

## 構造体リファレンス

名称

ADR\_CODE

用途

住所コードを格納する。

宣言

```
typedef struct {
    int codeLevel; /* コードが有効なレベル */
    int cityCode; /* 市区町村コード(JIS) 5桁 :レベル4 */
    int choCode; /* 町・大字コード(独自) :レベル5 */
    int azaCode; /* 小字コード(独自) :レベル6 */
    int gaikuCode; /* 地番、街区符号コード(独自) :レベル7 */
    int jukyoCode; /* 地番の枝番、住居番号 :レベル8 */
} ADR_CODE
```

説明

住所は都道府県から始まって、地番、住居番号まで、場所を表す文字列が階層化されて並んでいます。この構造体はその階層化された住所をコードで表します。

名称 **GYOSEI\_HEADER**

用途 行政区域レコードファイルのヘッダー

宣言

```
typedef struct {
    char fileId[8];          /* ファイルID */
    int version;            /* バージョン */
    char notUsed[6];        /* 未使用 */
    int numRecords;         /* 行政区域レコード数 */
} GYOSEI_HEADER;
```

説明 行政区域レコードファイルのヘッダーを読み書きする際に使用します。

名称 **GYOSEI\_RECORD**

用途 行政区域レコードを格納

宣言

```
typedef struct {
    int shubetsu;            /* 種別 */
    int kenCode;            /* 県コード2桁 */
    int shichoCode;        /* 支庁コード5桁 */
    int gunCode;            /* 郡コード5桁 */
    int cityCode;          /* 市区町村コード5桁 */
    int removeFlag;        /* 削除フラグ */
    int newGyoseiCode[NEW_CODE_MAX]; /*変更後の市区町村コード
*/
    char name[GYOSEI_NAME_LEN+1]; /* 漢字名称 */
    char kana[GYOSEI_KANA_LEN+1]; /* 仮名名称 */
    int choHashTableNum;    /* 町大字ハッシュテーブル
番号 */
    int latitude;           /* 緯度 (ミリ秒) */
    int longitude;         /* 経度 (ミリ秒) */
} GYOSEI_RECORD;
```

説明 都道府県から市区町村までの区域を表すための構造体です。  
行政区域レコードファイルのアクセスと主要な処理で使用します。

名称

**GYOSEI\_HASH\_HEADER**

用途

行政区域名ハッシュテーブルファイルのヘッダー

宣言

```
typedef struct {
    char fileId[8];          /* ファイルID */
    int version;            /* バージョン */
    char notUsed[4];        /* 未使用 */
    int numRecords;         /* ハッシュレコード数 */
    int hashSize;           /* ハッシュサイズ */
} GYOSEI_HASH_HEADER;
```

説明

行政区域レコードファイルのヘッダーを読み書きする際に使用します。

名称

**GYOSEI\_HASH\_BUCKET**

用途

行政区域検索用名称のレコードを格納

宣言

```
typedef struct {
    char name[GYOSEI_NAME_LEN+1]; /* 検索用名称 */
    int gyoseiCode;                /* 行政区域コード */
    int nextBucketNum;             /* 次のバケット番号 */
} GYOSEI_HASH_BUCKET;
```

説明

行政区域名の検索をハッシュ値で行いますが、その検索の際に使用します。  
通常のライブラリ使用ではこの構造体は使いません。

名称

**CHO\_HEADER**

用途

町大字レコードファイルのヘッダー

宣言

```
typedef struct {
    char fileId[8];          /* ファイルID */
    int version;            /* バージョン */
    char notUsed[8];        /* 未使用 */
    int numRecords;         /* 町大字レコード数 */
    int numBytes;           /* 1レコードのバイト数 */
    int offsetRecordTop;    /* レコード先頭のバイト位置 */
} CHO_HEADER;
```

説明

町大字レコードファイルのヘッダーを読み書きする際に使用します。

名称

**CHO\_OFFSET**

用途

町大字レコードの格納位置

宣言

```
typedef struct {
    int numRecords;         /* 町大字レコード数 */
    int startRecNum;        /* 先頭のレコード番号 */
} CHO_OFFSET;
```

説明

町大字レコードは市区町村毎に格納されていますが、各市区町村のレコードがどこから始まるか示します。

名称 **CHO\_RECORD**

用途 町大字レコードを格納

宣言

```
typedef struct {
    int choCode;           /* 町大字コード */
    int attribute;        /* 属性 */
    char name[CHO_NAME_LEN+1]; /* 町大字名 */
    char kana[CHO_KANA_LEN+1]; /* 読み仮名 */
    int zipCode;          /* 郵便番号 */
    int azaTableNum;      /* 丁目字テーブル番号 */
    int chibanTableNum;   /* 地番街区符号テーブル番号*/
    int latitude;         /* 緯度 (ミリ秒) */
    int longitude;        /* 経度 (ミリ秒) */
} CHO_RECORD;
```

説明 町大字のレコードを格納します。

町大字レコードファイルのアクセスと主要な処理で使われます。

名称

**CHO\_HASH\_HEADER**

用途

町大字名ハッシュテーブルファイルのヘッダー

宣言

```
typedef struct {
    char fileId[8];          /* ファイルID */
    int version;            /* バージョン */
    char notUsed[16];       /* 未使用 */
    int hashTableBytes;     /* ハッシュテーブルのバイト数 */
    int numRecords;        /* 名称レコード総数 */
} CHO_HASH_HEADER;
```

説明

町大字ハッシュテーブルファイルのヘッダーを読み書きする際に使用します。

名称

**CHO\_HASH\_BUCKET**

用途

町大字検索用名称のレコードを格納

宣言

```
typedef struct {
    char name[GYOSEI_NAME_LEN+1]; /* 検索用名称 */
    int gyoseiCode;                /* 行政区域コード */
    int nextBucketNum;             /* 次のバケット番号 */
} GYOSEI_HASH_BUCKET;
```

説明

行政区域名の検索をハッシュ値で行いますが、その検索の際に使用します。  
通常のライブラリ使用ではこの構造体は使いません。

名称

**AZA\_HEADER**

用途

丁目字レコードファイルのヘッダー

宣言

```
typedef struct{
    char fileId[8];        /* ファイルID */
    int version;          /* バージョン */
    char notUsed[8];      /* 未使用 */
    int maxNumTables;     /* 未使用 */
    int numTables;        /* 丁目字テーブル数 */
    int numTotalRecords; /* 丁目字レコード総数 */
} AZA_HEADER;
```

説明

丁目字レコードファイルのヘッダーを読み書きする際に使用します。

名称

**AZA\_TABLE\_LIST**

用途

丁目字テーブルのリスト

宣言

```
typedef struct {
    int cityCode;        /* 5桁市区町村コード */
    int choCode;         /* 町大字コード */
    int numRecords;     /* 丁目字レコード数 */
    int numSerialChomoku; /* 連続丁目の数 */
    int startRecNum;    /* テーブル先頭のレコード番号 */
} AZA_TABLE_LIST;
```

説明

丁目字レコードは町大字毎にまとまった形(テーブル)で格納されていますが、各町大字のテーブルがどこから始まるかなどの情報を格納します。

名称

AZA\_RECORD

用途

丁目字レコードを格納

宣言

```
typedef struct {
    int choazaCode;          /* 町大字コード+丁目字コード */
    char name[AZA_NAME_LEN+1]; /* 丁目字漢字名称 */
    char kana[AZA_KANA_LEN+1]; /* 丁目字仮名名称 */
    int chibanTableNum;     /* 地番街区符号テーブル番号 */
    int latitude;           /* 緯度 (ミリ秒) */
    int longitude;          /* 経度 (ミリ秒) */
} AZA_RECOR
```

説明

丁目字のレコードを格納します。

丁目字レコードファイルのアクセスと主要な処理で使用します。

名称

**GAIKU\_HEADER**

用途

地番街区符号レコードファイルのヘッダー

宣言

```
typedef struct{
    char fileId[8];        /* ファイルID */
    int version;          /* バージョン */
    char notUsed[8];      /* 未使用 */
    int maxNumTables;     /* 未使用 */
    int numTables;        /* 地番街区符号テーブル数 */
    int numTotalRecords; /* 地番街区符号レコード総数 */
} GAIKU_HEADER;
```

説明

地番街区符号レコードファイルのヘッダーを読み書きする際に使用します。

名称

**GAIKU\_TABLE\_LIST**

用途

地番街区符号テーブルのリスト

宣言

```
typedef struct {
    int cityCode;        /* 5桁市区町村コード */
    int choCode;         /* 町大字コード */
    int azaCode;         /* 丁目字コード */
    int numRecords;     /* 地番街区符号レコード数 */
    int startRecNum;    /* テーブル先頭のレコード番号 */
    short jukyoHyoji;   /* 住居表示フラグ */
} GAIKU_TABLE_LIST;
```

説明

地番街区符号レコードは丁目字毎にまとまった形(テーブル)で格納されていますが、各丁目字のテーブルがどこから始まるかなどの情報を格納します。

**名称**

**GAIKU\_RECORD**

**用途**

地番街区符号レコードを格納

**宣言**

```
typedef struct {  
    int gaikuCode;          /* 地番街区符号コード */  
    int jukyoTableNum;     /* 枝番住居番号テーブル番号 */  
    int latitude;          /* 緯度 (ミリ秒) */  
    int longitude;        /* 経度 (ミリ秒) */  
} GAIKU_RECORD;
```

**説明**

地番街区符号のレコードを格納します。

地番街区符号レコードファイルのアクセスと主要な処理で使します。

名称

JUKYO\_HEADER

用途

枝番住居番号レコードファイルのヘッダー

宣言

```
typedef struct {
    char fileId[8];          /* ファイルID */
    int version;            /* バージョン */
    char notUsed[8];        /* 未使用 */
    int maxNumTables;       /* 未使用 */
    int numTables;          /* 枝番住居番号テーブル数 */
    int numTotalRecords;    /* 枝番住居番号レコード総数 */
} JUKYO_HEADER;
```

説明

枝番住居番号レコードファイルのヘッダーを読み書きする際に使用します。

名称

JUKYO\_TABLE\_LIST

用途

枝番住居番号テーブルのリスト

宣言

```
typedef struct {
    int cityCode;           /* 5桁市区町村コード */
    int choCode;           /* 町大字コード */
    int azaCode;           /* 丁目字コード */
    int gaikuCode;         /* 地番街区符号コード */
    int numRecords;        /* 枝番住居番号レコード数 */
    int startRecNum;       /* テーブル先頭のレコード番号 */
} JUKYO_TABLE_LIST;
```

説明

枝番住居番号レコードは地番街区符号毎にまとまった形(テーブル)で格納されていますが、各地番街区符号のテーブルがどこから始まるかなどの情報を格納しま

名称

JUKYO\_RECORD

用途

枝番住居番号レコードを格納

宣言

```
typedef struct {  
    int jukyoCode;          /* 枝番住居番号コード */  
    int magobanTableNum;   /* 孫番テーブル番号 */  
    int latitude;          /* 緯度 (ミリ秒) */  
    int longitude;        /* 経度 (ミリ秒) */  
} JUKYO_RECORD;
```

説明

枝番住居番号のレコードを格納します。

枝番住居番号レコードファイルのアクセスと主要な処理で使われます。

名称

**STRING\_HEADER**

用途

地番文字列レコードファイルのヘッダー

宣言

```
typedef struct{
    char fileId[8];        /* ファイルID */
    int version;          /* バージョン */
    char notUsed[12];     /* 未使用 */
    int recordBytes;     /* 1レコードのバイト数 */
    int numRecords;      /* 文字列レコード数 */
} STRING_HEADER;
```

説明

地番文字列レコードファイルのヘッダーを読み書きする際に使用します。

名称

**STR\_RECORD**

用途

文字列レコードを格納

宣言

```
typedef struct {
    char str[CHIBANSTR_BYTES]; /* 文字列 */
} STR_RECORD;
```

説明

地番、枝番、街区符号など、通常は数字ですが、たまに文字列である場合があります。  
そのような場合にその文字列を格納します。