

LIBRERIA LDE

(liste di estrazione)

di Aldo Carpanelli – v1.0.1, 11/02/2020-12/11/2023

Descrizione

Questa libreria **C** rende disponibili liste numeriche di valori da **0** a **n** dalle quali estrarre in modo che nessun numero si ripeta finché non sono stati estratti tutti gli altri presenti nella lista.

La quantità massima dei valori ammessi è data dalla “capienza” massima del tipo **short int**, ovvero **0x7FFF (32767)**.

La lista dalla quale estrarre viene collocata in un *array* di valori di tipo **short int**, inizializzati al momento della creazione della lista stessa, ad ogni chiamata della funzione **LDE_Reset()**, e alla conclusione di ogni ciclo di estrazioni.

L'estrazione avviene impiegando la funzione di libreria **rand()** per identificare la posizione nella lista dalla quale “prelevare” il valore estratto.

N.B. È opportuno notare che questa libreria non è stata concepita per l'impiego in un contesto *multithread*.

Tipi di dati

In `liste_di_estrazione.h` vengono definiti due tipi di dati.

```
typedef struct {  
    short *v; // lista dei valori tra i quali estrarre  
    short qv; // quantita' dei valori nella lista v  
    short ue; // ultimo valore estratto (-1: nessuno)  
    short de; // valori ancora da estrarre  
    short ok; // vale 0x6B6F ('ok') se  
                // la struttura e' valida  
} LDE_struct;
```

```
typedef LDE_struct* LDE;
```

I campi della struttura `LDE_struct` sono liberamente accessibili, ma è sconsigliabile modificarli manualmente se non si sa bene quel che si sta facendo. Meglio usarli solo in lettura.

Il tipo `LDE` non è altro che un puntatore a una `LDE_struct`.

Le funzioni

Sono disponibili undici funzioni:

```
LDE LDE_Crea( short qVal );  
int LDE_Dimensiona( LDE lde, short qVal );  
LDE LDE_Duplica( const LDE originale );  
int LDE_Copia( LDE destinazione, const LDE origine );  
LDE LDE_Distruggi( LDE lde );  
int LDE_Valida( const LDE lde );  
int LDE_Reset( LDE lde );  
short LDE_Estrai( LDE lde );  
int LDE_UltimoErrore( void );  
const char *LDE_DescrizioneUltimoErrore( void );  
const char *LDE_DescrizioneErrore( int codice );
```

```
LDE LDE_Crea( short qVal );
```

Crea una `LDE_struct` in uno spazio di memoria dinamica allocato con `calloc()`, la popola con i dati necessari per una lista con `qVal` elementi, quindi restituisce il puntatore alla struttura creata e popolata. Qualora non siano ancora state create altre liste di estrazione, provvedere a effettuare una chiamata a `srand()`.

Parametri

<code>short qVal</code>	La quantità dei valori da inserire nella lista di estrazione creata.
-------------------------	----------------------------------------------------------------------

Valore di ritorno

<code>LDE</code>	Il puntatore alla struttura di tipo <code>LDE_struct</code> allocata e popolata. In caso di errore, <code>NULL</code> . Se viene restituito <code>NULL</code> , il codice dell'ultimo errore rilevato viene posto in una variabile statica leggibile tramite <code>LDE_UltimoErrore()</code> o <code>LDE_DescrizioneUltimoErrore()</code> .
------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

```
int LDE_Dimensiona( LDE lde, short qVal );
```

Ridimensiona un “oggetto” di tipo **LDE** valido, reinizializzandone la lista con **qVal** elementi.

Parametri

LDE lde	La lista di estrazione da ridimensionare. Deve trattarsi di una lista di estrazione valida.
short qVal	La quantità dei valori da inserire nella lista di estrazione ridimensionata.

Valore di ritorno

int	Un codice di errore che definisce l'esito del ridimensionamento.
------------	------------------------------------------------------------------

```
LDE LDE_Duplica( const LDE originale );
```

Duplica un “oggetto” di tipo **LDE** valido, creandone una copia identica.

Parametri

LDE originale La lista di estrazione da ridimensionare. **Deve** trattarsi di una lista di estrazione valida.

Valore di ritorno

LDE Il puntatore alla struttura di tipo **LDE_struct** allocata e popolata per creare il duplicato. In caso di errore, **NULL**. Se viene restituito **NULL**, il codice dell'ultimo errore rilevato viene posto in una variabile statica leggibile tramite **LDE_UltimoErrore()** o **LDE_DescrizioneUltimoErrore()**.

```
int LDE_Copia( LDE destinazione, const LDE origine );
```

Copia un “oggetto” di tipo **LDE** valido in un altro “oggetto” di tipo **LDE** valido.

Parametri

LDE destinazione La lista di estrazione destinata a ricevere i dati copiati dalla lista di estrazione puntata dal parametro **origine**. **Deve** trattarsi di una lista di estrazione valida.

LDE origine La lista di estrazione dalla quale copiare i dati. **Deve** trattarsi di una lista di estrazione valida.

Valore di ritorno

int Un codice di errore che definisce l’esito del ridimensionamento.


```
LDE LDE_Distruggi( LDE lde );
```

Libera tutta la memoria dinamica associata all'esistenza di un "oggetto" di tipo **LDE** valido. Restituisce **NULL** per permettere un pratico annullamento del puntatore passato come parametro in **lde**, in questo modo:

```
LDE lde = LDE_Crea( 10 );  
lde = LDE_Distruggi( lde );
```

Parametri

LDE lde La lista di estrazione da distruggere.

Valore di ritorno

LDE Sempre **NULL**.

```
int LDE_Valida( const LDE lde );
```

Tenta di appurare se il parametro `lde` punta a una lista di estrazione valida.

Parametri

`LDE lde` La lista di estrazione della quale verificare la validità.

Valore di ritorno

`int` Se il parametro `lde` punta a una lista valida il valore di ritorno è `1`; in caso contrario, il valore di ritorno è `0`. Se viene restituito `0`, il codice dell'ultimo errore rilevato viene posto in una variabile statica leggibile tramite `LDE_UltimoErrore()` o `LDE_DescrizioneUltimoErrore()`.

```
int LDE_Reset( LDE lde );
```

Riporta la lista di estrazione `lde` alla sua condizione originaria, come se fosse stata appena creata con `LDE_Crea()`.

Parametri

`LDE lde` La lista di estrazione da riportare alla condizione originaria. **Deve** trattarsi di una lista di estrazione valida.

Valore di ritorno

`int` Un codice di errore che definisce l'esito dell'operazione. Se il parametro `lde` punta a una lista di estrazione valida, l'operazione dà **sempre** esito positivo, senza errori di sorta.

```
short LDE_Estrai( LDE lde );
```

Estrae un valore a caso tra quelli compresi nella lista di estrazione e non ancora estratti. Qualora tutti i valori fossero stati già estratti, prima di procedere all'estrazione richiesta ripristina la condizione originale della lista tramite `LDE_Reset()`. Non restituisce **mai** lo stesso valore estratto nella chiamata immediatamente precedente.

Parametri

`LDE lde` La lista di estrazione dalla quale estrarre un dato numerico. **Deve** trattarsi di una lista di estrazione valida.

Valore di ritorno

`short` Il valore numerico estratto. Solo nel caso in cui venga passato un parametro che riguarda una lista di estrazione non valida, viene restituito **-1**. In tal caso, il codice dell'ultimo errore rilevato viene posto in una variabile statica leggibile tramite `LDE_UltimoErrore()` o `LDE_DescrizioneUltimoErrore()`.

```
int LDE_UltimoErrore( void );
```

Restituisce il valore corrente della variabile statica destinata a conservare il codice dell'ultimo errore rilevato dalla libreria. È opportuno notare nuovamente che nel caso di impieghi in contesti *multithread* il codice potrebbe non essere affidabile.

Valore di ritorno

`int` Un codice di errore.

```
const char *DescrizioneUltimoErrore( void );
```

Restituisce una stringa **C** che descrive il valore corrente della variabile statica destinata a conservare il codice dell'ultimo errore rilevato dalla libreria. È opportuno notare nuovamente che nel caso di impieghi in contesti *multithread* il codice potrebbe non essere affidabile, né (di conseguenza) la stringa descrittiva corrispondente.

Valore di ritorno

`const char*` Una stringa **C**.

```
const char *LDE_DescrizioneErrore( int codice );
```

Restituisce una stringa **C** che descrive il codice di errore passato in `codice`.

Parametri

<code>int codice</code>	Il codice di errore del quale si desidera conoscere la descrizione sotto forma di stringa C .
-------------------------	------------------------------------------------------------------------------------------------------

Valore di ritorno

<code>const char*</code>	Una stringa C .
--------------------------	------------------------

Esempio d'uso

```
#include "liste_di_estrazione.h"
```

```
int main() {  
    LDE lde = NULL, LDE dup = NULL;  
    int errore;
```

```
    if( NULL != (lde=LDE_Crea(10)) ) {  
        // LDE_Estrai() non puo' fallire, perche' lde e'  
        // qui sicuramente una lista di estrazione valida  
        printf( "Un numero tra 0 e 9: %d\n", LDE_Estrai(lde) );
```

```
    if( NULL != (dup=LDE_Duplica(lde)) ) {  
        // dup e' del tutto indipendente da lde, ma ne e' una  
        // copia esatta, per cui certamente non verra' estratto  
        // il numero gia' estratto nell'estrazione precedente  
        printf( "Un numero tra 0 e 9: %d\n", LDE_Estrai(dup) );
```

```
    if( LDEErr_NoErr == (errore=LDE_Dimensiona(dup,20)) ) {  
        // ora l'estrazione puo' dare numeri compresi tra 0 e 19,  
        // incluso uno dei due gia' estratti perche' la lista e'  
        // stata inizializzata al momento del ridimensionamento  
        printf( "Un numero tra 0 e 19: %d\n", LDE_Estrai(dup) );  
    } else printf( "ERRORE: %s\n\n", LDE_DescrizioneErrore(errore) );
```



```

    if( LDEErr_NoErr == (errore=LDE_Copia(dup,lde)) ) {
        // ora l'estrazione puo' dare numeri compresi tra 0 e 9,
        // esclusi i due gia' estratti perche' la lista contiene
        // gli stessi dati gia' presenti anche in lde
        printf( "Un numero tra 0 e 9: %d\n", LDE_Estrai(dup) );
    } else printf( "ERRORE: %s\n\n", LDE_DescrizioneErrore(errore) );

    dup = LDE_Distruggi( dup ); // lista distrutta, dup ora e' NULL
}

lde = LDE_Distruggi( lde ); // lista distrutta, lde ora e' NULL
}
}

```