Министерство высшего образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

**лабораторная работа №11 - списки**

Дисциплина: основы алгоритмизации и программирования

Выполнил:

Студент группы РИС-22-1б

Баженов Тимофей

Проверил:

Доцент кафедры ИТАС

Полякова Ольга Андреевна

**Пермь, 2023**

**1.Постановка задачи**

Создать основные функции для работы со списком.

**2. Анализ решения**

1. Объявляем структуру Node, которая имеет два поля: int data (значение узла) и Node\*next (указатель на след. Узел)
2. Объявляем структуру List, которая имеет два поля: указатели на головной и хвостовой узлы. Н.з. указателей nullptr.
3. Функция void pushback с параметрами: элемент структуры List и значение поля data. Динамически создаем новый узел, инициализируем поле data данного узла. Если указатель на головной элемент листа nullptr, значит лист пуст => головной и хвостовой указатели в списке будут указывать на один и тот же элемент. Иначе хвостовой указатель указывает на новый элемент списка, затем переставляем хвостовой указатель на последний элемент.
4. Функция void print выводит список.
5. Функция void delete\_node (параметр лист). Проверка: лист пустой?

Если да, то конец функции. Если нет, то проверяем первый текущий элемент % 2 == 0, если да = > записываем в remove, затем delete remove.

Переход к след. Узлу и перестановка головного указателя на след элемент.

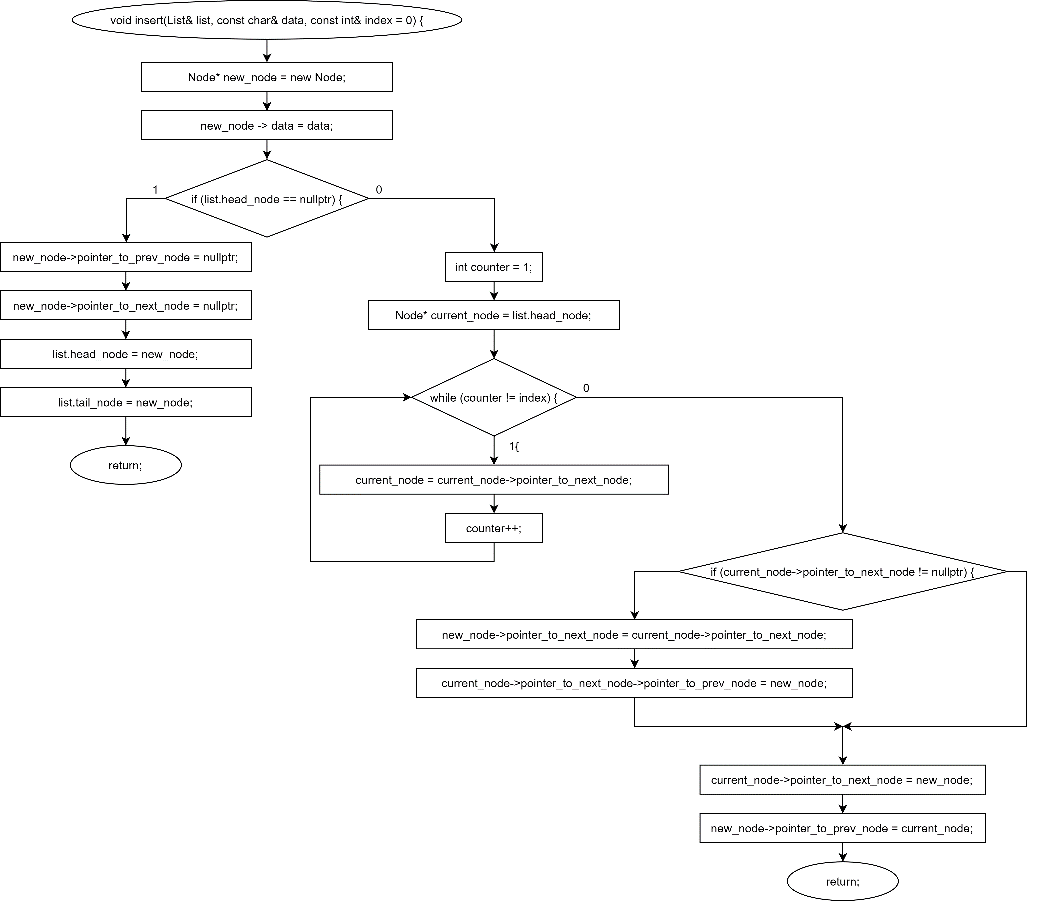
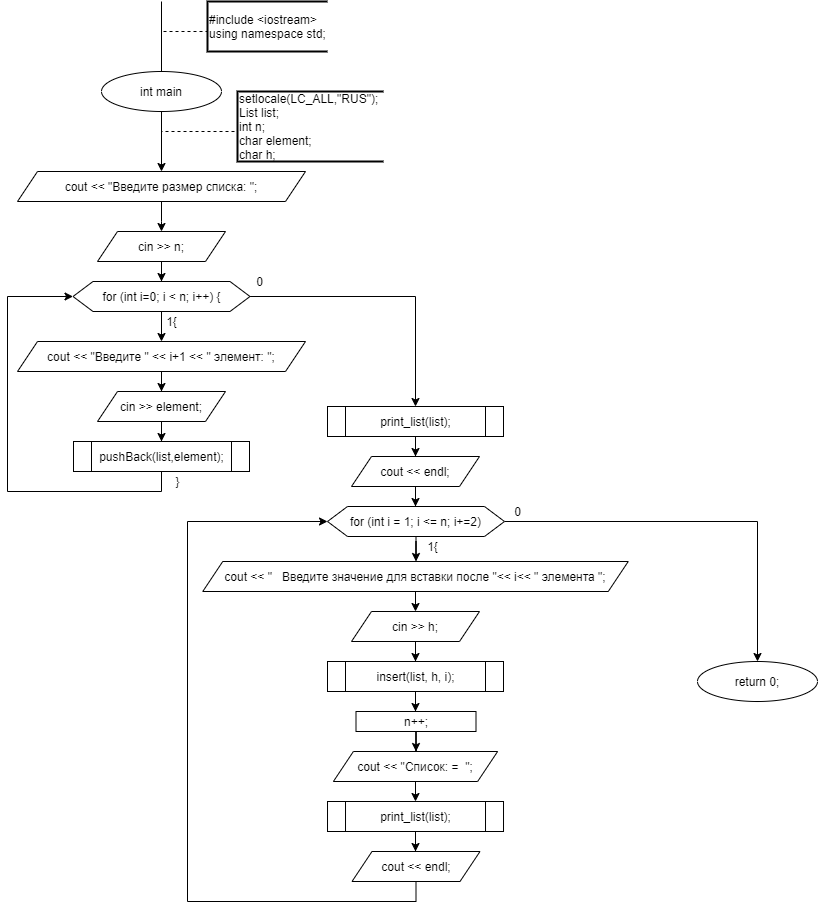
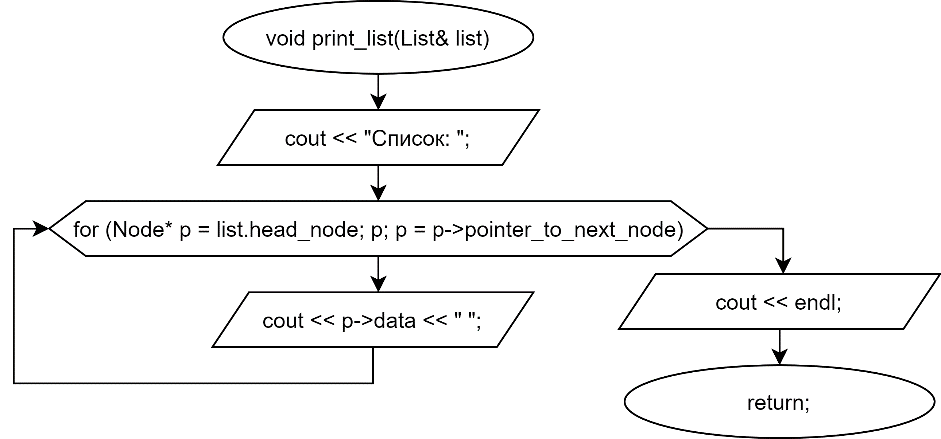
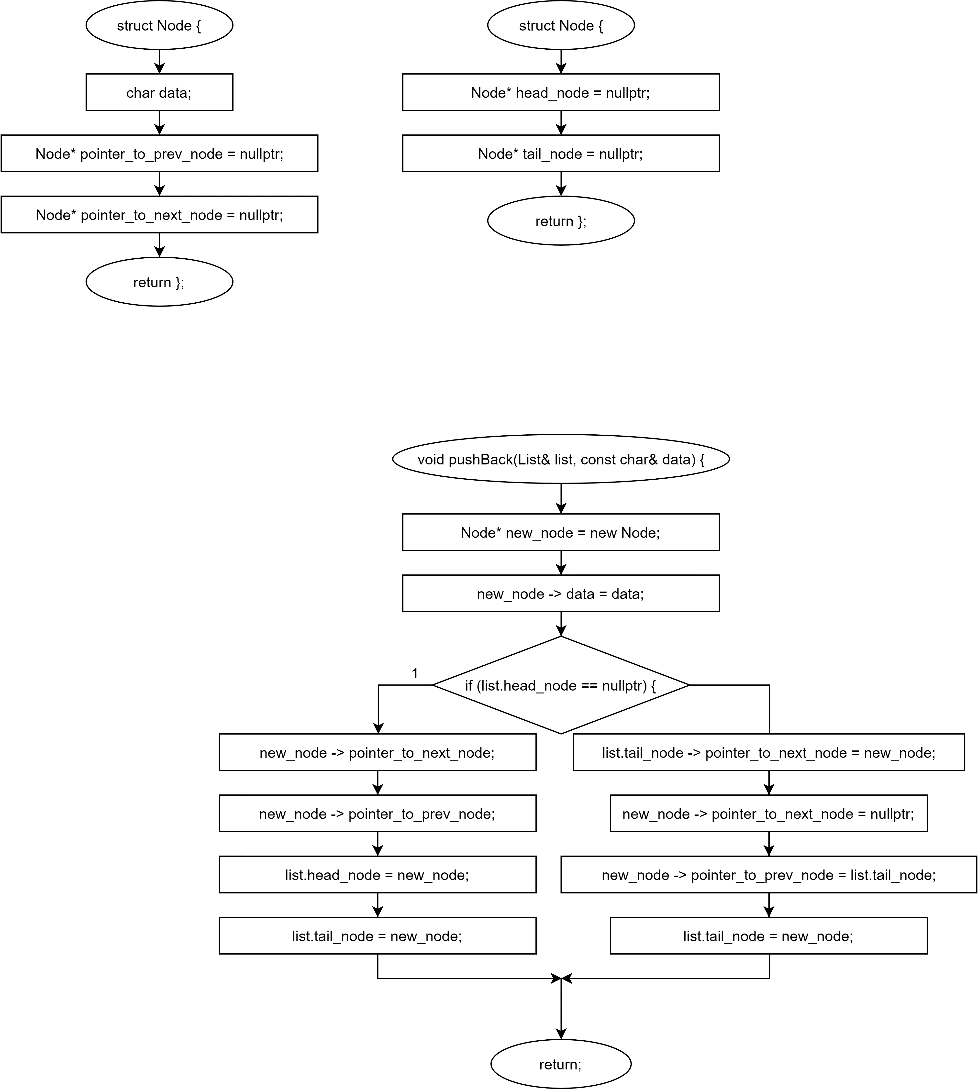
Если первый элемент списка нечетный, то просто переход к след элементу списка.

Начиная со второго элемента списка, запустим цикл while с условием пока текущий указатель ненулевой. Если четный элемент, то записываем в remove, затем delete remove. Текущий элемент становится предыдущим. Переход к след. Узлу.

Иначе(нечетный) => предыдущий. Переход к след. Элементу списка.

1. Вывод листа после обработки.
2. Конец программы.

**3. Блок-схема**



**4. Полный код программы**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct Point{

    string value;

    Point\* next = nullptr;

};

struct List{

    Point\* head = nullptr;

    Point\* tail = nullptr;

};

List list;

void popfirst(List& list){

    Point\* new\_point = list.head;

    list.head = new\_point->next;

    delete new\_point;

}

void poplast(List& list){

    Point\* new\_point = list.head;

    while (new\_point->next != list.tail){

        new\_point = new\_point->next;

    }

    new\_point->next = nullptr;

    delete list.tail;

    list.tail = new\_point;

}

void pushfront(List& list, string& value){

    Point\* new\_point = new Point;

    new\_point->value = value;

    if (list.head == nullptr){

        list.head = new\_point;

        list.tail = new\_point;

    }

    else{

        new\_point -> next = list.head;

        list.head = new\_point;

    }

}

void pushback(List& list, string& value){

    Point\* new\_point = new Point;

    new\_point->value = value;

    if (list.head == nullptr){

       list.head = new\_point;

       list.tail = new\_point;

    }

    else{

       list.tail -> next = new\_point;

       list.tail = new\_point;

    }

}

void pushq(List& list, string value, int elem){

    Point\* new\_point = new Point;

    new\_point->value = value;

    Point\* p2 = list.head;

    Point\* p3 = list.head;

    for(int i = 1; i <= elem - 2 ; i++){

        p2 = p2->next;

    }

    if (elem <= 1){

        pushfront(list, value);

    }

    else{

        for(int i = 1; i <= elem - 1 ; i++){

            p3 = p3->next;

        }

        p2->next = new\_point;

        new\_point->next = p3;

    }

}

void deleteq(List& list, int elem2){

    Point\* new\_point = list.head;

    Point\* p2 = list.head;

    for(int i = 1; i <= elem2 - 3; i++){

        new\_point = new\_point->next;

    }

    if (elem2 <= 2){

        popfirst(list);

    }

    else{

        for(int i = 1; i <= elem2 - 1; i++){

            p2 = p2->next;

        }

        new\_point->next = p2;

    }

}

void showmenu(){

    cout << "1. Создать список\n";

    cout << "2. Вывод списка на экран\n";

    cout << "3. Добавить элемент в начало\n";

    cout << "4. Добавить элемент в конец\n";

    cout << "5. Удалить начальный элемент\n";

    cout << "6. Удалить конечный элемент\n";

    cout << "7. Добавить несколько элементов\n";

    cout << "8. Удалить несколько элементов\n";

    cout << "9. Записать список в файл\n";

    cout << "10. Считать список из файла\n";

}

void printlist(List& list){

    Point\* current = list.head;

    while (current != nullptr){

        cout << current->value << " \t ";

        current = current->next;

    }

    cout << "\n";

}

void addlist(List & list){

    ofstream out("file.txt");

    Point\* current = list.head;

    while (current != nullptr){

        out << current->value << "\n";

        current = current->next;

    }

    out.close();

}

void createlist(int size){

    string str;

    cin.ignore();

    for (int i = 0; i < size; i++){

        getline(cin, str);

        pushback(list, str);

    }

}

void fromlist(List& list, int size){

    string buf;

    ifstream in("file.txt");

    for (int i = 0; i < size; i++){

        getline(in, buf);

        pushback(list, buf);

    }

    in.close();

}

void menu(){

    string str;

    int choice, size, quantity, index;

    while (choice != -1){

        system("pause");

        system("cls");

        showmenu();

        cin >> choice;

        switch(choice){

            case 1: {

                cout << "Введите количество элементов списка\n";

                cin >> size;

                createlist(size);

                break;

            }

            case 2: {

                printlist(list);

                break;

            }

            case 3: {

                cin.ignore();

                cout << "Введите добавляемый элемент\n";

                getline(cin, str);

                pushfront(list, str);

                break;

            }

            case 4: {

                cin.ignore();

                cout << "Введите добавляемый элемент\n";

                getline(cin, str);

                pushback(list, str);

                break;

            }

            case 5: {

                popfirst(list);

                break;

            }

            case 6: {

                poplast(list);

                break;

            }

            case 7: {

                cout << "Сколько элементов добавить?\n";

                cin >> quantity;

                cout << "Перед каким по счету?\n";

                cin >> index;

                cin.ignore();

                for (int i = 0; i < quantity; i++){

                    getline(cin, str);

                    pushq(list, str, index);

                    index++;

                }

                break;

            }

            case 8: {

                cout << "Сколько элементов удалить?\n";

                cin >> quantity;

                cout << "Перед каким по счету?\n";

                cin >> index;

                for (int i = 0; i < quantity; i++){

                    deleteq(list, index);

                    index--;

                }

                break;

            }

            case 9: {

                addlist(list);

                break;

            }

            case 10: {

                cout << "Введите количество символов списка\n";

                cin >> size;

                fromlist(list, size);

                break;

            }

            default: {

                cout << "??\n";

                break;

            }

        }

    }

}

int main(){

    system("chcp 1251 >> null");

    menu();

}

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct Point{

    string value;

    Point\* next = nullptr;

    Point\* prev = nullptr;

};

struct List{

    Point\* head = nullptr;

    Point\* tail = nullptr;

};

List list;

void insert(List& list, string& value, int& index) {

    Point\* new\_point = new Point;

    new\_point -> value = value;

    if (list.head == nullptr) {

        new\_point->prev = nullptr;

        new\_point->next = nullptr;

        list.head = new\_point;

        list.tail = new\_point;

        return;

    }

    int counter = 1;

    Point\* current = list.head;

    while (counter != index) {

        current = current->next;

        counter++;

    }

    if (current->next != nullptr) {

        new\_point->next = current->next;

        current->next->prev = new\_point;

    }

    current->next = new\_point;

    new\_point->prev = current;

}

void pushfront(List& list, string& value){

    Point\* new\_point = new Point;

    new\_point->value = value;

    if (list.head == nullptr){

        new\_point->prev = nullptr;

        new\_point->next = nullptr;

        list.head = new\_point;

        list.tail = new\_point;

    }

    else{

        list.head->next = new\_point;

        new\_point->prev = nullptr;

        new\_point->next = list.head;

        list.head = new\_point;

    }

}

void pushback(List& list, string& value){

    Point\* new\_point = new Point;

    new\_point->value = value;

    if (list.head == nullptr) {

        new\_point->prev = nullptr;

        new\_point->next = nullptr;

        list.head = new\_point;

        list.tail = new\_point;

    }

    else {

        list.tail->next = new\_point;

        new\_point->next = nullptr;

        new\_point->prev = list.tail;

        list.tail = new\_point;

    }

}

void popfirst(List& list){

    Point\* new\_point = list.head;

    list.head = new\_point->next;

    delete new\_point;

}

void pushq(List& list, string value, int elem){

    Point\* new\_point = new Point;

    new\_point->value = value;

    Point\* p2 = list.head;

    Point\* p3 = list.head;

    for(int i = 1; i <= elem - 2 ; i++){

        p2 = p2->next;

    }

    if (elem <= 1){

        pushfront(list, value);

    }

    else{

        for(int i = 1; i <= elem - 1 ; i++){

            p3 = p3->next;

        }

        p2->next = new\_point;

        new\_point->next = p3;

    }

}

void deleteq(List& list, int elem2){

    Point\* new\_point = list.head;

    Point\* p2 = list.head;

    for(int i = 1; i <= elem2 - 3; i++){

        new\_point = new\_point->next;

    }

    if (elem2 <= 2){

        popfirst(list);

    }

    else{

        for(int i = 1; i <= elem2 - 1; i++){

            p2 = p2->next;

        }

        new\_point->next = p2;

    }

}

void showmenu(){

    cout << "1. Создать список\n";

    cout << "2. Вывод списка на экран\n";

    cout << "3. Добавить элемент в начало\n";

    cout << "4. Добавить элемент в конец\n";

    cout << "5. Удалить начальный элемент\n";

    cout << "6. Вставить элемент перед заданным индексом\n";

    cout << "7. Добавить несколько элементов\n";

    cout << "8. Удалить несколько элементов\n";

    cout << "9. Записать список в файл\n";

    cout << "10. Считать список из файла\n";

}

void printlist(List& list){

    Point\* current = list.head;

    while (current != nullptr){

        cout << current->value << " \t ";

        current = current->next;

    }

    cout << "\n";

}

void addlist(List & list){

    ofstream out("file.txt");

    Point\* current = list.head;

    while (current != nullptr){

        out << current->value << "\n";

        current = current->next;

    }

    out.close();

}

void createlist(int size){

    string str;

    cin.ignore();

    for (int i = 0; i < size; i++){

        getline(cin, str);

        pushback(list, str);

    }

}

void fromlist(List& list, int size){

    string buf;

    ifstream in("file.txt");

    for (int i = 0; i < size; i++){

        getline(in, buf);

        pushback(list, buf);

    }

    in.close();

}

void menu(){

    string str;

    int choice = 1, size, quantity, index;

    while (choice != -1){

        system("pause");

        system("cls");

        showmenu();

        cin >> choice;

        switch(choice){

            case 1: {

                cout << "Введите количество элементов списка\n";

                cin >> size;

                createlist(size);

                break;

            }

            case 2: {

                printlist(list);

                break;

            }

            case 3: {

                cin.ignore();

                cout << "Введите добавляемый элемент\n";

                getline(cin, str);

                pushfront(list, str);

                break;

            }

            case 4: {

                cin.ignore();

                cout << "Введите добавляемый элемент\n";

                getline(cin, str);

                pushback(list, str);

                break;

            }

            case 5: {

                popfirst(list);

                break;

            }

            case 6: {

                cin.ignore();

                cout << "Введите добавляемый элемент\n";

                getline(cin, str);

                cout << "После какого по счету?\n";

                cin >> index;

                insert(list, str, index);

                break;

            }

            case 7: {

                cout << "Сколько элементов добавить?\n";

                cin >> quantity;

                cout << "Перед каким по счету?\n";

                cin >> index;

                cin.ignore();

                for (int i = 0; i < quantity; i++){

                    getline(cin, str);

                    pushq(list, str, index);

                    index++;

                }

                break;

            }

            case 8: {

                cout << "Сколько элементов удалить?\n";

                cin >> quantity;

                cout << "Перед каким по счету?\n";

                cin >> index;

                for (int i = 0; i < quantity; i++){

                    deleteq(list, index);

                    index--;

                }

                break;

            }

            case 9: {

                addlist(list);

                break;

            }

            case 10: {

                cout << "Введите количество символов списка\n";

                cin >> size;

                fromlist(list, size);

                break;

            }

            default: {

                cout << "??\n";

                break;

            }

        }

    }

}

int main(){

    system("chcp 1251 >> null");

    menu();

}

**5.Скриншоты результатов**

