Міністерство освіти і науки України

КПІ ім. Ігоря Сікорського

Кафедра ІПІ

ЗВІТ

з виконання лабораторної роботи № 6

з кредитного модуля

“Основи програмування-2. Методології програмування”

Варіант № 1

Виконав:

студент 1-го курсу

гр. ІП-22 ФІОТ

Андреєва Уляна Андріївна

Київ 2023

ПОСТАНОВКА ЗАДАЧІ

1. Спроектувати АТД "Однозв'язний список" для контейнера, що містить дані довільного типу. Інтерфейс АТД включає такі обов'язкові операції:

- перевірка списку на пустоту,
- очищення списку,
- визначення позиції у списку елемента із заданим значенням,
- видалення елемента з позиції із заданим номером,
- ітератор для доступу до елементів списку з операціями:
  1) встановлення на початок списку,
  2) перевірка кінця списку,
  3) доступ до значення поточного елемента,
  4) перехід до наступного елемента списку.

main.cpp

```
#include <iostream>
#include "LinkedList.h"

using namespace std;

#include "Functions.h"


int main() {
    LinkedList<string> list;

    printListStatus(list);
    addElements(list);
    printListStatus(list);
    printList(list);
    searchElement(list);
    removeElement(list);
    printList(list);
    list.clear();
    printListStatus(list);

    return 0;
}
```

Functions.cpp

```
#include <iostream>
using namespace std;
#include "Functions.h"
#include "LinkedList.h"
#include <limits>


void printListStatus(const LinkedList<string>& list) {
    cout << "Is the list empty? " << (list.isEmpty() ? "Yes" : "No") <<
endl;
}
```

```
void addElements(LinkedList<string>& list) {
    int numElements;
    cout << "Enter the number of elements to add: ";

    while (!(cin >> numElements)) {
        cout << "Invalid input. Please enter an integer: ";
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
    }

    cout << "Enter " << numElements << " elements:" << endl;
    for (int i = 0; i < numElements; i++) {
        string value;
        cin >> value;
        list.insertAtBeginning(value);
    }
}

void printList(const LinkedList<string>& list) {
    cout << "List elements: ";
    for (Iterator<string> it = list.begin(); !it.isEnd(); ++it) {
        cout << *it << " ";
    }
    cout << endl;
}

void searchElement(const LinkedList<string>& list) {
    string searchValue;
    cout << "Enter a value to search in the list: ";
    cin >> searchValue;

    int position = list.find(searchValue);
    if (position != -1) {
        cout << "Element " << searchValue << " found at position: " <<
position << endl;
    } else {
        cout << "Element " << searchValue << " not found" << endl;
    }
}

void removeElement(LinkedList<string>& list) {
    int removePosition;
    cout << "Enter a position to remove an element from: ";

    while (!(cin >> removePosition)) {
        cout << "Invalid input. Please enter an integer: ";
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
    }

    list.removeAt(removePosition);
}
```

Functions.h

```
#ifndef TEST_CPP3_FUNCTIONS_H
#define TEST_CPP3_FUNCTIONS_H

#include "LinkedList.h"
```

```
void printListStatus(const LinkedList<string>& list);

void addElements(LinkedList<string>& list);

void printList(const LinkedList<string>& list);

void searchElement(const LinkedList<string>& list);

void removeElement(LinkedList<string>& list);


#endif //TEST_CPP3_FUNCTIONS_H
```

LinkedList.tpp

```
#include "LinkedList.h"
#include <iostream>

template<typename T>
LinkedList<T>::~LinkedList() {
    clear();
}

template<typename T>
bool LinkedList<T>::isEmpty() const {
    return head == nullptr;
}

template<typename T>
void LinkedList<T>::clear() {
    while (head != nullptr) {
        Node<T>* temp = head;
        head = head->next;
        delete temp;
    }
}

template<typename T>
int LinkedList<T>::find(const T &value) const {
    int index = 0;
    Node<T>* current = head;
    while (current != nullptr) {
        if (current->data == value) {
            return index;
        }
        current = current->next;
        index++;
    }
    return -1;
}

template<typename T>
void LinkedList<T>::removeAt(int position) {
    if (position < 0)
        return;

    if (position == 0) {
        Node<T>* temp = head;
        head = head->next;
        delete temp;
        return;
```

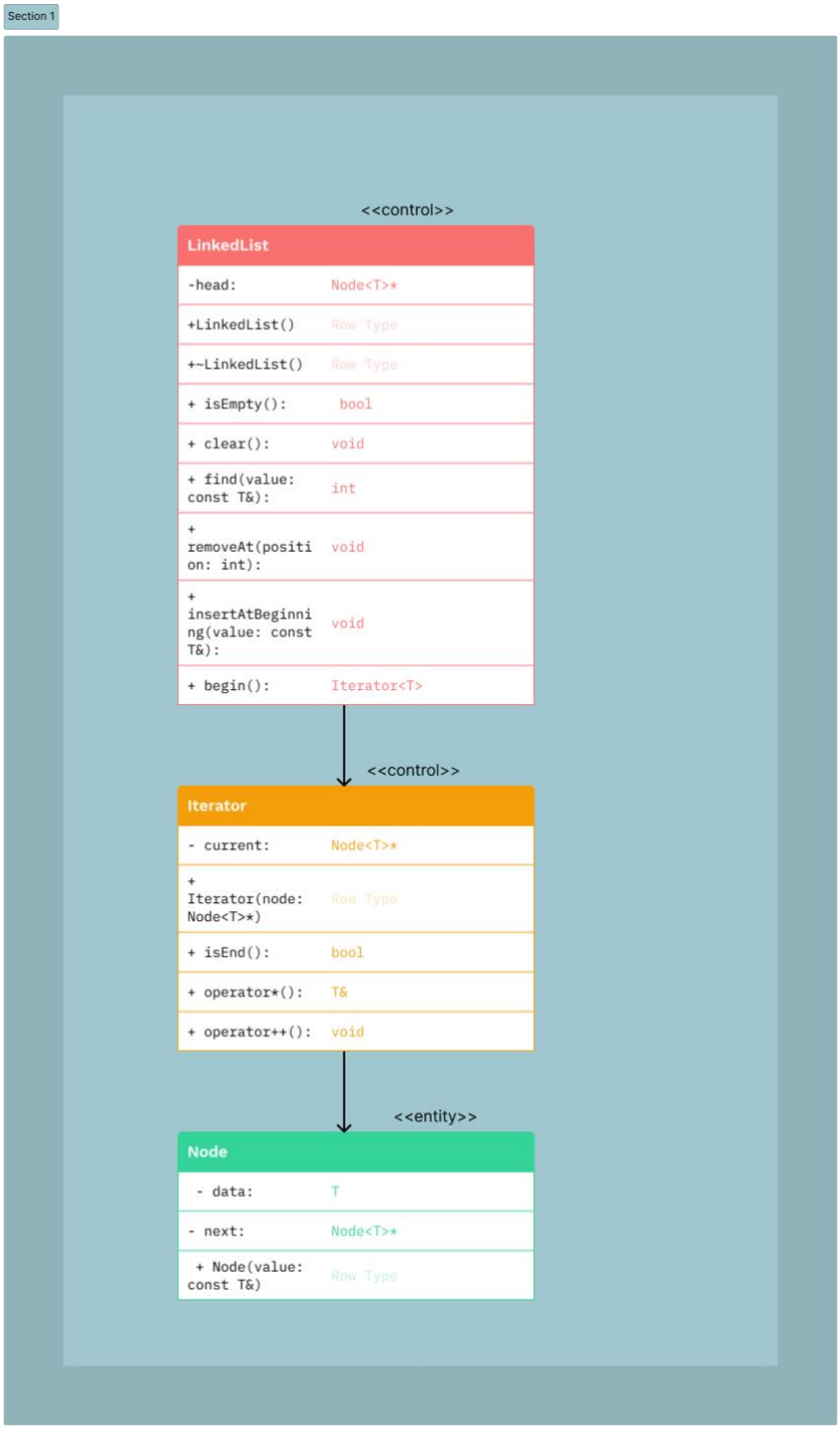
```
    }

    Node<T>* previous = nullptr;
    Node<T>* current = head;
    int index = 0;

    while (current != nullptr && index < position) {
        previous = current;
        current = current->next;
        index++;
    }

    if (current != nullptr) {
        previous->next = current->next;
        delete current;
    }
}

template<typename T>
void LinkedList<T>::insertAtBeginning(const T &value) {
    Node<T>* newNode = new Node<T>(value);
    newNode->next = head;
    head = newNode;
}

template<typename T>
Iterator<T> LinkedList<T>::begin() const {
    return Iterator<T>(head);
}


template<typename T>
bool Iterator<T>::isEnd() const  {
    return current == nullptr;
}

template<typename T>
T &Iterator<T>::operator*() const {
    return current->data;
}

template<typename T>
void Iterator<T>::operator++() {
    if (current != nullptr) {
        current = current->next;
    }
}
```

LinkedList.h

```
#ifndef TEST_CPP3_LINKEDLIST_H
#define TEST_CPP3_LINKEDLIST_H


template <typename T>
struct Node {
    T data;
    Node* next;
```

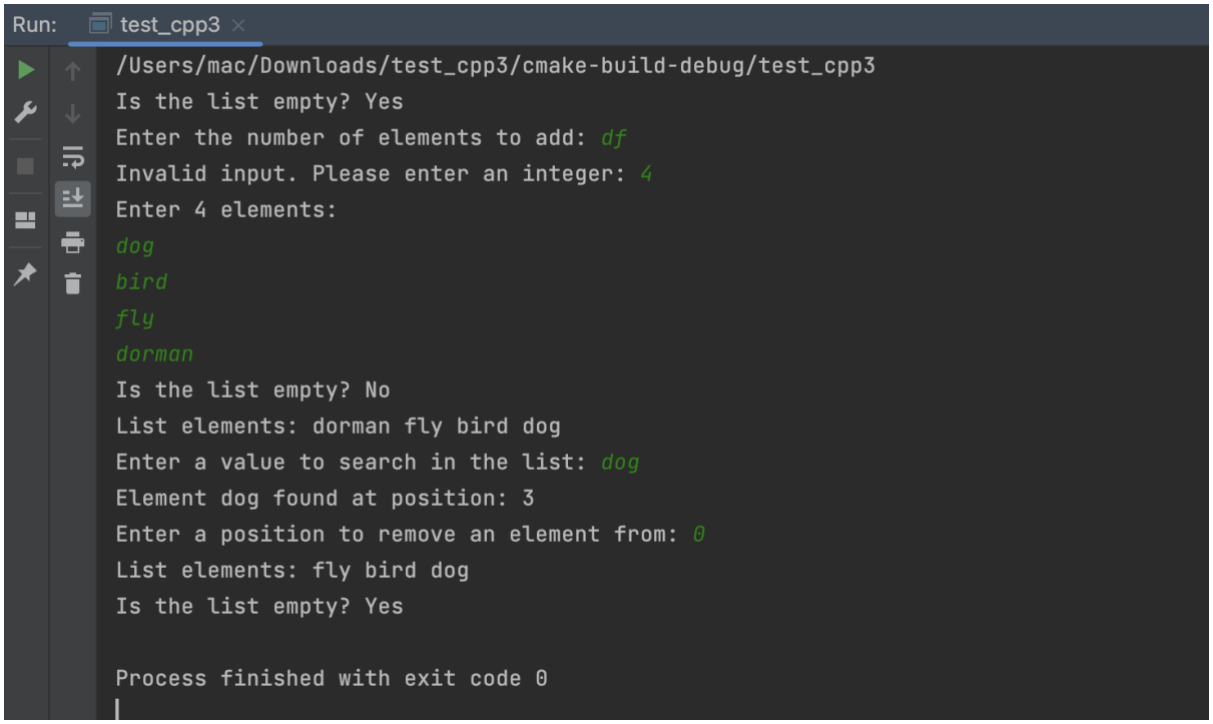
```
    Node(const T& value) : data(value), next(nullptr) {}
};

template <typename T>
class Iterator {
private:
    Node<T>* current;

public:
    Iterator(Node<T>* node) : current(node) {}
    bool isEnd() const;
    T& operator*() const;
    void operator++();
};

template <typename T>
class LinkedList {
private:
    Node<T>* head;

public:
    LinkedList() : head(nullptr) {}
    ~LinkedList();
    bool isEmpty() const;
    void clear();
    int find(const T& value) const;
    void removeAt(int position);
    void insertAtBeginning(const T& value);
    Iterator<T> begin() const;
};

#include "LinkedList.tpp"


#endif
```

Діаграма класів



РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Результат консолі



Лінк на репозиторій у GitHub:

https://github.com/Uliana200407/CppProjects-.git