My Project

Generated by Doxygen 1.8.14

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 AStack< ItemType > Class Template Reference

Public Member Functions

- bool isEmpty ()

  checks if stack is empty and returns the empty status
- bool push (const ItemType &newEntry)

  pushes new entry to the stack
- bool pop ()

  pops from the top of stack, checks if stack is not empty returns true, if empty returns false
- ItemType peek ()

  checks the top element

3.1.1 Member Function Documentation

3.1.1.1 isEmpty()

```
template<class ItemType >
bool AStack< ItemType >::isEmpty ( )  [inline]
```

checks if stack is empty and returns the empty status

3.1.1.2 peek()

```
template<class ItemType >
ItemType AStack< ItemType >::peek ( )  [inline]
```

checks the top element

3.1.1.3 pop()

```
template<class ItemType >
bool AStack< ItemType >::pop ( )  [inline]
```

pops from the top of stack, checks if stack is not empty returns true, if empty returns false

3.1.1.4 push()

```
template<class ItemType >
bool AStack< ItemType >::push (
            const ItemType & newEntry )  [inline]
```

pushes new entry to the stack

checks if the push was successful

push failed, return it false

The documentation for this class was generated from the following file:

- Users/renatnorderhaug/Desktop/CS302 project 3/FlightMap.v1.cpp

3.2 Map< ItemType > Class Template Reference

class to represent the flight map, basis of finding flight destinations

Public Member Functions

- Map (int v)
  constructor
- void SetV ()
  method which sets the cities to the Map
- bool checkCity (string City)
  method which checks if the city exists inside the city list
- void addEdge (string v, string w)
  method to add the edges
- void unvisitAll ()
  method that sets all cities as unvisited
- void markVisited (string visit)
  method to mark cities as visited
- string getNextCity (string topCity)
  method to get next unvisited city from the list
- bool isPath (string originCity, string destinationCity)
  the ispath method in the textbook, algorithm for finding the path from original city to destination city
- Map (int v)
- void SetV ()
- bool checkCity (string City)
- void addEdge (string v, string w)
- void unvisitAll ()
- void markVisited (string visit)
- string getNextCity (string topCity)
- bool isPath (string originCity, string destinationCity)

3.2.1 Detailed Description

template<class ItemType>
class Map< ItemType >

class to represent the flight map, basis of finding flight destinations

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Map() [1/2]

```
template<class ItemType >
Map< ItemType >::Map (
            int v )  [inline]
```

constructor

sets the adjacency list

initializes the array of adjacency list pointers

sets pointer to the beginning of the adjacency list

3.2.2.2 Map() [2/2]

```
template<class ItemType >
Map< ItemType >::Map (
            int v )  [inline]
```

3.2.3 Member Function Documentation

3.2.3.1 addEdge() [1/2]

```
template<class ItemType >
void Map< ItemType >::addEdge (
            string v,
            string w )  [inline]
```

3.2.3.2 addEdge() [2/2]

```
template<class ItemType >
void Map< ItemType >::addEdge (
            string v,
            string w )  [inline]
```

method to add the edges

declares variable for position

if city is found in the adjacency list

sets the position of the city in the city list

3.2.3.3 checkCity() [1/2]

```
template<class ItemType >
bool Map< ItemType >::checkCity (
            string City )  [inline]
```

3.2.3.4 checkCity() [2/2]

```
template<class ItemType >
bool Map< ItemType >::checkCity (
            string City )  [inline]
```

method which checks if the city exists inside the city list

3.2.3.5 getNextCity() [1/2]

```
template<class ItemType >
string Map< ItemType >::getNextCity (
            string topCity )  [inline]
```

3.2.3.6 getNextCity() [2/2]

```
template<class ItemType >
string Map< ItemType >::getNextCity (
            string topCity )  [inline]
```

method to get next unvisited city from the list

variable to set position

for loop to traverse city list

find adjacent list of the topcity

sets the position

for loop to traverse the entire adjacency list

for loop to traverse city list

if an unvisited city is found

update pointer

return unvisited city

else return empty value

3.2.3.7 isPath() [1/2]

```
template<class ItemType >
bool Map< ItemType >::isPath (
            string originCity,
            string destinationCity )  [inline]
```

3.2.3.8 isPath() [2/2]

```
template<class ItemType >
bool Map< ItemType >::isPath (
            string originCity,
            string destinationCity )  [inline]
```

the ispath method in the textbook, algorithm for finding the path from original city to destination city

to send to log file

for loop to set adjacency list pointer

pushes origin city onto stack and mark as visited

find top city

while loop repeat while stack becomes empty

string to get the next city

if no city as found as next

pops the stack to backtrack

else visits the city

push city into stack

mark city as visited

if stack is not empty

retrieve top of stack

returns opposite of stack isempty status

3.2.3.9 markVisited() [1/2]

```
template<class ItemType >
void Map< ItemType >::markVisited (
            string visit )  [inline]
```

3.2.3.10 markVisited() [2/2]

```
template<class ItemType >
void Map< ItemType >::markVisited (
            string visit )  [inline]
```

method to mark cities as visited

3.2.3.11 SetV() [1/2]

```
template<class ItemType >
void Map< ItemType >::SetV ( )  [inline]
```

3.2.3.12 SetV() [2/2]

```
template<class ItemType >
void Map< ItemType >::SetV ( )  [inline]
```

method which sets the cities to the Map

temporary variable

object of the class readfiles

while loop to get the names of the cities from file

get name from the file

when reach end of list

sets the city name

sets city as unvisited

increment tem

sets the number of cities

3.2.3.13 unvisitAll() [1/2]

```
template<class ItemType >
void Map< ItemType >::unvisitAll ( )  [inline]
```

3.2.3.14 unvisitAll() [2/2]

```
template<class ItemType >
void Map< ItemType >::unvisitAll ( )  [inline]
```

method that sets all cities as unvisited

The documentation for this class was generated from the following files:

- Users/renatnorderhaug/Desktop/CS302 project 3/FlightMap.v1.cpp
- Users/renatnorderhaug/Desktop/CS302 project 3/FlightMap.v2.cpp

3.3 OurStack< ItemType > Class Template Reference

Public Member Functions

- bool isEmpty ()

  checks if stack is empty and returns the empty status
- bool push (const ItemType &newEntry)

  pushes new entry to the stack
- bool pop ()

  pops from the top of stack, checks if stack is not empty returns true, if empty returns false
- ItemType peek ()

  checks the top element

3.3.1 Member Function Documentation

3.3.1.1 isEmpty()

```
template<class ItemType >
bool OurStack< ItemType >::isEmpty ( )  [inline]
```

checks if stack is empty and returns the empty status

3.3.1.2 peek()

```
template<class ItemType >
ItemType OurStack< ItemType >::peek ( )  [inline]
```

checks the top element

3.3.1.3 pop()

```
template<class ItemType >
bool OurStack< ItemType >::pop ( )  [inline]
```

pops from the top of stack, checks if stack is not empty returns true, if empty returns false

3.3.1.4 push()

```
template<class ItemType >
bool OurStack< ItemType >::push (
            const ItemType & newEntry )  [inline]
```

pushes new entry to the stack

checks if the push was successful

push failed, return it false

The documentation for this class was generated from the following file:

- Users/renatnorderhaug/Desktop/CS302 project 3/FlightMap.v2.cpp

3.4 ReadFiles Class Reference

class to read in the input data, list of cities, list of where flights work, list of where we want to fly to

Public Member Functions

- ReadFiles (string c, string f, string r)
  class constructor which opens files
- string getName ()
  method that gets city name from the file
- template<class ItemType >
  std::pair< ItemType, ItemType > getNamePair ()
  method to return pairs from the flight file
- template<class ItemType >
  std::pair< ItemType, ItemType > getRequestPair ()
  method that returns pairs from request file
- ReadFiles (string c, string f, string r)
- string getName ()
- template<class ItemType >
  std::pair< ItemType, ItemType > getNamePair ()
- template<class ItemType >
  std::pair< ItemType, ItemType > getRequestPair ()

3.4.1 Detailed Description

class to read in the input data, list of cities, list of where flights work, list of where we want to fly to

3.4.2 Constructor & Destructor Documentation

3.4.2.1 ReadFiles() [1/2]

```
ReadFiles::ReadFiles (
            string c,
            string f,
            string r )  [inline]
```

class constructor which opens files

3.4.2.2 ReadFiles() [2/2]

```
ReadFiles::ReadFiles (
            string c,
            string f,
            string r )  [inline]
```

3.4.3 Member Function Documentation

3.4.3.1 getName() [1/2]

```
string ReadFiles::getName ( )  [inline]
```

3.4.3.2 getName() [2/2]

```
string ReadFiles::getName ( )  [inline]
```

method that gets city name from the file

string variable

if there are lines to read from the first file return line

if there are no lines to read return end of file string

3.4.3.3 getNamePair() [1/2]

```
template<class ItemType >
std::pair<ItemType, ItemType> ReadFiles::getNamePair ( )  [inline]
```

3.4.3.4 getNamePair() [2/2]

```
template<class ItemType >
std::pair<ItemType, ItemType> ReadFiles::getNamePair ( )  [inline]
```

method to return pairs from the flight file

initialize string variable and the delimiter string which checks the limit of what to read.

declares a pair

initializes the pair

declare size type variable

if there are lines to read from the second file

retrive the string that is before the delimiter

sets string before the delimiter as first member of pair

sets string after the delimiter as second member of pair

returns pair edge

return null pair if there is no pair able to read.

3.4.3.5 getRequestPair() [1/2]

```
template<class ItemType >
std::pair<ItemType, ItemType> ReadFiles::getRequestPair ( )  [inline]
```

3.4.3.6 getRequestPair() [2/2]

```
template<class ItemType >
std::pair<ItemType, ItemType> ReadFiles::getRequestPair ( )  [inline]
```

method that returns pairs from request file

declair strings to read in file data and delimiter

declares pair

if there is lines to read from the file

retrieve substring before delimiter

sets string as first member of pair

The documentation for this class was generated from the following files:

- Users/renatnorderhaug/Desktop/CS302 project 3/FlightMap.v1.cpp
- Users/renatnorderhaug/Desktop/CS302 project 3/FlightMap.v2.cpp

3.5 StackInterface< ItemType > Class Template Reference

interface definition for stack

3.5.1 Detailed Description

template<class ItemType>
class StackInterface< ItemType >

interface definition for stack

The documentation for this class was generated from the following files:

- Users/renatnorderhaug/Desktop/CS302 project 3/FlightMap.v1.cpp
- Users/renatnorderhaug/Desktop/CS302 project 3/FlightMap.v2.cpp

Chapter 4

File Documentation

4.1 Users/renatnorderhaug/Desktop/CS302 project 3/FlightMap.v1.cpp File Reference

```
#include <vector>
#include <string>
#include <fstream>
#include <stack>
#include <map>
#include <iostream>
#include <list>
```

Classes

- class StackInterface< ItemType >

  interface definition for stack
- class AStack< ItemType >
- class ReadFiles

  class to read in the input data, list of cities, list of where flights work, list of where we want to fly to
- class Map< ItemType >

  class to represent the flight map, basis of finding flight destinations

4.2 Users/renatnorderhaug/Desktop/CS302 project 3/FlightMap.v2.cpp File Reference

```
#include <vector>
#include <string>
#include <fstream>
#include <stack>
#include <map>
#include <iostream>
#include <list>
```

Classes

- class StackInterface< ItemType >

  interface definition for stack
- class OurStack< ItemType >
- class ReadFiles

  class to read in the input data, list of cities, list of where flights work, list of where we want to fly to
- class Map< ItemType >

  class to represent the flight map, basis of finding flight destinations

4.3 Users/renatnorderhaug/Desktop/CS302 project 3/PA03.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <vector>
#include <map>
#include <sstream>
#include <list>
#include <stack>
#include "FlightMap.v2.cpp"
```

Functions

- int main ()

  main.cpp

4.3.1 Function Documentation

4.3.1.1 main()

```
int main ( )
```

main.cpp

main.cpp Project 3 Created by Renat Norderhaug on 2/27/18. Copyright © 2018 Renat Norderhaug. All rights reserved.tester file for the Flightmap declare pair variables

object of class readfiles parameter: the names of the files it reads returns: the data inside the files pre: none post: files read

object of class map as datatype string

parameter: an integer value regarding what cities in map returns: the cities in the map pre: empty map post: cities from the files into the map

while loop which sets flight paths in the adjacency list

assign the pair value

if end of file is reached

break the loop

adds the edge onto the adjacency list

the while loop to serve the requests

sets the request pair

if request is Empty

break loop

display the request

if first city is not in city list

display the result

if their is a path that exists

display result

if there is no path that exists

display result

Index