



МИНОБРНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение высшего
образования*

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания №6

Тема: Однонаправленный динамический список

Дисциплина Структуры и алгоритмы обработки данных

Выполнил студент Вагизов И.И.

группа ИКБО-01-20

Москва 2021

Содержание

1. Постановка задачи	3
2. Определение операций над списком.....	3
2.1. Определение структуры узла однонаправленного списка	3
2.2. Изображение процесса выполнения операций на списке	4
2.3. Используемая в операциях структура данных	14
3. Код программы.....	14
ВЫВОДЫ.....	16
СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ	16

1. Постановка задачи

Требуется реализовать программу решения следующих задач варианта №4 по использованию линейного однонаправленного списка:

1. Информационная часть узла содержит линейный однонаправленный список L1.
2. Разработать функцию для создания исходного списка, используя функцию вставки нового узла перед первым узлом.
3. Разработать функцию вывода списка.
4. Разработать функцию, которая переформирует список L1, переписав в начало списка его часть, начиная с заданной позиции.
5. Разработать функцию вставки узла в упорядоченный по не возрастанию список. Сформировать такой список L2.
6. Разработать функцию, которая удаляет из L2 все повторяющиеся значения, оставляя одно из них.
7. В основной программе выполнить тестирование каждой функции.

2. Определение операций над списком

2.1. Определение структуры узла однонаправленного списка

Согласно варианту №4 в качестве информационной части узла списка используются целые числа.

2.2. Изображение процесса выполнения операций на списке

1. Вставка нового узла перед первым:

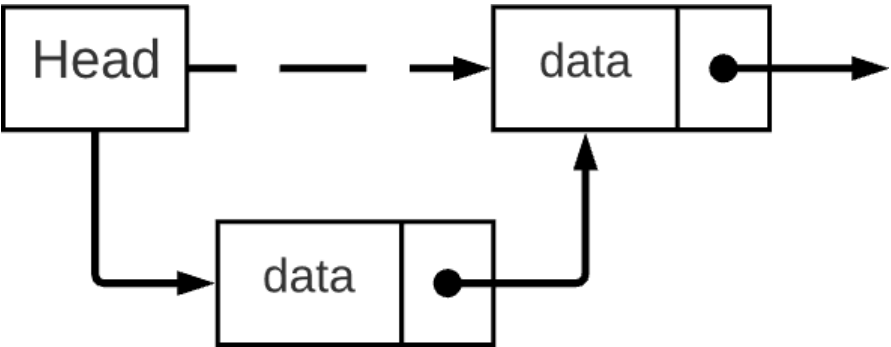


Рис. 1 Изображение вставки узла перед первым

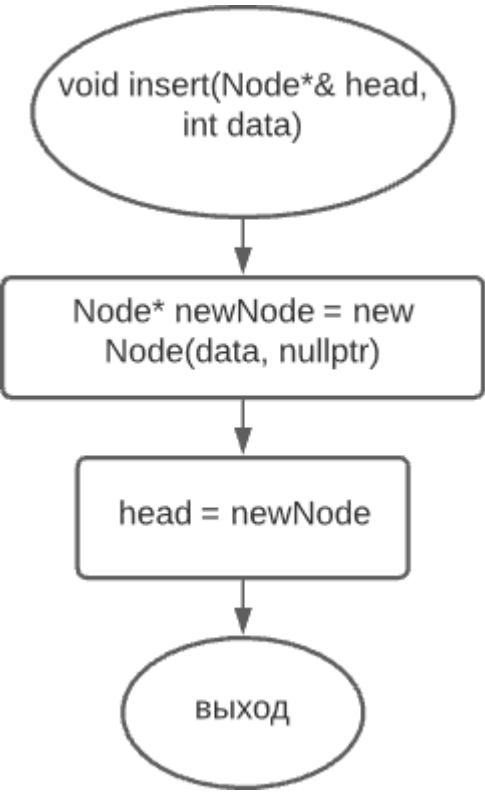
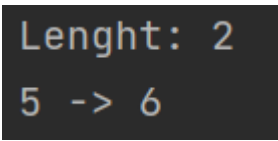


Рис. 2 Алгоритм вставки узла перед первым

Таблица 1 Набор тестов для функции

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Длина списка: 1 Значения: 5 Вставка: 6	5 6	

2	Длина списка: 2 Значения: 5 6 Вставка: 1	5 6 1	<code>Lenght: 3</code> <code>5 -> 6 -> 1</code>
3	Длина списка: 3 Значения: 5 6 1 Вставка: 9	5 6 1 9	<code>Lenght: 3</code> <code>5 -> 6 -> 1 -> 9</code>

2. Вывод списка на экран:

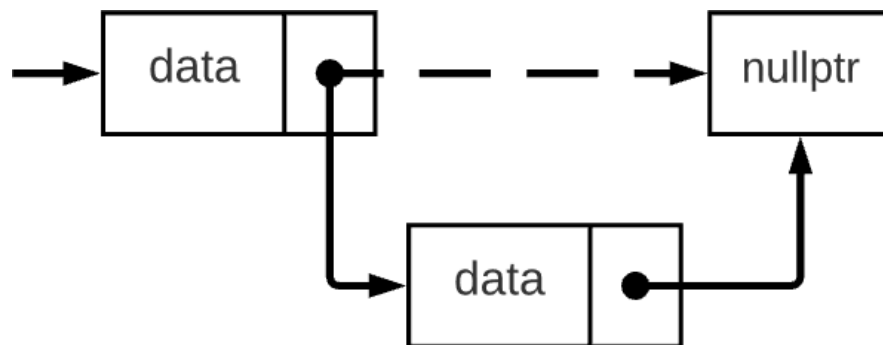


Рис. 3 Изображение вставки узла в конец списка

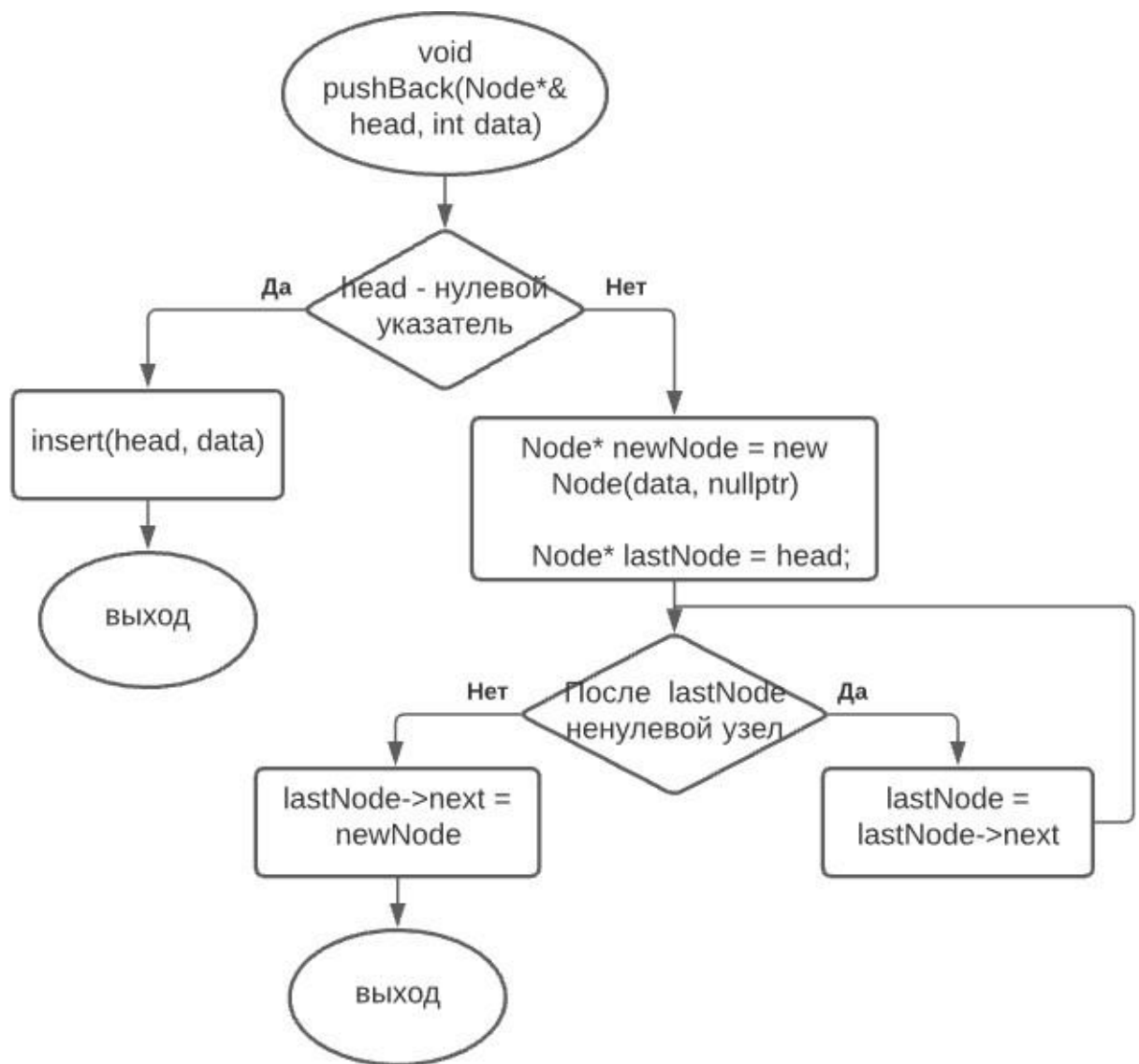


Рис. 4 Алгоритм вставки узла в конец списка

Таблица 2 Набор тестов для функции

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Длина списка: 1 Значения: 5	5	<code>Lenght: 1</code> <code>5</code>
2	Длина списка: 2 Значения: 5 6	5 -> 6	<code>Lenght: 2</code> <code>5 -> 6</code>
3	Длина списка: 3 Значения: 5 6 1	5 -> 6 -> 1	<code>Lenght: 3</code> <code>5 -> 6 -> 1</code>

3. Переформатирование списка:

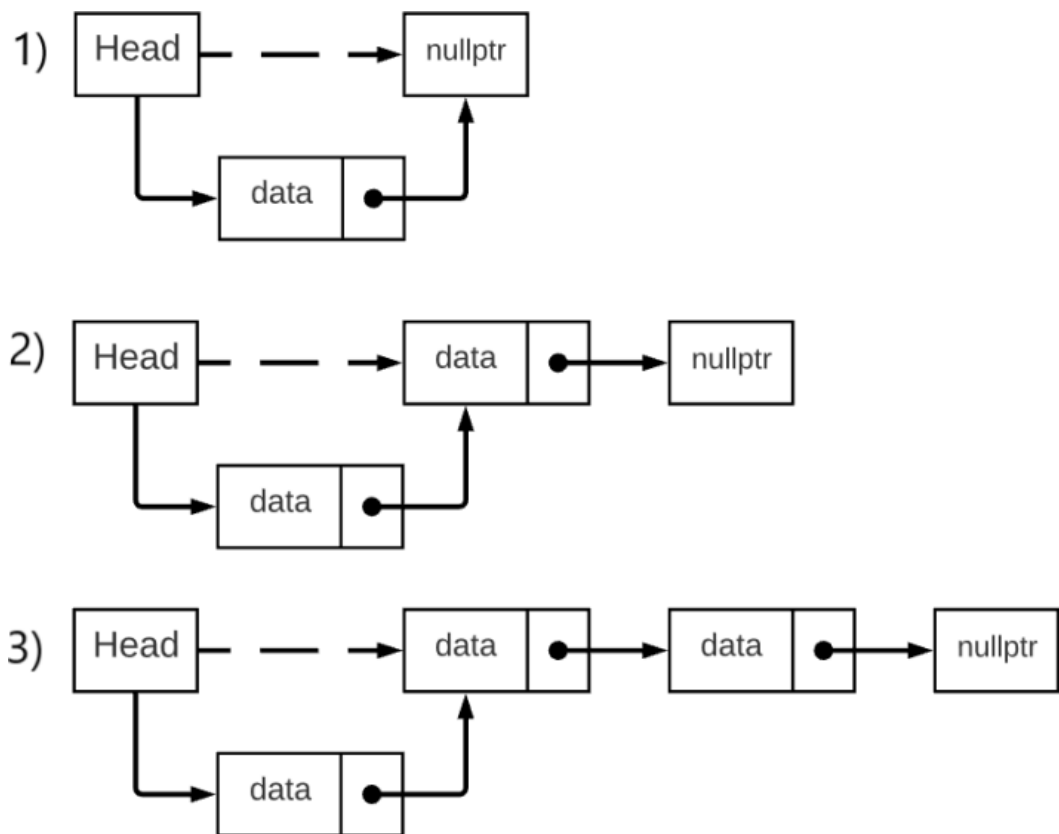


Рис. 5 Изображение создания списка на основе удаления элементов массива с заданного числа

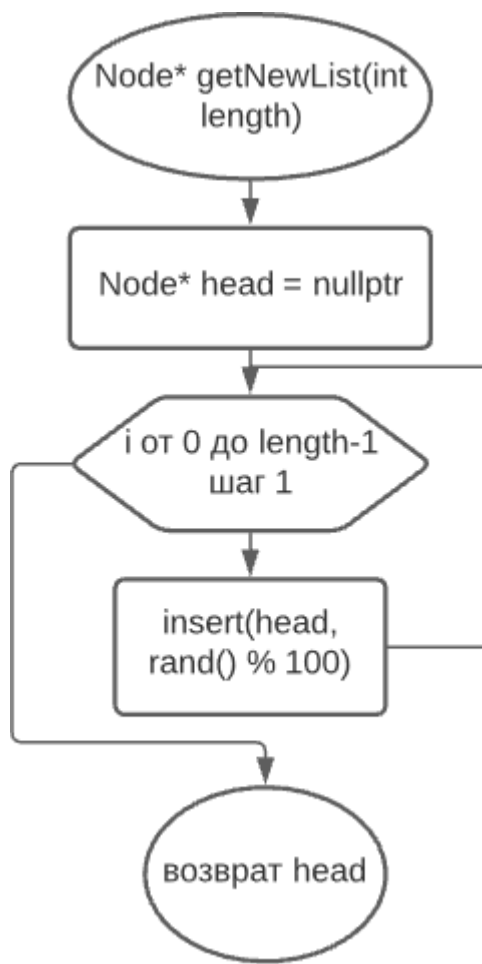


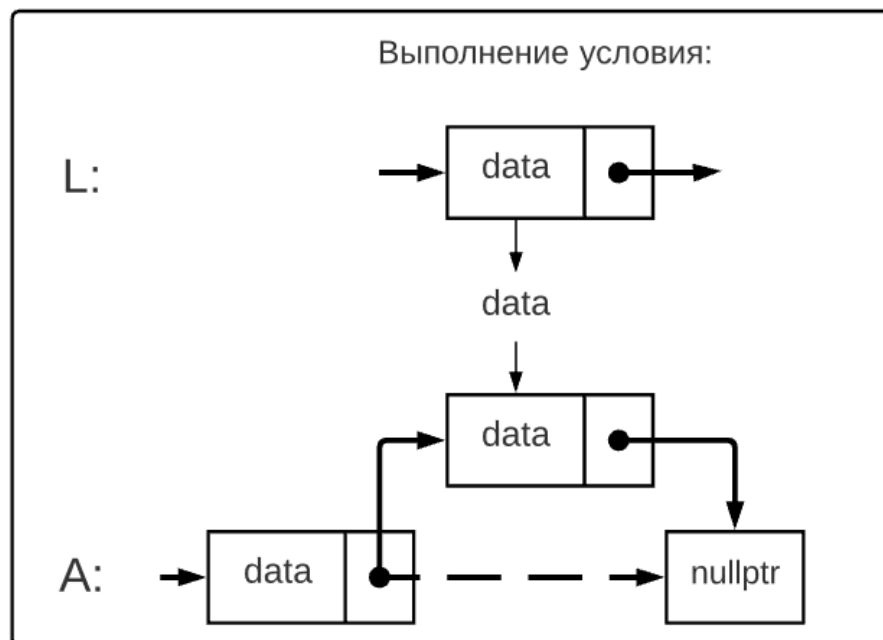
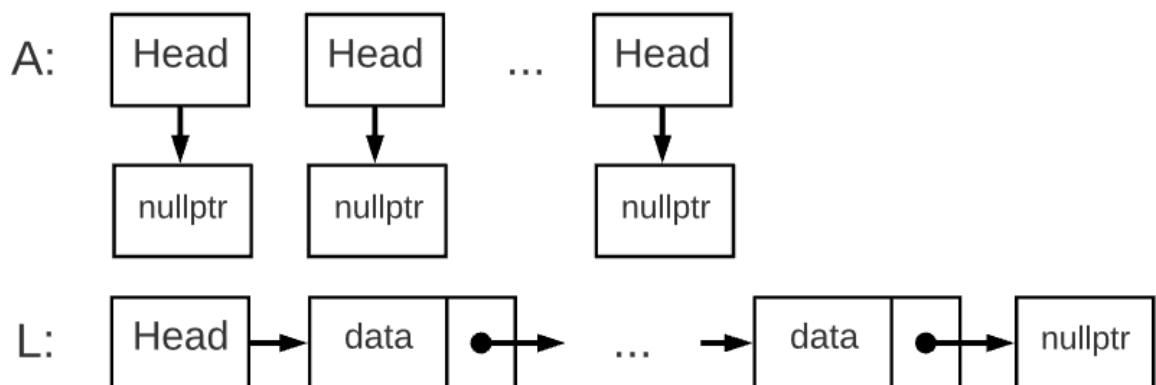
Рис. 6 Алгоритм переформатирования списка с заданного числа

Таблица 3 Набор тестов для функции

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Длина списка: 2 Значения: 5 6 Форматирование до: 1	5	<code>Lenght: 2</code> <code>5</code>
2	Длина списка: 3 Значения: 5 6 1 Форматирование до: 2	5 6	<code>Lenght: 3</code> <code>5 -> 6</code>
3	Длина списка: 4 Значения: 5 6 1 9 Форматирование до: 2	5 6	<code>Lenght: 4</code> <code>5 -> 6</code>

4. Вставка элемента в отсортированный список:

Рис. 7 Изображение вставки элемента в отсортированный список массива
Начальное состояние:



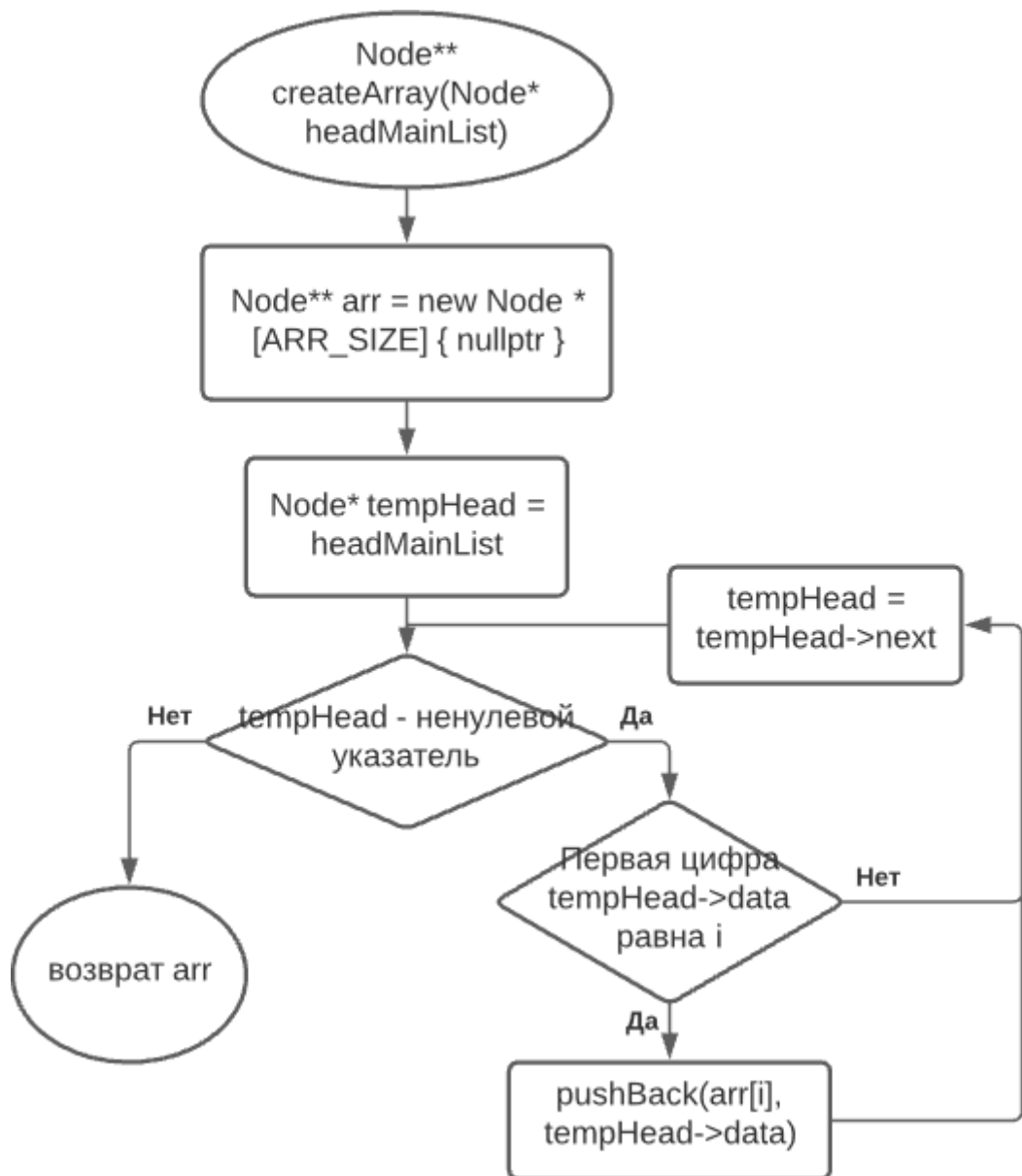


Рис. 8 Алгоритм включения числа из списка в список массива

Таблица 5 Набор тестов для функции

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Длина списка: 1 Значения: 5 Вставка: 6	6 -> 5	<code>Lenght: 1</code> <code>6 -> 5</code>
2	Длина списка: 2 Значения: 6 5 Вставка: 1	6 -> 5 -> 1	<code>Lenght: 2</code> <code>6 -> 5 -> 1</code>
3	Длина списка: 3 Значения: 6 5 1 Вставка: 9	9 -> 6 -> 5 -> 1	<code>Lenght: 3</code> <code>9 -> 6 -> 5 -> 1</code>

5. Удаление из списка повторяющихся элементов

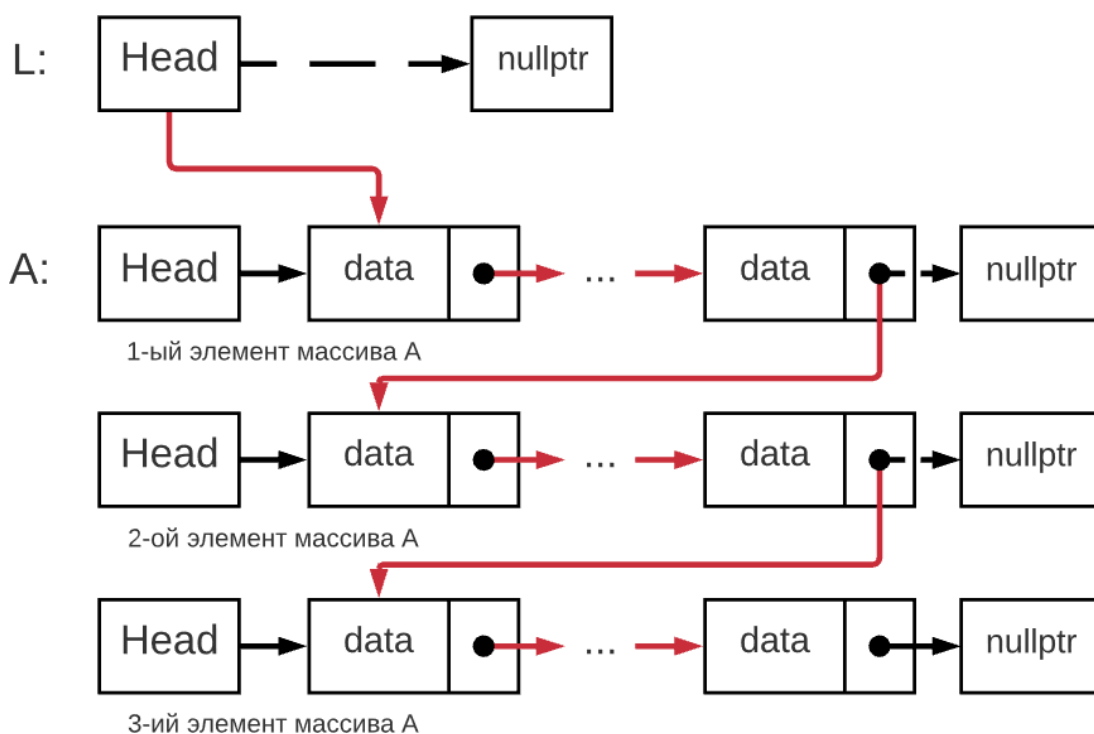


Рис. 9 Изображение удаления из списка повторяющихся элементов

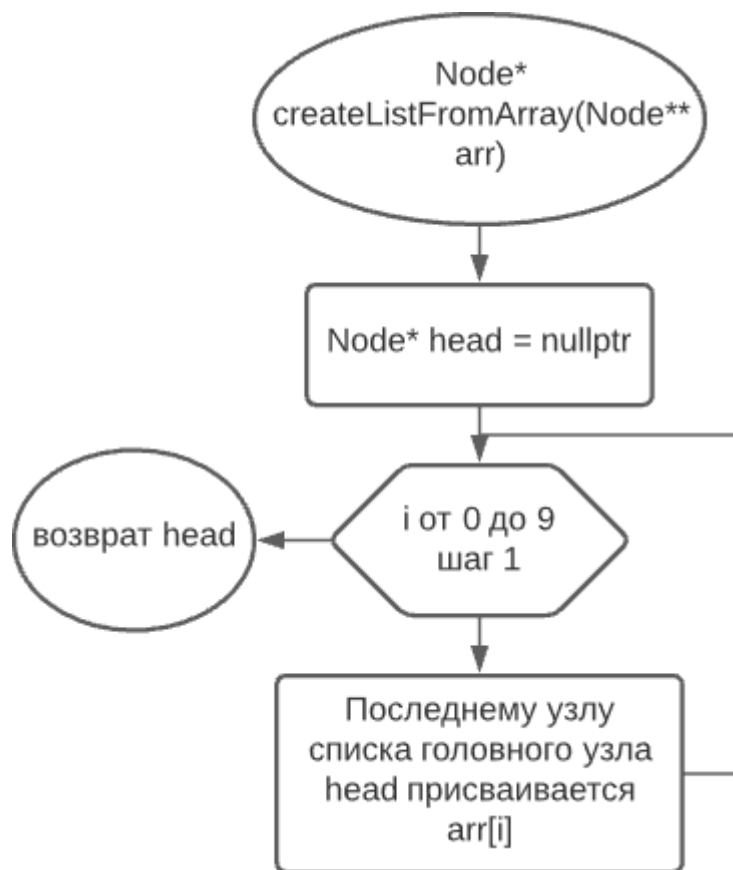


Рис. 10 Алгоритм удаления из списка повторяющихся элементов

Таблица 5 Набор тестов для функции

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Длина списка: 3 Значения: 5 1 1	5 -> 1	Lenght: 2 5 -> 1
2	Длина списка: 4 Значения: 5 2 2 2	5 -> 2	Lenght: 3 5 -> 2
3	Длина списка: 5 Значения: 5 2 2 2 1	5 -> 2 -> 1	Lenght: 4 5 -> 2 -> 1

2.3. Используемая в операциях структура данных

Структура данных – однонаправленный линейный список – состоит из узлов, каждый из которых включает информационную часть и указатель на следующий узел:

```
class Node{
public:
    int n;
    Node* next;
    Node (int n){
        this -> n = n;
        next = nullptr;
    }
}
```

3. Код программы

```
#include <iostream>

using namespace std;

class Node{
public:
    int n;
    Node* next;
    Node (int n){
        this -> n = n;
        next = nullptr;
    }
    // Node (int n, Node* next){
    // this -> next = next;
    // }
    void pushBack(Node* a,int n){
        if (a->next!= nullptr) pushBack(a->next,n);
        else {
            Node* next = new Node(n);
            a->next = next;
        }
    }
    void printArr(Node* a){
        cout << a -> n;
        if (a->next!= nullptr) {
            cout << " -> ";
            printArr(a -> next);
        }
    }
    void rewriteArr(int n,Node* a,int i){
        if (i<n-1) {
            i++;
            rewriteArr(n,a-> next,i);
        }
        else if (i==n-1){
            a -> next = nullptr;
        }
    }
    void push_in_sort(Node* a,int n){
        if (a->n>n){
            if (a->next == nullptr){
                Node* next = new Node(n);
            }
        }
    }
}
```

```

        a -> next = next;
    }
    else if (a->next -> n > n) {
        push_in_sort(a->next,n);
    }
    else {
        Node* next = new Node(n);
        next -> next = a -> next;
        a -> next = next;
    }
}
else {
    Node* next = new Node(a -> n);
    a -> n = n;
    Node* b = a -> next;
    a -> next = next;
    next -> next = b;
}
}

void delete_same(Node* a) {
    if (a -> next != nullptr) {
        if (a -> n == a->next->n) {
            if (getNext(a->next) != nullptr) {
                Node* next1 = getNext(a->next);
                while (next1 != nullptr and next1 -> n == a->n) {
                    a -> next = next1;
                    next1 = getNext(a->next);
                }
                if (next1 == nullptr) a->next = next1;
                else delete_same(a -> next);
            }
            else a -> next = nullptr;
        }
        else delete_same(a -> next);
    }
}

Node* getNext(Node* a) {
    return a->next;
}

};

int main()
{
    Node* a = new Node(5);
    int n,b;
    cin >> n;
    for (int i=0;i<n;++i) {
        cin >> b;
        a->push_in_sort(a,b);
    }
    cout <<"Lenght: "<<n<<endl;
    // a ->printArr(a);
    //cout <<endl;
    a ->delete_same(a);
    a ->printArr(a);
    return 0;
}

```

ВЫВОДЫ

В ходе практической работы был разработан однонаправленный динамический список, получены знания и практические навыки управления однонаправленным динамическим списком; реализованы необходимые функции взаимодействия со списком, включая задания индивидуального варианта. Каждая выполняющаяся над списком операция прошла тестирование.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Кораблин Ю.П., Сыромятников В.П., Скворцова Л.А. Учебно-методическое пособие Структуры и алгоритмы обработки данных, М.:МИРЭА, 2020
2. Никлаус Вирт Алгоритмы и структуры данных. Классика программирования – М.:ДМК Пресс, 2016. — 272 с.
3. Круз Р. Л. Структуры данных и проектирование программ / пер. с англ. — 3-е издание / Р.Л. Круз. – М.:Лаборатория знаний, 2017. — 768
4. Алгоритмы и структуры данных: связный список // tproger.ru [Электронный ресурс].- <https://tproger.ru/translations/linked-list-for-beginners/>- дата обращения(26.04.2021).