OptimalList

Generated by Doxygen 1.8.13

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 List< T > Class Template Reference

Collaboration diagram for List< T >:

2.2 OptimalList< T > Class Template Reference

```
#include <OptimalList.h>
```

Public Member Functions

- OptimalList (T PoisonValue_, size_t max_size_)
- List< T > * getFront ()
- List< T > * getBack ()
- void pushFront (List< T > &elem)
- void pushBack (List< T > &elem)
- void popFront ()

  Delete first element if exist.
- void popBack ()

  Delete last element if exist.
- void insert (List< T > *pos, List< T > &elem)
- List< T > * getNext (List< T > *elem)
- List< T > * getPrev (List< T > *elem)
- ~OptimalList ()

  Deleting allocated Memory.

2.2.1 Detailed Description

template<class T>
class OptimalList< T >

OPTIMAL LIST All nodes in one place

2.2.2 Constructor & Destructor Documentation

2.2.2.1 OptimalList()

```
template<class T>
OptimalList< T >::OptimalList (
            T PoisonValue_,
            size_t max_size_ )  [inline]
```

Constructor Of Optimal List

Parameters

*Poison↩ Value_*	- value to catch errors
*max_size_*	- size_t maximum count of elements in list

2.2.3 Member Function Documentation

2.2.3.1 getBack()

```
template<class T>
List<T>* OptimalList< T >::getBack ( )  [inline]
```

Return pointer to head, maybe null

Returns

tail

2.2.3.2 getFront()

```
template<class T>
List<T>* OptimalList< T >::getFront ( )  [inline]
```

Return pointer to head, maybe null

Returns

head

2.2.3.3 getNext()

```
template<class T>
List<T>* OptimalList< T >::getNext (
            List< T > * elem )  [inline]
```

Return next element after element

Parameters

elem	- node to get next position

Returns

pointer to next element

2.2.3.4 getPrev()

```
template<class T>
List<T>* OptimalList< T >::getPrev (
            List< T > * elem )  [inline]
```

Return prev element after element

Parameters

elem	- node to get prev position

Returns

pointer to prev element

2.2.3.5 insert()

```
template<class T>
void OptimalList< T >::insert (
            List< T > * pos,
            List< T > & elem )  [inline]
```

Add element after pos If there are now empty nodes, skip adding

Parameters

pos	- position after what we must insert
elem	- node to insert

2.2.3.6 pushBack()

```
template<class T>
```

```
void OptimalList< T >::pushBack (
            List< T > & elem )  [inline]
```

Add element to the tail If there are now empty nodes, skip adding

Parameters

elem	- node to insert

2.2.3.7 pushFront()

```
template<class T>
void OptimalList< T >::pushFront (
            List< T > & elem )  [inline]
```

Add element to the head If there are now empty nodes, skip adding

Parameters

elem	- node to insert

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

- OptimalList/OptimalList.h