Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

ПНИПУ

**Лабораторная работа №11. Списки (однонаправленные и двунаправленные**

Выполнил:   
студент группы РИС-23-1б   
Сингур Иван Сергеевич

Проверила:   
доцент кафедры ИТАС   
О.А. Полякова

Пермь, 2024 г

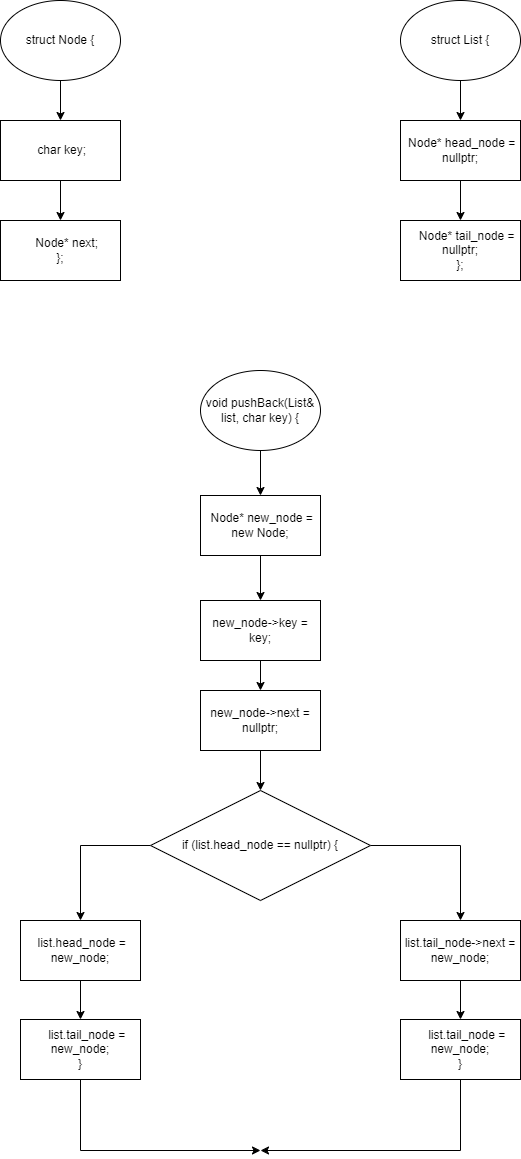
**Условие:**

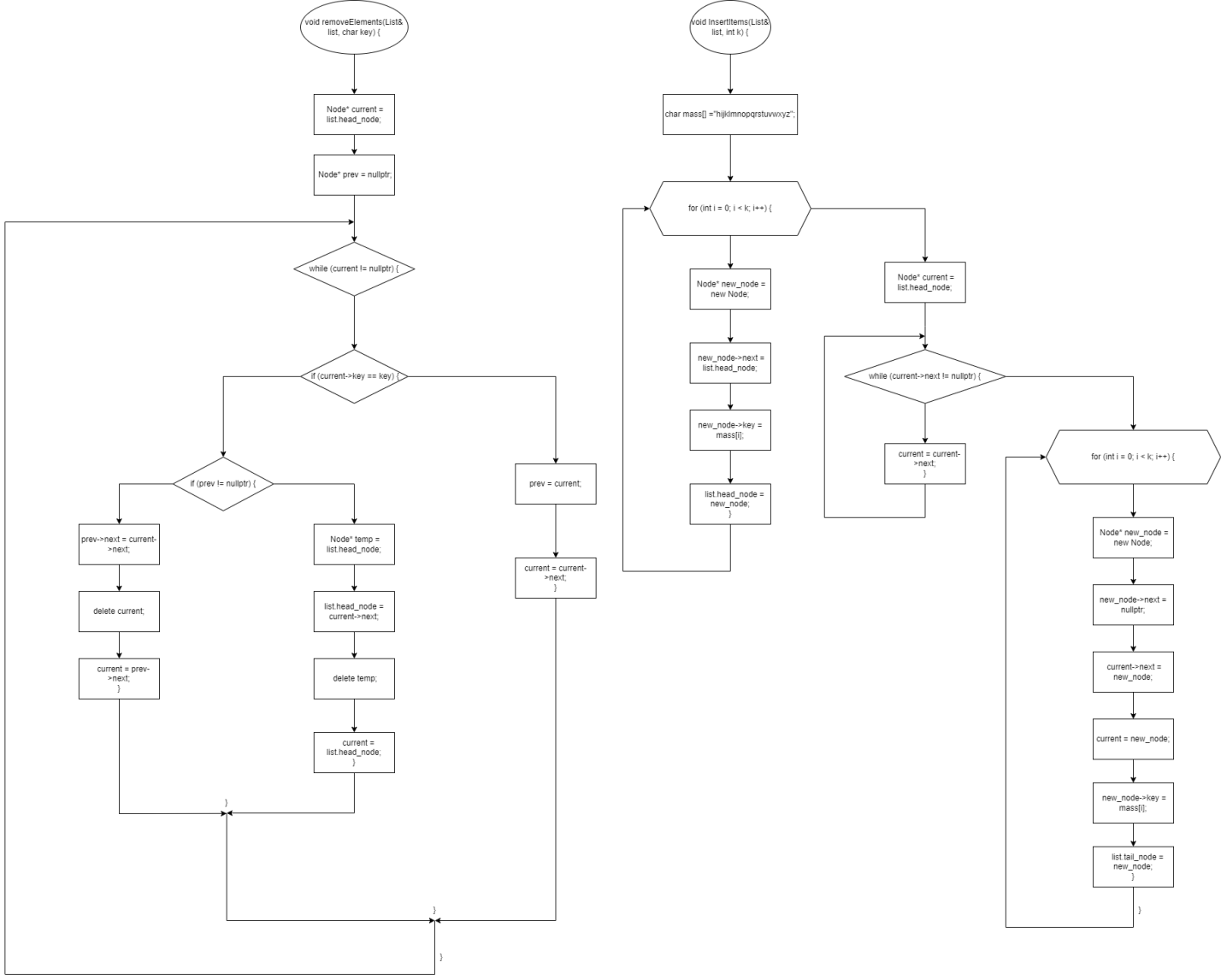
Записи в линейном списке содержат ключевое поле типа \*char (строка символов). Сформировать двунаправленный/однонаправленный список. Удалить элемент с заданным ключом. Добавить по К элементов в начало и в конец списка.

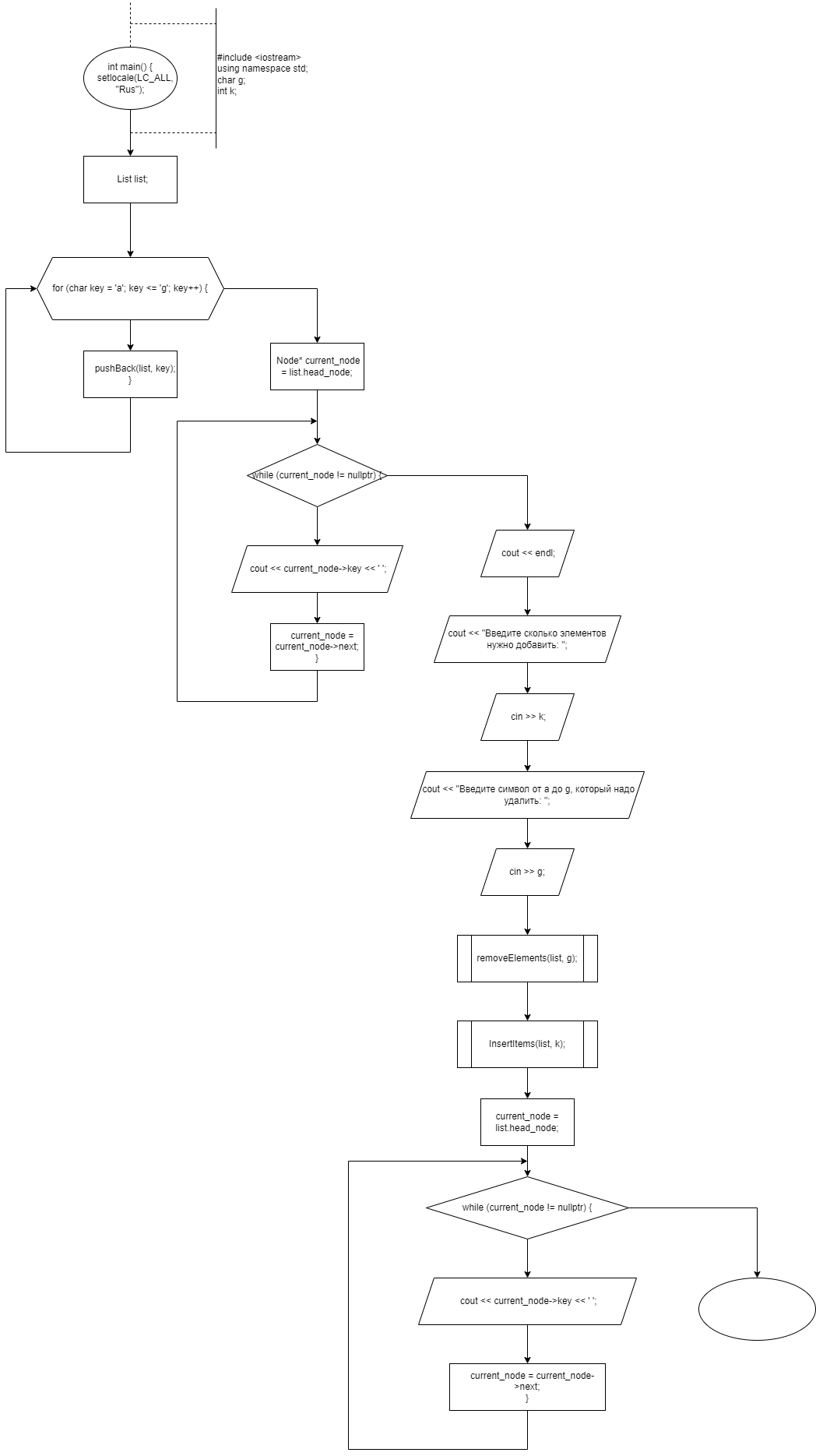
**Анализ задачи:**

**Блок-схема:**

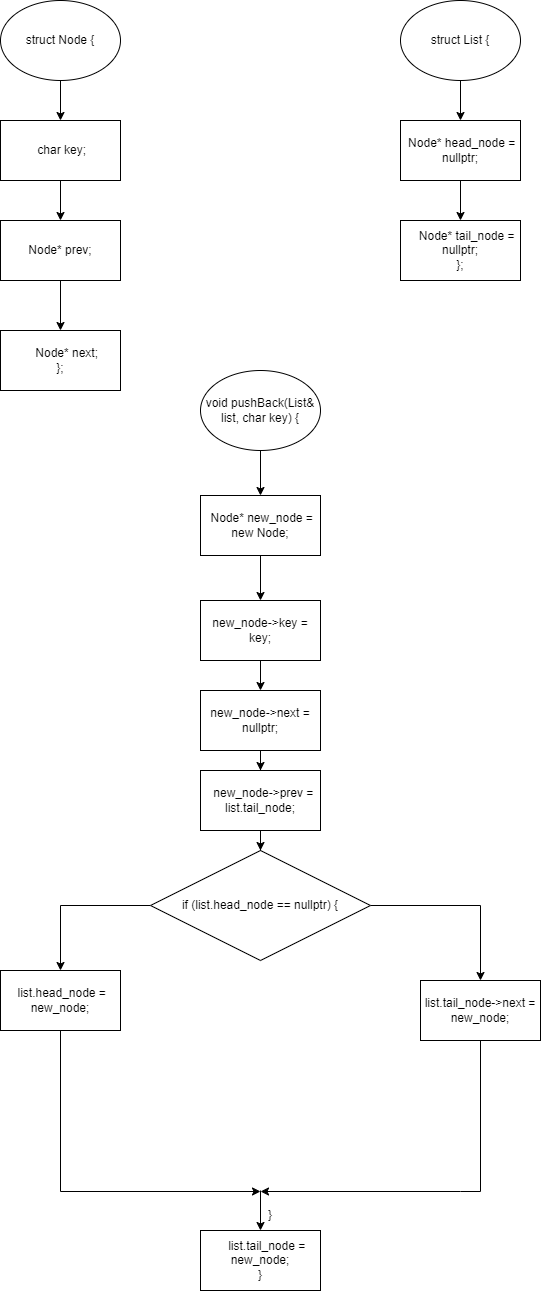
**Однонаправленный:**

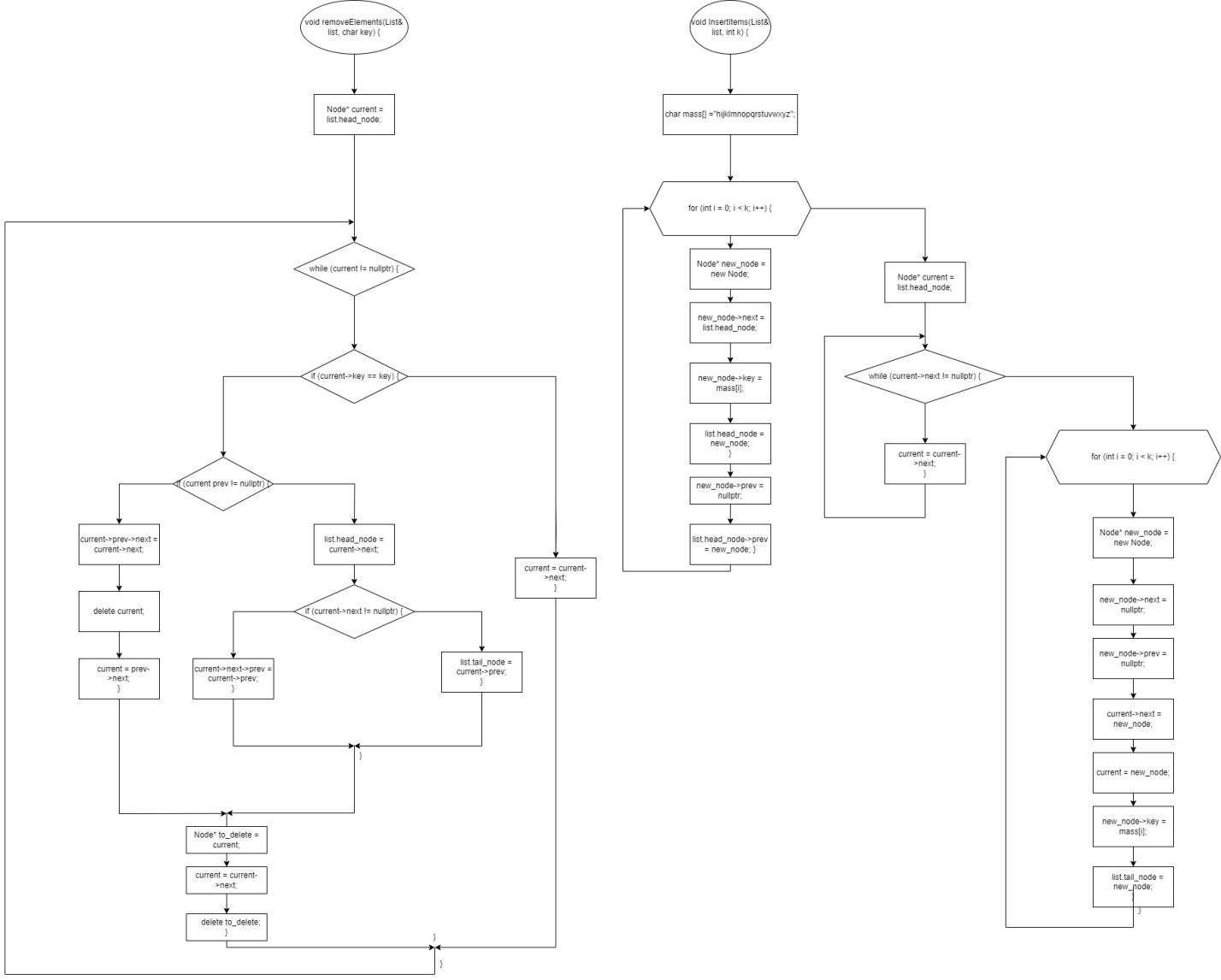
****

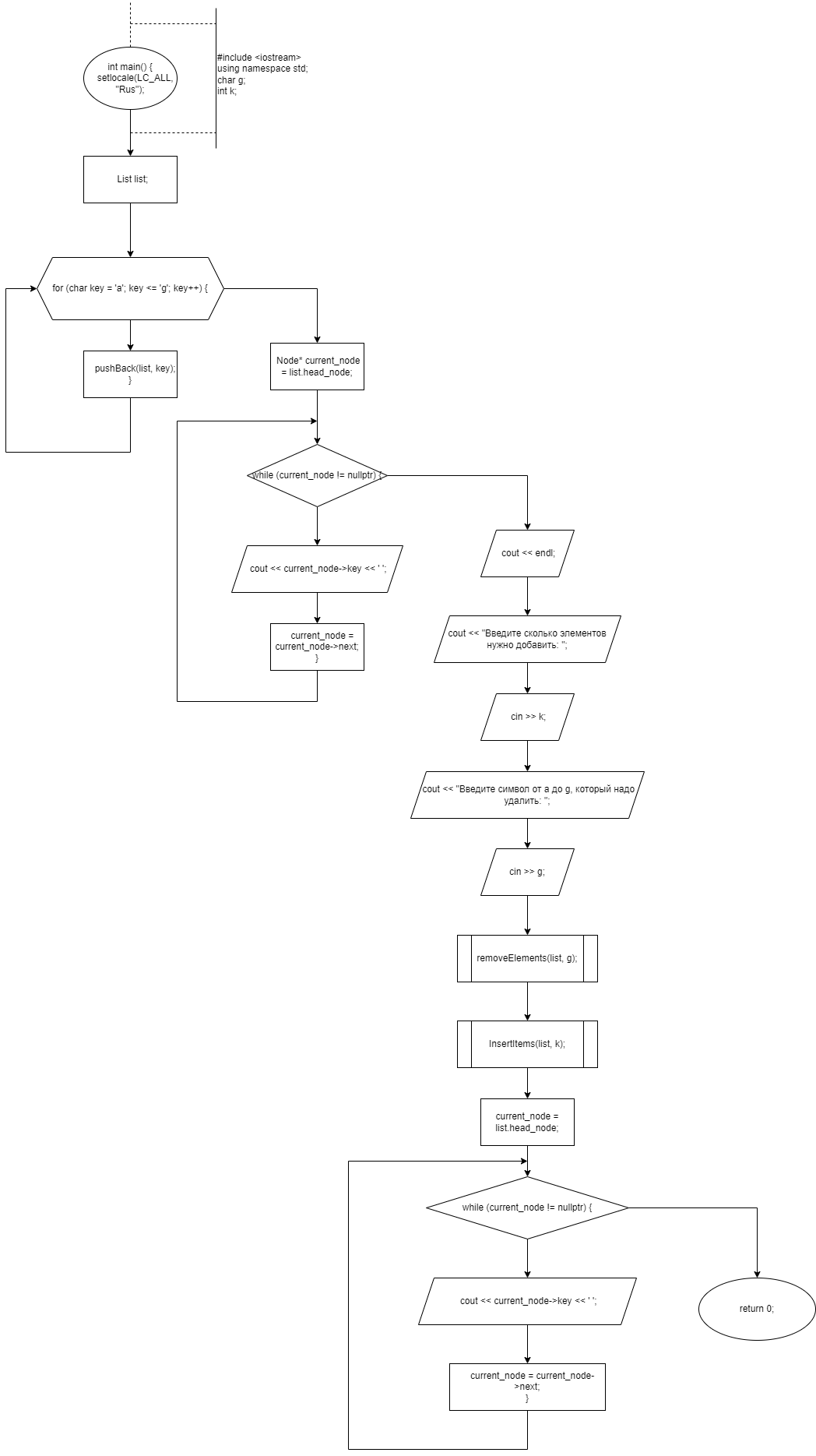
****

****

**Двунаправленный:**

****

****

****

**Код на языке C++:**

**Однонаправленный:**

#include <iostream>

using namespace std;

struct Node {

char key;

Node\* next;

};

struct List {

Node\* head\_node = nullptr;

Node\* tail\_node = nullptr;

};

void pushBack(List& list, char key) {

Node\* new\_node = new Node;

new\_node->key = key;

new\_node->next = nullptr;

if (list.head\_node == nullptr) {

list.head\_node = new\_node;

list.tail\_node = new\_node;

}

else {

list.tail\_node->next = new\_node;

list.tail\_node = new\_node;

}

}

void removeElements(List& list, char key) {

Node\* current = list.head\_node;

Node\* prev = nullptr;

while (current != nullptr) {

if (current->key == key) {

if (prev != nullptr) {

prev->next = current->next;

delete current;

current = prev->next;

}

else {

Node\* temp = list.head\_node;

list.head\_node = current->next;

delete temp;

current = list.head\_node;

}

}

else {

prev = current;

current = current->next;

}

}

}

void InsertItems(List& list, int k) {

char mass[] ="hijklmnopqrstuvwxyz";

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

Node\* new\_node = new Node;

new\_node->next = list.head\_node;

new\_node->key = mass[i];

list.head\_node = new\_node;

}

Node\* current = list.head\_node;

while (current->next != nullptr) {

current = current->next;

}

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

Node\* new\_node = new Node;

new\_node->next = nullptr;

current->next = new\_node;

current = new\_node;

new\_node->key = mass[i];

list.tail\_node = new\_node;

}

}

int main() {

setlocale(LC\_ALL, "Rus");

List list;

for (char key = 'a'; key <= 'g'; key++) {

pushBack(list, key);

}

Node\* current\_node = list.head\_node;

while (current\_node != nullptr) {

cout << current\_node->key << ' ';

current\_node = current\_node->next;

}

cout << endl;

char g;

int k;

cout << "Введите сколько элементов нужно добавить: ";

cin >> k;

cout << "Введите символ от a до g, который надо удалить: ";

cin >> g;

removeElements(list, g);

InsertItems(list, k);

current\_node = list.head\_node;

while (current\_node != nullptr) {

cout << current\_node->key << ' ';

current\_node = current\_node->next;

}

return 0;

}

**Двунаправленный:**

#include <iostream>

using namespace std;

struct Node {

char key;

Node\* next;

Node\* prev;

};

struct List {

Node\* head\_node = nullptr;

Node\* tail\_node = nullptr;

};

void pushBack(List& list, char key) {

Node\* new\_node = new Node;

new\_node->key = key;

new\_node->next = nullptr;

new\_node->prev = list.tail\_node;

if (list.head\_node == nullptr) {

list.head\_node = new\_node;

}

else {

list.tail\_node->next = new\_node;

}

list.tail\_node = new\_node;

}

void removeElements(List& list, char key) {

Node\* current = list.head\_node;

while (current != nullptr) {

if (current->key == key) {

if (current->prev != nullptr) {

current->prev->next = current->next;

}

else {

list.head\_node = current->next;

}

if (current->next != nullptr) {

current->next->prev = current->prev;

}

else {

list.tail\_node = current->prev;

}

Node\* to\_delete = current;

current = current->next;

delete to\_delete;

}

else {

current = current->next;

}

}

}

void InsertItems(List& list, int k) {

char mass[] = "hijklmnopqrstuvwxyz";

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

Node\* new\_node = new Node;

new\_node->next = list.head\_node;

new\_node->prev = nullptr;

new\_node->key = mass[i];

list.head\_node->prev = new\_node;

list.head\_node = new\_node;

}

Node\* current = list.head\_node;

while (current->next != nullptr) {

current = current->next;

}

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

Node\* new\_node = new Node;

new\_node->next = nullptr;

new\_node->prev = current;

current->next = new\_node;

current = new\_node;

new\_node->key = mass[i];

list.tail\_node = new\_node;

}

}

int main() {

setlocale(LC\_ALL, "Rus");

List list;

for (char key = 'a'; key <= 'g'; key++) {

pushBack(list, key);

}

Node\* current\_node = list.head\_node;

while (current\_node != nullptr) {

cout << current\_node->key << ' ';

current\_node = current\_node->next;

}

cout << endl;

char g;

int k;

cout << "Введите сколько элементов нужно добавить: ";

cin >> k;

cout << "Введите символ от a до g, который надо удалить: ";

cin >> g;

removeElements(list, g);

InsertItems(list, k);

current\_node = list.head\_node;

while (current\_node != nullptr) {

cout << current\_node->key << ' ';

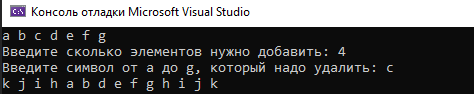
current\_node = current\_node->next;

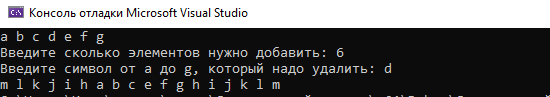
}

return 0;

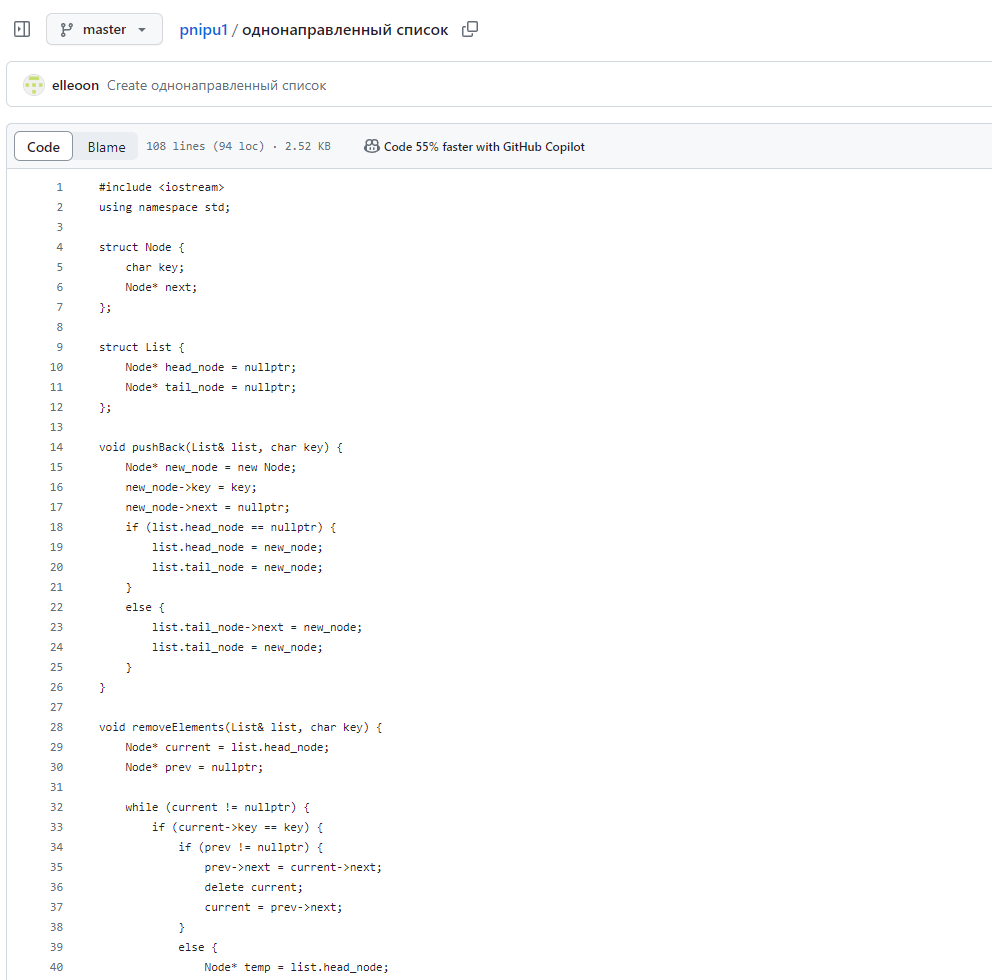
}

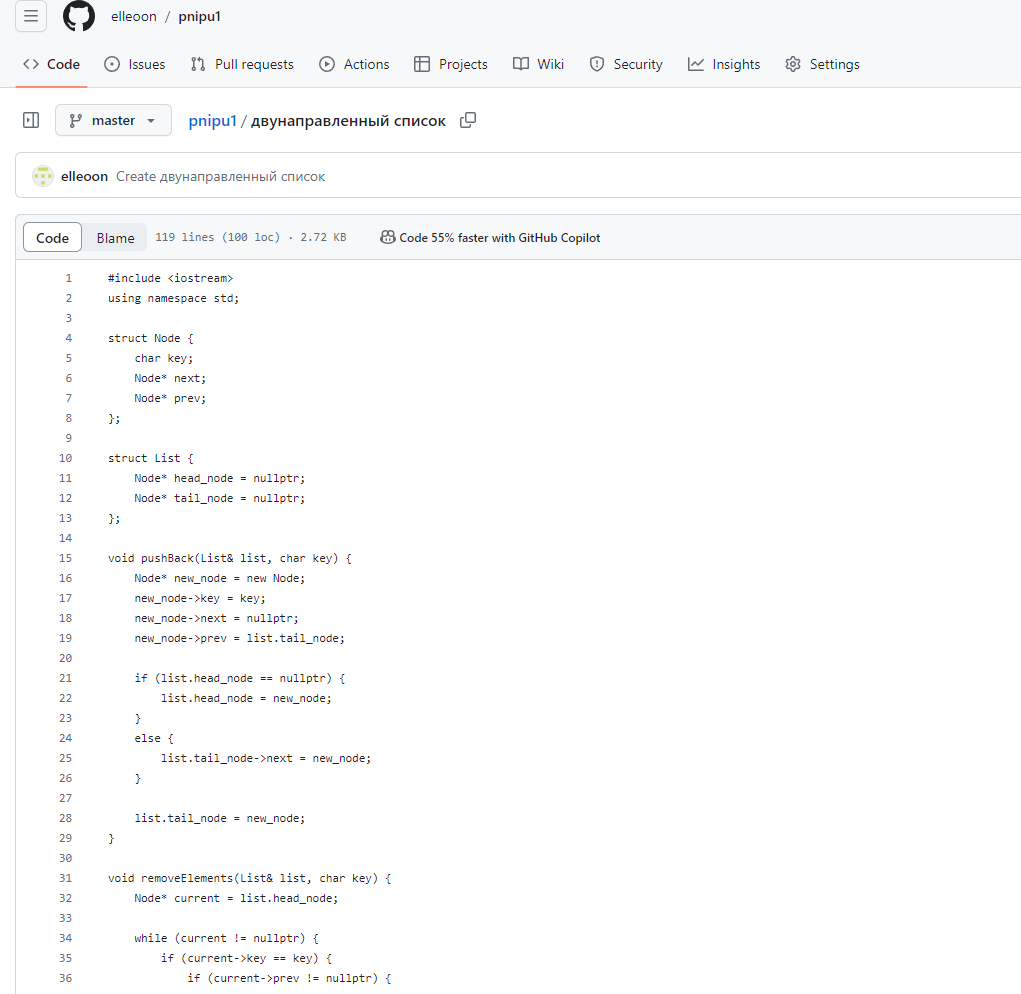
**Работа программы:**





**Cкрины из гита:**





**Ссылка на гит:**

https://github.com/elleoon/pnipu1

**Вывод:** Задача была выполнена. Всё получилось.