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

РТУ МИРЭА

Отчет по выполнению практического задания №5
Тема:
«Однонаправленный динамический список»
Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент: Подоплелов А.С.

Группа: ИНБО-21-23

Москва - 2024

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Цель: получить знания и практические навыки управления динамическим однонаправленным динамическим списком.

1.1 УСЛОВИЕ ЗАДАНИЯ

Разработать реализацию односвязного списка средствами языка C++ с информационной частью узла списка согласно индивидуальному варианту. Реализовать три функции для работы над односвязным списком согласно индивидуальному варианту. Мой вариант под номером 6.

Тип информационной части узла — double.

Дан линейный однонаправленный список L.

Функции:

1. Разработать функцию, которая вставляет перед последним узлом два новых узла.

2. Удаляет из списка L первое отрицательное значение, если оно присутствует в списке.

3. Найти в списке L максимальное значение и перенести его узел в конец списка.

2 РЕШЕНИЕ ЗАДАЧИ

2.1 РЕАЛИЗАЦИЯ ОДНОСВЯЗНОГО СПИСКА И ЕГО ОСНОВНЫХ ФУНКЦИЙ

Разработаем схему узла однонаправленного списка, а также его реализацию на языке C++ (См. Рисунок 1).

Представим схему узла в качестве таблицы (См. Таблица 1.).

Таблица 1 — Схема узла однонаправленного списка.

Node	
Тип	**Имя свойства**
double	val
Указатель	next

Теперь реализуем данную схему с помощью языка C++. Здесь поле «val» - информационное, «next» - связь, «Node» - конструктор экземпляра узла списка.

```
struct Node {
    double val; //Тип данных согласно варианту
    Node* next;
    Node(double _val): val(_val), next(nullptr) {}
};
```

Рисунок 1 — Реализация узла с помощью языка C++

Далее реализуем сам односвязный список, включающий функции проверки наличия узлов в списке - «is_empty», добавления узла в конец списка - «_add», а также вывод списка в консоль - «print» (См. Рисунок 2).

```
struct list {
    Node* first;
    Node* last;
    list() : first(nullptr), last(nullptr) {}
    bool is_empty() {
        return first == nullptr;
    }
    void _add(double _val) {
        Node* ref = new Node(_val);
        if (is_empty()) {
            first = ref;
            last = ref;
            return;
        }
        last->next = ref;
        last = ref;
    }
    void print() {
        if (is_empty()) {
            return;
        };
        Node* ref = first;
        while (ref) {
            cout << ref -> val << " ";
            ref = ref->next;
        }
        cout << "\n";
    }
};
```

Рисунок 2 — Реализация односвязного списка с помощью языка C++

Для поиска узла в списке по ключевому значению в структуре «list» добавим функцию обхода списка, пока указатель «ref» не пустой и пока

значение узла «ref» не равно ключу, после чего возвращаем найденный узел, если он есть (См. Рисунок 3).

```
Node* find(double _val) {
    Node* ref = first;
    while (ref && ref->val != _val) ref = ref->next;
        return (ref && ref->val == _val) ? ref : nullptr;
}
```

Рисунок 3 — Реализация поиска по ключу с помощью языка C++

Остается последняя базовая функции, которая позволит нам удалять узел из списка по заданному ключу (См. Рисунок 4).

```
void remove(double _val) {
    if (is_empty()) return;
    if (first->val == _val) {
        Node* ref = first;
        first = ref->next;
        delete ref;
        return;
    }
    else if (last->val == _val){
        if (is_empty()) {
            return;
        }
        if (first == last) {
            Node* ref = first;
            first = ref->next;
            delete ref;
            return;
        }
        Node* ref = first;
        while (ref->next != last) ref = ref->next;
        ref->next = nullptr;
        delete last;
        last = ref;
        return;
    }
    Node* slow = first;
    Node* fast = first->next;
    while (fast && fast-> val != _val) {
        fast = fast->next;
        slow = slow->next;
    }
    if (!fast) {
        cout << "Cancel." << "\n";
        return;
    }
    slow->next = fast->next;
    delete fast;
}
```

Рисунок 4 — Реализация функции удаления по ключу с помощью языка C++

2.2 РАЗРАБОТКА И РЕАЛИЗАЦИЯ ФУНКЦИЙ ИНДИВИДУАЛЬНОГО ВАРИАНТА

Реализация функций ручного и автоматического заполнения односвязного динамического списка (См. Рисунок 5).

```
list Create(int len) {
    list M;
    double val;
    cout << "1./ -> Руками." << "\n";
    for (int i = 0; i < len; i++) {
        cout << "Введите " << len - i << " элемент." << "\n";
        cin >> val;
        M._add(val);
    }
    M.print();
    cout << "Успех." << "\n";
    return M;
}
list autoCreate(int len) {
    list M;
    cout << "2./ -> Генератором." << "\n";
    srand(time(NULL));
    for (int i = 0; i < len; i++) {
        M._add(rand() % 10);
    }
    M.print();
    cout << "Успех." << "\n";
    return M;
}
```

Рисунок 5 — Реализация функций заполнения списка с помощью языка C++

Реализация функции, которая вставляет перед последним узлом два новых узла (См. Рисунок 6).

```
//Разработать функцию, которая вставляет перед последним узлом два новых узла.
void addingtwo() {
    int k = 0;
    while (k != 2) {
        int j_ = k + 1;
        cout << "Введите элемент #" << j_ << "\n";
        double _j;
        cin >> _j;
        _add(_j);
        k++;
    }
}
```

Рисунок 6 — Реализация функции, которая вставляет перед последним узлом два новых узла с помощью языка C++

Реализация функции, которая удаляет из списка L первое отрицательное значение, если оно присутствует в списке (См. Рисунок 7).

```
//Удаляет из списка L первое отрицательное значение, если оно присутствует в списке.
void removeMINUS() {
    if (is_empty()) {
        return;
    };
    Node* ref = first;
    while (ref != nullptr) {
        if (ref->val >= 0) {
            ref = ref->next;
        }
        else {
            remove(ref->val);
            break;
        }
    }
}
```

Рисунок 7 — Реализация функции, которая удаляет из списка L первое отрицательное значение, если оно присутствует в списке с помощью языка C++

Реализация функции, которая позволяет найти в списке L максимальное значение и перенести его узел в конец списка (См. Рисунок 8).

```
//Найти в списке L максимальное значение и перенести его узел в конец списка.
void findMAX() {
    int _cnt = 0, cnt_ = 0;
    if (is_empty()) {
        return;
    };
    Node* ref = first;
    while (ref != nullptr) {
        ref = ref->next;
        _cnt++;
    }
    double* M = new double[_cnt];
    Node* ref_ = first;
    while (ref_ != nullptr) {
        M[cnt_] = ref_->val;
        ref_ = ref_->next;
        cnt_++;
    }
    double temp;
    for (int i = 0; i < _cnt - 1; i++) {
        for (int j = 0; j < _cnt - i - 1; j++) {
            if (M[j] > M[j + 1]) {
                // меняем элементы местами
                temp = M[j];
                M[j] = M[j + 1];
                M[j + 1] = temp;
            }
        }
    }
    double MAX = M[_cnt - 1]; //Максимальный элемент
    remove(MAX);
    _add(MAX);
    delete[] M;
}
```

Рисунок 8 — Реализация функции, которая позволяет найти в списке L максимальное значение и перенести его узел в конец списка. с помощью языка C++

2.3 ТЕСТИРОВАНИЕ РАБОТЫ АЛГОРИТМА

Результаты тестирования функций программы при различных входных данных представлены в таблице 2.

Таблица 2 — Результаты тестирования функций и методов программы

Функция/Метод	Входные данные	Ожидаемый результат	Полученный результат
is_empty()	1	false	false
	[]	true	true
_add(double _val)	8 | 1, 5, 7	1, 5, 7, 8	1, 5, 7, 8
print()	1, 5, 7, 8	1, 5, 7, 8	1, 5, 7, 8
	[]	«Cancel.»	«Cancel.»
remove(double _val)	5 | 4,7,5	4, 7	4, 7
	4 | 7, 5, 3	«Cancel»	«Cancel»
Create()	2 | 6, 7	6, 7	6, 7
autoCreate()	4	a, b, c, d	42, -4, 203,45
addingtwo()	4,5 | 3, 12, 8	3, 12, 8, 4, 5	3, 12, 8, 4, 5
removeMINUS()	4, 5, -1, 6	4, 5, 6	4, 5, 6
findMAX()	4, 15, 5, 0	4, 5, 0, 15	4, 5, 0, 15

Результаты выполнения программы в консоли (См. Рисунок 9 — 11).

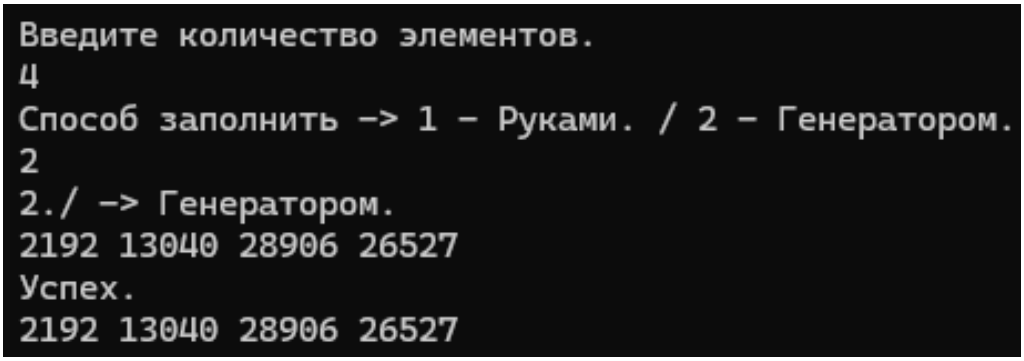

Рисунок 9 — Результат программы, где генератор случайных чисел вставляет элементы в односвязный список

```
Введите количество элементов.
3
Способ заполнить -> 1 - Руками. / 2 - Генератором.
1
1./ -> Руками.
Введите 3 элемент.
4
Введите 2 элемент.
6
Введите 1 элемент.
-2
4 6 -2
Успех.
Односвязный список.
4 6 -2
```

Рисунок 10 — Результат программы, где идет реализация ручного способа заполнения односвязного списка

```
Введите количество элементов.
5
Способ заполнить -> 1 - Руками. / 2 - Генератором.
1
1./ -> Руками.
Введите 5 элемент.
8
Введите 4 элемент.
5
Введите 3 элемент.
24
Введите 2 элемент.
-9
Введите 1 элемент.
-3
8 5 24 -9 -3
Успех.
Односвязный список.
8 5 24 -9 -3
Задание #2. Вариант 6.
8 5 24 -3
Задание #3. Вариант 6.
8 5 -3 24
```

Рисунок 11 — Результат программы, где идет реализация ручного способа заполнения односвязного списка, а также двух функций, которые нужно было реализовать в индивидуальном варианте

По результатам тестирования можно сделать вывод, что все функции и методы являются рабочими.

ВЫВОД

В ходе выполнения работы были получены знания и практические навыки управления динамическим однонаправленным списком. Были реализованы все необходимые методы для работы с однонаправленным списком, а также функции согласно индивидуальному варианту.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Структуры данных и проектирование программ : Пер. с англ. / Р. Круз. — М.: БИНОМ. Лаборатория знаний, 2017. — 766 с.

2. Полный справочник по C++ : Пер. с англ. / Г. Шилдт. — М.: ООО "И.Д.Вильямс", 2016. — 796 с.: ил. — Предм. указ.: с. 787-796