map.h File Reference

```
#include "status.h"
#include "List.h"
```

Go to the source code of this file.

Data Structures

struct	**City**
struct	**Neighbour**

Macros

#define	**MAX_LENGTH**	1024

Typedefs

typedef struct **City**	**City**	
typedef struct **Neighbour**	**Neighbour**	

Functions

List *	**parseMapFile** (char *)
City *	**findCity** (**List** *, char *)
status	**findPath** (**List** *, char *, char *)
City *	**createCity** ()
Neighbour *	**createNeighbour** ()
void	**destroyCities** (**List** *)
void	**destroyCity** (**City** **)
void	**destroyNeighbour** (**Neighbour** **)

Variables

static char *	**mapFileName** = "FRANCE.MAP"

Macro Definition Documentation

#define MAX_LENGTH 1024

Responsible for loading map details from the user input file to a data structure which using the list implementation from **list.h**. Map file contain information about cities and it's neighbouring cities. Also implemented path finder which returns lowest path between the cities using A start algorithm

Typedef Documentation

typedef struct City City

The struct holds the information about city in a Map file

typedef struct Neighbour Neighbour

The struct holds information of **Neighbour(name and distance)** city from parent city

Function Documentation

City* createCity ( )

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

Returns

a new (empty) city if memory allocation OK

0 otherwise

Neighbour* createNeighbour ( )

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

Returns

a new (empty) **Neighbour** if memory allocation OK

0 otherwise

void destroyCities (List *)

destroy the list of Cities by deallocation of the the used memory.

Parameters

void destroyCity (City **)

destroy the **City** by deallocation of the the used memory.

Parameters

void destroyNeighbour (Neighbour **)

destroy the **City** by deallocation of the the used memory.

Parameters

```
City* findCity ( List *  ,
                 char *
               )
```

Find **City** details from list bu using comparison function provided during list creation.

Parameters

```
status findPath ( List *  ,
                  char *  ,
                  char *
                )
```

Finds shortest path between two cities using A Star algorithm and display using function provided during list creation .

Parameters

if successor city is on the OPEN list but the existing

if successor city is on the CLOSED list but the existing

```
List* parseMapFile ( char *  )
```

Function loads the Map details about cities and connected neighboring cities to a **List** .

Parameters

Variable Documentation

```
char* mapFileName = "FRANCE.MAP"
```

static