

MyVensim

Generated by Doxygen 1.9.6

1 bcc-322	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	9
5.1 Flow Class Reference	9
5.1.1 Detailed Description	10
5.1.2 Constructor & Destructor Documentation	10
5.1.2.1 Flow() [1/3]	11
5.1.2.2 Flow() [2/3]	11
5.1.2.3 Flow() [3/3]	11
5.1.2.4 ~Flow()	12
5.1.3 Member Function Documentation	12
5.1.3.1 execute()	12
5.1.3.2 getName()	12
5.1.3.3 getSource()	13
5.1.3.4 getTarget()	13
5.1.3.5 operator!=(())	14
5.1.3.6 operator=()	14
5.1.3.7 operator==(())	15
5.1.3.8 setName()	15
5.1.3.9 setSource()	16
5.1.3.10 setTarget()	16
5.1.4 Member Data Documentation	16
5.1.4.1 name	16
5.1.4.2 source	16
5.1.4.3 target	17
5.2 FlowExponential Class Reference	17
5.2.1 Detailed Description	19
5.2.2 Constructor & Destructor Documentation	19
5.2.2.1 FlowExponential() [1/3]	19
5.2.2.2 FlowExponential() [2/3]	19
5.2.2.3 FlowExponential() [3/3]	20
5.2.2.4 ~FlowExponential()	20
5.2.3 Member Function Documentation	20
5.2.3.1 execute()	20
5.3 FlowLogistical Class Reference	21

5.3.1 Detailed Description	22
5.3.2 Constructor & Destructor Documentation	22
5.3.2.1 FlowLogistical() [1/3]	23
5.3.2.2 FlowLogistical() [2/3]	23
5.3.2.3 FlowLogistical() [3/3]	23
5.3.2.4 ~FlowLogistical()	24
5.3.3 Member Function Documentation	24
5.3.3.1 execute()	24
5.4 Model Class Reference	24
5.4.1 Detailed Description	26
5.4.2 Member Typedef Documentation	26
5.4.2.1 itFlow	26
5.4.2.2 itSystem	26
5.4.3 Constructor & Destructor Documentation	26
5.4.3.1 Model() [1/3]	26
5.4.3.2 Model() [2/3]	26
5.4.3.3 Model() [3/3]	27
5.4.3.4 ~Model()	27
5.4.4 Member Function Documentation	27
5.4.4.1 add() [1/2]	27
5.4.4.2 add() [2/2]	27
5.4.4.3 clear()	29
5.4.4.4 getFlowBegin()	30
5.4.4.5 getFlowEnd()	30
5.4.4.6 getFlowSize()	30
5.4.4.7 getName()	30
5.4.4.8 getSystemBegin()	31
5.4.4.9 getSystemEnd()	31
5.4.4.10 getSystemSize()	31
5.4.4.11 remove() [1/2]	31
5.4.4.12 remove() [2/2]	32
5.4.4.13 run()	32
5.4.4.14 setName()	33
5.4.4.15 show()	33
5.4.5 Member Data Documentation	34
5.4.5.1 flows	34
5.4.5.2 name	34
5.4.5.3 systems	34
5.5 System Class Reference	35
5.5.1 Detailed Description	35
5.5.2 Constructor & Destructor Documentation	35
5.5.2.1 System() [1/5]	36

5.5.2.2 System() [2/5]	36
5.5.2.3 System() [3/5]	36
5.5.2.4 System() [4/5]	36
5.5.2.5 System() [5/5]	37
5.5.2.6 ~System()	37
5.5.3 Member Function Documentation	37
5.5.3.1 getName()	37
5.5.3.2 getValue()	38
5.5.3.3 operator=()	38
5.5.3.4 setName()	39
5.5.3.5 setValue()	39
5.5.4 Member Data Documentation	40
5.5.4.1 name	40
5.5.4.2 value	40
6 File Documentation	41
6.1 README.md File Reference	41
6.2 src/flow.cpp File Reference	41
6.3 src/flow.h File Reference	41
6.3.1 Detailed Description	42
6.4 flow.h	43
6.5 src/main.cpp File Reference	43
6.6 test/funcional/main.cpp File Reference	43
6.6.1 Macro Definition Documentation	44
6.6.1.1 MAIN_FUNCIONAL_TESTS	44
6.6.2 Function Documentation	44
6.6.2.1 main()	44
6.7 src/model.cpp File Reference	45
6.8 src/model.h File Reference	45
6.8.1 Detailed Description	47
6.9 model.h	47
6.10 src/system.cpp File Reference	48
6.11 src/system.h File Reference	48
6.11.1 Detailed Description	49
6.12 system.h	49
6.13 test/funcional/flowExponential.cpp File Reference	50
6.14 test/funcional/flowExponential.h File Reference	50
6.14.1 Detailed Description	51
6.15 flowExponential.h	52
6.16 test/funcional/flowLogistical.cpp File Reference	52
6.17 test/funcional/flowLogistical.h File Reference	53
6.17.1 Macro Definition Documentation	54

6.17.1.1 FLOWLOGISTIC_H	54
6.18 flowLogistical.h	54
6.19 test/funcional/funcional_tests.cpp File Reference	54
6.19.1 Function Documentation	55
6.19.1.1 complexFuncionalTest()	55
6.19.1.2 exponentialFuncionalTest()	56
6.19.1.3 logisticalFuncionalTest()	56
6.20 test/funcional/funcional_tests.h File Reference	57
6.20.1 Detailed Description	58
6.20.2 Function Documentation	59
6.20.2.1 complexFuncionalTest()	59
6.20.2.2 exponentialFuncionalTest()	60
6.20.2.3 logisticalFuncionalTest()	60
6.21 funcional_tests.h	61
Index	63

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:

Flow	9
FlowExponential	17
FlowLogistical	21
Model	24
System	35

Chapter 3

Class Index

3.1 Class List

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

Flow	Define the interface with the methods to be implemented	9
FlowExponential	This flow class connects two systems to evaluate their final values after running an equation for a given time	17
FlowLogistical	This flow class connects two systems to evaluate their final value after running an equation for a given time	21
Model	Store vectors containing the name of the model and flows and systems related to it	24
System	Define the interface with the methods to be implemented	35

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/ flow.cpp	41
src/ flow.h	
Contains the specifications of the flow class	41
src/ main.cpp	43
src/ model.cpp	45
src/ model.h	
Represents the simulation model	45
src/ system.cpp	48
src/ system.h	
Describes the system class	48
test/funcional/ flowExponential.cpp	50
test/funcional/ flowExponential.h	
Contains the code used to run the functional exponential tests	50
test/funcional/ flowLogistical.cpp	52
test/funcional/ flowLogistical.h	53
test/funcional/ funcional_tests.cpp	54
test/funcional/ funcional_tests.h	
Runs the functional tests	57
test/funcional/ main.cpp	43

Chapter 5

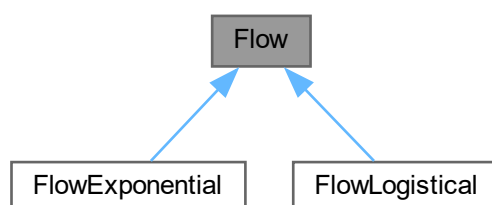
Class Documentation

5.1 Flow Class Reference

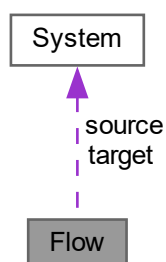
Define the interface with the methods to be implemented.

```
#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]

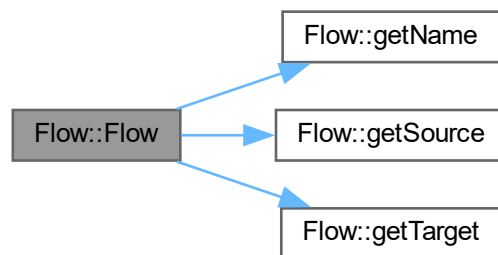
```
Flow::Flow (
    Flow & obj )
```

Copy a [Flow](#) object.

Parameters

<i>obj</i>	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

<i>name</i>	of the flow
<i>source</i>	system
<i>target</i>	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()

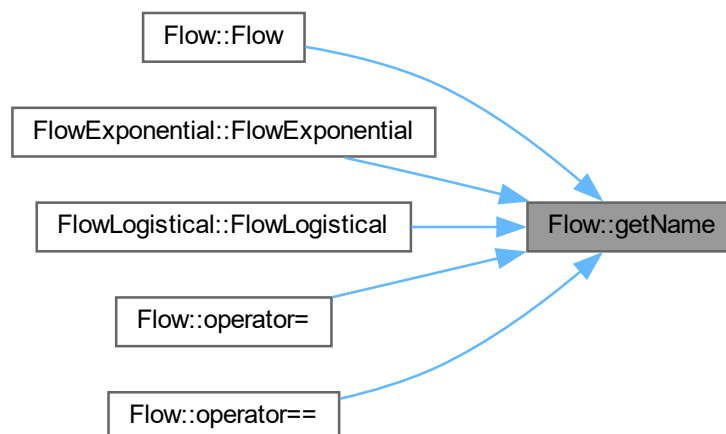
```
string Flow::getName ( ) const
```

Get the Name object.

Returns

string The name of a flow

Here is the caller graph for this function:



5.1.3.3 getSource()

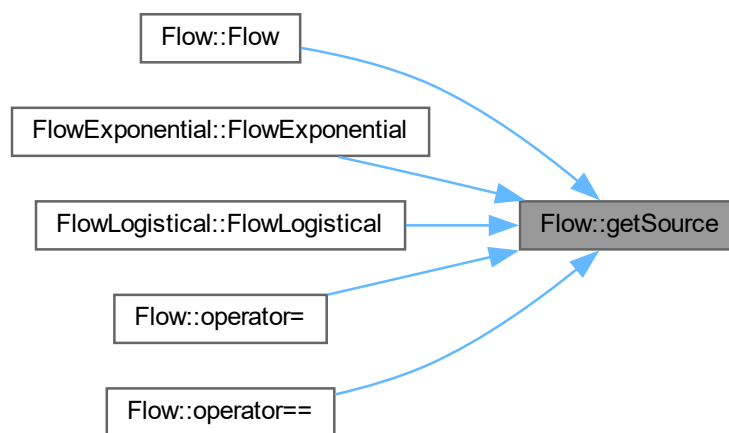
```
System * Flow::getSource ( ) const
```

Get the Source object.

Returns

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

Here is the caller graph for this function:



5.1.3.4 getTarget()

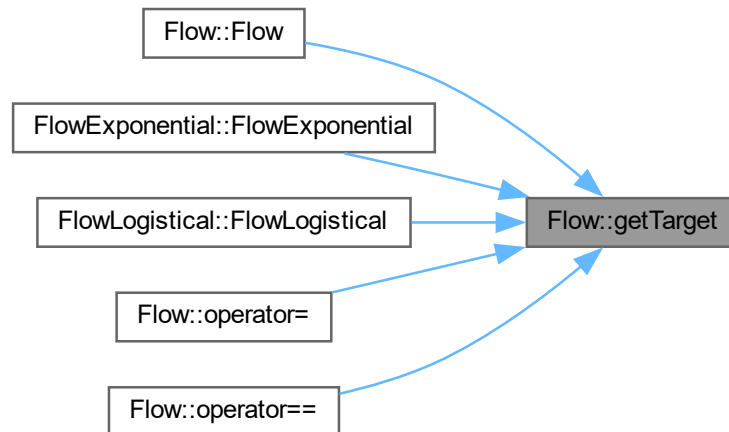
```
System * Flow::getTarget ( ) const
```

Get the Target object.

Returns

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

Here is the caller graph for this function:

**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.

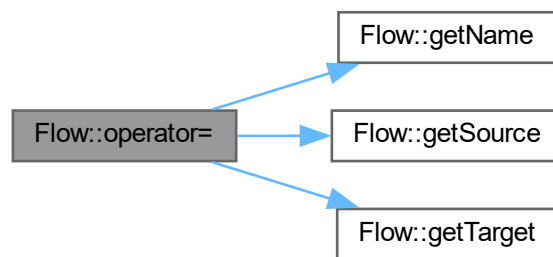
Parameters

<i>obj</i>	flow to be cloned
------------	-------------------

Returns

Flow& A clone of the flow

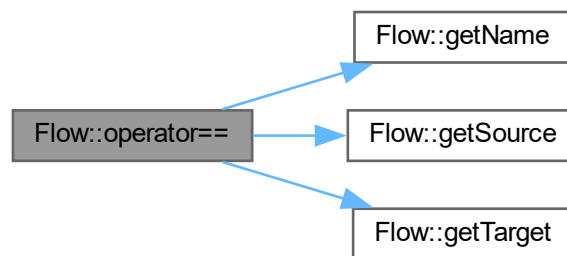
Here is the call graph for this function:



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

<i>name</i>	the flow
-------------	----------

5.1.3.9 setSource()

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

Set the Source object.

Parameters

<i>source</i>	system
---------------	--------

5.1.3.10 setTarget()

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

Set the Target object.

Parameters

<i>target</i>	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:

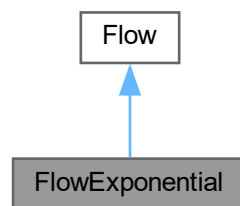
- [src/flow.h](#)
- [src/flow.cpp](#)

5.2 FlowExponential Class Reference

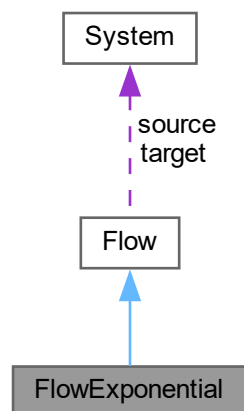
This flow class connects two systems to evaluate their final values after running an equation for a given time.

```
#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]

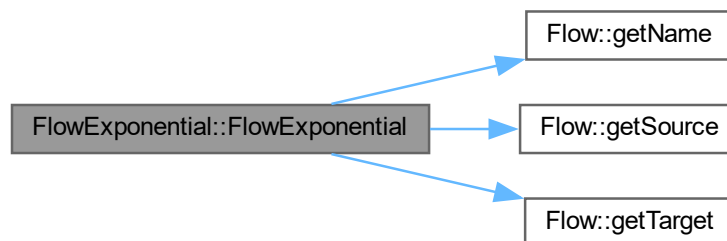
```
FlowExponential::FlowExponential (
    Flow & obj )
```

Copy a [Flow](#) Exponential object.

Parameters

<i>obj</i>	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

<i>name</i>	of the flow
<i>source</i>	system
<i>target</i>	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:

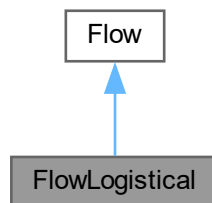
- test/funcional/[flowExponential.h](#)
- test/funcional/[flowExponential.cpp](#)

5.3 FlowLogistical Class Reference

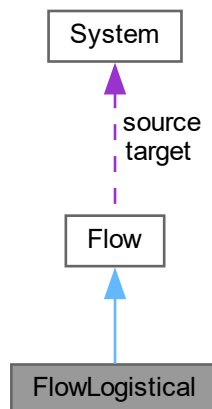
This flow class connects two systems to evaluate their final value after running an equation for a given time.

```
#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]

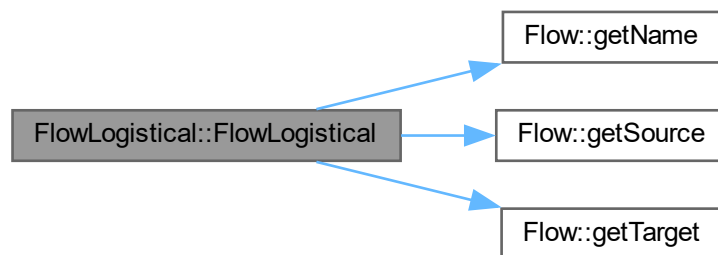
```
FlowLogistical::FlowLogistical (
    Flow & obj )
```

Construct a new [Flow](#) Logistical object.

Parameters

<i>obj</i>	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

<i>name</i>	of the flow
<i>source</i>	system
<i>target</i>	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

<i>name</i>	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

<i>name</i>	of the model
<i>flows</i>	array of pointer-to-flow variables
<i>systems</i>	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

<i>pointer</i>	to a flow
----------------	-----------

5.4.4.2 add() [2/2]

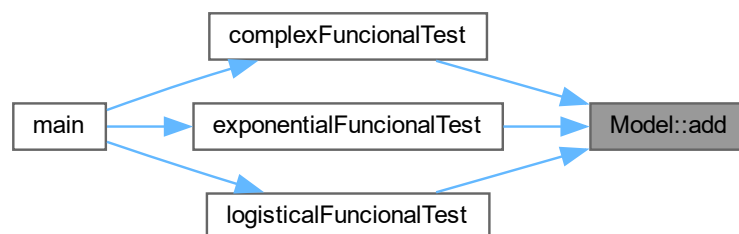
```
void Model::add (
    System * subSystem )
```

Add a system to the model.

Parameters

<i>pointer</i>	to a system
----------------	-------------

Here is the caller graph for this function:

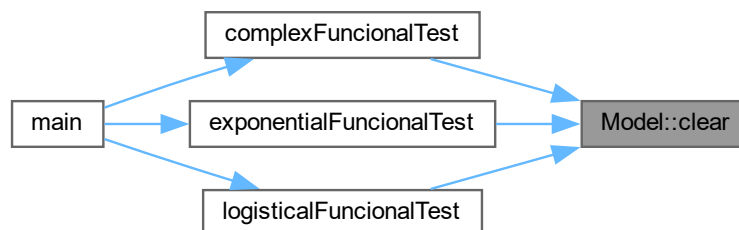


5.4.4.3 clear()

```
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()

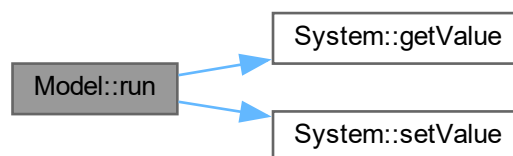
```
void Model::run (
    int start,
    int finish,
    int increment )
```

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

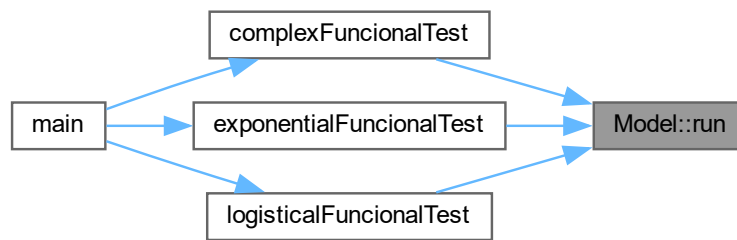
Parameters

<i>int</i>	start time
<i>int</i>	end time
<i>int</i>	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()

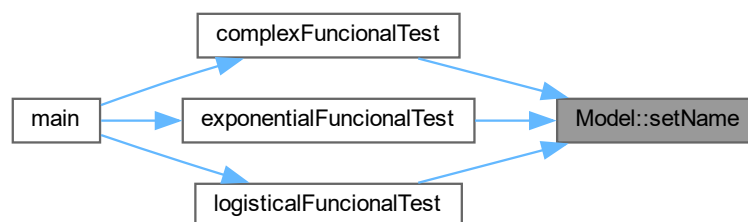
```
void Model::setName (
    const string name )
```

Set the Name object.

Parameters

<i>name</i>	the model
-------------	-----------

Here is the caller graph for this function:

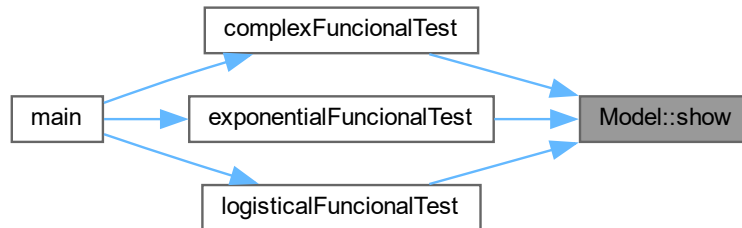


5.4.4.15 show()

```
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

<i>name</i>	of the system
-------------	---------------

5.5.2.3 `System()` [3/5]

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

Construct a new [System](#) object.

Parameters

<i>value</i>	to be contained in that system
--------------	--------------------------------

5.5.2.4 `System()` [4/5]

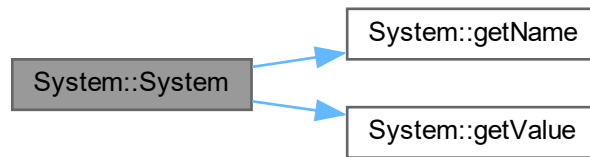
```
System::System (
    System & obj )
```

Copy a [System](#) object.

Parameters

<i>obj</i>	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

<i>name</i>	assigned to the system
<i>value</i>	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()

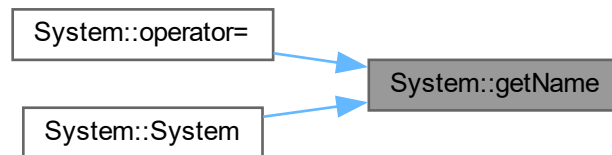
```
string System::getName ( ) const
```

Get the Name object.

Returns

string The name of a system

Here is the caller graph for this function:

**5.5.3.2 getValue()**

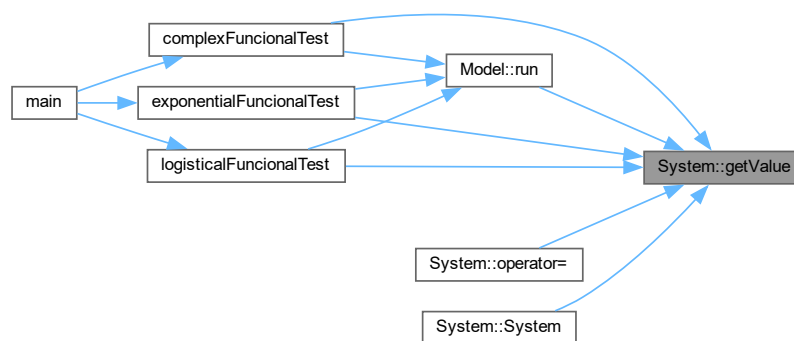
```
float System::getValue ( ) const
```

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.

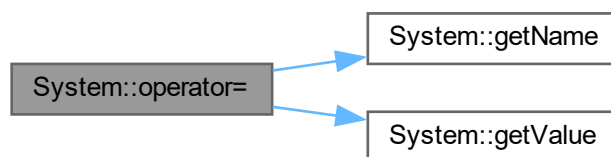
Parameters

<i>obj</i>	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

<i>name</i>	the system
-------------	------------

5.5.3.5 setValue()

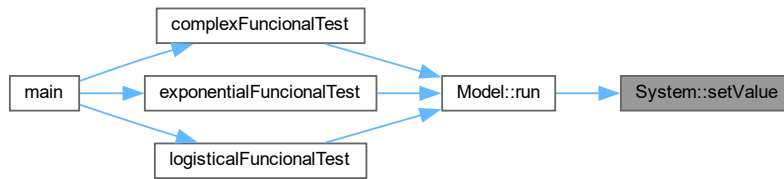
```
void System::setValue (  
    float value )
```

Set the Value object.

Parameters

<i>value</i>	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

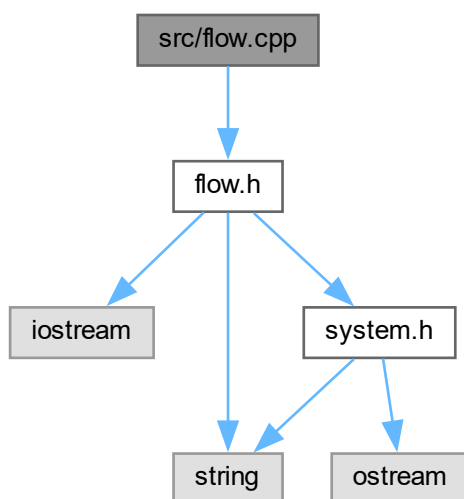
File Documentation

6.1 README.md File Reference

6.2 src/flow.cpp File Reference

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

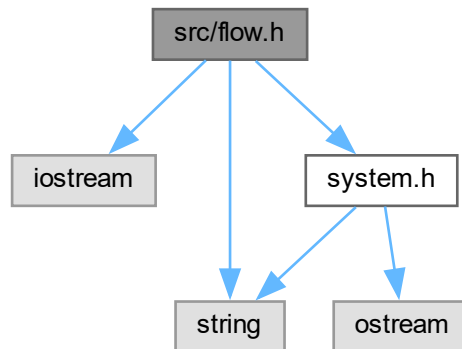
Include dependency graph for flow.cpp:



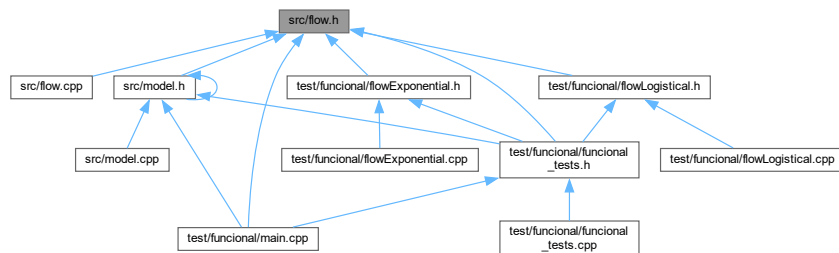
6.3 src/flow.h File Reference

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:



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.](#)

```

00001
00010 #ifndef FLOW_H
00011 #define FLOW_H
00012
00013 #include <iostream>
00014 #include <string>
00015 #include "system.h"
00016
00021 class Flow {
00022     protected:
00027         string name;
00032         System *source;
00037         System *target;
00038     public:
00043         Flow();
00049         Flow(Flow &obj);
00057         Flow(const string name, System *source, System *target);
00062         virtual ~Flow();
00063
00069         string getName() const;
00075         void setName(const string name);
00081         System *getSource() const;
00087         void setSource(System *source);
00093         System *getTarget() const;
00099         void setTarget(System *target);
00100
00101         bool operator==(const Flow &obj) const;
00102         bool operator!=(const Flow &obj) const;
00109         Flow &operator= (const Flow &obj);
00115         virtual float execute() = 0;
00116 };
00117
00118 #endif

```

6.5 src/main.cpp File Reference

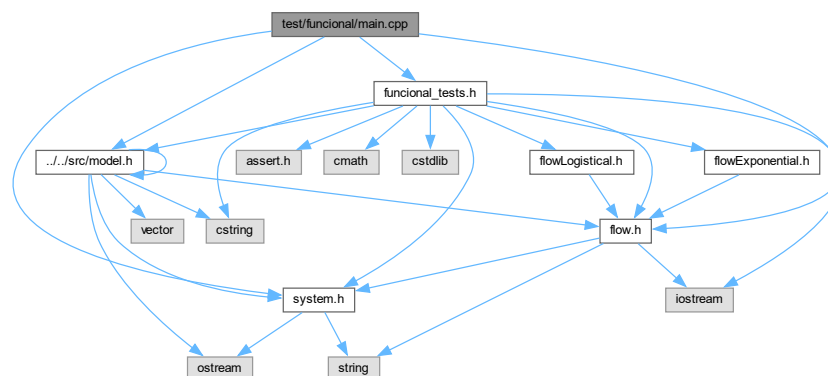
6.6 test/funcional/main.cpp File Reference

```

#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

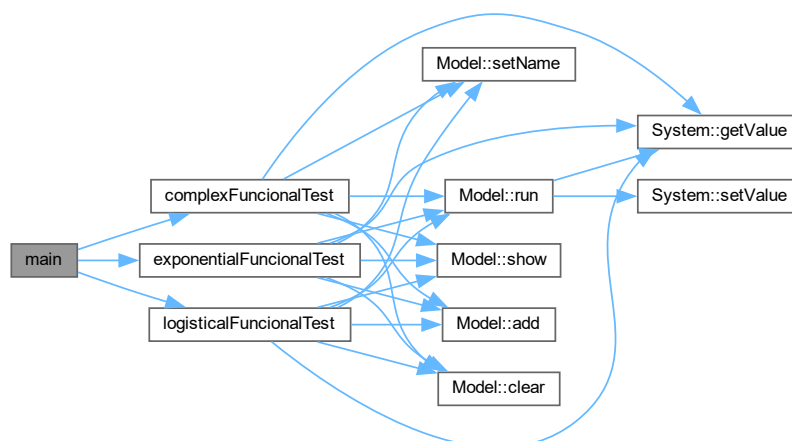
```
#define MAIN_FUNCIONAL_TESTS
```

6.6.2 Function Documentation

6.6.2.1 main()

```
int main ( )
```

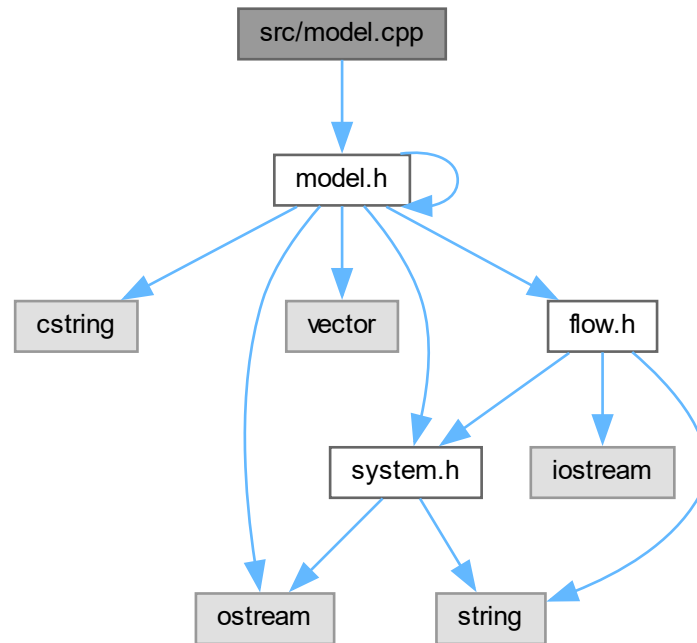
Here is the call graph for this function:



6.7 src/model.cpp File Reference

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

Include dependency graph for model.cpp:

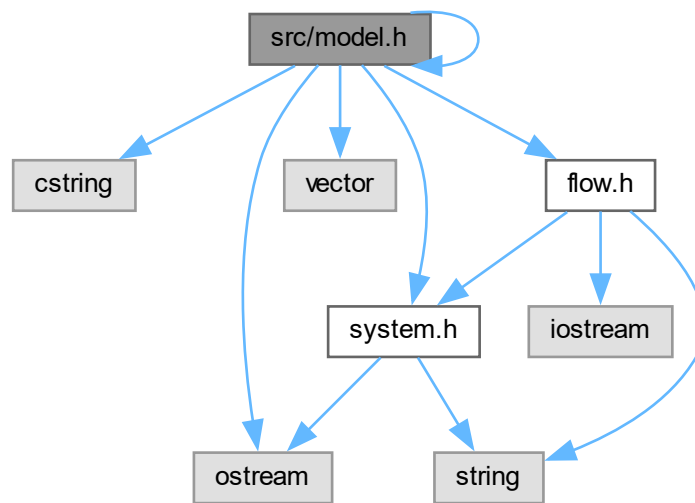


6.8 src/model.h File Reference

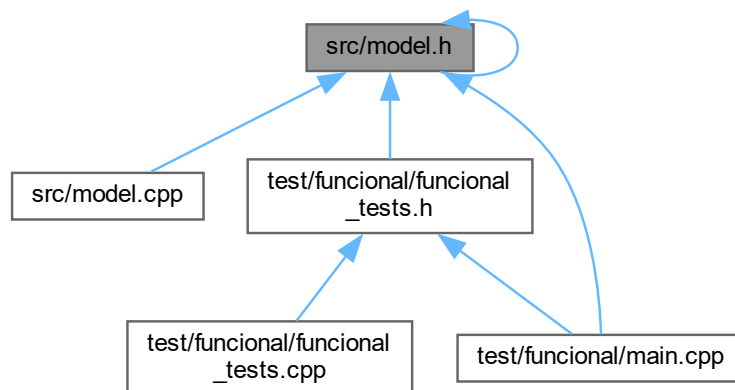
Represents the simulation model.

```
#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.](#)

```

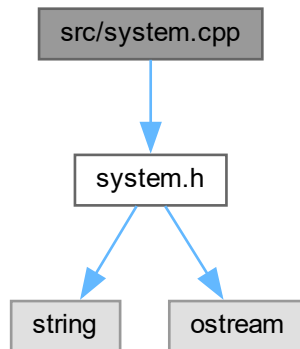
00001
00010 #ifndef MODEL_H
00011 #define MODEL_H
00012 #include <cstring>
00013 #include <ostream>
00014 #include <vector>
00015 #include "flow.h"
00016 #include "system.h"
00017 #include "model.h"
00018
00023 class Model {
00024     protected:
00029         string name;
00034         vector<Flow*> flows;
00039         vector<System*> systems;
00040
00041     private:
00047         Model(Model& obj);
00054         Model& operator= (const Model& obj);
00055
00056     public:
00061         Model();
00067         Model(const string name);
00075         Model(const string name, vector<Flow*> &flows, vector<System*> &systems);
00080         virtual ~Model();
00081
00086         typedef typename vector<Flow*> :: iterator itFlow;
00091         typedef typename vector<System*> :: iterator itSystem;
00092
00098         string getName() const;
00104         void setName(const string name);
00105
00111         itFlow getFlowBegin();
00117         itFlow getFlowEnd();
00123         int getFlowSize();
00124
00130         itSystem getSystemBegin();
00136         itSystem getSystemEnd();
00142         int getSystemSize();
00143
00149         void add(System*);
00155         void add(Flow*);
00162         bool remove(System*);
00169         bool remove(Flow*);
00174         void clear();
00179         void show();
00187         void run(int, int, int);
00188 };
00189
00190 #endif

```

6.10 src/system.cpp File Reference

```
#include "system.h"
```

Include dependency graph for system.cpp:



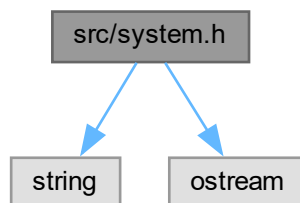
6.11 src/system.h File Reference

Describes the system class.

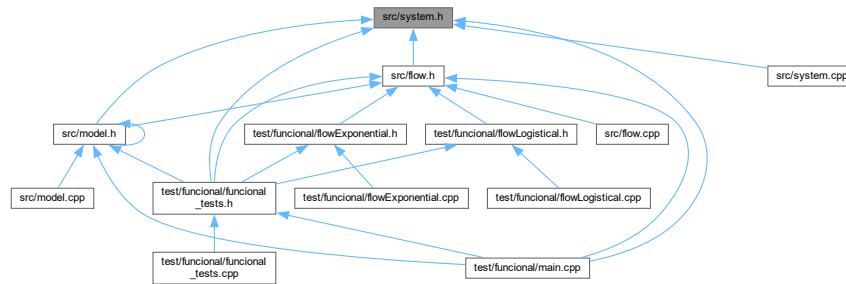
```
#include <string>
```

```
#include <ostream>
```

Include dependency graph for system.h:



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



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.](#)

```

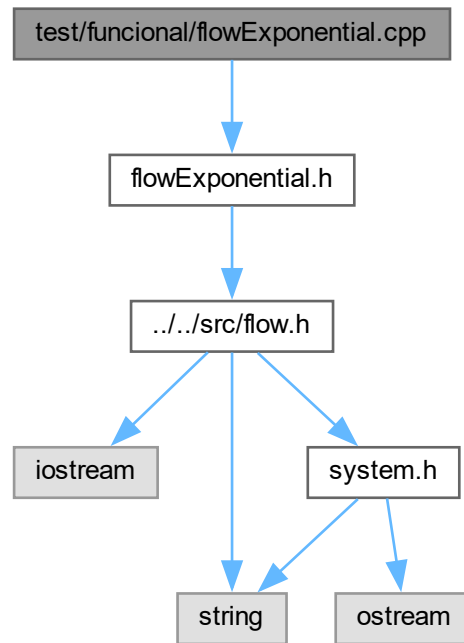
00001
00010 #ifndef SYSTEM_H
00011 #define SYSTEM_H
00012
00013 #include <string>
00014 #include <ostream>
00015
00016 using namespace std;
00017
00022 class System {
00023     protected:
00028         string name;
00033         float value;
00034
00035     public:
00040         System();
00046         System(const string name);
00052         System(float value);
00058         System(System& obj);
00065         System(const string name, float value);
00070         virtual ~System();
00071
00077         string getName() const;
00083         void setName(const string name);
00089         float getValue() const;
00095         void setValue(float value);
00096
00103         System& operator= (const System& obj);
00104 };
00105
00106 #endif

```

6.13 test/funcional/flowExponential.cpp File Reference

```
#include "flowExponential.h"
```

Include dependency graph for flowExponential.cpp:

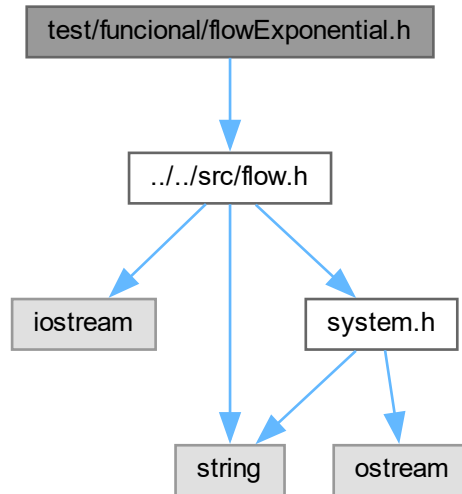


6.14 test/funcional/flowExponential.h File Reference

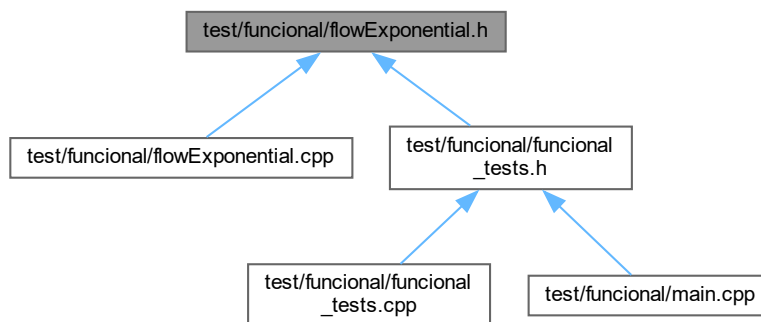
Contains the code used to run the functional exponential tests.


```
#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

Copyright (c) 2023

6.15 flowExponential.h

[Go to the documentation of this file.](#)

```

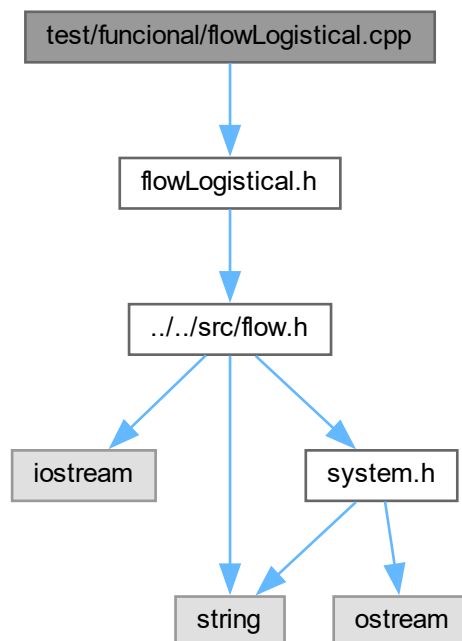
00001
00010 #ifndef FLOWEXPONENTIAL_H
00011 #define FLOWEXPONENTIAL_H
00012
00013 #include "../src/flow.h"
00014
00020 class FlowExponential : public Flow {
00021     public:
00026         FlowExponential();
00031         FlowExponential(Flow &obj);
00039         FlowExponential(const string name, System *source, System *target);
00044         virtual ~FlowExponential();
00050         virtual float execute();
00051 };
00052
00053 #endif

```

6.16 test/funcional/flowLogistical.cpp File Reference

```
#include "flowLogistical.h"
```

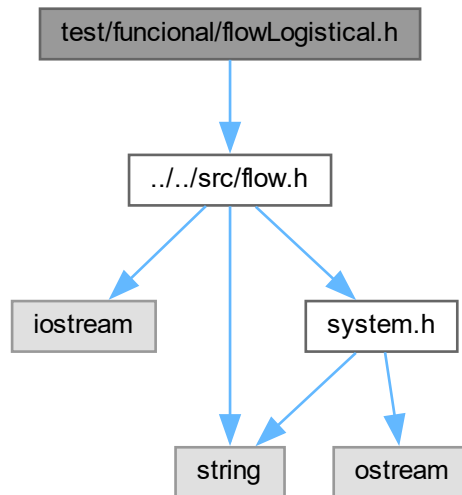
Include dependency graph for flowLogistical.cpp:



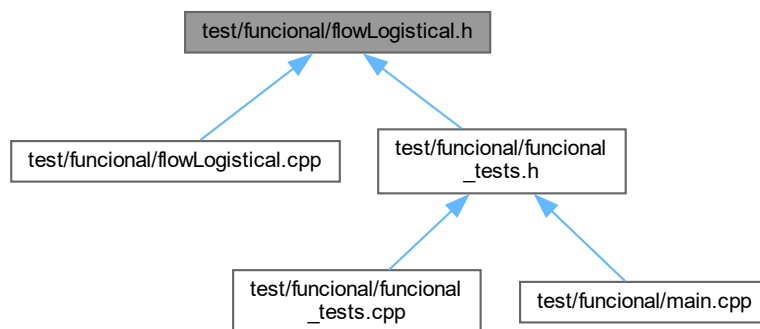
6.17 test/funcional/flowLogistical.h File Reference

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

Include dependency graph for flowLogistical.h:



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



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

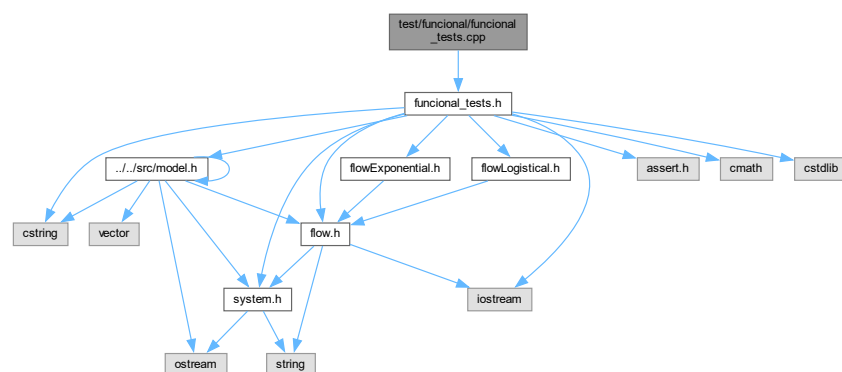
[Go to the documentation of this file.](#)

```
00001
00010 #ifndef FLOWLOGISTICAL_H
00011 #define FLOWLOGISTIC_H
00012
00013 #include "../src/flow.h"
00014
00020 class FlowLogistical : public Flow {
00021 public:
00026     FlowLogistical();
00032     FlowLogistical(Flow &obj);
00040     FlowLogistical(const string name, System *source, System *target);
00045     virtual ~FlowLogistical();
00051     virtual float execute();
00052 };
00053
00054 #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.

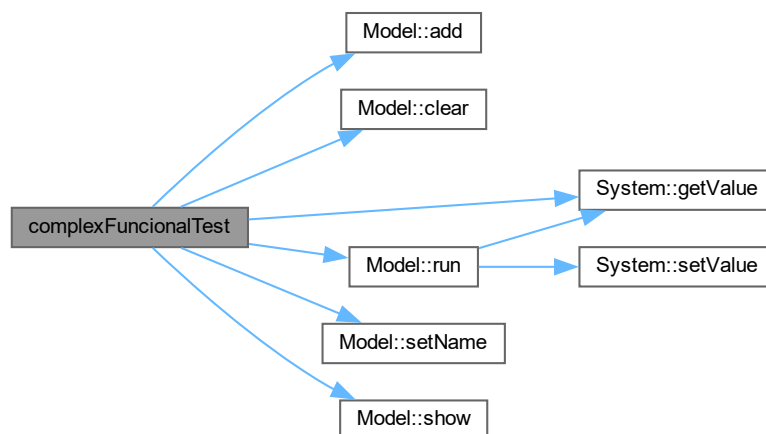
6.19.1 Function Documentation

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:

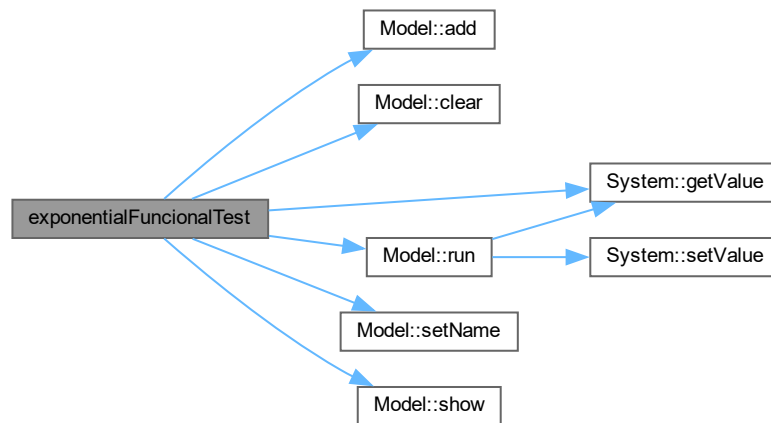


6.19.1.2 exponentialFuncionalTest()

```
void exponentialFuncionalTest ( )
```

Run the exponential test.

Here is the call graph for this function:



Here is the caller graph for this function:

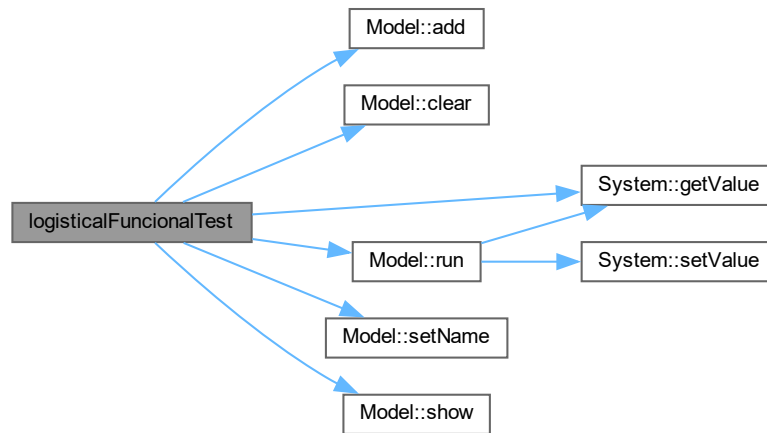


6.19.1.3 logisticalFuncionalTest()

```
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

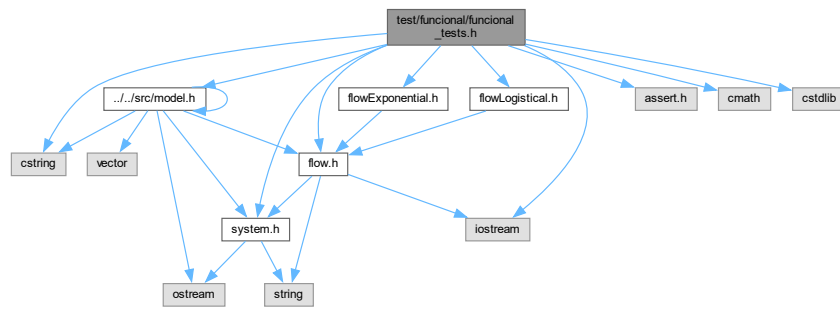
Runs the functional tests.

```

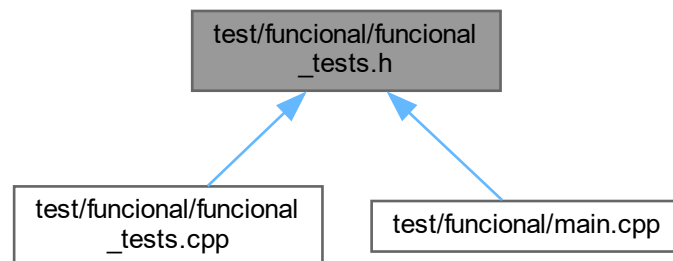
#include "../src/model.h"
#include "../src/system.h"
#include "../src/flow.h"
#include "flowExponential.h"
#include "flowLogistical.h"
#include <assert.h>
#include <cmath>
#include <iostream>
#include <cstdlib>
#include <cstring>

```

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

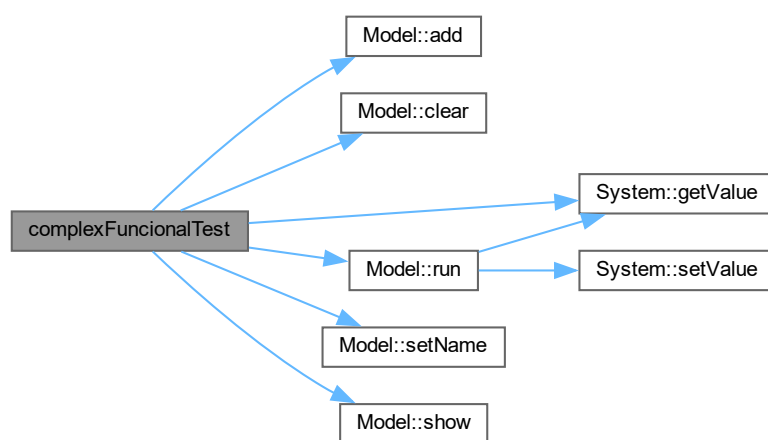
6.20.2 Function Documentation

6.20.2.1 complexFuncionalTest()

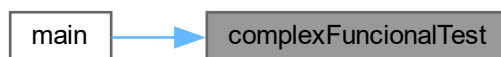
```
void complexFuncionalTest ( )
```

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:

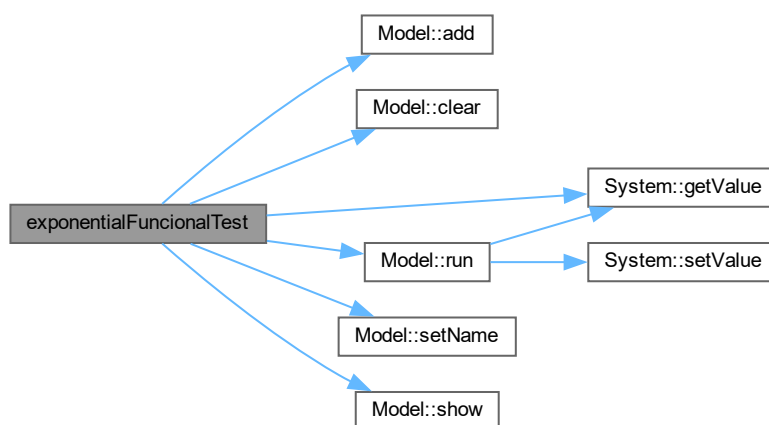


6.20.2.2 exponentialFuncionalTest()

```
void exponentialFuncionalTest ( )
```

Run the exponential test.

Here is the call graph for this function:



Here is the caller graph for this function:

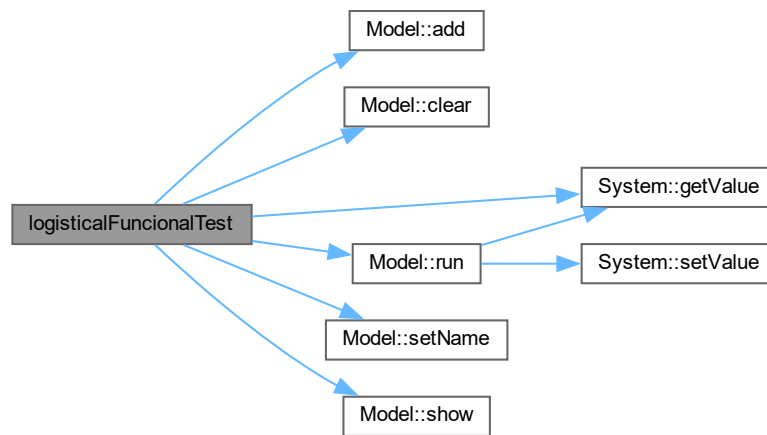


6.20.2.3 logisticalFuncionalTest()

```
void logisticalFuncionalTest ( )
```

Run the logistical test.

Here is the call graph for this function:



Here is the caller graph for this function:



6.21 functional_tests.h

[Go to the documentation of this file.](#)

```

00001
00010 #include "../src/model.h"
00011 #include "../src/system.h"
00012 #include "../src/flow.h"
00013 #include "flowExponential.h"
00014 #include "flowLogistical.h"
00015
00016 #include <assert.h>
00017 #include <cmath>
00018 #include <iostream>
00019 #include <cstdlib>
00020 #include <cstring>
00021
00022 #ifndef FUNCTIONAL_TESTS
00023 #define FUNCTIONAL_TESTS
00024
00029 void exponentialFunctionalTest();
00034 void logisticalFunctionalTest();
00039 void complexFunctionalTest();
00040
00041 #endif
  
```


Index

- ~Flow
 - Flow, [12](#)
- ~FlowExponential
 - FlowExponential, [20](#)
- ~FlowLogistical
 - FlowLogistical, [24](#)
- ~Model
 - Model, [27](#)
- ~System
 - System, [37](#)
- add
 - Model, [27](#)
- clear
 - Model, [29](#)
- complexFuncionalTest
 - funcional_tests.cpp, [55](#)
 - funcional_tests.h, [59](#)
- execute
 - Flow, [12](#)
 - FlowExponential, [20](#)
 - FlowLogistical, [24](#)
- exponentialFuncionalTest
 - funcional_tests.cpp, [55](#)
 - funcional_tests.h, [59](#)
- Flow, [9](#)
 - ~Flow, [12](#)
 - execute, [12](#)
 - Flow, [10](#), [11](#)
 - getName, [12](#)
 - getSource, [12](#)
 - getTarget, [13](#)
 - name, [16](#)
 - operator!=, [14](#)
 - operator=, [14](#)
 - operator==, [15](#)
 - setName, [15](#)
 - setSource, [16](#)
 - setTarget, [16](#)
 - source, [16](#)
 - target, [16](#)
- FlowExponential, [17](#)
 - ~FlowExponential, [20](#)
 - execute, [20](#)
 - FlowExponential, [19](#)
- FLOWLOGISTIC_H
 - flowLogistical.h, [54](#)
- FlowLogistical, [21](#)
 - ~FlowLogistical, [24](#)
 - execute, [24](#)
 - FlowLogistical, [22](#), [23](#)
- flowLogistical.h
 - FLOWLOGISTIC_H, [54](#)
- flows
 - Model, [34](#)
- funcional_tests.cpp
 - complexFuncionalTest, [55](#)
 - exponentialFuncionalTest, [55](#)
 - logisticalFuncionalTest, [56](#)
- funcional_tests.h
 - complexFuncionalTest, [59](#)
 - exponentialFuncionalTest, [59](#)
 - logisticalFuncionalTest, [60](#)
- getFlowBegin
 - Model, [29](#)
- getFlowEnd
 - Model, [30](#)
- getFlowSize
 - Model, [30](#)
- getName
 - Flow, [12](#)
 - Model, [30](#)
 - System, [37](#)
- getSource
 - Flow, [12](#)
- getSystemBegin
 - Model, [30](#)
- getSystemEnd
 - Model, [31](#)
- getSystemSize
 - Model, [31](#)
- getTarget
 - Flow, [13](#)
- getValue
 - System, [38](#)
- itFlow
 - Model, [26](#)
- itSystem
 - Model, [26](#)
- logisticalFuncionalTest
 - funcional_tests.cpp, [56](#)
 - funcional_tests.h, [60](#)
- main

- main.cpp, 44
- main.cpp
 - main, 44
 - MAIN_FUNCIONAL_TESTS, 44
- MAIN_FUNCIONAL_TESTS
 - main.cpp, 44
- Model, 24
 - ~Model, 27
 - add, 27
 - clear, 29
 - flows, 34
 - getFlowBegin, 29
 - getFlowEnd, 30
 - getFlowSize, 30
 - getName, 30
 - getSystemBegin, 30
 - getSystemEnd, 31
 - getSystemSize, 31
 - itFlow, 26
 - itSystem, 26
 - Model, 26, 27
 - name, 34
 - remove, 31
 - run, 32
 - setName, 33
 - show, 33
 - systems, 34
- name
 - Flow, 16
 - Model, 34
 - System, 40
- operator!=
 - Flow, 14
- operator=
 - Flow, 14
 - System, 38
- operator==
 - Flow, 15
- README.md, 41
- remove
 - Model, 31
- run
 - Model, 32
- setName
 - Flow, 15
 - Model, 33
 - System, 39
- setSource
 - Flow, 16
- setTarget
 - Flow, 16
- setValue
 - System, 39
- show
 - Model, 33
- source
 - Flow, 16
 - src/flow.cpp, 41
 - src/flow.h, 41, 43
 - src/main.cpp, 43
 - src/model.cpp, 45
 - src/model.h, 45, 47
 - src/system.cpp, 48
 - src/system.h, 48, 49
 - System, 35
 - ~System, 37
 - getName, 37
 - getValue, 38
 - name, 40
 - operator=, 38
 - setName, 39
 - setValue, 39
 - System, 35–37
 - value, 40
- systems
 - Model, 34
- target
 - Flow, 16
- test/funcional/flowExponential.cpp, 50
- test/funcional/flowExponential.h, 50, 52
- test/funcional/flowLogistical.cpp, 52
- test/funcional/flowLogistical.h, 53, 54
- test/funcional/funcional_tests.cpp, 54
- test/funcional/funcional_tests.h, 57, 61
- test/funcional/main.cpp, 43
- value
 - System, 40