MyVensim

Generated by Doxygen 1.9.6

Chapter 1

bcc-322

Código referente ao trabalho prático desenvolvido para a disciplina Engenharia de Software I.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

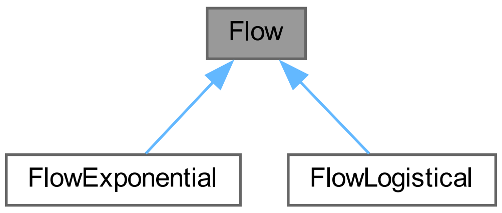
Class Documentation

5.1 Flow Class Reference

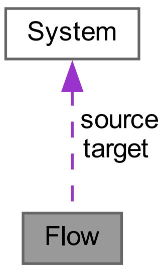
Define the interface with the methods to be implemented.

`#include <flow.h>`

Inheritance diagram for Flow:



Collaboration diagram for Flow:



Public Member Functions

- Flow ()

  Construct a new Flow object.
- Flow (Flow &obj)

  Copy a Flow object.
- Flow (const string name, System ∗source, System ∗target)

  Construct a new Flow object.
- virtual ∼Flow ()

  Destroy the Flow object.
- string getName () const

  Get the Name object.
- void setName (const string name)

  Set the Name object.
- System ∗ getSource () const

  Get the Source object.
- void setSource (System ∗source)

  Set the Source object.
- System ∗ getTarget () const

  Get the Target object.
- void setTarget (System ∗target)

  Set the Target object.
- bool operator== (const Flow &obj) const
- bool operator!= (const Flow &obj) const
- Flow & operator= (const Flow &obj)

  Overload the '=' operator, cloning from one to the other.
- virtual float execute ()=0

  Purely virtual method to be inherited by subclasses created by the user.

Protected Attributes

- string name

  Name the flow.
- System ∗ source

  Assign the flow's source.
- System ∗ target

  Assign the flow's target.

5.1.1 Detailed Description

Define the interface with the methods to be implemented.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Flow() [1/3]

```
Flow::Flow ( )
```

Construct a new Flow object.

5.1.2.2 Flow() [2/3]

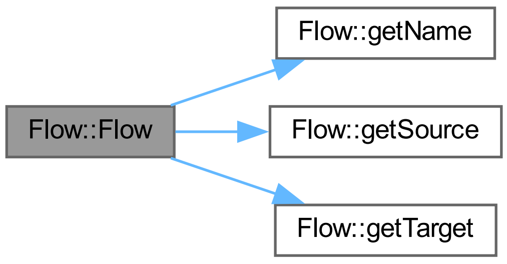
```
Flow::Flow (
            Flow & obj )
```

Copy a Flow object.

Parameters

obj	flow to be copied

Here is the call graph for this function:



5.1.2.3 Flow() [3/3]

```
Flow::Flow (
            const string name,
            System * source,
            System * target )
```

Construct a new Flow object.

Parameters

name	of the flow
source	system
target	system

5.1.2.4 ∼Flow()

```
Flow::∼Flow ( )  [virtual]
```

Destroy the Flow object.

5.1.3 Member Function Documentation

5.1.3.1 execute()

```
virtual float Flow::execute ( )  [pure virtual]
```

Purely virtual method to be inherited by subclasses created by the user.

Returns

float

Implemented in FlowExponential, and FlowLogistical.

5.1.3.2 getName()

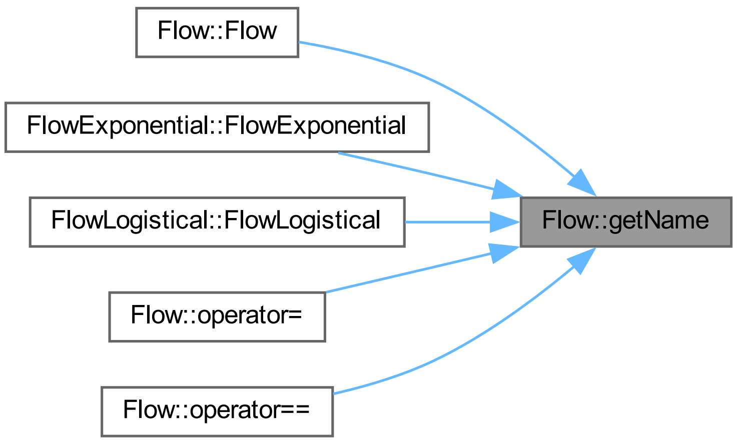
```
string Flow::getName ( ) const
```

Get the Name object.

Returns

string The name of a flow

Here is the caller graph for this function:


Flow::Flow
FlowExponential::FlowExponential
FlowLogistical::FlowLogistical
Flow::operator=
Flow::operator==
Flow::getName

5.1.3.3 getSource()

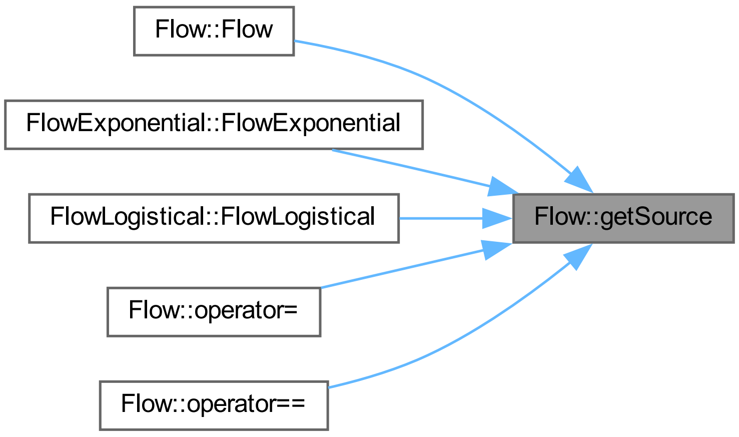
```
System * Flow::getSource ( ) const
```

Get the Source object.

Returns

System∗ The system that acts as a source for the flow

Here is the caller graph for this function:


Flow::Flow
FlowExponential::FlowExponential
FlowLogistical::FlowLogistical
Flow::operator=
Flow::operator==
Flow::getSource

5.1.3.4 getTarget()

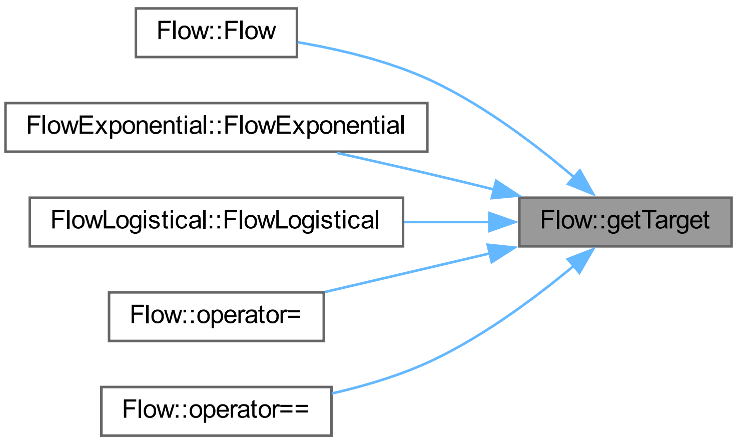
```
System * Flow::getTarget ( ) const
```

Get the Target object.

Returns

System∗ The system that acts as a target for the flow

Here is the caller graph for this function:


Flow::Flow
FlowExponential::FlowExponential
FlowLogistical::FlowLogistical
Flow::operator=
Flow::operator==
Flow::getTarget

5.1.3.5 operator"!=()

```
bool Flow::operator!= (
             const Flow & obj ) const
```

5.1.3.6 operator=()

```
Flow & Flow::operator= (
             const Flow & obj )
```

Overload the '=' operator, cloning from one to the other.

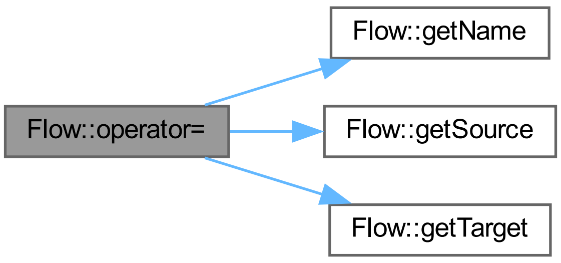
Parameters

obj	flow to be cloned

Returns

Flow& A clone of the flow

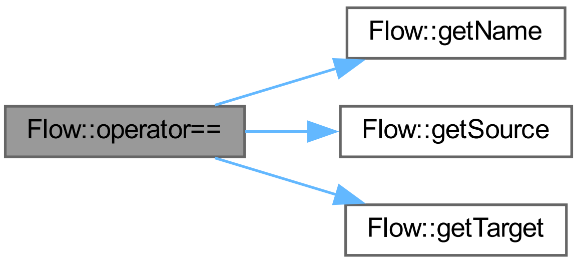
Here is the call graph for this function:



5.1.3.7 operator==()

```
bool Flow::operator== (
             const Flow & obj ) const
```

Here is the call graph for this function:



5.1.3.8 setName()

```
void Flow::setName (
             const string name )
```

Set the Name object.

Parameters

name	the flow

5.1.3.9 setSource()

```
void Flow::setSource (
             System * source )
```

Set the Source object.

Parameters

source	system

5.1.3.10 setTarget()

```
void Flow::setTarget (
             System * target )
```

Set the Target object.

Parameters

target	system

5.1.4 Member Data Documentation

5.1.4.1 name

```
string Flow::name  [protected]
```

Name the flow.

5.1.4.2 source

```
System* Flow::source  [protected]
```

Assign the flow's source.

5.1.4.3 target

```
System* Flow::target  [protected]
```

Assign the flow's target.

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

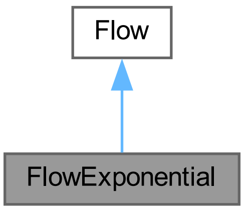
- src/flow.h
- src/flow.cpp

5.2 FlowExponential Class Reference

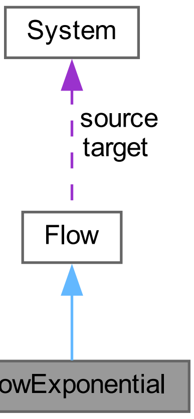
This flow class connects two systems to evaluate their final values after running an equation for a given time.

```
#include <flowExponential.h>
```

Inheritance diagram for FlowExponential:



Collaboration diagram for FlowExponential:



Public Member Functions

- FlowExponential ()
  Construct a new Flow Exponential object.
- FlowExponential (Flow &obj)
  Copy a Flow Exponential object.
- FlowExponential (const string name, System ∗source, System ∗target)
  Construct a new Flow Exponential object.
- virtual ∼FlowExponential ()
  Destroy the Flow Exponential object.
- virtual float execute ()
  Run the flow's equation.

Public Member Functions inherited from Flow

- Flow ()
  Construct a new Flow object.
- Flow (Flow &obj)
  Copy a Flow object.
- Flow (const string name, System ∗source, System ∗target)
  Construct a new Flow object.
- virtual ∼Flow ()
  Destroy the Flow object.
- string getName () const
  Get the Name object.
- void setName (const string name)
  Set the Name object.
- System ∗ getSource () const
  Get the Source object.
- void setSource (System ∗source)
  Set the Source object.
- System ∗ getTarget () const
  Get the Target object.
- void setTarget (System ∗target)
  Set the Target object.
- bool operator== (const Flow &obj) const
- bool operator!= (const Flow &obj) const
- Flow & operator= (const Flow &obj)
  Overload the '=' operator, cloning from one to the other.
- virtual float execute ()=0
  Purely virtual method to be inherited by subclasses created by the user.

Additional Inherited Members

Protected Attributes inherited from Flow

- string name
  Name the flow.
- System ∗ source
  Assign the flow's source.
- System ∗ target
  Assign the flow's target.

5.2.1 Detailed Description

This flow class connects two systems to evaluate their final values after running an equation for a given time.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 FlowExponential() [1/3]

```
FlowExponential::FlowExponential ( )
```

Construct a new Flow Exponential object.

5.2.2.2 FlowExponential() [2/3]

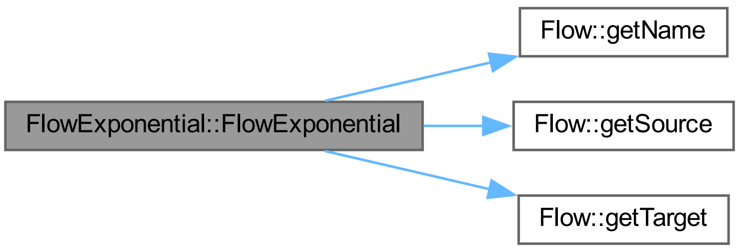
```
FlowExponential::FlowExponential (
             Flow & obj )
```

Copy a Flow Exponential object.

Parameters

obj	flow to be copied

Here is the call graph for this function:



5.2.2.3 FlowExponential() [3/3]

```
FlowExponential::FlowExponential (
            const string name,
            System * source,
            System * target )
```

Construct a new Flow Exponential object.

Parameters

name	of the flow
source	system
target	system

5.2.2.4 ∼FlowExponential()

```
FlowExponential::∼FlowExponential ( ) [virtual]
```

Destroy the Flow Exponential object.

5.2.3 Member Function Documentation

5.2.3.1 execute()

```
float FlowExponential::execute ( ) [virtual]
```

Run the flow's equation.

Returns

float The result of the calculations

Implements Flow.

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

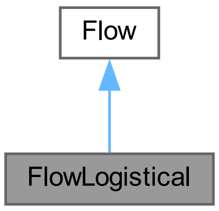
- test/funcional/flowExponential.h
- test/funcional/flowExponential.cpp

5.3 FlowLogistical Class Reference

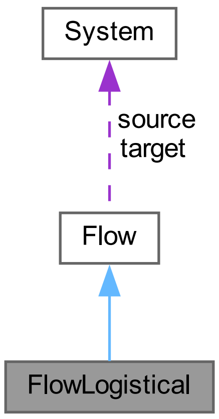
This flow class connects two systems to evaluate their final value after running an equation for a given time.

```
#include <flowLogistical.h>
```

Inheritance diagram for FlowLogistical:



Collaboration diagram for FlowLogistical:



Public Member Functions

- FlowLogistical ()

  Construct a new Flow Logistical object.
- FlowLogistical (Flow &obj)

  Construct a new Flow Logistical object.
- FlowLogistical (const string name, System ∗source, System ∗target)

  Construct a new Flow Logistical object.
- virtual ∼FlowLogistical ()

  Destroy the Flow Logistical object.
- virtual float execute ()

  Run the flow's equation.

Public Member Functions inherited from Flow

- Flow ()

  Construct a new Flow object.
- Flow (Flow &obj)

  Copy a Flow object.
- Flow (const string name, System ∗source, System ∗target)

  Construct a new Flow object.
- virtual ∼Flow ()

  Destroy the Flow object.
- string getName () const

  Get the Name object.
- void setName (const string name)

  Set the Name object.
- System ∗ getSource () const

  Get the Source object.
- void setSource (System ∗source)

  Set the Source object.
- System ∗ getTarget () const

  Get the Target object.
- void setTarget (System ∗target)

  Set the Target object.
- bool operator== (const Flow &obj) const
- bool operator!= (const Flow &obj) const
- Flow & operator= (const Flow &obj)

  Overload the '=' operator, cloning from one to the other.
- virtual float execute ()=0

  Purely virtual method to be inherited by subclasses created by the user.

Additional Inherited Members

Protected Attributes inherited from Flow

- string name

  Name the flow.
- System ∗ source

  Assign the flow's source.
- System ∗ target

  Assign the flow's target.

5.3.1 Detailed Description

This flow class connects two systems to evaluate their final value after running an equation for a given time.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 FlowLogistical() [1/3]

```
FlowLogistical::FlowLogistical ( )
```

Construct a new Flow Logistical object.

5.3.2.2 FlowLogistical() [2/3]

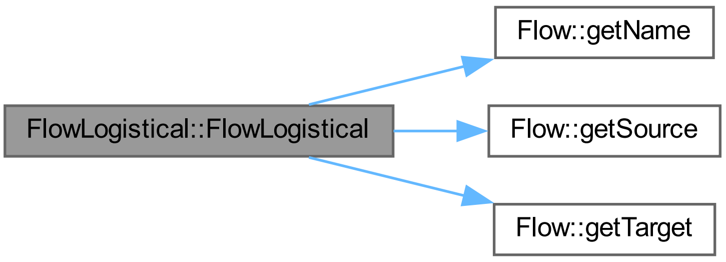
```
FlowLogistical::FlowLogistical (
                Flow & obj )
```

Construct a new Flow Logistical object.

Parameters

obj	flow to be copied

Here is the call graph for this function:



5.3.2.3 FlowLogistical() [3/3]

```
FlowLogistical::FlowLogistical (
                const string name,
                System * source,
                System * target )
```

Construct a new Flow Logistical object.

Parameters

name	of the flow
source	system
target	system

5.3.2.4 ∼FlowLogistical()

```
FlowLogistical::∼FlowLogistical ( )  [virtual]
```

Destroy the Flow Logistical object.

5.3.3 Member Function Documentation

5.3.3.1 execute()

```
float FlowLogistical::execute ( )  [virtual]
```

Run the flow's equation.

Returns

float The result of the calculations

Implements Flow.

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

- test/funcional/flowLogistical.h
- test/funcional/flowLogistical.cpp

5.4 Model Class Reference

Store vectors containing the name of the model and flows and systems related to it.

```
#include <model.h>
```

Public Types

- typedef vector< Flow ∗ >::iterator itFlow

  Set the flow vector type.
- typedef vector< System ∗ >::iterator itSystem

  Set the system vector type.

Public Member Functions

- Model ()

  Empty constructor of the class.
- Model (const string name)

  Construct a new Model object.
- Model (const string name, vector< Flow ∗ > &flows, vector< System ∗ > &systems)

  Construct a new Model object.
- virtual ∼Model ()

  Destroy the Model object.
- string getName () const

  Get the Name object.
- void setName (const string name)

  Set the Name object.
- itFlow getFlowBegin ()

  Get the flow from the beginning of the vector.
- itFlow getFlowEnd ()

  Get the flow from the end of the vector.
- int getFlowSize ()

  Get the size of the flow vector.
- itSystem getSystemBegin ()

  Get the system from the beginning of the vector.
- itSystem getSystemEnd ()

  Get the system from the end of the vector.
- int getSystemSize ()

  Get the size of the system vector.
- void add (System ∗)

  Add a system to the model.
- void add (Flow ∗)

  Add a flow to the model.
- bool remove (System ∗)

  Remove a system from the model.
- bool remove (Flow ∗)

  Remove a flow from the model.
- void clear ()

  Clean the model.
- void show ()

  Display the model.
- void run (int, int, int)

  Execute the model based on initial time, end time and time intervals.

Protected Attributes

- string name

  Name of the model.
- vector< Flow ∗ > flows

  Store an array of pointer-to-flow variables.
- vector< System ∗ > systems

  Store an array of pointer-to-system variables.

5.4.1 Detailed Description

Store vectors containing the name of the model and flows and systems related to it.

5.4.2 Member Typedef Documentation

5.4.2.1 itFlow

```
typedef vector<Flow*>::iterator Model::itFlow
```

Set the flow vector type.

5.4.2.2 itSystem

```
typedef vector<System*>::iterator Model::itSystem
```

Set the system vector type.

5.4.3 Constructor & Destructor Documentation

5.4.3.1 Model() [1/3]

```
Model::Model ( )
```

Empty constructor of the class.

5.4.3.2 Model() [2/3]

```
Model::Model (
              const string name )
```

Construct a new Model object.

Parameters

name	of the model

5.4.3.3 Model() [3/3]

```
Model::Model (
             const string name,
             vector< Flow * > & flows,
             vector< System * > & systems )
```

Construct a new Model object.

Parameters

name	of the model
flows	array of pointer-to-flow variables
systems	array of pointer-to-system variables

5.4.3.4 ∼Model()

```
Model::∼Model ( )  [virtual]
```

Destroy the Model object.

5.4.4 Member Function Documentation

5.4.4.1 add() [1/2]

```
void Model::add (
             Flow * flow )
```

Add a flow to the model.

Parameters

pointer	to a flow

5.4.4.2 add() [2/2]

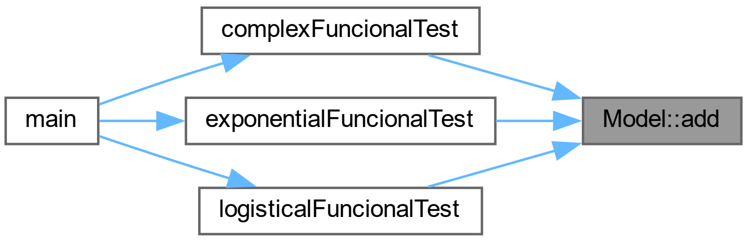
```
void Model::add (
             System * subSystem )
```

Add a system to the model.

Parameters

pointer	to a system

Here is the caller graph for this function:

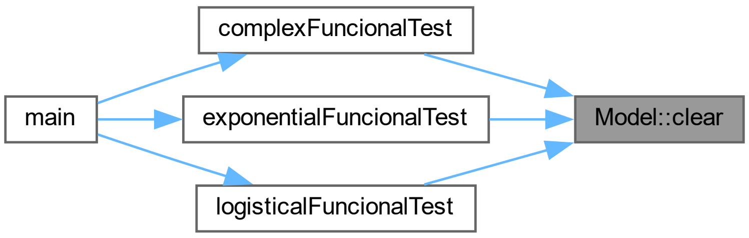


5.4.4.3 clear()

```
void Model::clear ( )
```

Clean the model.

Here is the caller graph for this function:



5.4.4.4 getFlowBegin()

```
Model::itFlow Model::getFlowBegin ( )
```

Get the flow from the beginning of the vector.

Returns

itFlow The flow from the beginning

5.4.4.5 getFlowEnd()

```
Model::itFlow Model::getFlowEnd ( )
```

Get the flow from the end of the vector.

Returns

itFlow The flow from the end

5.4.4.6 getFlowSize()

```
int Model::getFlowSize ( )
```

Get the size of the flow vector.

Returns

int The size of the flow vector

5.4.4.7 getName()

```
string Model::getName ( ) const
```

Get the Name object.

Returns

string The name of a model

5.4.4.8 getSystemBegin()

```
Model::itSystem Model::getSystemBegin ( )
```

Get the system from the beginning of the vector.

Returns

itSystem The system from the beginning

5.4.4.9 getSystemEnd()

```
Model::itSystem Model::getSystemEnd ( )
```

Get the system from the end of the vector.

Returns

itSystem The system from the end

5.4.4.10 getSystemSize()

```
int Model::getSystemSize ( )
```

Get the size of the system vector.

Returns

int The size of the system vector

5.4.4.11 remove() [1/2]

```
bool Model::remove (
             Flow * obj )
```

Remove a flow from the model.

Returns

true If the object and item have the same memory address

false If the object and item have different memory addresses

5.4.4.12 remove() `[2/2]`

```
bool Model::remove (
             System * obj )
```

Remove a system from the model.

Returns

true If the object and item have the same memory address

false If the object and item have different memory addresses

5.4.4.13 run()

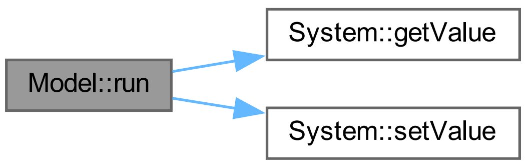
```
void Model::run (
             int start,
             int finish,
             int increment )
```

Execute the model based on initial time, end time and time intervals.

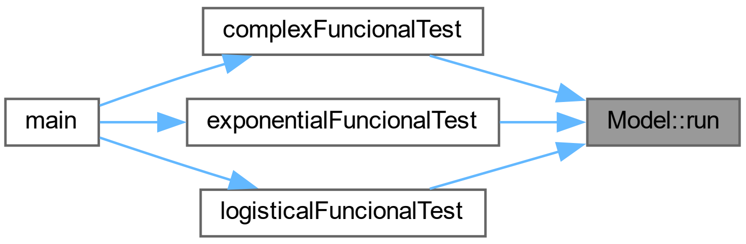
Parameters

int	start time
int	end time
int	how many units of time shall pass between one execution and the next

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.4.14 setName()

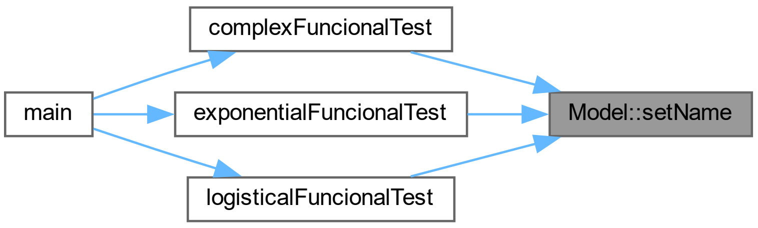
```
void Model::setName (
             const string name )
```

Set the Name object.

Parameters

name	the model

Here is the caller graph for this function:

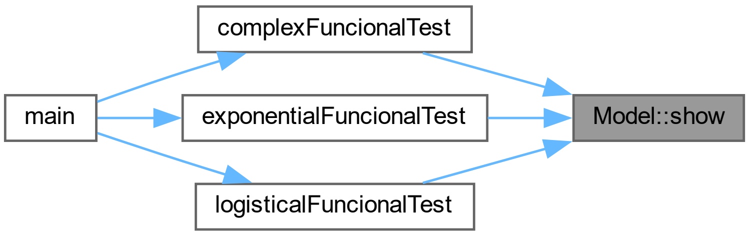


5.4.4.15 show()

```
void Model::show ( )
```

Display the model.

Here is the caller graph for this function:



5.4.5 Member Data Documentation

5.4.5.1 flows

```
vector<Flow*> Model::flows  [protected]
```

Store an array of pointer-to-flow variables.

5.4.5.2 name

```
string Model::name  [protected]
```

Name of the model.

5.4.5.3 systems

```
vector<System*> Model::systems  [protected]
```

Store an array of pointer-to-system variables.

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

- src/model.h
- src/model.cpp

5.5 System Class Reference

Define the interface with the methods to be implemented.

```
#include <system.h>
```

Public Member Functions

- System ()

  Empty constructor of the class.
- System (const string name)

  Construct a new System object.
- System (float value)

  Construct a new System object.
- System (System &obj)

  Copy a System object.
- System (const string name, float value)

  Assigns name and value to a system.
- virtual ∼System ()

  Destroy the System object.
- string getName () const

  Get the Name object.
- void setName (const string name)

  Set the Name object.
- float getValue () const

  Get the Value object.
- void setValue (float value)

  Set the Value object.
- System & operator= (const System &obj)

  Overload the '=' operator, cloning from one to the other.

Protected Attributes

- string name

  Name the system.
- float value

  Store a value for the system.

5.5.1 Detailed Description

Define the interface with the methods to be implemented.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 System() [1/5]

```
System::System ( )
```

Empty constructor of the class.

5.5.2.2 System() [2/5]

```
System::System (
             const string name )
```

Construct a new System object.

Parameters

name	of the system

5.5.2.3 System() [3/5]

```
System::System (
             float value )
```

Construct a new System object.

Parameters

value	to be contained in that system

5.5.2.4 System() [4/5]

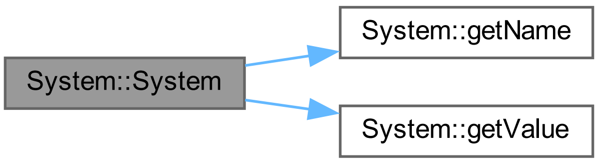
```
System::System (
             System & obj )
```

Copy a System object.

Parameters

obj	system to be copied

Here is the call graph for this function:



5.5.2.5 System() [5/5]

```
System::System (
              const string name,
              float value )
```

Assigns name and value to a system.

Parameters

name	assigned to the system
value	assigned to the system

5.5.2.6 ∼System()

```
System::∼System ( )  [virtual]
```

Destroy the System object.

5.5.3 Member Function Documentation

5.5.3.1 getName()

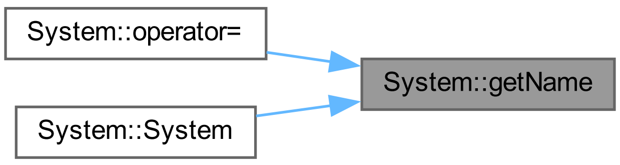
```
string System::getName ( ) const
```

Get the Name object.

Returns

string The name of a system

Here is the caller graph for this function:



5.5.3.2 getValue()

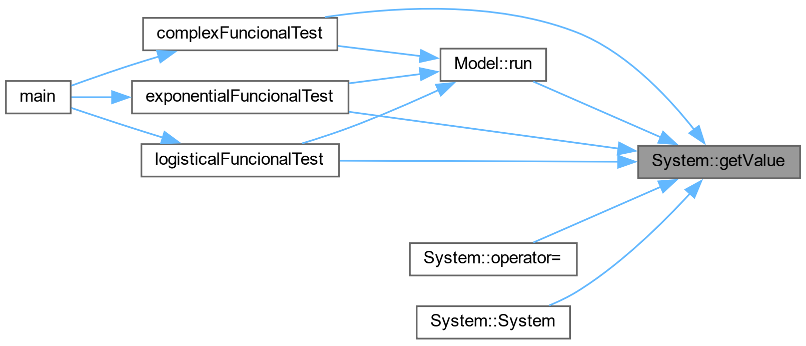
```
float System::getValue ( ) const
```

Get the Value object.

Returns

float Value assigned to a system

Here is the caller graph for this function:



5.5.3.3 operator=()

```
System & System::operator= (
             const System & obj )
```

Overload the '=' operator, cloning from one to the other.

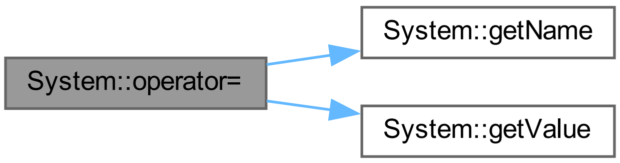
Parameters

obj	system to be cloned

Returns

System& A clone of the system

Here is the call graph for this function:



5.5.3.4 setName()

```
void System::setName (
            const string name )
```

Set the Name object.

Parameters

name	the system

5.5.3.5 setValue()

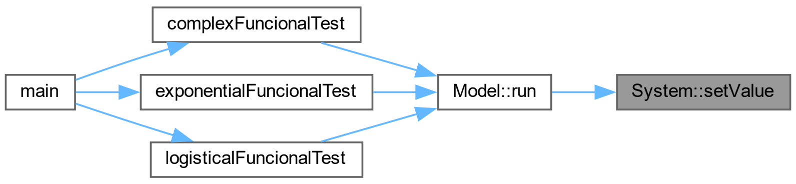
```
void System::setValue (
            float value )
```

Set the Value object.

Parameters

value	float value to be assigned to a system

Here is the caller graph for this function:



5.5.4 Member Data Documentation

5.5.4.1 name

```
string System::name [protected]
```

Name the system.

5.5.4.2 value

```
float System::value [protected]
```

Store a value for the system.

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

- src/system.h
- src/system.cpp

Chapter 6

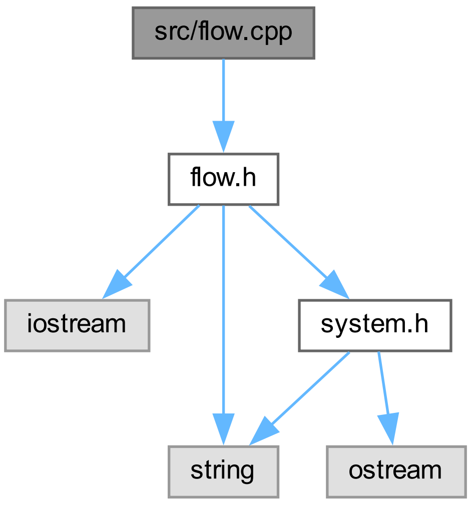
File Documentation

6.1 README.md File Reference

6.2 src/flow.cpp File Reference

```
#include "flow.h"
```

Include dependency graph for flow.cpp:

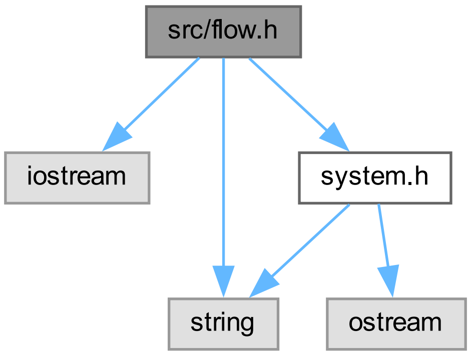


6.3 src/flow.h File Reference

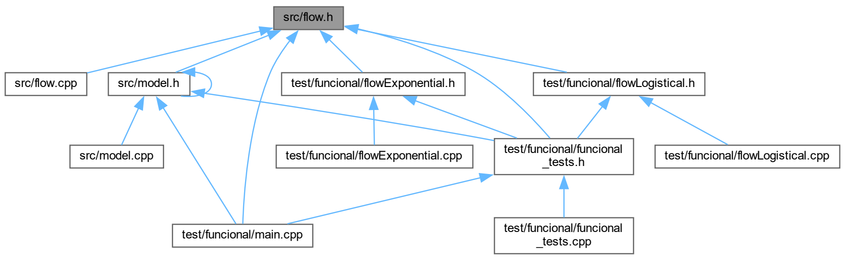
Contains the specifications of the flow class.

```
#include <iostream>
#include <string>
#include "system.h"
```

Include dependency graph for flow.h:



This graph shows which files directly or indirectly include this file:


src/flow.h
src/flow.cpp
src/model.h
test/funcional/flowExponential.h
test/funcional/flowLogistical.h
src/model.cpp
test/funcional/flowExponential.cpp
test/funcional/funcional_tests.h
test/funcional/flowLogistical.cpp
test/funcional/main.cpp
test/funcional/funcional_tests.cpp

Classes

- class Flow

  Define the interface with the methods to be implemented.

6.3.1 Detailed Description

Contains the specifications of the flow class.

Author

Iago Andrade (`iago.andrade@aluno.ufop.edu.br`)

Copyright

Copyright (c) 2023

6.4 flow.h

Go to the documentation of this file.

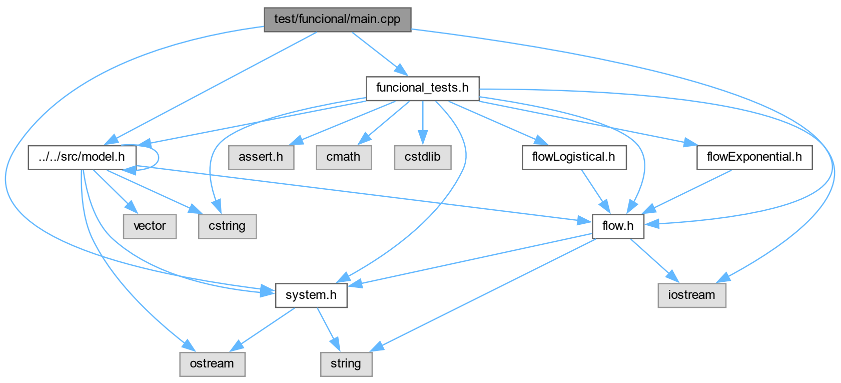
```
#ifndef FLOW_H
#define FLOW_H

#include <iostream>
#include <string>
#include "system.h"

class Flow {
    protected:
        string name;
        System *source;
        System *target;
    public:
        Flow();
        Flow(Flow &obj);
        Flow(const string name, System *source, System *target);
        virtual ~Flow();

        string getName() const;
        void setName(const string name);
        System *getSource() const;
        void setSource(System *source);
        System *getTarget() const;
        void setTarget(System *target);

        bool operator==(const Flow &obj) const;
        bool operator!=(const Flow &obj) const;
        Flow &operator= (const Flow &obj);
        virtual float execute() = 0;
};

#endif
```

6.5 src/main.cpp File Reference

6.6 test/funcional/main.cpp File Reference

```
#include "funcional_tests.h"
#include "..\..\src\model.h"
#include "..\..\src\system.h"
#include "..\..\src\flow.h"
```

Include dependency graph for main.cpp:



Macros

- #define MAIN_FUNCIONAL_TESTS

Functions

- int main ()

6.6.1 Macro Definition Documentation

6.6.1.1 MAIN_FUNCIONAL_TESTS

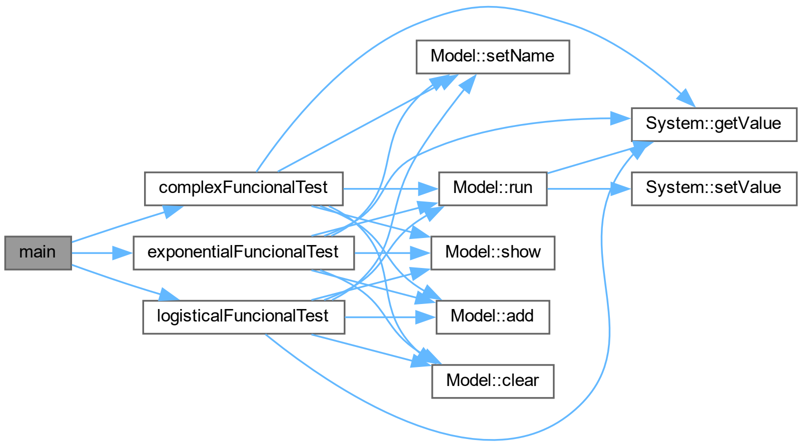
```
#define MAIN_FUNCIONAL_TESTS
```

6.6.2 Function Documentation

6.6.2.1 main()

```
int main ( )
```

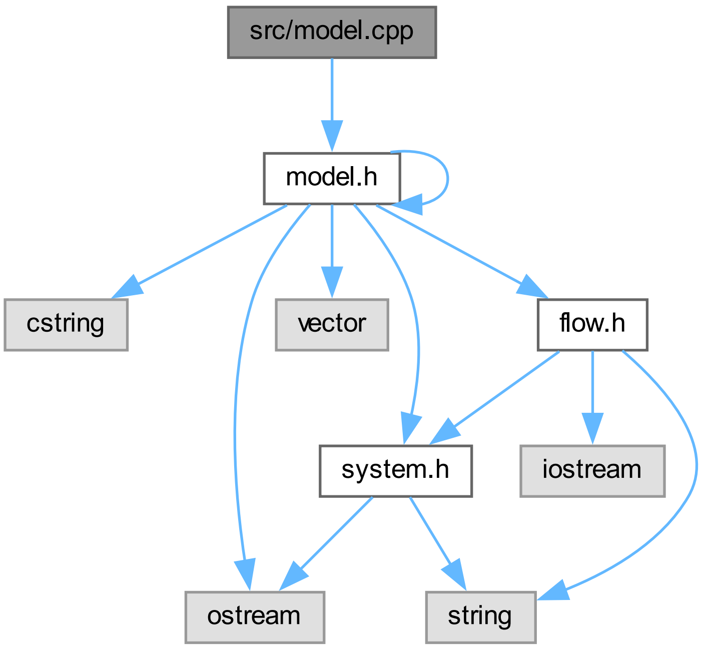
Here is the call graph for this function:



6.7 src/model.cpp File Reference

```
#include "model.h"
```

Include dependency graph for model.cpp:

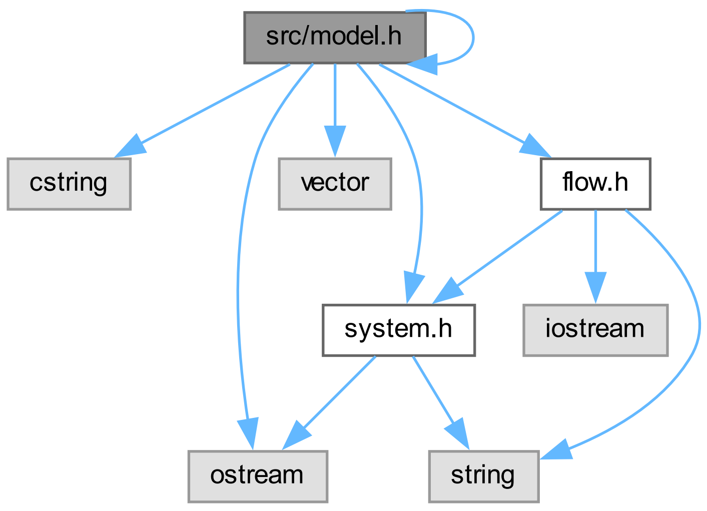


6.8 src/model.h File Reference

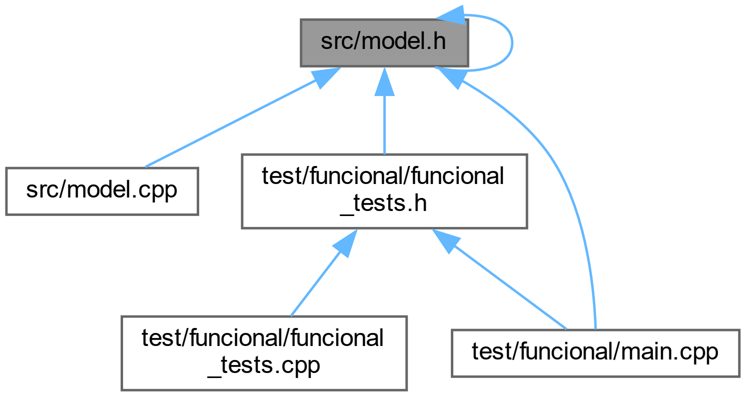
Represents the simulation model.

```
#include <cstring>
#include <ostream>
#include <vector>
#include "flow.h"
#include "system.h"
#include "model.h"
```

Include dependency graph for model.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Model

  Store vectors containing the name of the model and flows and systems related to it.

6.8.1 Detailed Description

Represents the simulation model.

Author

Iago Andrade (iago.andrade@aluno.ufop.edu.br)

Copyright

Copyright (c) 2023

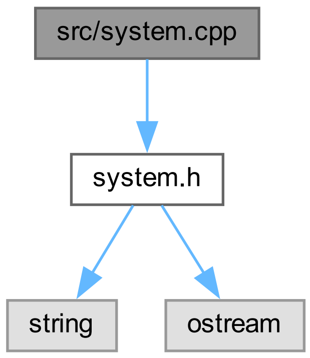
6.9 model.h

Go to the documentation of this file.

```
#ifndef MODEL_H
#define MODEL_H
#include <cstring>
#include <ostream>
#include <vector>
#include "flow.h"
#include "system.h"
#include "model.h"

class Model {
    protected:
        string name;
        vector<Flow*> flows;
        vector<System*> systems;

    private:
        Model(Model& obj);
        Model& operator= (const Model& obj);

    public:
        Model();
        Model(const string name);
        Model(const string name, vector<Flow*> &flows, vector<System*> &systems);
        virtual ~Model();

        typedef typename vector<Flow*> :: iterator itFlow;
        typedef typename vector<System*> :: iterator itSystem;

        string getName() const;
        void setName(const string name);

        itFlow getFlowBegin();
        itFlow getFlowEnd();
        int getFlowSize();

        itSystem getSystemBegin();
        itSystem getSystemEnd();
        int getSystemSize();

        void add(System*);
        void add(Flow*);
        bool remove(System*);
        bool remove(Flow*);
        void clear();
        void show();
        void run(int, int, int);
};

#endif
```

6.10 src/system.cpp File Reference

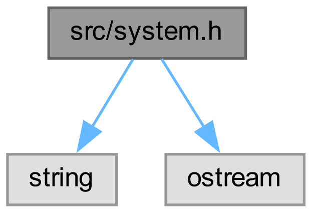
```
#include "system.h"
```
Include dependency graph for system.cpp:



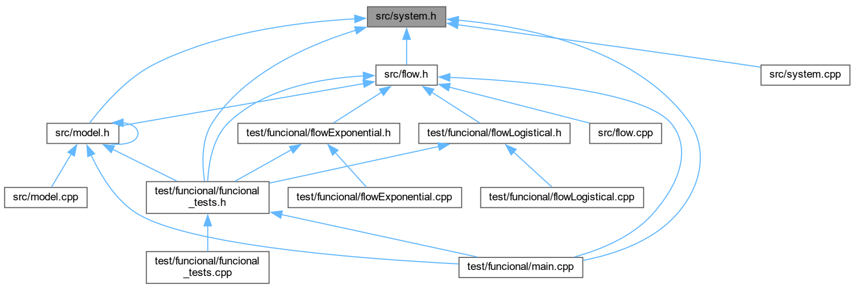
6.11 src/system.h File Reference

Describes the system class.

```
#include <string>
#include <ostream>
```
Include dependency graph for system.h:



This graph shows which files directly or indirectly include this file:


src/system.h
src/flow.h
src/system.cpp
src/model.h
test/funcional/flowExponential.h
test/funcional/flowLogistical.h
src/flow.cpp
src/model.cpp
test/funcional/funcional_tests.h
test/funcional/flowExponential.cpp
test/funcional/flowLogistical.cpp
test/funcional/funcional_tests.cpp
test/funcional/main.cpp

Classes

- class System

  Define the interface with the methods to be implemented.

6.11.1 Detailed Description

Describes the system class.

Author

Iago Andrade (iago.andrade@aluno.ufop.edu.br)

Copyright

Copyright (c) 2023

6.12 system.h

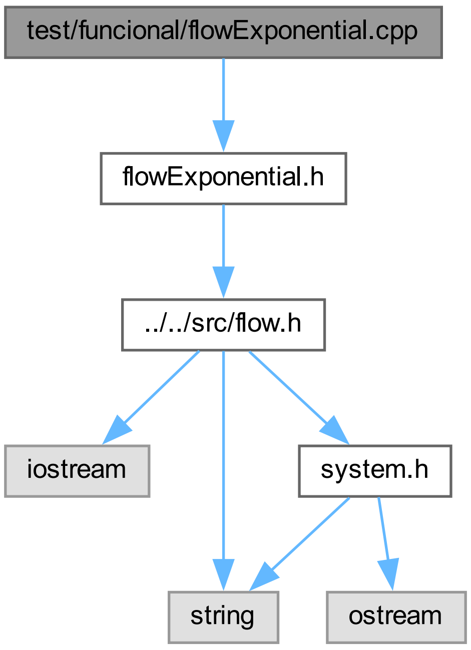
Go to the documentation of this file.

```

#ifndef SYSTEM_H
#define SYSTEM_H

#include <string>
#include <ostream>

using namespace std;

class System {
    protected:
        string name;
        float value;

    public:
        System();
        System(const string name);
        System(float value);
        System(System& obj);
        System(const string name, float value);
        virtual ~System();

        string getName() const;
        void setName(const string name);
        float getValue() const;
        void setValue(float value);

        System& operator= (const System& obj);
};

#endif
```

6.13 test/funcional/flowExponential.cpp File Reference

`#include "flowExponential.h"`

Include dependency graph for flowExponential.cpp:

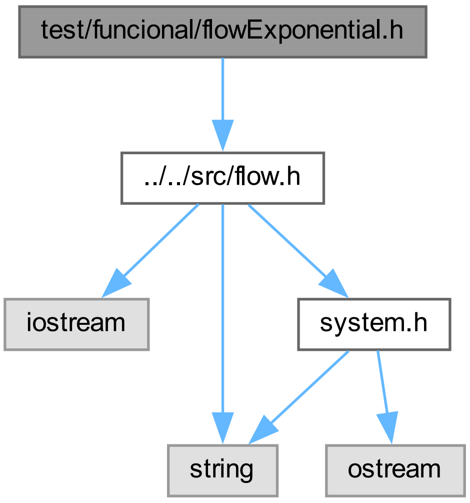


6.14 test/funcional/flowExponential.h File Reference

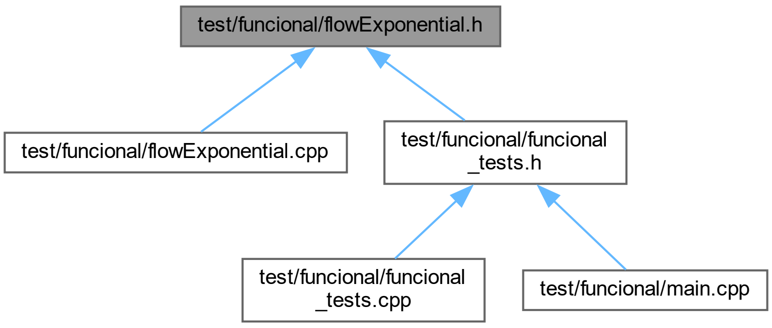
Contains the code used to run the functional exponential tests.

```
#include "../../src/flow.h"
```

Include dependency graph for flowExponential.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FlowExponential

  This flow class connects two systems to evaluate their final values after running an equation for a given time.

6.14.1 Detailed Description

Contains the code used to run the functional exponential tests.

Contains the code used to run the functional logistical tests.

Author

Iago Andrade (iago.andrade@aluno.ufop.edu.br)

Copyright

6.15 flowExponential.h

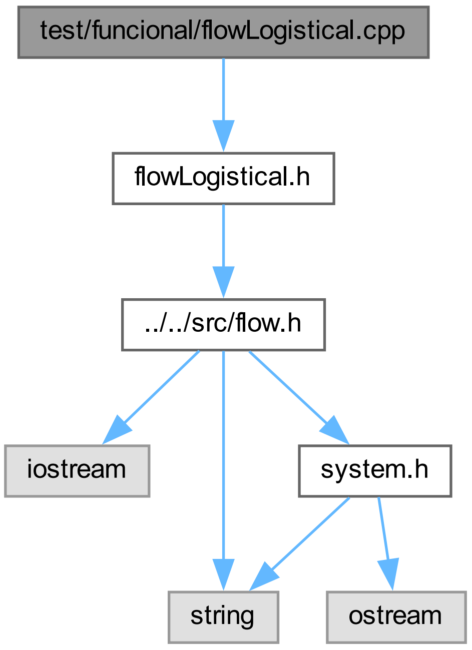
Go to the documentation of this file.

```
#ifndef FLOWEXPONENTIAL_H
#define FLOWEXPONENTIAL_H

#include "../../src/flow.h"

class FlowExponential : public Flow {
    public:
        FlowExponential();
        FlowExponential(Flow &obj);
        FlowExponential(const string name, System *source, System *target);
        virtual ~FlowExponential();
        virtual float execute();
};

#endif
```

6.16 test/funcional/flowLogistical.cpp File Reference

`#include "flowLogistical.h"`

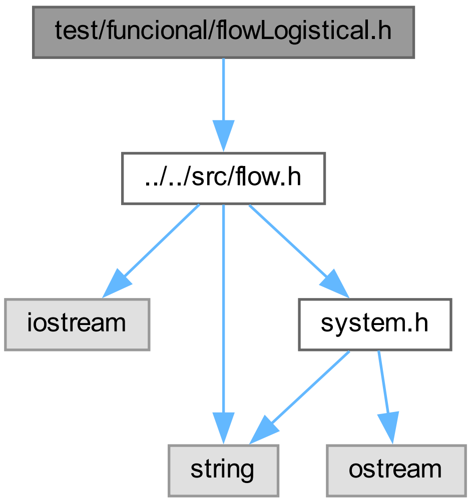
Include dependency graph for flowLogistical.cpp:



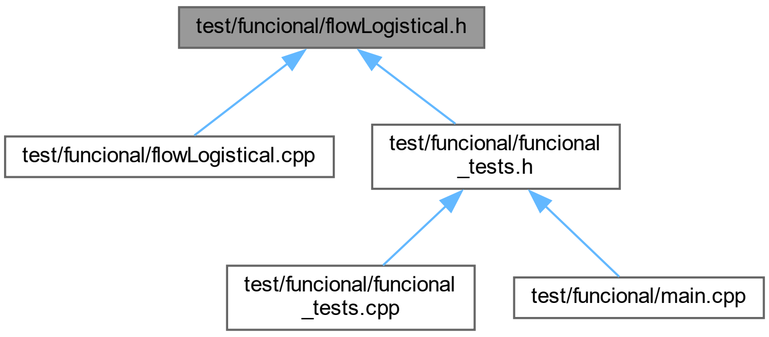
6.17 test/funcional/flowLogistical.h File Reference

```
#include "../../src/flow.h"
```

Include dependency graph for flowLogistical.h:



This graph shows which files directly or indirectly include this file:


test/funcional/flowLogistical.h
test/funcional/flowLogistical.cpp
test/funcional/funcional _tests.h
test/funcional/funcional _tests.cpp
test/funcional/main.cpp

Classes

- class FlowLogistical

    This flow class connects two systems to evaluate their final value after running an equation for a given time.

Macros

- #define FLOWLOGISTIC_H

6.17.1 Macro Definition Documentation

6.17.1.1 FLOWLOGISTIC_H

```
#define FLOWLOGISTIC_H
```

6.18 flowLogistical.h

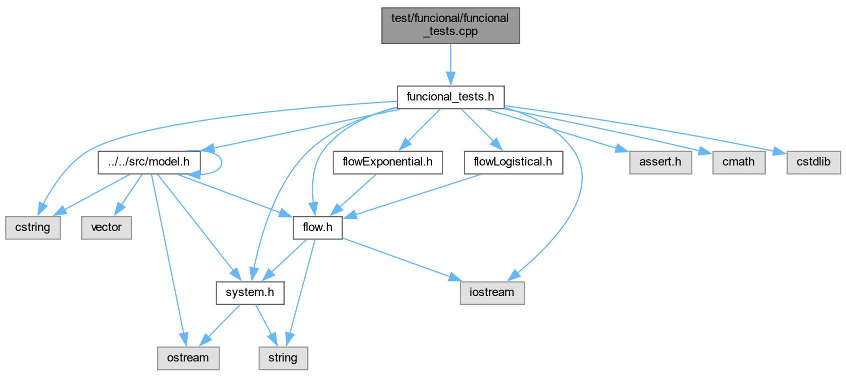
Go to the documentation of this file.

```
#ifndef FLOWLOGISTICAL_H
#define FLOWLOGISTIC_H

#include "../../src/flow.h"

class FlowLogistical : public Flow {
public:
    FlowLogistical();
    FlowLogistical(Flow &obj);
    FlowLogistical(const string name, System *source, System *target);
    virtual ~FlowLogistical();
    virtual float execute();
};

#endif
```

6.19 test/funcional/funcional_tests.cpp File Reference

```
#include "funcional_tests.h"
```

Include dependency graph for funcional_tests.cpp:



Functions

- void exponentialFuncionalTest ()

  Run the exponential test.
- void logisticalFuncionalTest ()

  Run the logistical test.
- void complexFuncionalTest ()

  Run the complex test, with multiple systems and flows.

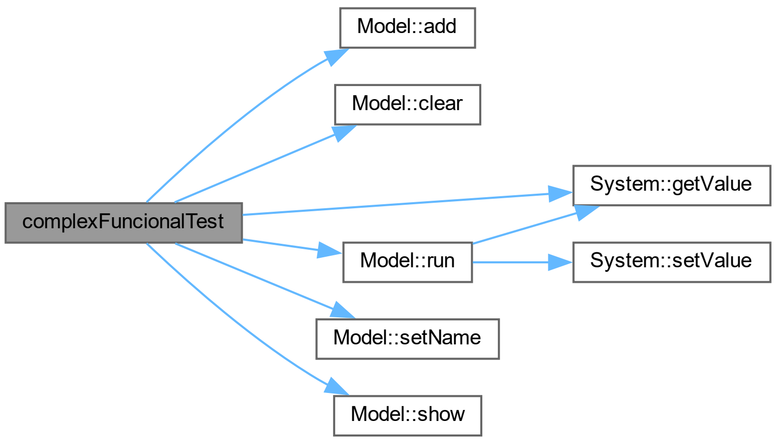
6.19.1 Function Documentation

6.19.1.1 complexFuncionalTest()

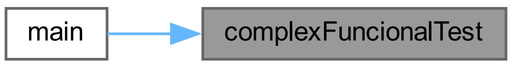
```
void complexFuncionalTest ( )
```

Run the complex test, with multiple systems and flows.

Here is the call graph for this function:



Here is the caller graph for this function:

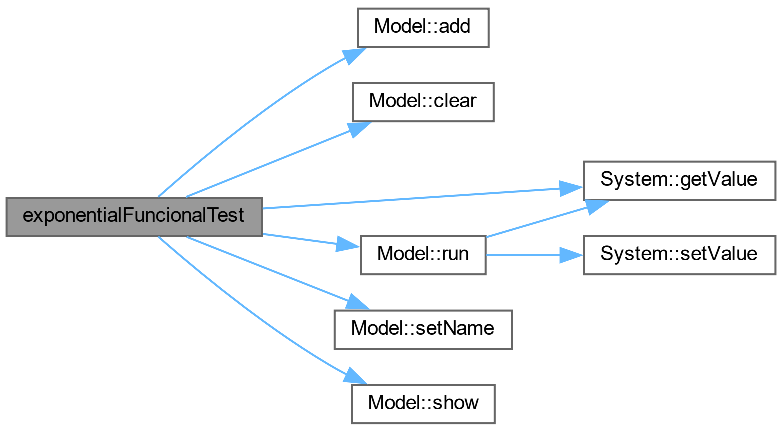


6.19.1.2 exponentialFuncionalTest()

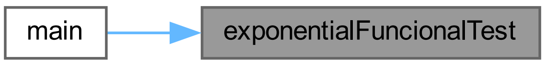
```
void exponentialFuncionalTest ( )
```

Run the exponential test.

Here is the call graph for this function:



Here is the caller graph for this function:

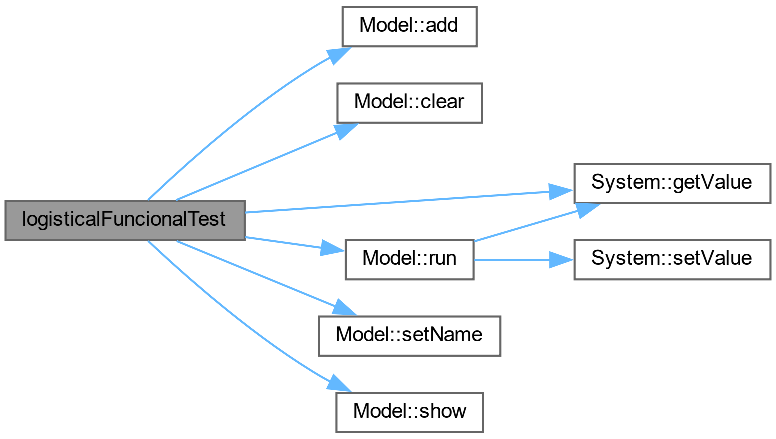


6.19.1.3 logisticalFuncionalTest()

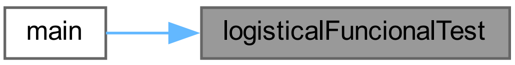
```
void logisticalFuncionalTest ( )
```

Run the logistical test.

Here is the call graph for this function:



Here is the caller graph for this function:

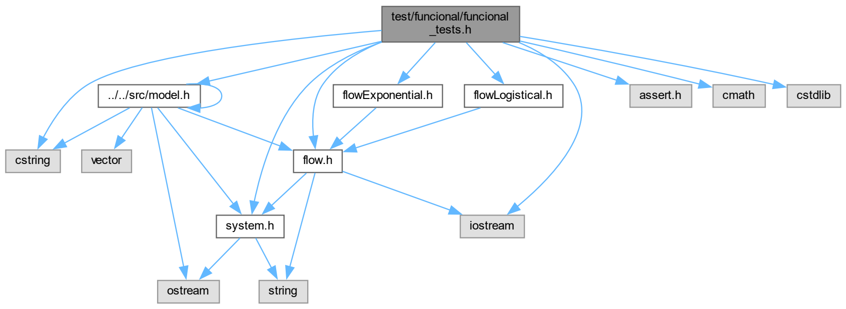


6.20 test/funcional/funcional_tests.h File Reference

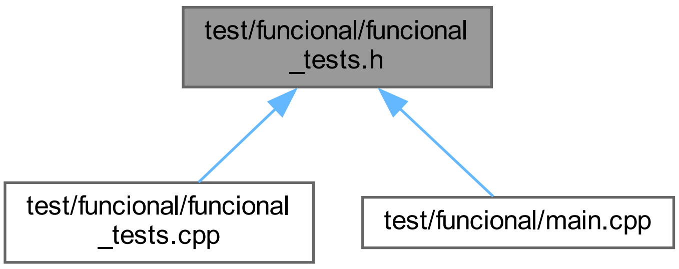
Runs the functional tests.

```
#include "../../src/model.h"
#include "../../src/system.h"
#include "../../src/flow.h"
#include "flowExponential.h"
#include "flowLogistical.h"
#include <assert.h>
#include <cmath>
#include <iostream>
#include <cstdlib>
#include <cstring>
```

Include dependency graph for funcional_tests.h:



This graph shows which files directly or indirectly include this file:



Functions

- void exponentialFuncionalTest ()

  Run the exponential test.
- void logisticalFuncionalTest ()

  Run the logistical test.
- void complexFuncionalTest ()

  Run the complex test, with multiple systems and flows.

6.20.1 Detailed Description

Runs the functional tests.

Author

Iago Andrade (`iago.andrade@aluno.ufop.edu.br`)

Copyright

Copyright (c) 2023

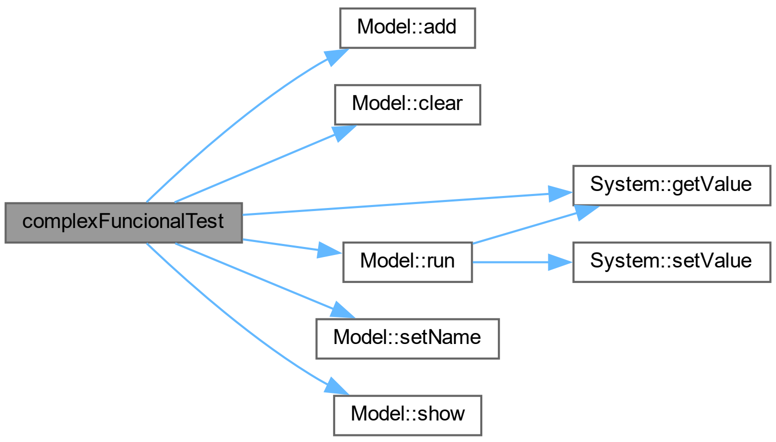
6.20.2 Function Documentation

6.20.2.1 complexFuncionalTest()

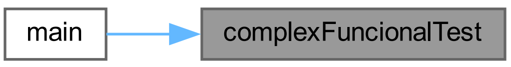
```
void complexFuncionalTest ( )
```

Run the complex test, with multiple systems and flows.

Here is the call graph for this function:



Here is the caller graph for this function:

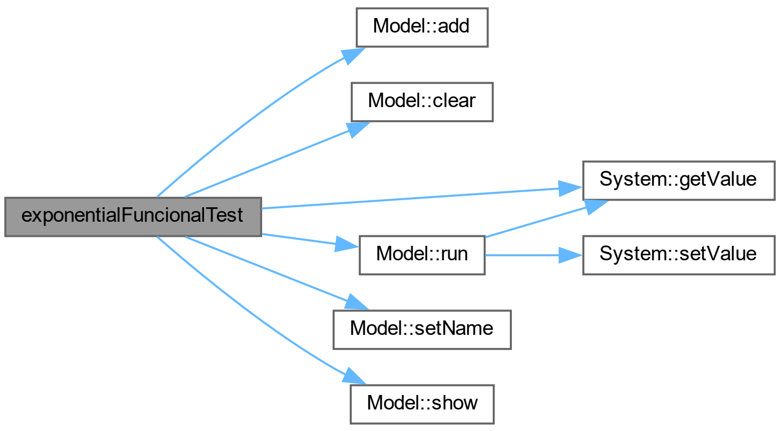


6.20.2.2 exponentialFuncionalTest()

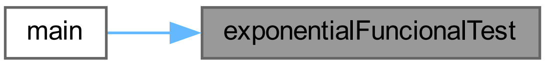
```
void exponentialFuncionalTest ( )
```

Run the exponential test.

Here is the call graph for this function:



Here is the caller graph for this function:

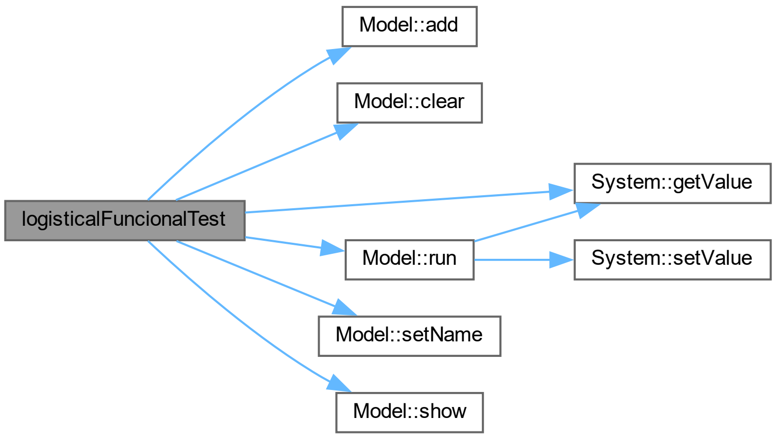


6.20.2.3 logisticalFuncionalTest()

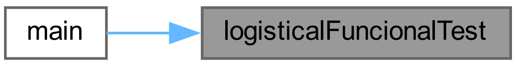
```
void logisticalFuncionalTest ( )
```

Run the logistical test.

Here is the call graph for this function:



Here is the caller graph for this function:



6.21 funcional_tests.h

Go to the documentation of this file.

```
#include "../../src/model.h"
#include "../../src/system.h"
#include "../../src/flow.h"
#include "flowExponential.h"
#include "flowLogistical.h"

#include <assert.h>
#include <cmath>
#include <iostream>
#include <cstdlib>
#include <cstring>

#ifndef FUNCIONAL_TESTS
#define FUNCIONAL_TESTS

void exponentialFuncionalTest();
void logisticalFuncionalTest();
void complexFuncionalTest();

#endif
```

Index