**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Систем автоматизированного проектирования**

отчет

**по лабораторной работе №1**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: Списки

Вариант: 6

|  |  |  |
| --- | --- | --- |
| Студент гр. 1302 |  | Королев П.С. |
| Преподаватель |  | Родионова Е.А. |

Санкт-Петербург

2022

Цель работы:

Реализовать объект в виде односвязного списка с набором методов/функций. Данные, хранящиеся в списке, имеют целочисленный тип данных.

Список методов/функций, реализуемых в программе:

1. добавление в конец списка
2. добавление в начало списка
3. удаление последнего элемента
4. удаление первого элемента
5. добавление элемента по индексу (вставка перед элементом, который был ранее доступен по этому индексу)
6. получение элемента по индексу
7. удаление элемента по индексу
8. получение размера списка
9. удаление всех элементов списка
10. замена элемента по индексу на передаваемый элемент
11. проверка на пустоту списка
12. вставка другого списка в начало

**Оценка временной сложности каждого метода:**

1. Добавление в конец списка

Надо проходить через весь список, временная сложность зависит от кол-ва элементов в списке =>O(n).

1. Добавление в начало списка

Каждый раз мы обращаемся только к начальному элементу =>O(1).

1. Удаление последнего элемента

Сложности 1 и 3 одинаковы =>O(n).

1. Удаление первого элемента

Сложности 2 и 4 одинаковы =>O(1).

1. Добавление элемента по индексу (вставка перед элементом, который был ранее доступен по этому индексу)

Сложность зависит от выбранного индекса =>O(x), где x–элемент, который был ранее доступен по этому индексу.

1. Получение элемента по индексу

Сложность зависит от выбранного индекса, =>O(x), где x – элемент, который доступен по этому индексу.

1. Удаление элемента по индексу

Сложности 6 и 7 одинаковы =>O(x).

1. Получение размера списка

Сложности 1 и 8 одинаковы =>O(n).

1. Удаление всех элементов списка

Сложности 1 и 9 одинаковы =>O(n).

1. Замена элемента по индексу на передаваемый элемент

Сложности 6 и 10 одинаковы =>O(n).

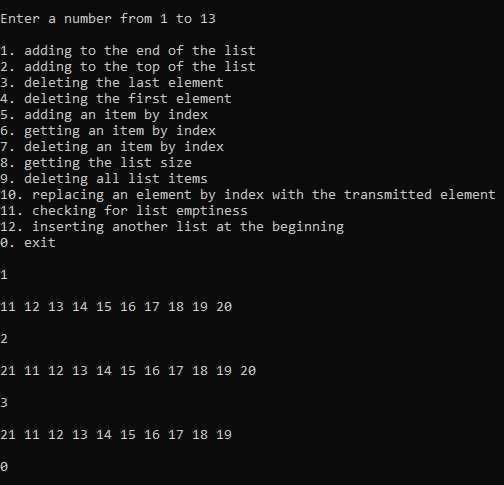
1. Проверка на пустоту списка

Так как проверяется только начальный элемент =>O(1)

12. Вставка другого списка в начало

Так как надо проходить весь список =>O(n)

**Пример работы:**

****

**Листинг:**

#include <fstream>

#include <iostream>

using namespace std;

class Node

{

public:

int data;

int index;

Node\* next = nullptr;

Node\* previous = nullptr;

Node(int count)

{

index = count;

data = 10 + index;

}

};

class List

{

public:

Node\* head = nullptr;

Node\* tail = nullptr;

Node\* safeH = nullptr;

Node\* safeT = nullptr;

int count = 9;

int indexT = 0;

int meaningT = 0;

void addTOback()

{

Node\* curr = new Node(count);

if (head == nullptr)

{

head = curr;

tail = curr;

curr->index = 1;

}

else

{

curr->previous = tail;

tail->next = curr;

tail = curr;

}

}

void addTOfront()

{

Node\* curr = new Node(count);

head->previous = curr;

curr->next = head;

curr->index = 1;

head = curr;

while (curr->next != nullptr)

{

curr = curr->next;

curr->index += 1;

}

}

void deleteBACK()

{

if (head == nullptr)

return;

if (head == tail)

{

deleteFRONT();

}

else

{

Node\* curr = tail;

tail = curr->previous;

tail->next = nullptr;

delete curr;

}

}

void deleteFRONT()

{

if (head == nullptr)

return;

else

{

Node\* curr;

curr = head;

head = curr->next;

delete curr;

curr = head;

if (curr == nullptr)

return;

while (curr != nullptr)

{

curr->index -= 1;

curr = curr->next;

}

}

}

void addBYindex()

{

if (head == nullptr)

return;

else

{

int flag = 0;

Node\* curr = head;

while (flag != 1)

{

if (indexT == curr->index)

{

Node\* tmp = new Node(count);

tmp->next = curr;

tmp->previous = curr->previous;

curr->previous->next = tmp;

curr->previous = tmp;

tmp->index = indexT;

while (curr != nullptr)

{

curr->index += 1;

curr = curr->next;

}

flag = 1;

}

if (flag != 1)

{

curr = curr->next;

if (curr == nullptr)

return;

}

}

}

}

void getBYindex()

{

if (head == nullptr)

return;

else

{

int flag = 0;

Node\* curr = head;

while (flag != 1)

{

if (indexT == curr->index)

{

cout << curr->data;

flag = 1;

}

curr = curr->next;

if (curr == nullptr)

return;

}

}

}

void deleteBYindex()

{

if (head == nullptr)

return;

else

{

int flag = 0;

Node\* curr = head;

while (flag != 1)

{

if (indexT == curr->index)

{

if (curr->next != nullptr)

{

curr->next->previous = curr->previous;

}

if (curr->previous != nullptr)

{

curr->previous->next = curr->next;

}

while (curr != nullptr)

{

curr->index -= 1;

curr = curr->next;

}

delete curr;

flag = 1;

}

if (flag != 1)

{

curr = curr->next;

if (curr == nullptr)

return;

}

}

}

}

void getsize()

{

Node\* curr = head;

int counter = 0;

while (curr != nullptr)

{

counter++;

curr = curr->next;

}

cout << counter;

}

void deleteList()

{

Node\* curr = head;

Node\* tmp;

while (curr != nullptr)

{

tmp = curr->next;

delete curr;

curr = tmp;

}

}

void replacetByindex()

{

if (head == nullptr)

return;

else

{

int flag = 0;

Node\* curr = head;

while (flag != 1)

{

if (indexT == curr->index)

{

curr->data = meaningT;

flag = 1;

}

if (flag != 1)

{

curr = curr->next;

if (curr == nullptr)

return;

}

}

}

}

void checkingFORemptiness()

{

if (head == nullptr)

cout << "Список пуст";

else

cout << "В списке есть элементы";

}

void addingNEWlist()

{

safeT->next = head;

head->previous = safeT;

head = safeH;

}

void out()

{

Node\* curr;

curr = head;

if (head != nullptr)

{

while (curr != nullptr)

{

cout << curr->data;

cout << ' ';

curr = curr->next;

}

cout << endl;

}

else

return;

}

};

int main()

{

setlocale(LC\_ALL, "Russian");

List tech;

Node\* previous = nullptr;

for (int count = 1; count < 10; count++)

{

if (tech.head == nullptr)

{

tech.head = new Node(count);

previous = tech.head;

}

else

{

Node\* curr;

curr = new Node(count);

previous->next = curr;

curr->previous = previous;

previous = curr;

if (count == 9)

tech.tail = curr;

}

}

cout << "Initial list:\n";

tech.out();

cout << endl;

cout << "Enter a number from 1 to 13\n\n";

cout << "1. adding to the end of the list\n";

cout << "2. adding to the top of the list\n";

cout << "3. deleting the last element\n";

cout << "4. deleting the first element\n";

cout << "5. adding an item by index\n";

cout << "6. getting an item by index\n";

cout << "7. deleting an item by index\n";

cout << "8. getting the list size\n";

cout << "9. deleting all list items\n";

cout << "10. replacing an element by index with the transmitted element\n";

cout << "11. checking for list emptiness\n";

cout << "12. inserting another list at the beginning\n";

cout << "0. exit\n\n";

int keyboard;

while (1)

{

cin >> keyboard;

if (keyboard == 1)

{

cout << endl;

tech.count++;

tech.addTOback();

tech.out();

cout << endl;

}

if (keyboard == 2)

{

cout << endl;

tech.count++;

tech.addTOfront();

tech.out();

cout << endl;

}

if (keyboard == 3)

{

cout << endl;

tech.deleteBACK();

tech.out();

cout << endl;

}

if (keyboard == 4)

{

cout << endl;

tech.deleteFRONT();

tech.out();

cout << endl;

}

if (keyboard == 5)

{

cout << endl;

cin >> keyboard;

if (keyboard == 1)

{

cout << endl;

tech.count++;

tech.addTOfront();

tech.out();

cout << endl;

}

else

{

cout << endl;

tech.indexT = keyboard;

tech.count++;

tech.addBYindex();

tech.out();

cout << endl;

}

}

if (keyboard == 6)

{

cout << endl;

cin >> keyboard;

cout << endl;

tech.indexT = keyboard;

tech.getBYindex();

cout << endl;

cout << endl;

}

if (keyboard == 7)

{

cout << endl;

cin >> keyboard;

if (keyboard == 1)

{

cout << endl;

tech.deleteFRONT();

tech.out();

cout << endl;

}

else

{

cout << endl;

tech.indexT = keyboard;

tech.deleteBYindex();

tech.out();

cout << endl;

cout << endl;

}

}

if (keyboard == 8)

{

cout << endl;

tech.getsize();

cout << endl;

cout << endl;

}

if (keyboard == 9)

{

cout << endl;

tech.deleteList();

cout << endl;

cout << endl;

return 0;

}

if (keyboard == 10)

{

cout << endl;

cin >> keyboard;

tech.indexT = keyboard;

cout << endl;

cin >> keyboard;

cout << endl;

tech.meaningT = keyboard;

tech.replacetByindex();

tech.out();

cout << endl;

cout << endl;

}

if (keyboard == 11)

{

cout << endl;

tech.checkingFORemptiness();

cout << endl;

cout << endl;

}

if (keyboard == 12)

{

for (int count = 1; count < 10; count++)

{

if (tech.safeH == nullptr)

{

tech.safeH = new Node(count);

previous = tech.safeH;

}

else

{

Node\* curr;

curr = new Node(count);

previous->next = curr;

curr->previous = previous;

previous = curr;

if (count == 9)

tech.safeT = curr;

}

}

cout << endl;

tech.addingNEWlist();

tech.out();

cout << endl;

cout << endl;

}

if (keyboard == 0)

{

return 0;

}

}

}

**Вывод**

В ходе данной работы я вспомнил множество различной информации, которая нужна для работы со списками.