**ФЕДЕРАЛЬНОЕ ГОСУДАСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА» (РУТ (МИИТ))**

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ

ПО ПРАКТИЧЕСКОЙ РАБОТЕ

По дисциплине «Языки программирования»

Проверила: доц. Васильева М. А.

Выполнил: ст. гр. ТКИ‑241 Негодаева Т. А.

**Формулировка задания:**

Написать класс Single Linked List.

**Файлы, содержащиеся в проекте:**

Main.cpp

Source.cpp

Source.h

**Код программы:**

Source.cpp:

#include "source.h"

#include <stdexcept>

template<typename T>

List<T>::List() : size(0), head(nullptr) {} /\*конструктор\*/

template<typename T>

List<T>::~List() /\*деструктор\*/

{

    if (head == nullptr)

    {

        return; }

        Node\* temp = head;

        head = head->next;

        delete temp;

        --size;

}

template<typename T>

void List<T>::push\_back(T data) /\*вставить элемент в конец списка\*/

{

    if (head == nullptr) /\*если список пустой\*/

    {

        head = new Node(data); /\*создаём новый элемент и передать в конструктор данные\*/

    }

    else /\*если список не пустой\*/

    {

        Node\* current = head;

        while (current->next != nullptr) /\*если указатель на следующий элемент не нулевой\*/

        {

            current = current->next;

        }

        current->next = new Node(data);

    }

++size;

}

template<typename T>

void List<T>::insert(T data, int index)

{

    if (index > size)

    {

        throw std::out\_of\_range("Index out of range"); // Правильный выброс исключения.

    }

    if(index == 0)

    {

        push\_front(data);

    }

    else

    {

        Node\* previous = head;

        for (int i = 0; i < index - 1; ++i)

        {

            previous = previous->next;

        }

        Node\* newNode = new Node(data, previous->next);

        previous->next = newNode;

        ++size;

    }

}

template<typename T>

void List<T>::pop\_back()

{

    removeAt(size - 1);

}

template<typename T>

void List<T>::removeAt(int index)

{

    if(index == 0)

    {

        if (head == nullptr)

        {

            return;

        }

            Node\* temp = head;

            head = head->next;

            delete temp;

            --size

    }

    else

    {

        Node\* previous = head;

        for (int i = 0; i < index - 1; ++i)

        {

            previous = previous->next;

        }

        Node\* toDelete = previous->next;

        previous->next = toDelete->next;

        delete toDelete;

        --size;

    }

}

template<typename T>

void List<T>::push\_front(T data)

{

    head = new Node(data, head);

    ++size;

}

Source.h:

template<typename T>

class List

{

public:

    List();

    ~List();

    void push\_back(T data); /\*добавить элемент в конец списка\*/

    int getsize() {return size;}; /\*возвращает количество элементов в списке\*/

    void push\_front(T data);

    void insert(T value, int index);

    void removeAt(int index);

    void pop\_back();

private:

    struct Node

    {

        Node\* next;

        T data;

        explicit Node(const T data, Node\* next = nullptr)

            : data{data}, next{next}

        {}

    };

    int size; /\*количество элементов\*/

    Node \*head; /\*указатель на начало списка\*/

};

Main.cpp:

#include "source.h"

#include "source.cpp"

int main()

{

    List<int> list;

    list.push\_front(4);

    list.push\_front(8);

    list.push\_front(7);

    list.push\_front(1);

    list.push\_front(3);

}

**UML диаграмма:**