List.h File Reference

```
#include <stdlib.h>
#include "status.h"
```

Go to the source code of this file.

Data Structures

struct	**Node**
struct	**List**

Typedefs

typedef struct **Node**	**Node**
typedef int(*	**compFun**) (void *, void *)
typedef void(*	**prFun**) (void *)
typedef struct **List**	**List**

Functions

List *	**newList** (**compFun**, **prFun**)
void	**delList** (**List** *)
status	**nthInList** (**List** *, int, void **)
status	**addListAt** (**List** *, int, void *)
status	**remFromListAt** (**List** *, int, void **)
status	**remFromList** (**List** *, void *)
status	**displayList** (**List** *)
void	**forEach** (**List** *, void(*)(void *))
int	**lengthList** (**List** *)
status	**addList** (**List** *, void *)
Node *	**isInList** (**List** *, void *)

Typedef Documentation

typedef int(* compFun) (void *, void *)

Comparison function for list elements. Must follow the "strcmp" convention: result is negative if first is less than second, null if both are equal, and positive otherwise.

typedef struct List List

The list embeds a counter for its size and the two function pointers

typedef struct Node Node

Creation of a generic (simply linked) **List** structure.

To create a list, one must provide two functions (one function to compare / order elements, one function to display them). Unlike arrays, indices begins with 1.Typical simple link structure: a **Node** is a "value / next" pair

typedef void(* prFun) (void *)

Display function for list elements

Function Documentation

```
status addList ( List *   ,
                 void *
               )
```

add given element to given list according to comparison function.

Parameters

```
status addListAt ( List *   ,
                   int      ,
                   void *
                 )
```

Insert element at a given position in the list.

Parameters

```
void delList ( List *   )
```

destroy the list by deallocating used memory.

Parameters

```
status displayList ( List *   )
```

display list the elements according to display function provided during list creation

Parameters

```
void forEach ( List *              ,
               void(*)(void *)
             )
```

sequentially call given function with each element of given list (O(NxF)).

Parameters

```
Node* isInList ( List *   ,
                 void *
               )
```

tests whether the list contains given element.

Parameters

```
int lengthList ( List *   )
```

compute and return the number of elements in given list.

Parameters

```
List* newList ( compFun  ,
                prFun
              )
```

Empty **List** creation by dynamic memory allocation.

Parameters

```
status nthInList ( List *    ,
                   int       ,
                   void **
                 )
```

get the Nth element of the list.

Parameters

```
status remFromList ( List *   ,
                     void *
                   )
```

remove given element from given list. user should provide a comparison function during list creation.

Parameters

```
status remFromListAt ( List *    ,
                       int       ,
                       void **
                     )
```

remove the element located at a given position in list.

Parameters

Generated by doxygen 1.8.11