МИНОБРНАУКИ РОСИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания №1.5
Тема:
Однонаправленный динамический список
Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент: Павлов Н.С.

Группа: ИКБО-30-23

Москва 2024

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Получить знания и практические навыки управления динамическим однонаправленным списком

2. ВЫПОЛНЕНИЕ ЗАДАНИЯ

2.1 ФОРМУЛИРОВКА ЗАДАЧИ

Реализуйте программу решения задачи варианта по использованию линейного однонаправленного списка.

Требования для всех вариантов:

1. Информационная часть узла определена вариантом
2. Разработать функции вставки нового узла перед первым узлом и удаления узла по ключу.
3. Реализуйте возможность а) создания нового списка вручную, а также б) использования уже готового списка для тестирования заданий индивидуального варианта.
4. Разработать функцию вывода списка в консоль.
5. Разработать функции согласно индивидуальному варианту. При необходимости можно добавлять вспомогательные функции, декомпозируя задачу.
6. Реализуйте текстовое пользовательское меню.
7. В основной программе выполните тестирование каждой функции (пункты 2-5).

Персональный вариант: Дан массив из n указателей на вершины списков. Структура узла списка содержит ключ (информационная часть узла) и ссылку на следующий узел.

1. Разработать функцию, которая вставляет переданный в качестве параметра ключ в i-ый список массива. Индекс i определяется по правилу: i=key%n. Некоторые элементы массива могут остаться nullptr.
2. Разработать функцию для удаления значение ключа из списка.
3. Разработать функцию, которая находит узел со значением ключа и возвращает указатель на найденный узел.

2.2 СПИСОК ОПЕРАЦИЙ НАД СПИСКОМ

2.2.1 Структура узла

У каждого узла (Node) будет информационное поле val со значением типа int, информационное поле next типа указателя на объект Node, а также конструктор экземпляра узла, принимающий значение _val типа int.

2.2.2 Схемы операций над списком

1. Функция вставки ключа в i-й список массива

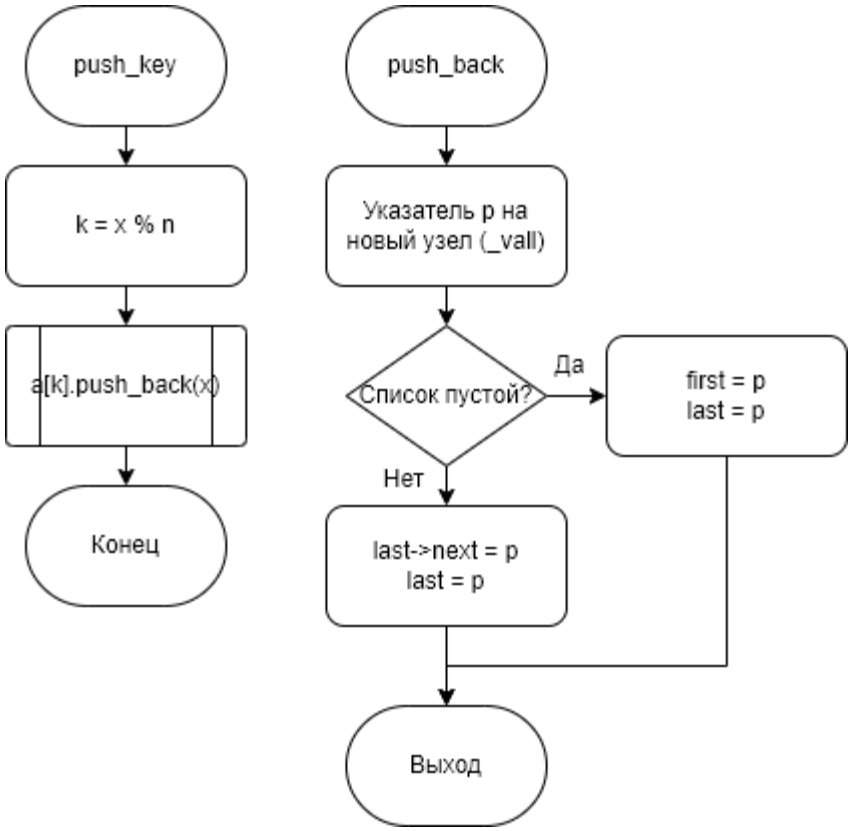

Рисунок 1 – Блок-схема функции push_key

2. Функция удаления ключа из списка

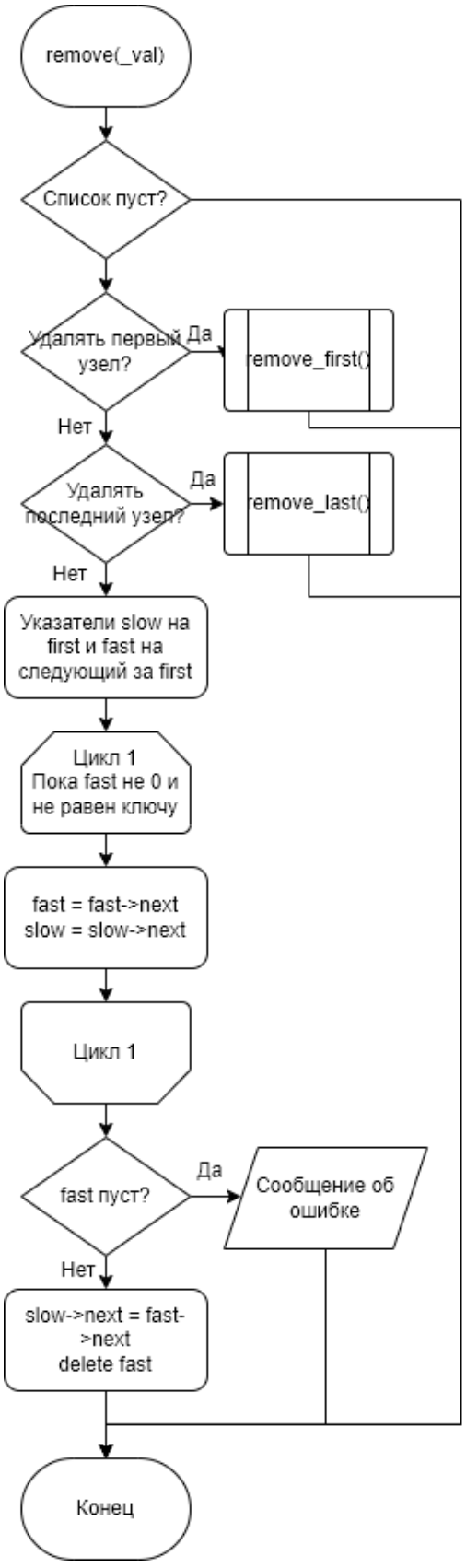

Рисунок 2 – Блок-схема функции remove

3. Функция нахождения узла со значением узла и возврата указателя на данный узел

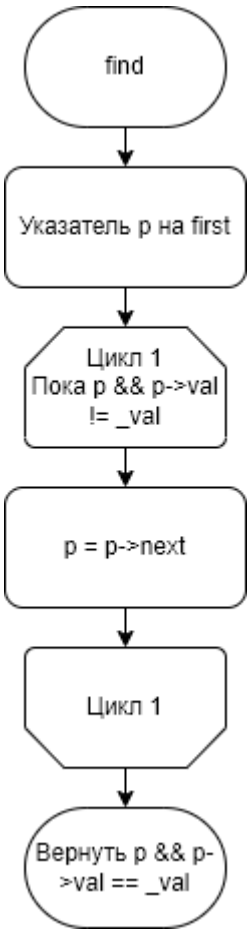

Рисунок 3 – Блок-схема функции find

2.2.3 Структура данных

Ниже представлен схематичный рисунок используемой структуры

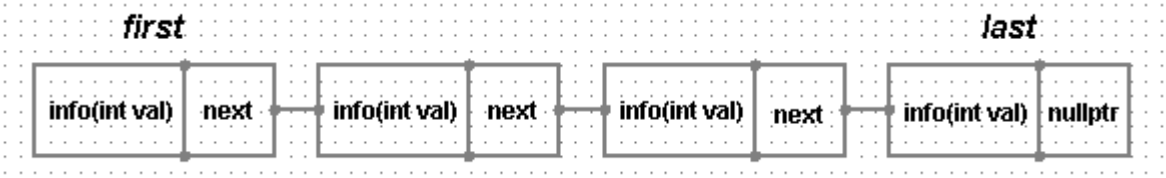

Рисунок 4 – Структура данных

2.2.4 Алгоритм выполнения операций

1. Функция вставки ключа в i-й список массива

Функция определяет номер списка, в который требуется выполнить вставку, и приписывает ключевой элемент к концу списка. Номер списка в

массиве определяется остатком от деления ключевого элемента на количество списков в массиве.

2. Функция удаления ключа из списка

Функция является встроенной в структуру list. Если список не пуст, то возможны три случая:

- узел с искомым значением равен первому;
- узел с искомым значением равен последнему;
- не первый и не второй случаи.

Первый случай: сравниваем значение первого узла с ключом, если значения совпадают, тогда вызываем функцию удаления первого узла.

Второй случай: сравниваем значение последнего узла с ключом, если значения совпадают, тогда вызываем функцию удаления последнего узла.

Третий случай:

Создаются указатели slow на первый узел, и fast – на следующий после первого. Затем, пока fast не пустой и пока значение текущего узла fast не равно ключу, при каждой итерации перенаправляем slow и fast на следующий после них узел. Если указатель fast пустой, то сообщаем об ошибке, иначе просто удаляем узел fast.

3. Функция нахождения узла со значением узла и возврата указателя на данный узел

Функция является встроенной в структуру list. Для поиска узла в списке по ключевому значению в структуре list добавим функцию обхода списка, пока указатель p не пустой и пока значение узла p не равно ключу, после чего возвращаем найденный узел, если он есть

2.2.5 Контрольные тесты для операций

1. Функция вставки ключа в i-й список массива

Входные данные	Выходные данные
1 2 3 4 5 6 7 8 9 0 55	1 2 3 4 5 6 7 8 9 0 55
345 567 34 34	345 567 34 34
8 8 8 8 8 12	8 8 8 8 8 12

2. Функция удаления ключа из списка

Входные данные	Выходные данные
1 2 3 4 5 6 7 8 9 0 6	1 2 3 4 5 7 8 9 0
345 567 34 56	345 567 34
8 8 8 8 8 8	

3. Функция нахождения узла со значением узла и возврата указателя на данный узел

Входные данные	Выходные данные
1 2 3 4 5 6 7 8 9 0 6	Адрес 6
345 567 34 56	Не найдено
8 8 8 8 8 8	Адрес первой 8

2.3 РЕАЛИЗАЦИЯ АЛГОРИТМА

```
struct Node
{
    int val;
    Node* next;
    Node(int _val) : val(_val), next(nullptr) {}
};

struct list
{
    Node* first;
    Node* last;
    list() : first(nullptr), last(nullptr) {}
    bool is_empty()
    {
        return first == nullptr;
    }
    void push_back(int _val) {
        Node* p = new Node(_val);
        if (is_empty()) {
            first = p;
            last = p;
            return;
        }
        last->next = p;
        last = p;
    }
```

Рисунок 5 – Реализация алгоритма (Часть 1)

```
    void push_behind(int _val) {
        Node* p = new Node(_val);
        if (is_empty()) {
            first = p;
            last = p;
            return;
        }
        p->next = first;
        first = p;
    }
    Node* find(int _val) {
        Node* p = first;
        while (p && p->val != _val)
            p = p->next;
        return (p && p->val == _val) ? p : nullptr;
    }
    void remove_first() {
        if (is_empty())
            return;
        Node* p = first;
        first = p->next;
        delete p;
    }
```

Рисунок 6 – Реализация алгоритма (Часть 2)

```
void remove_last() {
    if (is_empty())
        return;
    if (first == last) {
        remove_first();
        return;
    }
    Node* p = first;
    while (p->next != last)
        p = p->next;
    p->next = nullptr;
    delete last;
    last = p;
}
void remove(int _val) {
    if (is_empty())
        return;
    if (first->val == _val) {
        remove_first();
        return;
    }
    else if (last->val == _val) {
        remove_last();
        return;
    }
    Node* slow = first;
    Node* fast = first->next;
    while (fast && fast->val != _val) {
        fast = fast->next;
        slow = slow->next;
    }
    if (!fast) {
        cout << "Этот элемет не найден" << endl;
        return;
    }
    slow->next = fast->next;
    delete fast;
}
```

```
    void print() {
        if (is_empty()) return;
        Node* p = first;
        while (p) {
            cout << p->val << " ";
            p = p->next;
        }
        cout << endl;
    }
};

list create(int n) {
    list b;
    int k;
    for (int i = 0; i < n; i++)
    {
        cout << "Введите элемент: ";
        cin >> k;
        b.push_back(k);
    }
    return b;
}

void push_key(list* a, int x, int n)
{
    int k;
    k = x % n;
    a[k].push_back(x);
}
```

Рисунок 7-8 – Реализация алгоритма (Часть 3-4)

2.4 ТЕСТИРОВАНИЕ АЛГОРИТМА

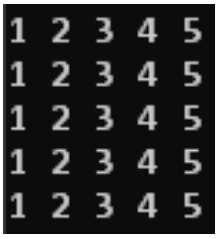

Рисунок 9 – Исходное заполнение списков числами

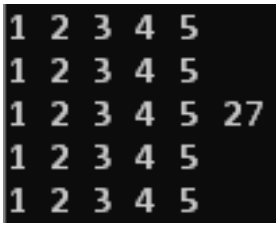

Рисунок 10 – Результат добавление числа, используя функцию

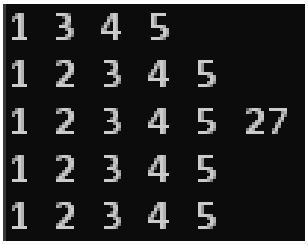

Рисунок 11 – Результат удаления ключа из списка

Рисунок 12 – Получений адреса узла со значением 4

3 ВЫВОД

Изучена такая структура как односвязный список, а также работа с ней. Реализовано несколько функций для работы с этой структурой. Проверена корректность их работы.

СПИСОК ИНФОРМАЦТОННЫХ ИСТОЧНИКОВ

1. Рысин, М. Л. Введение в структуры и алгоритмы обработки данных. Часть 1. Сложность алгоритмов. Сортировки. Линейные структуры данных. Поиск в таблице. : учеб. пособие / М. Л. Рысин, М. В. Сартаков, М. Б. Туманова ; МИРЭА – Российский технологический университет, 2022. – 109 с. – ISBN 978-5-7339-1612-5.

2. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения : межгосударственный стандарт : дата введения 1992-01-01 / Федеральное агентство по техническому регулированию. – Изд. официальное. – Москва : Стандартинформ, 2010. – 23 с.