

Reptéri csomagok

dokumentáció

1 Class Index	1
1.1 Class List	1
2 Class Documentation	3
2.1 Belt Class Reference	3
2.1.1 Detailed Description	4
2.1.2 Constructor & Destructor Documentation	4
2.1.2.1 Belt() [1/2]	4
2.1.2.2 Belt() [2/2]	4
2.1.3 Member Function Documentation	4
2.1.3.1 addPackage()	4
2.1.3.2 getPackage()	4
2.1.3.3 operator!=(())	5
2.1.3.4 operator=()	5
2.1.3.5 operator==(())	5
2.1.3.6 removePackage()	6
2.2 bfs Class Reference	6
2.2.1 Detailed Description	7
2.2.2 Constructor & Destructor Documentation	7
2.2.2.1 bfs() [1/2]	7
2.2.2.2 bfs() [2/2]	7
2.2.3 Member Function Documentation	7
2.2.3.1 createTable()	7
2.2.3.2 makePath()	7
2.2.3.3 operator!=(())	8
2.2.3.4 operator=()	8
2.2.3.5 operator==(())	8
2.3 bfsData Class Reference	9
2.3.1 Detailed Description	9
2.3.2 Constructor & Destructor Documentation	9
2.3.2.1 bfsData() [1/2]	9
2.3.2.2 bfsData() [2/2]	10
2.3.3 Member Function Documentation	10
2.3.3.1 display()	10
2.3.3.2 operator!=(())	10
2.3.3.3 operator=()	10
2.3.3.4 operator==(())	11
2.3.3.5 setDistance()	11
2.3.3.6 setPrev()	11
2.4 CC Class Reference	11
2.4.1 Detailed Description	12
2.4.2 Member Function Documentation	12

2.4.2.1 allDone()	12
2.4.2.2 hasPath()	12
2.4.2.3 makeBFS()	12
2.4.2.4 makePath()	13
2.4.2.5 nextBelt()	13
2.5 Junction Class Reference	13
2.5.1 Detailed Description	14
2.5.2 Constructor & Destructor Documentation	15
2.5.2.1 Junction() [1/2]	15
2.5.2.2 Junction() [2/2]	15
2.5.3 Member Function Documentation	15
2.5.3.1 addBelt()	15
2.5.3.2 addPackage()	15
2.5.3.3 display()	16
2.5.3.4 existingBelt()	16
2.5.3.5 existingPackage()	16
2.5.3.6 getBelt()	16
2.5.3.7 getNextPackage()	17
2.5.3.8 getPackage()	17
2.5.3.9 operator!=()	17
2.5.3.10 operator=()	17
2.5.3.11 operator==(())	17
2.5.3.12 send()	18
2.6 List< T > Class Template Reference	18
2.6.1 Detailed Description	19
2.6.2 Constructor & Destructor Documentation	19
2.6.2.1 List() [1/2]	19
2.6.2.2 List() [2/2]	19
2.6.3 Member Function Documentation	20
2.6.3.1 add()	20
2.6.3.2 find()	20
2.6.3.3 findDataByChar()	20
2.6.3.4 findJunctionByChar()	21
2.6.3.5 operator!=()	21
2.6.3.6 operator=()	21
2.6.3.7 operator==(())	22
2.6.3.8 operator[]() [1/2]	22
2.6.3.9 operator[]() [2/2]	22
2.6.3.10 remove() [1/2]	23
2.6.3.11 remove() [2/2]	24
2.6.3.12 search()	24
2.7 listItem< T > Class Template Reference	24

2.7.1 Detailed Description	25
2.7.2 Constructor & Destructor Documentation	25
2.7.2.1 listItem() [1/2]	25
2.7.2.2 listItem() [2/2]	25
2.7.3 Member Function Documentation	25
2.7.3.1 operator=()	25
2.8 Package Class Reference	26
2.8.1 Detailed Description	26
2.8.2 Constructor & Destructor Documentation	27
2.8.2.1 Package() [1/2]	27
2.8.2.2 Package() [2/2]	27
2.8.3 Member Function Documentation	27
2.8.3.1 display()	27
2.8.3.2 operator!=()	27
2.8.3.3 operator=()	28
2.8.3.4 operator==(())	28
2.9 Path Class Reference	28
2.9.1 Detailed Description	29
2.9.2 Constructor & Destructor Documentation	29
2.9.2.1 Path() [1/2]	29
2.9.2.2 Path() [2/2]	29
2.9.3 Member Function Documentation	30
2.9.3.1 addStop()	30
2.9.3.2 nextStop()	30
2.9.3.3 operator!=()	30
2.9.3.4 operator=()	30
2.9.3.5 operator==(())	31
2.9.3.6 operator[]()	31
Index	33

Chapter 1

Class Index

1.1 Class List

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

Belt	Objects of this class are "belts" between two junctions.	3
bfs	Objects of this class store the BFS algorithm's data	6
bfsData	Objects of this class store data which is used in the bfs class's table	9
CC	Objects of this class store package's paths and also junction's bfs objects	11
Junction	Objects of this class store packages and are connected to one another with belts	13
List< T >	A linked list of listItems	18
listItem< T >	A list item consisting of a T type item and a pointer to the next element	24
Package	Objects of this class are "packages" that go from their starting point to their destination.	26
Path	Objects of this class store a package's path that it follows to its destination	28

Chapter 2

Class Documentation

2.1 Belt Class Reference

Objects of this class are "belts" between two junctions.

Public Member Functions

- **Belt** ()
A basic belt object (ID -1)
- **Belt** (int id, char start, char end)
A belt object that connects to junctions.
- **Belt** (const **Belt** &b)
Copy constructor of belt.
- int **getId** () const
Returns the id of the belt.
- char **getStart** () const
Returns the starting junction's name of the belt.
- char **getEnd** () const
Returns the ending junction's name of the belt.
- **Belt** & **operator=** (const **Belt** &b)
Copies a belt object.
- bool **operator==** (const **Belt** &b) const
Checks if the two belts are the same (have the same id)
- bool **operator!=** (const **Belt** &b) const
Checks if the two belts are not the same (don't have the same id)
- void **addPackage** (const **Package** &p)
Adds a package to the belts package list.
- **Package** **removePackage** ()
Removes the returns the first element of the packages list.
- size_t **getNoOfPackages** () const
Returns the number of packages stored in the list.
- **Package** & **getPackage** (size_t i)
Returns the 'i'th element of the packages list.

2.1.1 Detailed Description

Objects of this class are "belts" between two junctions.

Every belt has a unique id, a starting and an ending point and also a list of packages currently "on" the belt

2.1.2 Constructor & Destructor Documentation

2.1.2.1 Belt() [1/2]

```
Belt::Belt (
    int id,
    char start,
    char end) [inline]
```

A belt object that connects to junctions.

Parameters

<i>id</i>	the belt id
<i>start</i>	the starting junction's name
<i>end</i>	the ending junction's name

2.1.2.2 Belt() [2/2]

```
Belt::Belt (
    const Belt & b) [inline]
```

Copy constructor of belt.

Parameters

<i>b</i>	the belt object to copy
----------	-------------------------

2.1.3 Member Function Documentation

2.1.3.1 addPackage()

```
void Belt::addPackage (
    const Package & p) [inline]
```

Adds a package to the belts package list.

Parameters

<i>p</i>	the package to add to the list
----------	--------------------------------

2.1.3.2 getPackage()

```
Package & Belt::getPackage (
    size_t i) [inline]
```

Returns the 'i'th element of the packages list.

Parameters

<i>i</i>	index
----------	-------

Returns

A reference to the object

2.1.3.3 operator!=()

```
bool Belt::operator!= (
    const Belt & b) const [inline]
```

Checks if the two belts are not the same (don't have the same id)

Parameters

<i>b</i>	the belt object to check
----------	--------------------------

Returns

True if the two id's don't match

2.1.3.4 operator=()

```
Belt & Belt::operator= (
    const Belt & b)
```

Copies a belt object.

Parameters

<i>b</i>	the belt object to copy
----------	-------------------------

Returns

A reference to this object

2.1.3.5 operator==()

```
bool Belt::operator== (
    const Belt & b) const [inline]
```

Checks if the two belts are the same (have the same id)

Parameters

<i>b</i>	the belt object to check
----------	--------------------------

Returns

True if the two id's match

2.1.3.6 removePackage()

```
Package Belt::removePackage ()
```

Removes the returns the first element of the packages list.

Returns

The first element of the packages list or a generic package (ID -1) if the list is empty

2.2 bfs Class Reference

Objects of this class store the BFS algorithm's data.

Public Member Functions

- **bfs** ()
Default bfs object.
- **bfs** (const **Junction** &junction)
A bfs object starting from the given junction with an empty table.
- **bfs** (const **bfs** &newBFS)
Copy constructor of the bfs class.
- char **getJunction** () const
Returns the junction's name where the algorithm has started.
- **bfs** & **operator=** (const **bfs** &newBFS)
Assigns the values of the bfs object to this.
- bool **operator==** (const **bfs** &otherBFS) const
Checks if two bfs objects are the same (starting from the same junction)
- bool **operator!=** (const **bfs** &otherBFS) const
Checks if two bfs objects are not the same (starting from different junctions)
- void **createTable** (List< **Junction** > &junctions)
The algorithm creates the table containing which junctions are how far from the start junction.
- List< char > **makePath** (char destination) const
Creates a path to the given destination.
- void **display** () const
Displays the table.

2.2.1 Detailed Description

Objects of this class store the BFS algorithm's data.

Starting from a junction, the algorithm creates a table that include all other junctions in the given list and calculates their distances from the starting junction and where each junction is visited from.

From this table with a function the object can create a path to any other junction.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 bfs() [1/2]

```
bfs::bfs (
    const Junction & junction) [inline]
```

A bfs object starting from the given junction with an empty table.

Parameters

<i>junction</i>	the junction where the algorithm will start from
-----------------	--

2.2.2.2 bfs() [2/2]

```
bfs::bfs (
    const bfs & newBFS) [inline]
```

Copy constructor of the bfs class.

Parameters

<i>newBFS</i>	the bfs object to copy
---------------	------------------------

2.2.3 Member Function Documentation

2.2.3.1 createTable()

```
void bfs::createTable (
    List< Junction > & junctions)
```

The algorithm creates the table containing which junctions are how far from the start junction.

and the previous junction before them where they were discovered from

Parameters

<i>junctions</i>	list of junctions
------------------	-------------------

2.2.3.2 makePath()

```
List< char > bfs::makePath (
    char destination) const
```

Creates a path to the given destination.

from this object's starting junction based on this object's table

Parameters

<i>destination</i>	the destination junction's name
--------------------	---------------------------------

Returns

A list of junction names in the order of visiting them

2.2.3.3 operator!=()

```
bool bfs::operator!= (
    const bfs & otherBFS) const [inline]
```

Checks if two bfs objects are not the same (starting from different junctions)

Parameters

<i>otherBFS</i>	the other bfs object to check
-----------------	-------------------------------

Returns

True if the junctions don't match

2.2.3.4 operator=()

```
bfs & bfs::operator= (
    const bfs & newBFS)
```

Assigns the values of the bfs object to this.

Parameters

<i>newBFS</i>	the bfs object to copy
---------------	------------------------

Returns

A reference to this object

2.2.3.5 operator==()

```
bool bfs::operator== (
    const bfs & otherBFS) const [inline]
```

Checks if two bfs objects are the same (starting from the same junction)

Parameters

<i>otherBFS</i>	the other bfs object to check
-----------------	-------------------------------

Returns

True if the junctions match

2.3 bfsData Class Reference

Objects of this class store data which is used in the bfs class's table.

Public Member Functions

- **bfsData** ()
*Default constructor (distance 0, previous *, junction \0)*
- **bfsData** (char junctionName, int distance, char previous)
Constructor of the [bfsData](#) class.
- **bfsData** (const **bfsData** &d)
Copy constructor.
- **bfsData** & **operator=** (const **bfsData** &d)
Assigns the values of the given [bfsData](#) object to this one.
- bool **operator==** (const **bfsData** &data) const
- bool **operator!=** (const **bfsData** &data) const
- char **getJunction** () const
Returns the junction's name.
- int **getDistance** () const
Returns the distance from the starting junction.
- char **getPrev** () const
Returns the previous junction's name.
- void **setDistance** (int d)
Sets the distance to the given number.
- void **setPrev** (char p)
Sets the previous junction's name to the given name.
- std::ostream & **display** (std::ostream &os) const
Outputs the junction's name, the distance and the previous junction's name to the given output stream.

2.3.1 Detailed Description

Objects of this class store data which is used in the bfs class's table.

2.3.2 Constructor & Destructor Documentation

2.3.2.1 bfsData() [1/2]

```
bfsData::bfsData (
    char junctionName,
    int distance,
    char previous) [inline]
```

Constructor of the [bfsData](#) class.

Parameters

<i>junctionName</i>	this junction's name
<i>distance</i>	distance from the starting junction
<i>previous</i>	the previous junction's name where this was visited from

2.3.2.2 bfsData() [2/2]

```
bfsData::bfsData (  
    const bfsData & d) [inline]
```

Copy constructor.

Parameters

<i>d</i>	the <code>bfsData</code> object to copy
----------	---

2.3.3 Member Function Documentation**2.3.3.1 display()**

```
std::ostream & bfsData::display (  
    std::ostream & os) const [inline]
```

Outputs the junction's name, the distance and the previous junction's name to the given output stream.

Returns

The output stream

2.3.3.2 operator!=()

```
bool bfsData::operator!= (  
    const bfsData & data) const [inline]
```

Parameters

<i>data</i>	the <code>bfsData</code> object to check
-------------	--

Returns

True if the two objects are not the same

2.3.3.3 operator=()

```
bfsData & bfsData::operator= (  
    const bfsData & d) [inline]
```

Assigns the values of the given `bfsData` object to this one.

Parameters

<i>d</i>	the bfsData object to copy
----------	--

Returns

A reference to this object

2.3.3.4 operator==()

```
bool bfsData::operator== (
    const bfsData & data) const [inline]
```

Parameters

<i>data</i>	the bfsData object to check
-------------	---

Returns

True if the two objects are the same

2.3.3.5 setDistance()

```
void bfsData::setDistance (
    int d) [inline]
```

Sets the distance to the given number.

Parameters

<i>d</i>	new distance
----------	--------------

2.3.3.6 setPrev()

```
void bfsData::setPrev (
    char p) [inline]
```

Sets the previous junction's name to the given name.

Parameters

<i>p</i>	the new name
----------	--------------

2.4 CC Class Reference

Objects of this class store package's paths and also junction's bfs objects.

Public Member Functions

- **CC** ()
Constructor of the controllcenter class (empty path and bfs lists)
- bool **hasPath** (int packageID) const
Checks if the package with the given packageID has a path.
- **Belt** & **nextBelt** (int packageID, char junction, List< **Belt** > &belts)
Searches for the belt object where the package should go from the given junction.
- void **makeBFS** (List< **Junction** > &junctions, size_t i)
Makes a bfs object and its table for the 'i'th element of the list of junctions.
- void **makePath** (char junction, char destination, int packageID)
Makes a path for a given package from the starting junction to the destination.
- bool **allDone** (List< **Junction** > &junctions) const

2.4.1 Detailed Description

Objects of this class store package's paths and also junction's bfs objects.

2.4.2 Member Function Documentation

2.4.2.1 allDone()

```
bool CC::allDone (
    List< Junction > & junctions) const
```

Parameters

<i>junctions</i>	list of junctions
------------------	-------------------

Returns

True if every package has arrived to its destination

2.4.2.2 hasPath()

```
bool CC::hasPath (
    int packageID) const
```

Checks if the package with the given packageID has a path.

Parameters

<i>packageID</i>	
------------------	--

Returns

True if the package has a path already

2.4.2.3 makeBFS()

```
void CC::makeBFS (
    List< Junction > & junctions,
    size_t i)
```

Makes a bfs object and its table for the 'i'th element of the list of junctions.

Parameters

<i>junctions</i>	list of junctions
<i>i</i>	index of element

2.4.2.4 makePath()

```
void CC::makePath (
    char junction,
    char destination,
    int packageId)
```

Makes a path for a given package from the starting junction to the destination.

Parameters

<i>junction</i>	starting junction name
<i>destination</i>	destination junction name
<i>packageId</i>	

2.4.2.5 nextBelt()

```
Belt & CC::nextBelt (
    int packageID,
    char junction,
    List< Belt > & belts)
```

Searches for the belt object where the package should go from the given junction.

Parameters

<i>packageID</i>	the id of the package
<i>junction</i>	the current junction where the package is
<i>belts</i>	list of the belts that the package can go to from the current junction

Returns

A reference to the belt

2.5 Junction Class Reference

Objects of this class store packages and are connected to one another with belts.

Public Member Functions

- **Junction** ()
Default constructor (name \0, empty belt and package lists, reachable and hasBelt false)
- **Junction** (char name)
Constructor for a junction object.
- **Junction** (const **Junction** &j)
Copy constructor for junction.
- char **getName** () const
Returns the name of the junction.
- **Package** **getNextPackage** () const
- **Package** & **getPackage** (size_t i)
Returns the i'th element in the packages list.
- **Belt** & **getBelt** (size_t i)
Returns the i'th element in the belts list.
- **List**< **Belt** > & **getBeltList** ()
Return the whole packages list.
- **Junction** & **operator=** (const **Junction** &j)
Assigns the values of the given junction to this object.
- bool **operator==** (const **Junction** &j) const
Checks if two junctions are the same (have the same name)
- bool **operator!=** (const **Junction** &j) const
Checks if two junctions are not the same (don't have the same name)
- int **getNOfBelts** () const
Returns the number of belts in the list.
- int **getNOfPackages** () const
Returns the number of packages in the list.
- void **addBelt** (const **Belt** &b)
Adds the given belt to the belt list of this junction.
- void **addPackage** (const **Package** &p)
Adds a package to the package list of this junction.
- bool **existingBelt** (const **Belt** &b) const
Checks if a belt is already in the belts list.
- bool **existingPackage** (const **Package** &p) const
Checks if a package is already in the package list.
- bool **getReachable** ()
Returns true if this junction is reachable from an other.
- bool **getHasBelt** ()
Returns true if this junction has at least one belt.
- void **setReachable** ()
Sets the reachable value to true.
- void **setHasBelt** ()
Sets the hasBelt value to true.
- void **send** (const **Package** &p, **Belt** &b)
Moves a package from the junction's packages list to the belt.
- std::ostream & **display** (std::ostream &os)
Displays the junction's name, belts and packages to the given ostream.

2.5.1 Detailed Description

Objects of this class store packages and are connected to one another with belts.

2.5.2 Constructor & Destructor Documentation

2.5.2.1 Junction() [1/2]

```
Junction::Junction (  
    char name) [inline]
```

Constructor for a junction object.

Parameters

<i>name</i>	The junction's name
-------------	---------------------

2.5.2.2 Junction() [2/2]

```
Junction::Junction (  
    const Junction & j) [inline]
```

Copy constructor for junction.

Parameters

<i>j</i>	the junction to copy
----------	----------------------

2.5.3 Member Function Documentation

2.5.3.1 addBelt()

```
void Junction::addBelt (  
    const Belt & b)
```

Adds the given belt to the belt list of this junction.

Parameters

<i>b</i>	the belt to add
----------	-----------------

2.5.3.2 addPackage()

```
void Junction::addPackage (  
    const Package & p)
```

Adds a package to the package list of this junction.

Parameters

<i>p</i>	the package to add
----------	--------------------

2.5.3.3 display()

```
std::ostream & Junction::display (  
    std::ostream & os)
```

Displays the junction's name, belts and packages to the given ostream.

Returns

a reference given ostream

2.5.3.4 existingBelt()

```
bool Junction::existingBelt (  
    const Belt & b) const
```

Checks if a belt is already in the belts list.

Parameters

<i>b</i>	the belt object
----------	-----------------

2.5.3.5 existingPackage()

```
bool Junction::existingPackage (  
    const Package & p) const
```

Checks if a package is already in the package list.

Parameters

<i>p</i>	the package object
----------	--------------------

2.5.3.6 getBelt()

```
Belt & Junction::getBelt (  
    size_t i) [inline]
```

Returns the *i*'th element in the belts list.

Parameters

<i>i</i>	index
----------	-------

2.5.3.7 getNextPackage()

```
Package Junction::getNextPackage () const
```

Returns

The next package in the list which hasn't arrived yet \

If it has no packages or all of them have arrived it returns a basic package (ID -1)

2.5.3.8 getPackage()

```
Package & Junction::getPackage (
    size_t i)
```

Returns the 'i'th element in the packages list.

Parameters

<i>i</i>	index
----------	-------

2.5.3.9 operator!=(())

```
bool Junction::operator!= (
    const Junction & j) const [inline]
```

Checks if two junctions are not the same (don't have the same name)

Parameters

<i>j</i>	the other junction to check
----------	-----------------------------

Returns

True if the two names don't match

2.5.3.10 operator=()

```
Junction & Junction::operator= (
    const Junction & j)
```

Assigns the values of the given junction to this object.

Parameters

<i>j</i>	the junction to copy
----------	----------------------

Returns

A reference to this object

2.5.3.11 operator==(())

```
bool Junction::operator== (
    const Junction & j) const [inline]
```

Checks if two junctions are the same (have the same name)

Parameters

<i>j</i>	the other junction to check
----------	-----------------------------

Returns

True if the two names match

2.5.3.12 send()

```
void Junction::send (
    const Package & p,
    Belt & b)
```

Moves a package from the junction's packages list to the belt.

Parameters

<i>p</i>	the package object
<i>b</i>	the belt object

2.6 List< T > Class Template Reference

A linked list of listItems.

Public Member Functions

- `List (size_t size=0)`
Constructor for the list.
- `void add (const T &item)`
Adds an item to the list.
- `void remove (size_t i)`
Removes the 'i'th element from the list if it exists.
- `void remove (const T &item)`
Removes an item from the list if it's in there.
- `T pop ()`
Removes and returns the first element of the list.
- `bool search (const T &item) const`
Searches for an item in the list.
- `template<typename K >`
`listItem< T > * find (const K &item) const`
Finds an item in the list if it can be compared to the items in the list.
- `listItem< bfsData > * findDataByChar (char name) const`
Specifically made for bfsData lists
- `listItem< Junction > * findJunctionByChar (char name) const`

Specifically made for junction lists

- **bool empty ()**
Returns true if the list is empty.
- **size_t size () const**
Returns the number of items in the list.
- **listItem< T > & operator[] (size_t i)**
- **const listItem< T > & operator[] (size_t i) const**
- **listItem< T > * first () const**
Returns a pointer to the first element of the list.
- **List (const List &l)**
Copy constructor of the list.
- **List & operator= (const List &l)**
Assigns the values of the other list to this object.
- **bool operator== (const List< T > &l) const**
- **bool operator!= (const List< T > &l) const**
- **void reverse ()**
Reverses the list.
- **listItem< bfsData > * findDataByChar (char name) const**
- **listItem< Junction > * findJunctionByChar (char name) const**

2.6.1 Detailed Description

template<typename T>
class List< T >

A linked list of listItems.

Template Parameters

<i>T</i>	type of items
----------	---------------

2.6.2 Constructor & Destructor Documentation

2.6.2.1 List() [1/2]

```
template<typename T >
List< T >::List (
    size_t size = 0) [inline]
```

Constructor for the list.

Parameters

<i>size</i>	the size which the list should be (default 0)
-------------	---

2.6.2.2 List() [2/2]

```
template<typename T >
List< T >::List (
    const List< T > & l) [inline]
```

Copy constructor of the list.

Parameters

<i>/</i>	the other list to copy
----------	------------------------

2.6.3 Member Function Documentation

2.6.3.1 add()

```
template<typename T >
void List< T >::add (
    const T & item) [inline]
```

Adds an item to the list.

Parameters

<i>item</i>	the item to add
-------------	-----------------

2.6.3.2 find()

```
template<typename T >
template<typename K >
listItem< T > * List< T >::find (
    const K & item) const [inline]
```

Finds an item in the list if it can be compared to the items in the list.

Template Parameters

<i>K</i>	type of items
----------	---------------

Parameters

<i>item</i>	item to find
-------------	--------------

Returns

A pointer to the element

2.6.3.3 findDataByChar()

```
template<typename T >
listItem< bfsData > * List< T >::findDataByChar (
    char name) const
```

Specifically made for [bfsData](#) lists\

.

It finds a [bfsData](#) item with the junction's name

Parameters

<i>name</i>	the junction's name
-------------	---------------------

Returns

A pointer to the element

2.6.3.4 findJunctionByChar()

```
template<typename T >
listItem< Junction > * List< T >::findJunctionByChar (
    char name) const
```

Specifically made for junction lists\

.

Finds a junction in the list by its name

Parameters

<i>name</i>	The junction's name
-------------	---------------------

Returns

A pointer to the element

2.6.3.5 operator!=()

```
template<typename T >
bool List< T >::operator!= (
    const List< T > & l) const [inline]
```

Parameters

/	the other list object
---	-----------------------

Returns

True if the two lists don't match

2.6.3.6 operator=()

```
template<typename T >
List & List< T >::operator= (
    const List< T > & l) [inline]
```

Assigns the values of the other list to this object.

Parameters

<i>l</i>	the other list object
----------	-----------------------

Returns

A reference to this object

2.6.3.7 operator==()

```
template<typename T >
bool List< T >::operator== (
    const List< T > & l) const [inline]
```

Parameters

<i>l</i>	the other list object
----------	-----------------------

Returns

True if the two lists match

2.6.3.8 operator[]() [1/2]

```
template<typename T >
ListItem< T > & List< T >::operator[] (
    size_t i) [inline]
```

Parameters

<i>i</i>	index
----------	-------

Returns

A reference to the 'i'th element of the list

2.6.3.9 operator[]() [2/2]

```
template<typename T >
const ListItem< T > & List< T >::operator[] (
    size_t i) const [inline]
```

Parameters

<i>i</i>	index
----------	-------

Returns

A constant reference to the 'i'th element of the list

2.6.3.10 remove() [1/2]

```
template<typename T >  
void List< T >::remove (  
    const T & item) [inline]
```

Removes an item from the list if it's in there.

Parameters

<i>item</i>	item to remove
-------------	----------------

2.6.3.11 remove() [2/2]

```
template<typename T >
void List< T >::remove (
    size_t i) [inline]
```

Removes the 'i'th element from the list if it exists.

Parameters

<i>i</i>	index
----------	-------

2.6.3.12 search()

```
template<typename T >
bool List< T >::search (
    const T & item) const [inline]
```

Searches for an item in the list.

Parameters

<i>item</i>	item to search for
-------------	--------------------

Returns

True if it exist in the list

2.7 listItem< T > Class Template Reference

A list item consisting of a T type item and a pointer to the next element.

Public Member Functions

- **listItem** ()
Default constructor (no name, next is nullptr)
- **listItem** (const T &newItem)
Constructor for the list item.
- **listItem** & **operator=** (const listItem< T > &newItem)
Assings the item and next of the given list item to this object.
- **listItem** (const listItem< T > &newItem)
Copy constructor for the list item.
- T & **getItem** ()
Returns a reference to the item of this object.
- const T & **getItem** () const
Returns a constant reference to the item of this object.
- listItem< T > * **getNext** () const
Returns a pointer to the next element.

Friends

- template<typename U >
class **List**

2.7.1 Detailed Description

template<typename T>
class listItem< T >

A list item consisting of a T type item and a pointer to the next element.

Template Parameters

<i>T</i>	type of items
----------	---------------

2.7.2 Constructor & Destructor Documentation

2.7.2.1 listItem() [1/2]

```
template<typename T >
listItem< T >::listItem (
    const T & newItem) [inline]
```

Constructor for the list item.

Parameters

<i>newItem</i>	the item to contain (next is nullptr)
----------------	---------------------------------------

2.7.2.2 listItem() [2/2]

```
template<typename T >
listItem< T >::listItem (
    const listItem< T > & newItem) [inline]
```

Copy constructor for the list item.

Parameters

<i>newItem</i>	the list item to copy
----------------	-----------------------

2.7.3 Member Function Documentation

2.7.3.1 operator=()

```
template<typename T >
listItem & listItem< T >::operator= (
    const listItem< T > & newItem) [inline]
```

Assings the item and next of the given list item to this object.

Parameters

<i>newItem</i>	the list item to copy
----------------	-----------------------

Returns

A reference to this object

2.8 Package Class Reference

Objects of this class are "packages" that go from their starting point to their destination.\

Public Member Functions

- **Package** ()
Default constructor (id: -1, start and destination: \0, done: default false)
- **Package** (int id, char start, char destination)
Constructor for the package.
- **Package** (const **Package** &p)
Copy constructor.
- **Package** & **operator=** (const **Package** &p)
Assigns the values of the other package to this object.
- bool **operator==** (const **Package** &p) const
- bool **operator!=** (const **Package** &p) const
- bool **arrived** () const
Returns true if the package has arrived.
- void **setArrived** ()
Sets the arrived value of the package to true.
- int **getId** () const
Returns the package's id.
- char **getStart** () const
Returns the package's starting junction's name.
- char **getDest** () const
Returns the package's destination junction's name.
- std::ostream & **display** (std::ostream &os)
Displays the package's id, starting and destination point with a tab in the beginning for readability.

2.8.1 Detailed Description

Objects of this class are "packages" that go from their starting point to their destination.\

.

Each package has a unique id.

Parameters

<i>done</i>	indicates if the package has arrived to its destination
-------------	---

2.8.2 Constructor & Destructor Documentation

2.8.2.1 Package() [1/2]

```
Package::Package (
    int id,
    char start,
    char destination) [inline]
```

Constructor for the package.

Parameters

<i>id</i>	package id
<i>start</i>	starting junction's name
<i>destination</i>	destination junction's name
<i>done</i>	default set to false

2.8.2.2 Package() [2/2]

```
Package::Package (
    const Package & p) [inline]
```

Copy constructor.

Parameters

<i>p</i>	the package object to copy
----------	----------------------------

2.8.3 Member Function Documentation

2.8.3.1 display()

```
std::ostream & Package::display (
    std::ostream & os)
```

Displays the package's id, starting and destination point with a tab in the beginning for readability.

Parameters

<i>os</i>	the ostream where to write
-----------	----------------------------

Returns

A reference to the ostream

2.8.3.2 operator!==(

```
bool Package::operator!= (
    const Package & p) const
```

Parameters

<i>p</i>	the other package object
----------	--------------------------

Returns

True if the two packages don't match

2.8.3.3 operator=()

```
Package & Package::operator= (  
    const Package & p)
```

Assigns the values of the other package to this object.

Parameters

<i>p</i>	the other package object
----------	--------------------------

Returns

A reference to this object

2.8.3.4 operator==()

```
bool Package::operator== (  
    const Package & p) const
```

Parameters

<i>p</i>	the other package object
----------	--------------------------

Returns

True if the two packages match

2.9 Path Class Reference

Objects of this class store a package's path that it follows to its destination.

Public Member Functions

- **Path** ()
Default constructor (package id -1, empty stops list)
- `template<size_t n>`
Path (int id, const char(&newStops)[n])
Constructor for path.
- **Path** (int id, `List< char >` newStops)
Constructor for path.
- **Path** & **operator=** (const **Path** &p)
Assigns the values of the other path to this object.
- bool **operator==** (const **Path** &p) const
- bool **operator!=** (const **Path** &p) const
- char **operator[]** (size_t i)
- void **addStop** (char stop)
Adds a stop to the stops list.
- char **nextStop** (char current) const
Returns the next stop after the current stop in the list.
- int **getID** () const
Returns the package id.
- `List< char >` **getStops** () const
Returns the list of the stops.

2.9.1 Detailed Description

Objects of this class store a package's path that it follows to its destination.

2.9.2 Constructor & Destructor Documentation

2.9.2.1 Path() [1/2]

```
template<size_t n>
Path::Path (
    int id,
    const char(&) newStops[n]) [inline]
```

Constructor for path.

Parameters

<i>id</i>	the package id
<i>newStops</i>	a char array of the stops

2.9.2.2 Path() [2/2]

```
Path::Path (
    int id,
    List< char > newStops) [inline]
```

Constructor for path.

Parameters

<i>id</i>	the package id
<i>newStops</i>	a list of the stops

2.9.3 Member Function Documentation

2.9.3.1 addStop()

```
void Path::addStop (  
    char stop)
```

Adds a stop to the stops list.

Parameters

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

2.9.3.2 nextStop()

```
char Path::nextStop (  
    char current) const
```

Returns the next stop after the current stop in the list.

Parameters

<i>current</i>	the current stop
----------------	------------------

2.9.3.3 operator!=()

```
bool Path::operator!= (  
    const Path & p) const [inline]
```

Parameters

<i>p</i>	the other patch object
----------	------------------------

Returns

True if the two path objects are not the same

2.9.3.4 operator=()

```
Path & Path::operator= (  
    const Path & p)
```

Assigns the values of the other path to this object.

Parameters

<i>p</i>	the other path object
----------	-----------------------

Returns

A reference to this object

2.9.3.5 operator==()

```
bool Path::operator== (
    const Path & p) const [inline]
```

Parameters

<i>p</i>	the other patch object
----------	------------------------

Returns

True if the two path objects are the same

2.9.3.6 operator[]()

```
char Path::operator[] (
    size_t i) [inline]
```

Parameters

<i>i</i>	index
----------	-------

Returns

The 'i'th element of the stops

Index

add
List< T >, 20

addBelt
Junction, 15

addPackage
Belt, 4
Junction, 15

addStop
Path, 30

allDone
CC, 12

Belt, 3
addPackage, 4
Belt, 4
getPackage, 4
operator
operator!=, 5
operator=, 5
operator==, 5
removePackage, 6

bfs, 6
bfs, 7
createTable, 7
makePath, 7
operator!=, 8
operator=, 8
operator==, 8

bfsData, 9
bfsData, 9, 10
display, 10
operator!=, 10
operator=, 10
operator==, 11
setDistance, 11
setPrev, 11

CC, 11
allDone, 12
hasPath, 12
makeBFS, 12
makePath, 13
nextBelt, 13

createTable
bfs, 7

display
bfsData, 10
Junction, 15
Package, 27

existingBelt
Junction, 16
existingPackage
Junction, 16

find
List< T >, 20
findDataByChar
List< T >, 20
findJunctionByChar
List< T >, 21

getBelt
Junction, 16
getNextPackage
Junction, 16
getPackage
Belt, 4
Junction, 17

hasPath
CC, 12

Junction, 13
addBelt, 15
addPackage, 15
display, 15
existingBelt, 16
existingPackage, 16
getBelt, 16
getNextPackage, 16
getPackage, 17
Junction, 15
operator!=, 17
operator=, 17
operator==, 17
send, 18

List
List< T >, 19

List< T >, 18
add, 20
find, 20
findDataByChar, 20
findJunctionByChar, 21
List, 19
operator!=, 21
operator=, 21
operator==, 22
operator[], 22
remove, 22, 24

- search, [24](#)
- listItem
 - listItem< T >, [25](#)
- listItem< T >, [24](#)
 - listItem, [25](#)
 - operator=, [25](#)
- makeBFS
 - CC, [12](#)
- makePath
 - bfs, [7](#)
 - CC, [13](#)
- nextBelt
 - CC, [13](#)
- nextStop
 - Path, [30](#)
- operator!=
 - Belt, [5](#)
 - bfs, [8](#)
 - bfsData, [10](#)
 - Junction, [17](#)
 - List< T >, [21](#)
 - Package, [27](#)
 - Path, [30](#)
- operator=
 - Belt, [5](#)
 - bfs, [8](#)
 - bfsData, [10](#)
 - Junction, [17](#)
 - List< T >, [21](#)
 - listItem< T >, [25](#)
 - Package, [28](#)
 - Path, [30](#)
- operator==
 - Belt, [5](#)
 - bfs, [8](#)
 - bfsData, [11](#)
 - Junction, [17](#)
 - List< T >, [22](#)
 - Package, [28](#)
 - Path, [31](#)
- operator[]
 - List< T >, [22](#)
 - Path, [31](#)
- Package, [26](#)
 - display, [27](#)
 - operator!=, [27](#)
 - operator=, [28](#)
 - operator==, [28](#)
 - Package, [27](#)
- Path, [28](#)
 - addStop, [30](#)
 - nextStop, [30](#)
 - operator!=, [30](#)
 - operator=, [30](#)
 - operator==, [31](#)
- operator[], [31](#)
 - Path, [29](#)
- remove
 - List< T >, [22](#), [24](#)
- removePackage
 - Belt, [6](#)
- search
 - List< T >, [24](#)
- send
 - Junction, [18](#)
- setDistance
 - bfsData, [11](#)
- setPrev
 - bfsData, [11](#)