|  |
| --- |
|  |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«МИРЭА** - **Российский технологический университет»**  **РТУ МИРЭА** |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2**  Стандартные типы данных для представления многоэлементных не однородных структур данных | |
| **по дисциплине** | |
| «Структуры и алгоритмы обработки данных» | |
| Выполнил студент группы ИКБО-02-22 | Вагулич А.Р. |
|  |  |
| Принял | Сорокин А.В. |
|  |  |

Практическая работа выполнена «\_\_» 2023 г.

(подпись студента)

«Зачтено» «\_\_» 2023 г.

(подпись преподавателя)

Москва 2023

**1 Постановка задачи**

Вариант 2

Даны два линейных однонаправленных списка L1 и L2. Разработать функцию для создания исходного списка, используя функцию вставки нового узла перед первым узлом. Разработать функцию вывода списка. Разработать функцию, которая формирует список L, включив в него по одному разу элементы, значения которых входят одновременно в оба списка L1 и L2. Разработать функцию, которая удаляет узел списка L2, расположенный перед узлом, содержащим отрицательное значение. И так для всех узлов, содержащих отрицательное значение. Разработать функцию, которая вставляет новый узел с заданным значением перед каждым узлом списка L1, содержащим нечетное значение.

**2 Модель решения**

Список операций над списками:

1. Создание списков L1 и L2
2. Поиск одинаковых элементов списков L1 и L2
3. Внесение в список L одинаковых элементов списков L1 и L2
4. Удаление узла списка L2
5. Вставка нового узла в список L1

**2.1 Определить структуру узла однонаправленного списка в соответствии с вариантом.**

struct Node {

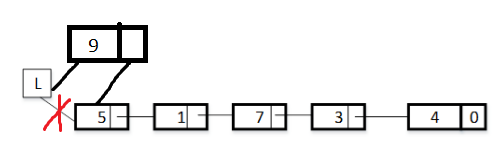
float data;

Node\* next;

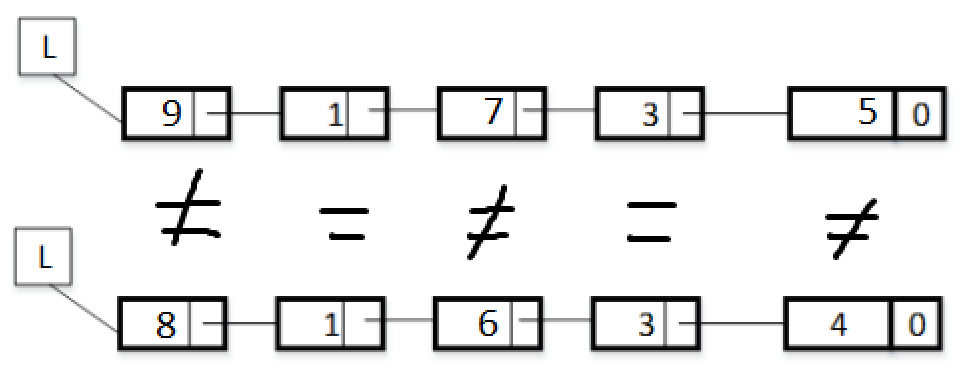
};

**2.2 Изобразить (рисунок) для каждой операции полученного списка процесс выполнения операции на существующем однонаправленном списке.**

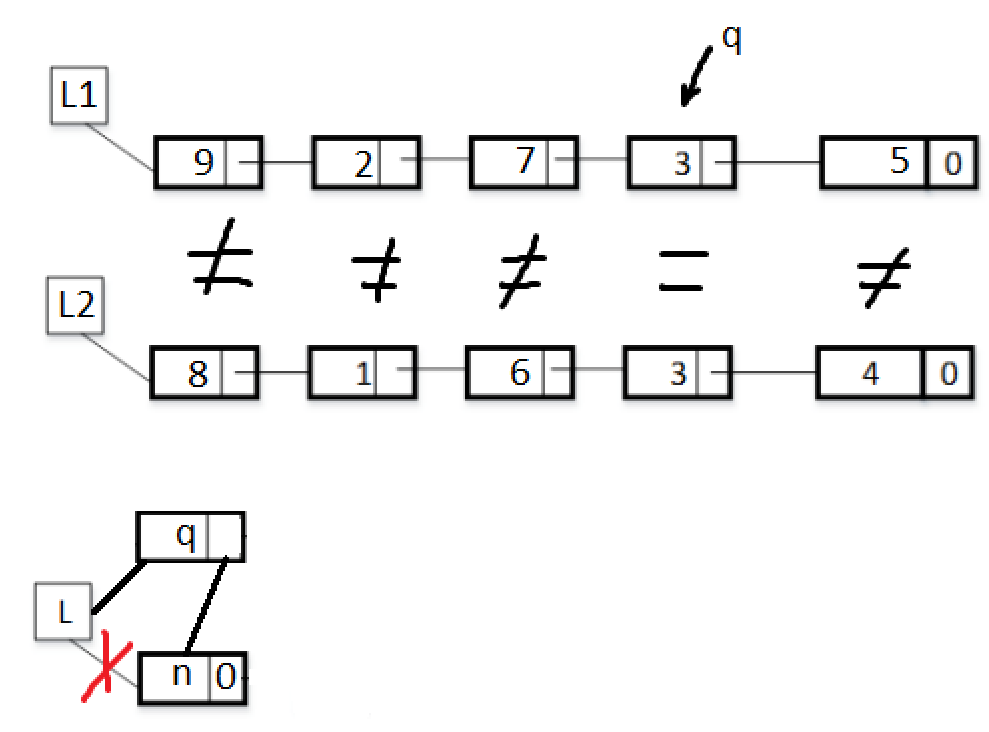
1) Создание списка L1 (или L2)



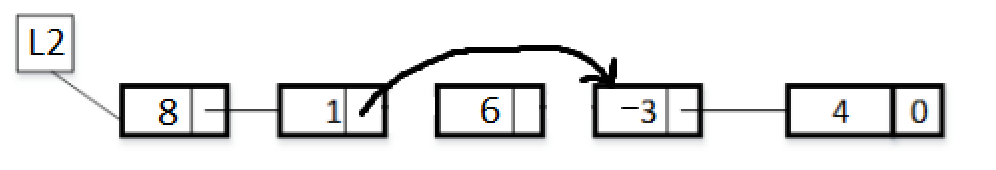
2) Поиск одинаковых элементов списков L1 и L2



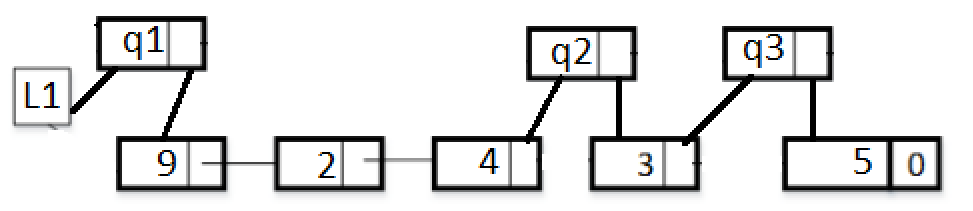
3) Внесение в список L одинаковых элементов списков L1 и L2



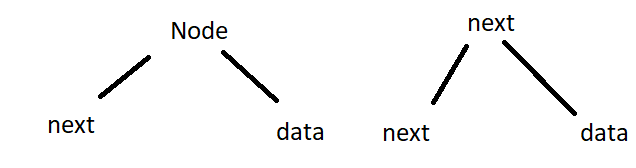
4) Удаление узла списка L2 расположенный перед узлом, содержащим отрицательное значение



5) Вставка узла в список L1 перед каждым узлом списка, содержащим нечетное значение



**2.3. Изобразить структуру данных, которая будет использоваться в операциях.**

****

**2.4. Пример алгоритма выполнения операции**

1) Создание списков, добавляя последующие элементы перед первым

2) Создание списка, добавляя общие элементы первых двух списков

3) Удаление узла в списке L2, расположенного перед узлом, содержащим отрицательное значение; Создаем связь элемента до удаляемого и после удаляемого

4) Добавление узла в список L1 перед узлом, содержащим нечетное значение

**2.5 Таблица тестов для тестирования каждой операции**

1) Создание списков

1.1) Создание списка L1

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 8.15 | -1.24 3.17 -5.74 16.08 -3.32 8.15 |
| -3.32 |  |
| 16.08 |  |
| -5.74 |  |
| 3.17 |  |
| -1.24 |  |

1.2) Создание списка L2

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 13.09 | 5.7 -10.01 3.17 -5.74 13.09 |
| -5.74 |  |
| 3.17 |  |
| -10.01 |  |
| 5.7 |  |

2) Создание списка L

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
|  | -5.74 3.17 |

3) Удаление узла в списке L2

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
|  | -10.01 -5.74 13.09 |

4) Добавление узла в список L1

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 4.4 | -1.24 4.4 3.17 -5.74 16.08 -3.32 4.4 8.15 |

**3 Код основной задачи**

**4 Тестирование**

Для подтверждения правильности работы, было проведено тестирование, в ходе которого была проверена правильность выполнения всех операций: создание списка, вывод, поиск общих узлов у двух списков, добавление и удаление узлов разных списков по заданным критериям.

**5 Ответы на вопросы**

1) Расскажите о трех уровнях представления данных в программной  
системе.

Существуют тру уровня представления данных: логический, хранения и физический. На логическом уровне работают с логическими структурами данных, отражающими реальные отношения, которые существуют между объектами и их характеристиками. Единицей информации на этом уровне является логическая запись. На логическом уровне устанавливается перечень признаков, полностью характеризующий описываемый класс объектов

На уровне хранения оперируют со структурами хранения – представлениями логической структуры данных в памяти ЭВМ. Структура хранения должна полностью отображать логическую структура данных и поддерживать ее в процессе функционирования АИС. Единицей информации на этом уровне также является логическая запись.

На физическом уровне представления данных оперируют с физическими структурами данных. На этом уровне решается задача реализации структуры хранения непосредственно в конкретной памяти конкретной ЭВМ. Единицей информации на этом уровне является физическая запись, представляющая участков носителя, на котором размещаются одна или несколько логических записей.

2) Что определяет тип данных?

Тип данных определяет формат представления данных в памяти компьютера, множество допустимых операций, применимых к этому типу.

3) Что определяет структура данных?

Структура данных определяет семантику, а также способы организации и управления ими

4) Расскажите о структуры хранения данных в компьютерных  
технологиях.

Основных структур три: линейная(список), табличная, иерархическая(дерево).

5) Дайте определение линейной структуре данных.

Линейная структура данных(список) – упорядоченная структура, в которой адрес данного однозначно определяется его номером (индексом).

6) Дайте определение структуре данных линейный список.

Линейный список – линейная последовательность элементов. Каждый из которых содержит указатели(ссылается) на своих соседей.

7) Дайте определение структуре данных стек.

Стек – абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO (Last In - First Out, «последним пришел – первым вышел»).

8) Дайте определение структуре данных очередь.

Очередь – структура данных, которая работает по принципу FIFO (First In – First Out, «первый пришел – первый вышел»)

9) Чем стек отличается от структуры данных линейный список?

В стеке невозможен доступ к элементам, кроме верхнего, а в линейном списке доступ возможен.

10) Какой из видов линейных списков лучше использовать, если  
нужно введенную последовательность вывести наоборот?

Стек, так как элементы будут извлекаться из структуры данных в обратном порядке их попадания внутрь.

11) Определите сложность алгоритма операции вставки элемента в  
i-ую позицию: а) массива; б) линейного списка.

а) O(n). Поиск сдвиг элементов правее i на один вправо за O(n) и сама вставка за O(1). Итого O(n).

б) O(n). Нужно получить указатель на старый i-ый элемент за O(n) и сама вставка элемента за O(1). Итого O(n).

12) Определите сложность алгоритма операции удаления элемента из  
i-ой позиции: а) массива; б) линейного однонаправленного списка.

а) O(n). Затирание следующим за текущим элементом текущего элемента и уменьшение размера на 1 за O(n).

б) O(n). Получить указатель на i-ый объект за O(n) и за O(1) изменить связи так, чтобы i-1 указывал на i+1. Итого O(n).

13) В чем суть трюка Вирта при выполнении операции удаления  
элемента из линейного однонаправленного списка?

Суть трюка Вирта описана в п. 12), т.е. нужно указатель на i-ый элемент перенаправить на i+1-ый элемент

14) Определите структуру узла однонаправленного списка.

В однонаправленном списке каждый узел должен содержать поле данных и поле указателя на следующий элемент.

15) Реализуйте алгоритм вывода линейного однонаправленного  
списка на языке С++.

void show\_list() {

for (Node\* node = head; node != nullptr; node = node->next)

std::cout << node->data << ' ';

}

16) Приведите фрагмент кода программы на языке С++ выполнения  
операции перемещения последнего элемента в начало списка.

void move\_last\_to\_start() {

Node\* node = head;

for (; node->next->next != nullptr; node = node->next) {}

node->next->next = head;

head = node->next;

node->next = nullptr;

}

17) Какой из действий лишнее в следующем фрагменте кода? Куда  
вставляется новый узел?  
struct Node {

int info;

Node\* next;

};

typedef Node\* List;

List L = new List;

void insertToList(List LL, int x) {

List q = new Node; q->info = x; q->next = 0;

if (LL == nullptr)

LL->next = q;

else {

q->next = LL->next;

LL->next = q;

}

}

Вставка в однонаправленный линейный список узла после LL.

List L=new List; - Лишнее действие