Class documentation*List*

Description

ClassList is an implementation of a doubly linked list that can used to store and manage data of any type. It provides a variety of methods for adding, deleting, searching, sorting, and working with files.

Transfers

enum FiLa

Specifies how to search for values   in a list:

- FIRST: Indicates the first occurrence of the search value.
- LAST: Indicates the last occurrence of the search value.

enum AsDe

Specifies the sorting mode:

- ASCENDING: Sort in ascending order.
- DESCENDING: Sort in descending order.

enum MiMa

Used to find the minimum or maximum value in a list:

- MIN: Indicates that the minimum value is to be retrieved.
- MAX: Indicates that the maximum value is obtained.

enum VeHo

Specifies the output format for the method.primDisplay:

- VERTICAL: Displays the list vertically.
- HORIZONTAL: Displays the list horizontally.

enum FSS

Specifies the type of getting the middle element of the list:

- Fast_Slow: Uses a "slow" and "fast" movement method to search middle element.
- SizeFM: Determines the middle element based on the list size.

Constructor and destructor

- **List()**: Constructor that creates an empty doubly linked list.
- **~List()**: A destructor that frees the resources used by the list.

Methods

Control methods

- **void push_back(T data)**: Adds an element to the end of the list.
- **void pop_back()**: Removes the last element from the list.
- **void clear()**: Clears the list by removing all elements.
- **void push_front(T data)**: Adds an element to the beginning of the list.
- **void pop_front()**: Removes the first element from the list.
- **void insert(int index, T data)**: Inserts an element at the specified index, if the index is valid.
- **void removeAt(int index)**: Removes the element at the specified index.
- **void reverse()**: Expands all list items.

Display methods

- **void primDisplay(VeHo type)**: Displays list items in horizontal or vertical format.

- **void infDisplay()**: Displays detailed information about each list item, including indexes and addresses.
- **void DisplayInfo()**: Prints general information about a list, including its size, head and tail addresses, and values.

Search and Removal Methods

- **int FindByValueINDEX(const T value, FiLa find)**: Returns the index of the first or the last occurrence of the specified value.
- **void DeleteByValue(const T value, FiLa find)**: Removes the first or last occurrence of the specified value from the list.
- **Node* FindByValueADRESS(const T value, FiLa find)**: Returns a pointer to the node containing the specified value.

Filling methods

- **void RandomFill(T start, T end, bool repeat)**: Fills the list with random values in the specified range. Ifrepeat is false, the values will be unique.

- **void Initialize_Fill(int size, T value)**: Fills the list with the given the value of the specified size.

Sorting and mixing methods

- **void Sort(AsDe type)**: Sorts the list items in ascending or descending order.
- **void Mesh()**: Shuffles the elements of a list.

Methods for working with files

- **void SaveData_F(std::string path)**: Saves all list items to the specified file.
- **void Clear_F(std::string path)**: Clears the contents of the specified file.
- **void Remove_F(std::string path)**: Deletes the file at the given path.
- **void ImportData_F(std::string path)**: Imports data from a file and adds them to the list.

Methods for managing elements

- **void DeleteAll(T value)**: Removes all list elements with the specified value.
- **void Cutter(int startIndex, int lastIndex)**: Cuts a portion of the list from startIndex to lastIndex.
- **void Swap(int firstIndex, int secondIndex)**: Swaps data two elements at given indices.
- **T GetDataByIndex(int   index)**: Returns the node data at the specified index.

Getters

- **int&GetSize()**: Returns the current size of the list.
- **Node* GetHead()**: Returns a pointer to the head of the list.
- **Node* GetTail()**: Returns a pointer to the tail of the list.
- **Node* GetMiddle(FSS type)**: Returns a pointer to the node in the middle of the list, depending on the specified method (if the list is even, the element on the left).

Iterators

ClassList provides iterators to conveniently traverse the elements of a list:

- **Iterator**: An iterator for iterating over the elements of a list, with the ability to modify them.
- **const_Iterator**: An iterator for iterating over the elements of a list without the ability to modify the data.
- **Reverse_Iterator**: An iterator for iterating backwards through the elements of a list.

Examples of using iterators:

```
for (const auto& item : myList) {

    std::cout << item << " "; // Output each element of the list
}
```

Notes

- When working with methodspush_back, push_front and insert nodes are not created will be automatically released. The user must monitor the allocation and release of memory.

- Methods that require the presence of elements in the list (e.g.pop_front, pop_back, FindByValueINDEX, DeleteByValue and others) throw exceptions in case, if the list is empty.
- To ensure type consistency when importing data from a file, ensure that compatible types are used.

Example of use

```
int main()

{
        List<int> myList; // Create a list of integers myList.push_back(10); // Add
        an element to the end myList.push_front(5); // Add an element to the
        front myList.push_back(20); // Add another element
        myList.Sort(ASCENDING); // Sort the list in ascending order
        myList.infDisplay(); // Display all the elements of the list

        std::cout << "Middle element: "<< myList.GetMiddle(Fast_Slow)->data << std::endl; //
Get and print the middle element
        return 0;
}
```