリア */

    /* 空きコード表 READ -> NG ? */
    if( (ret = akicode_tbl_read( )) == NG ) {
        return ret;
    }

    /* 空きコードあり ? */
    if( akicode_tbl[ 0 ] <= 0 ) {
        printf( "%n 残念ながらただ今メンバーの空きがありません%n" );
        ret = OK;
        return ret;
    }

    /* 空きコード確認 */
    sprintf( msg, "%n 会員コードは %d です。よろしいですか( Y/N )", akicode_tbl[ 1 ] );

    if( (ret = kakunin_input( msg )) == OK ) {
        /* 登録会員コード下退避 */
        kaiin_code = akicode_tbl[ 1 ];

        /* 空きコード表更新 -> OK ? */
        if( (ret = akicode_tbl_update( )) == OK ) {

            /* 計測データ表追加 -> OK ? */
            if( (ret = keisoku_tbl_add( &fptr, kaiin_code )) == OK ) {

                /* コード・データ対照表更新 */
                ret = codedata_tbl_update( kaiin_code, fptr );
            }
        }
    }

    if( ret == OK ) {
        printf( "%n 入会登録処理が終了しました" );
    }

    return ret;
}

```

```

/*****
/* 入会登録処理                                     */
/* 空きコード表 更新処理                             */
/*                                                     */
/* パラメータ : なし                                 */
/* リターン   : 0:OK                                  */
/*           -1:NG                                    */
/*****
static int akicode_tbl_update( void )
{
    int    ret;                /* リターンコード          */
    int    i;                  /* インデックス           */
    int    cnt;                /* 空きコード件数         */
    FILE   *fp;                /* ファイルポインタ       */
    char   *fname = AKICODE_TBL_NAME; /* 空きコード表ファイル */

    /* 空きコード件数セット */
    cnt = akicode_tbl[ 0 ];

    /* 空きコード編集 */
    for( i = 1; i < cnt; i++ ) {
        if( akicode_tbl[ i + 1 ] == 0 ) {
            break;
        }
        akicode_tbl[ i ] = akicode_tbl[ i + 1 ];
    }

    akicode_tbl[ i ] = 0;

    /* 空きコード件数セット */
    akicode_tbl[ 0 ] = cnt - 1;

    /* 空きコード表ファイル OPEN -> NULL ? */
    if( (fp = fopen( fname, "wb" )) == NULL ) {
        printf( "%n 空きコード表ファイル OPEN エラー" );
        return NG;
    }

    /* 空きコード表ファイル WRITE -> 1以外 ? */
    if( (ret = fwrite( (char *)akicode_tbl,
        sizeof( int ) * (akicode_tbl[ 0 ] + 1), 1, fp )) != 1 ) {
        printf( "%n 空きコード表ファイル WRITE エラー" );
        ret = NG;
    }
    else {
        ret = OK;
    }

    /* 空きコード表ファイル CLOSE */
    fclose( fp );

    return ret;
}

/*****
/* 入会登録処理                                     */
/* 計測データ表 追加処理                             */
/*                                                     */
/* パラメータ : 計測データポインタ                 */
/*           登録会員コード                         */
/* リターン   : 0:OK                                  */
/*           -1:NG                                    */
/*****
static int keisoku_tbl_add( long *fptr, int kaiin_code )
{
    int    ret;                /* リターンコード          */
    FILE   *fp;                /* ファイルポインタ       */
    char   *fname = KEISOKU_TBL_NAME; /* 計測データ表ファイル */

```

```

/* 計測データ表ファイル OPEN -> NULL ? */
if( (fp = fopen( fname, "a+b" )) == NULL ) {
    printf( "\n 計測データ表ファイル OPEN エラー" );
    return NG;
}

/* 計測データ表ファイル SEEK -> OK でない ? */
if( (ret = fseek( fp, 0L, SEEK_END )) != OK ) {
    printf( "\n 計測データ表ファイル SEEK エラー" );
    /* 計測データ表ファイル CLOSE */
    fclose( fp );
    return NG;
}

/* ファイルポインタ取得 */
*fptr = ftell( fp );

/* 計測データ表クリア */
kojin_keisoku_tbl = init_kojin_keisoku_tbl();

/* 会員コードセット */
kojin_keisoku_tbl.kaiin_code = kaiin_code;

/* 計測データ表ファイル WRITE -> 1以外 ? */
if( (ret = fwrite( (char *)&kojin_keisoku_tbl, sizeof( koin_keisoku_tbl ),
    1, fp )) != 1 ) {
    printf( "\n 計測データ表ファイル WRITE エラー" );
    ret = NG;
}
else {
    ret = OK;
}

/* 計測データ表ファイル CLOSE */
fclose( fp );

return ret;
}

```

```

/*****
/* 入会登録処理 */
/* コード・データ対照表 更新処理 */
/*
/* パラメータ : 登録会員コード */
/* 計測データポインタ */
/* リターン : 0:OK */
/* -1:NG */
*****/
static int codedata_tbl_update( int kaiin_code, long fptr )
{
    int ret; /* リターンコード */
    FILE *fp; /* ファイルポインタ */
    char *fname = CODEDATA_TBL_NAME; /* コード・データ対照表ファイル */

    /* コード・データ対照表ファイル OPEN -> NULL ? */
    if( (fp = fopen( fname, "r+b" )) == NULL ) {
        printf( "\n コード・データ対照表ファイル OPEN エラー" );
        return NG;
    }

    /* コード・データ対照表ファイル READ -> 1以外 ? */
    if( (ret = fread( (char *)codedata_tbl, sizeof( codedata_tbl ), 1, fp ))
        != 1 ) {
        printf( "\n コード・データ対照表ファイル READ エラー" );
        ret = NG;
    }
    else {
        /* 該当データポインタセット */
        codedata_tbl[ kaiin_code - 1 ] =

```

```

    (int)( (fptr / sizeof( struct KEISOKU_TBL )) + 1 );
/* ファイルポインタを先頭に SEEK -> OK でない ? */
if( (ret = fseek( fp, 0L, SEEK_SET )) != OK ) {
    printf( "%n コード・データ対照表ファイル SEEK エラー" );

    /* コード・データ対照表ファイル CLOSE */
    fclose( fp );
    return NG;
}

/* コード・データ対照表ファイル WRITE -> 1以外 ? */
if( (ret = fwrite( (char *)codedata_tbl, sizeof( codedata_tbl ), 1,
    fp )) != 1 ) {
    printf( "%n コード・データ対照表ファイル WRITE エラー" );
    ret = NG;
}
else {
    ret = OK;
}
}

/* コード・データ対照表ファイル CLOSE */
fclose( fp );
return ret;
}

```

```

/*****
/* keisoku.c */
/* 計測記録入力処理プログラム */
/*****
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

#include "common.h"
#include "main.h"
#include "keisoku.h"

extern int akicode_tbl[ MEMBER_MAX + 1 ]; /* 空きコード表 */
extern int codedata_tbl[ MEMBER_MAX ]; /* コード・データ対照表 */
extern struct KEISOKU_TBL kojiri_keisoku_tbl; /* 個人別計測データ表 */
extern struct KEISOKU_TBL sort_keisoku_tbl[ MEMBER_MAX ]; /* ソート用計測データ表 */

/* 入力計測データテーブル */
static struct KEISOKU_INPUT keisoku_indata[ 5 ];

/*****
/* 計測記録入力処理 */
/* メインルーチン */
/* */
/* パラメータ : なし */
/* リターン : 0:OK */
/* -1:NG */
/*****
int keisoku_input( void )
{
    int ret; /* リターンコード */
    int i; /* インデックス */
    int kaiin_code; /* 計測会員コード */
    int undou_sisuu; /* 算出運動指数 */
    char msg[ 64 ]; /* メッセージエリア */
    char keisoku_date[ 9 ]; /* 計測日付 */

    /* 空きコード表 READ -> NG ? */
    if( (ret = akicode_tbl_read( )) == NG ) {
        return ret;
    }

    /* 入会者チェック */
    if( akicode_tbl[ 0 ] >= MEMBER_MAX ) {
        printf( "\n 現在, 入会者がいません" );
        return NG;
    }

    /* コード・データ対照表ファイル READ -> NG ? */
    if( (ret = codedata_tbl_read( )) == NG ) {
        return ret;
    }

    /* 計測会員コード入力 */
    kaiin_code_input( &kaiin_code );

    /* 計測日付入力 */
    keisoku_date_input( keisoku_date );

    /* 計測値入力処理 */
    for( i = 0; i < 5; i++ ) {
        keisoku_data_input( i );
    }

    /* 入力データ表示 */
    keisoku_data_disp( kaiin_code, keisoku_date );

    /* 入力確認 */
    strcpy( msg, "\n 入力はよろしいですか( Y/N )" );

```

```

/* 'N' OR 'n'入力 -> OK でない ? */
if( (ret = kakunin_input( msg )) != OK ) {
    return NG;
}

/* 計測データ表 READ -> NG ? */
if( (ret = kojim_data_read( kaiin_code )) == NG ) {
    return ret;
}

/* 計測値計算処理 */
undou_sisuu_keisan( &undou_sisuu );

/* 計測データ表 SET */
kojin_data_update( kaiin_code, keisoku_date, undou_sisuu );

/* 計測結果データ表示 */
kojin_data_disp( kaiin_code, "%n ** 計測結果データ **" );

/* キー入力待ち */
while( getchar( ) != '\n' );
printf( "%n リターンキーを押してください" );
getchar( );

/* 計測結果順位ソート表示 */
ret = keisoku_rank( );

return ret;
}

/*****
/* 計測記録入力処理 */
/* コード・データ対照表 読み込み処理 */
/* */
/* パラメータ : なし */
/* リターン : 0:OK */
/* -1:NG */
*****/
static int codedata_tbl_read( void )
{
    int ret; /* リターンコード */
    FILE *fp; /* ファイルポインタ */
    char *fname = CODEDATA_TBL_NAME; /* コード・データ対照表ファイル */

    /* コード・データ対照表ファイル OPEN -> NULL ? */
    if( (fp = fopen( fname, "rb" )) == NULL ) {
        printf( "%n コード・データ対照表ファイル OPEN エラー" );
        return NG;
    }

    /* コード・データ対照表ファイル READ -> 1以外 ? */
    if( (ret = fread( (char *)codedata_tbl, sizeof( codedata_tbl ), 1, fp ))
        != 1 ) {
        printf( "%n コード・データ対照表ファイル READ エラー" );
        ret = NG;
    }
    else {
        ret = OK;
    }

    /* コード・データ対照表ファイル CLOSE */
    fclose( fp );

    return ret;
}

```

```

/*****
/* 計測記録入力処理          */
/*  会員コード入力処理      */
/*                          */
/*  パラメータ  : 入力会員コード  */
/*  リターン   : なし          */
/*****
static void kaiin_code_input( int *kaiin_code )
{
    int    loop = TRUE;        /* ループフラグ      */
    char   work[ 128 ];       /* 入力ワーク        */

    while( loop ) {
        printf( "\n 計測者の会員コードを入力してください" );
        printf( "\n ? " );

        /* 会員コード入力 */
        work[ 0 ] = '\0';
        scanf( "%s", work );

        /* ニューメリック・チェック -> 数値以外 ? */
        if( strstr( work, "1234567890" ) < strlen( work ) ) {
            printf( "\n 数値以外が入力されました" );
            continue;
        }

        /* 入力範囲チェック( 0 < kaiin_code <= MEMBER_MAX ) */
        *kaiin_code = atoi( work );
        if( *kaiin_code > MEMBER_MAX || *kaiin_code <= 0 ) {
            printf( "\n 入力ミスです" );
            continue;
        }

        /* 会員コード登録チェック -> 未登録 ? */
        if( codedata_tbl[ *kaiin_code - 1 ] == 0 ) {
            printf( "\n この会員コードは未登録です" );
            continue;
        }
        break;
    }
    return;
}

```

```

/*****
/* 計測記録入力処理          */
/*  計測日付入力処理      */
/*                          */
/*  パラメータ  : 入力日付    */
/*  リターン   : なし          */
/*****
static void keisoku_date_input( char *keisoku_date )
{
    int    loop = TRUE;        /* ループフラグ      */
    int    chk_date;          /* 日付数値          */
    char   conv[ 3 ];         /* 数値変換用        */
    char   work[ 128 ];       /* 入力ワーク        */

    while( loop ) {
        printf( "\n 日付を入力してください( YYYYMMDD )" );
        printf( "\n ? " );

        /* 日付入力 */
        work[ 0 ] = '\0';
        scanf( "%s", work );

        /* 入力桁数チェック -> 8 以外 ? */
        if( strlen( work ) != 8 ) {
            printf( "\n 入力ミスです" );
            continue;
        }
    }
}

```

```

    }

    /* ニューメリック・チェック -> 数値以外 ? */
    if( strspn( work, "1234567890" ) < strlen( work ) ) {
        printf( "¥n 数値以外が入力されました" );
        continue;
    }

    /* 月チェック */
    conv[0] = work[4];
    conv[1] = work[5];
    conv[2] = '¥0';

    chk_date = atoi( conv );
    if( chk_date > 12 || chk_date < 1 ) {
        printf( "¥n 日付( 月 )入力エラーです" );
        continue;
    }

    /* 日チェック */
    conv[0] = work[6];
    conv[1] = work[7];
    conv[2] = '¥0';
    chk_date = atoi( conv );
    if( chk_date > 31 || chk_date < 1 ) {
        printf( "¥n 日付( 日 )入力エラーです" );
        continue;
    }

    break;
}

/* 入力データ セット */
strcpy(keisoku_date, work);
return;
}

/*****
/* 計測記録入力処理
/* 計測データ入力処理
/*
/* パラメータ : インデックス
/* リターン : なし
*****/
static void keisoku_data_input( int idx )
{
    int i; /* インデックス */
    int loop = TRUE; /* ループフラグ */
    char work[ 3 ][ 128 ]; /* 入力ワーク */

    while( loop ) {
        printf( "¥n 運動 %d の計測データを入力してください", idx + 1 );
        printf( "¥n 負荷( 999 ) セット( 100 ) 回数( 100 )" );
        printf( "¥n ? " );

        /* 計測データ入力 */
        work[ 0 ][ 0 ] = '¥0';
        work[ 1 ][ 0 ] = '¥0';
        work[ 2 ][ 0 ] = '¥0';
        scanf( "%s %s %s", work[ 0 ], work[ 1 ], work[ 2 ] );

        for( i = 0; i < 3; i++ ) {

            /* 入力桁数チェック -> 3 より大きい ? */
            if( strlen( work[ i ] ) > 3 ) {
                printf( "¥n 入力ミスです" );
                break;
            }
        }
    }
}

```

```

/* ニューメリック・チェック -> 数値以外? */
if( strstr( work[ i ], "1234567890" ) < strlen( work[ i ] ) ) {
    printf( "%n 数値以外が入力されました" );
    break;
}

/* 負荷以外の上限チェック -> 100 より大きい? */
if( i != 0 ) {
    if( atoi( work[ i ] ) > 100 ) {
        printf( "%n 上限( 100 )を超えています" );
        break;
    }
}

if( i < 3 ) {
    continue;
}

break;
}

/* 入力データテーブル セット */
keisoku_indata[ idx ].huka = atoi( work[ 0 ] );
keisoku_indata[ idx ].set = atoi( work[ 1 ] );
keisoku_indata[ idx ].kaisuu = atoi( work[ 2 ] );

return;
}

```

```

/*****
/* 計測記録入力処理 */
/* 入力計測データ 表示処理 */
/* */
/* パラメータ : 会員コード */
/* 日付 */
/* リターン : なし */
/*****
static void keisoku_data_disp( int kaiin_code, char *keisoku_date )
{
    int i; /* インデックス */

    printf( "%n ** 入力計測値データ **" );
    printf( "%n 会員コード %3d", kaiin_code );
    printf( "%n 日付 %4.4s-%2.2s-%2.2s",
        ( keisoku_date + 0 ), ( keisoku_date + 4 ), ( keisoku_date + 6 ) );

    printf( "%n%n 運動 負荷 セット 回数" );

    for( i = 0; i < 5; i++ ) {
        printf( "%n %d %3d %3d %3d", i + 1,
            keisoku_indata[ i ].huka,
            keisoku_indata[ i ].set,
            keisoku_indata[ i ].kaisuu );
    }

    return;
}

```

```

/*****
/* 計測記録入力処理 */
/* 運動指数計算処理 */
/* */
/* パラメータ : 運動指数 */
/* リターン : なし */
/*****
static void undou_sisuu_keisan( int *undou_sisuu )

```

```

{
    int    i;                /* インデックス */
    double sisuu;           /* 計算ワーク */
    double sisuu_total;     /* 計算値合計 */
    static double huka_sisuu[ 5 ] = { 0.24, 0.36, 0.52, 1.05, 2.13 };

    sisuu_total = 0.0;
    for( i = 0; i < 5; i++ ) {

        /* 回数 1 以下 ? */
        if( keisoku_indata[ i ].kaisuu <= 1 ) {
            continue;
        }

        sisuu = huka_sisuu[ i ] * (double)keisoku_indata[ i ].huka *
                (double)keisoku_indata[ i ].set *
                ((pow( (double)keisoku_indata[ i ].kaisuu, 2.0 )) /
                 ((double)keisoku_indata[ i ].kaisuu - 1.0));

        /* シグマ(累計)の計算 */
        sisuu_total += sisuu;
    }

    /* 平方根の算出 */
    *undou_sisuu = (int)sqrt( sisuu_total );

    return;
}

/*****
/* 計測記録入力処理 */
/* 個人計測データ表 更新処理 */
/*
/* パラメータ : 会員コード */
/*                日付 */
/*                運動指数 */
/* リターン   : 0:OK */
/*                -1:NG */
*****/
static int kojinn_data_update( int kaiin_code, char *keisoku_date, int undou_sisuu )
{
    int    ret;                /* リターンコード */
    long   fptr;              /* 計測データポインタ */
    FILE   *fptr;             /* ファイルポインタ */
    char   *fname = KEISOKU_TBL_NAME; /* 計測データ表ファイル */
    int    i;                /* インデックス */

    /* 1 回目 ? */
    if( kojinn_keisoku_tbl.count <= 0 ) {
        strcpy( kojinn_keisoku_tbl.first_date, keisoku_date );
        kojinn_keisoku_tbl.first_data = undou_sisuu;

        strcpy( kojinn_keisoku_tbl.max_date, keisoku_date );
        kojinn_keisoku_tbl.max_data = undou_sisuu;
    }
    else {

        /* 最高記録 ? */
        if( kojinn_keisoku_tbl.max_data < undou_sisuu ) {
            strcpy( kojinn_keisoku_tbl.max_date, keisoku_date );
            kojinn_keisoku_tbl.max_data = undou_sisuu;
        }
        for( i = sizeof kojinn_keisoku_tbl.soku_data / sizeof(int) - 1; i > 0; i-- )
            kojinn_keisoku_tbl.soku_data[ i ] = kojinn_keisoku_tbl.soku_data[ i - 1 ];
    }

    strcpy( kojinn_keisoku_tbl.soku_date, keisoku_date );
    kojinn_keisoku_tbl.soku_data[ 0 ] = undou_sisuu;
    kojinn_keisoku_tbl.count++;
}

```

```

/* 計測データ表ファイル OPEN -> NULL ? */
if( (fp = fopen( fname, "r+b" )) == NULL ) {
    printf( "¥n 計測データ表ファイル OPEN エラー" );
    return NG;
}

/* 該当データポインタ セット */
fptr = ( codedata_tbl[ kain_code - 1 ] - 1 ) *
        sizeof( struct KEISOKU_TBL );

/* 計測データ表ファイル SEEK -> OK でない ? */
if( (ret = fseek( fp, fptr, SEEK_SET )) != OK ) {
    printf( "¥n 計測データ表ファイル SEEK エラー" );

    /* 計測データ表ファイル CLOSE */
    fclose( fp );
    return NG;
}

/* 計測データ表ファイル WRITE -> 1以外 ? */
if( (ret = fwrite( (char *)&kojin_keisoku_tbl, sizeof( koin_keisoku_tbl ),
                1, fp )) != 1 ) {
    printf( "¥n 計測データ表ファイル WRITE エラー" );
    ret = NG;
}
else {
    ret = OK;
}

/* 計測データ表ファイル CLOSE */
fclose( fp );

return ret;
}

```

```

/*****/
/* 計測記録入力処理 */
/* 計測結果順位ソート・表示処理 */
/* */
/* パラメータ : なし */
/* リターン : 0:OK */
/* -1:NG */
/*****/
static int keisoku_rank( void )
{
    int ret; /* リターンコード */
    int i; /* インデックス */
    FILE *fp; /* ファイルポインタ */
    char *fname = KEISOKU_TBL_NAME; /* 計測データ表ファイル */

    /* 計測データ表ファイル OPEN -> NULL ? */
    if( (fp = fopen( fname, "rb" )) == NULL ) {
        printf( "¥n 計測データ表ファイル OPEN エラー" );
        return NG;
    }

    i = 0;
    for( ; ; ) {

        /* 計測データ表ファイル READ -> 1以外 ? */
        if( (ret = fread( (char *)&sort_keisoku_tbl[ i ],
                sizeof( struct KEISOKU_TBL ), 1, fp )) != 1 ) {

            /* READ エラー ? */
            if( ferrror( fp ) != 0 ) {
                printf( "¥n 計測データ表ファイル READ エラー" );
                ret = NG;
            }
        }
    }
}

```

```

else {
    /* ファイル EOF でない ? */
    if( feof( fp ) == 0 ) {
        printf( "%n 計測データ表ファイル READ エラー" );
        ret = NG;
    }
    else {
        ret = OK;
    }
}
}

/* READ エラー ? */
if( ret == NG ) {
    break;
}

/* 計測データあり ? */
if( sort_keisoku_tbl[ i ].count != 0 ) {
    i++;
}

/* eof ? */
if( ret == OK ) {
    break;
}
}

/* 計測データ表ファイル CLOSE */
fclose( fp );

if( ret == OK ) {
    /* 計測データソート */
    keisoku_data_sort( i );
    /* 計測データソート結果表示 */
    keisoku_sort_disp( i );
}

return ret;
}

```

```

/*****
/* 計測記録入力処理 */
/* 計測結果順位ソート処理 */
/* */
/* パラメータ : 計測データ件数 */
/* リターン : なし */
*****/
static void keisoku_data_sort( int cnt )
{
    int i; /* インデックス */
    int j; /* インデックス */
    struct KEISOKU_TBL work; /* スワップ用エリア */

    /* データソート */
    for( i = 0; i < cnt - 1; i++ ) {
        for( j = i + 1; j < cnt; j++ ) {
            if( sort_keisoku_tbl[ i ].max_data <
                sort_keisoku_tbl[ j ].max_data ) {

                work = sort_keisoku_tbl[ i ];
                sort_keisoku_tbl[ i ] = sort_keisoku_tbl[ j ];
                sort_keisoku_tbl[ j ] = work;
            }
        }
    }
    return;
}

```

```

/*****
/* 計測記録入力処理          */
/* 計測結果順位表示処理      */
/*                             */
/* パラメータ : 計測データ件数 */
/* リターン   : なし           */
/*****
static void keisoku_sort_disp( int cnt )
{
    int    i;                /* インデックス */

    printf( "\n\n ** 順位表 **" );
    printf( "\n 順位 会員コード 最高データ   日付" );

    for( i = 0; i < cnt; i++ ) {
        /* 10位まで表示する */
        if( i >= 10 ) {
            break;
        }

        printf( "\n  %2d  %3d          %4d  %4.4s-%2.2s-%2.2s", i + 1,
                sort_keisoku_tbl[ i ].kaiin_code,
                sort_keisoku_tbl[ i ].max_data,
                &sort_keisoku_tbl[ i ].max_date[ 0 ],
                &sort_keisoku_tbl[ i ].max_date[ 4 ],
                &sort_keisoku_tbl[ i ].max_date[ 6 ] );
    }
    return;
}

```

```

/*****
/* sakujyo.c */
/* 登録削除処理プログラム */
/*****
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include "common.h"
#include "main.h"
#include "sakujyo.h"

extern int akicode_tbl[ MEMBER_MAX + 1 ]; /* 空きコード表 */
extern int codedata_tbl[ MEMBER_MAX ]; /* コード・データ対照表 */
extern struct KEISOKU_TBL kojcin_keisoku_tbl; /* 個人別計測データ表 */

/*****
/* 登録削除処理 */
/* メインルーチン */
/* パラメータ : なし */
/* リターン : 0:OK */
/* -1:NG */
/*****
int touroku_sakujyo( void )
{
    int ret; /* リターンコード */
    int loop = TRUE; /* ループフラグ */
    int kaiin_code; /* 削除会員コード */
    char work[ 128 ]; /* 入力ワーク */

    while( loop ) {
        printf( "\n 退会者の会員コードを入力してください" );
        printf( "\n ? " );

        /* 会員コード入力 */
        work[ 0 ] = '\0';
        scanf( "%s", work );

        /* ニューメリック・チェック -> 数値以外 ? */
        if( strstr( work, "1234567890" ) < strlen( work ) ) {
            printf( "\n 数値以外が入力されました" );
            continue;
        }

        /* 入力範囲チェック -> ( 0 < kaiin_code <= MEMBER_MAX ) ? */
        kaiin_code = atoi( work );
        if( kaiin_code > MEMBER_MAX || kaiin_code <= 0 ) {
            printf( "\n 入力ミスです" );
            continue;
        }

        /* コード・データ対照表削除 -> OK ? */
        if( (ret = codedata_tbl_delete( kaiin_code )) == OK ) {
            /* 空きコード表追加 -> OK ? */
            if( (ret = akicode_tbl_add( kaiin_code )) == OK ) {
                /* メインに戻る */
                loop = FALSE;
            }
        }
        else {
            /* メインに戻る */
            loop = FALSE;
        }
    }

    if( ret == OK ) {
        printf( "\n 入会登録削除処理が終了しました" );
    }

    return ret;
}

```

```
}
```

```
/******  
/* 登録削除処理 */  
/* コード・データ対照表 削除処理 */  
/* */  
/* パラメータ : 削除会員コード */  
/* リターン : 0:OK */  
/* 1:CANCEL */  
/* -1:NG */  
/******  
static int codedata_tbl_delete( int kaiin_code )  
{  
    int ret; /* リターンコード */  
    int i; /* インデックス */  
    char msg[ 64 ]; /* メッセージエリア */  
    FILE *fp; /* ファイルポインタ */  
    char *fname = CODEDATA_TBL_NAME; /* コード・データ対照表ファイル */  
  
    /* コード・データ対照表ファイル OPEN -> NULL ? */  
    if( (fp = fopen( fname, "r+b" )) == NULL ) {  
        printf( "%n コード・データ対照表ファイル OPEN エラー" );  
        return NG;  
    }  
  
    /* コード・データ対照表ファイル READ -> 1以外 ? */  
    if( (ret = fread( (char *)codedata_tbl, sizeof( codedata_tbl ), 1, fp ))  
        != 1 ) {  
        /* READ エラー */  
        printf( "%n コード・データ対照表ファイル READ エラー" );  
  
        /* コード・データ対照表ファイル CLOSE */  
        fclose( fp );  
        return NG;  
    }  
  
    /* 会員登録チェック -> 未登録 ? */  
    if( codedata_tbl[ kaiin_code - 1 ] == 0 ) {  
        printf( "%n この会員コードは未登録です" );  
  
        /* コード・データ対照表ファイル CLOSE */  
        fclose( fp );  
        return CANCEL;  
    }  
  
    /* 計測データ表 READ -> NG ? */  
    if( (ret = kojim_data_read( kaiin_code )) == NG ) {  
        /* コード・データ対照表ファイル CLOSE */  
        fclose( fp );  
        return ret;  
    }  
  
    /* 計測データ表示 */  
    kojim_data_disp( kaiin_code, "%n ** 削除データ **" );  
  
    /* 計測データ表削除確認 */  
    sprintf( msg, "%n%n 上のデータを削除します。よろしいですか( Y/N )" );  
  
    if( (ret = kakunin_input( msg )) == OK ) {  
        /* 計測データ表削除 -> OK ? */  
        if( (ret = kojim_data_delete( kaiin_code )) == OK ) {  
            /* 計測データポインタ更新 */  
            for( i = 0; i < MEMBER_MAX; i++ ) {  
                if( codedata_tbl[ i ] > codedata_tbl[ kaiin_code - 1 ] ) {  
                    codedata_tbl[ i ]--;  
                }  
            }  
        }  
    }  
}
```

```

/* 計測データポインタクリア */
codedata_tbl[ kaiin_code - 1 ] = 0;

/* コード・データ対照表ファイルの先頭位置に SEEK -> OK ? */
if( (ret = fseek( fp, 0L, SEEK_SET )) == OK ) {

    /* コード・データ対照表ファイル WRITE -> 1以外 ? */
    if( (ret = fwrite( (char *)codedata_tbl,
                      sizeof( codedata_tbl ),1, fp )) != 1 ) {
        printf( "%n コード・データ対照表ファイル WRITE エラー" );
        ret = NG;
    }
    else {
        ret = OK;
    }
}
else {
    printf( "%n コード・データ対照表ファイル SEEK エラー" );
    ret = NG;
}
}
else {
    /* 処理中止 */
    ret = CANCEL;
}

/* コード・データ対照表ファイル CLOSE */
fclose( fp );

return ret;
}

/*****
/* 登録削除処理 */
/* 個人計測データ表 削除処理 */
/*
/* パラメータ : 削除会員コード */
/* リターン : 0:OK */
/* -1:NG */
*****/
static int kojim_data_delete( int kaiin_code )
{
    int ret; /* リターンコード */
    int i; /* インデックス */
    FILE *fp; /* 計測データ表ファイルポインタ */
    FILE *tmp; /* テンポラリファイルポインタ */
    char *fname = KEISOKU_TBL_NAME; /* 計測データ表ファイル */
    char *tmpfl = "keisoku.tmp"; /* テンポラリファイル */

    /* 計測データ表ファイル OPEN -> NULL ? */
    if( (fp = fopen( fname, "rb" )) == NULL ) {
        printf( "%n 計測データ表ファイル OPEN エラー" );
        return NG;
    }

    /* テンポラリファイル OPEN -> NULL ? */
    if( (tmp = fopen( tmpfl, "wb" )) == NULL ) {
        printf( "%n テンポラリファイル OPEN エラー" );
        fclose( fp );
        return NG;
    }

    i = 0;
    for( ; ; ) {

        /* 計測データ表ファイル READ -> 1以外 ? */
        if( (ret = fread( (char *)&kojin_keisoku_tbl,

```

```

        sizeof( kojinkaisoku_tbl ), 1, fp )) != 1 ) {
/* READ エラー ? */
if( ferror( fp ) != 0 ) {
    printf( "%n 計測データ表ファイル READ エラー" );
    ret = NG;
}
else {
/* ファイル EOF でない ? */
if( feof( fp ) == 0 ) {
    printf( "%n 計測データ表ファイル READ エラー" );
    ret = NG;
}
else {
    ret = OK;
}
}
break;
}

/* 削除データ ? */
if( kaimin_code == kojinkaisoku_tbl.kaimin_code ) {
    continue;
}

/* テンポラリファイル WRITE -> 1以外 ? */
if( (ret = fwrite( (char *)&kojinkaisoku_tbl,
    sizeof( kojinkaisoku_tbl ), 1, tmp )) != 1 ) {
    printf( "%n 計測データ表ファイル WRITE エラー" );
    ret = NG;
    break;
}
i++;
}

/* テンポラリファイル CLOSE */
fclose( tmp );

/* 計測データ表ファイル CLOSE */
fclose( fp );

/* 正常終了 ? */
if( ret == OK ) {
/* 計測データ表ファイル DELETE -> 0以外 ? */
if( (ret = remove( fname )) != 0 ) {
    printf( "%n 計測データ表ファイル削除エラー" );
    ret = NG;
}
else {
/* 書き込みデータあり ? */
if( i > 0 ) {
/* テンポラリファイルを計測データ表ファイルにリネームする -> 0以外 ? */
if( (ret = rename( tmpfl, fname )) != 0 ) {
    printf( "%n 計測データ表ファイルリネームエラー" );
    ret = NG;
}
}
else {
/* テンポラリファイル削除 */
remove( tmpfl );
}
}
}
else {
/* テンポラリファイル削除 */
remove( tmpfl );
}
}

return ret;

```

```
}
```

```
/* **** */
/* 登録削除処理 */
/* 空きコード表 追加処理 */
/*
/* パラメータ : 削除会員コード */
/* リターン : 0:OK */
/*          -1:NG */
/* **** */
static int akicode_tbl_add( int kaiin_code )
{
    int    ret;                /* リターンコード */
    int    cnt;                /* 空きコード件数 */
    FILE   *fp;                /* ファイルポインタ */
    char   *fname = AKICODE_TBL_NAME; /* 空きコード表ファイル */

    /* 空きコード表 READ -> NG ? */
    if( (ret = akicode_tbl_read( )) == NG ) {
        return ret;
    }

    /* 空きコード件数セット */
    cnt = akicode_tbl[ 0 ];

    /* 空きコードテーブルセット */
    akicode_tbl[ cnt + 1 ] = kaiin_code;

    /* 空きコード件数セット */
    akicode_tbl[ 0 ] = cnt + 1;

    /* 空きコード表ファイル OPEN -> NULL ? */
    if( (fp = fopen( fname, "wb" )) == NULL ) {
        printf( "\n 空きコード表ファイル OPEN エラー" );
        return NG;
    }

    /* 空きコード表ファイル WRITE -> 1以外 ? */
    if( (ret = fwrite( (char *)akicode_tbl,
                      sizeof( int ) * ( akicode_tbl[ 0 ] + 1 ), 1, fp )) != 1 ) {
        printf( "\n 空きコード表ファイル WRITE エラー" );
        ret = NG;
    }
    else {
        ret = OK;
    }

    /* 空きコード表ファイル CLOSE */
    fclose( fp );

    return ret;
}
}
```

