Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
Пермский национальный исследовательский политехнический университет
Электротехнический факультет
Кафедра информационных технологий и автоматизированных систем

Лабораторная работа №11

"однонаправленные списки"

Вариант: 12

Выполнил студент ИВТ-24-2б:
Шишкин Максим Григорьевич

_____
(дата, подпись)

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

_____
(дата, подпись)

Пермь 2025

Содержание

1 Постановка задачи

Записи в линейном списке содержат ключевое поле типа *char(строка символов). Сформировать однонаправленный список. Удалить из него Элементы, с одинаковыми ключевыми полями. Добавить элемент после элемента с заданным ключевым полем.

2 Код на C++

```cpp
#include <iostream>
#include <string>
#include <clocale>

using namespace std;

// Структура узла однонаправленного списка (без конструктора)
struct Node {
    string key;
    Node* next;
};

// Функция для создания узла
Node* createNode(const string& k) {
    Node* newNode = new Node;
    newNode->key = k;
    newNode->next = nullptr;
    return newNode;
}

// Функция для добавления элемента в конец однонаправленного списка
void push_back(Node*& head, const string& k) {
    Node* newNode = createNode(k);

    if (head == nullptr) {
        head = newNode;
        return;
    }

    Node* curr = head;
    while (curr->next != nullptr) {
        curr = curr->next;
    }

    curr->next = newNode;
}

// Функция для формирования однонаправленного списка из массива строк
Node* createList(string* strArr, int sz) {
    Node* head = nullptr;
    for (int i = 0; i < sz; ++i) {
        push_back(head, strArr[i]);
    }
    return head;
}

// Функция для вывода однонаправленного списка
```

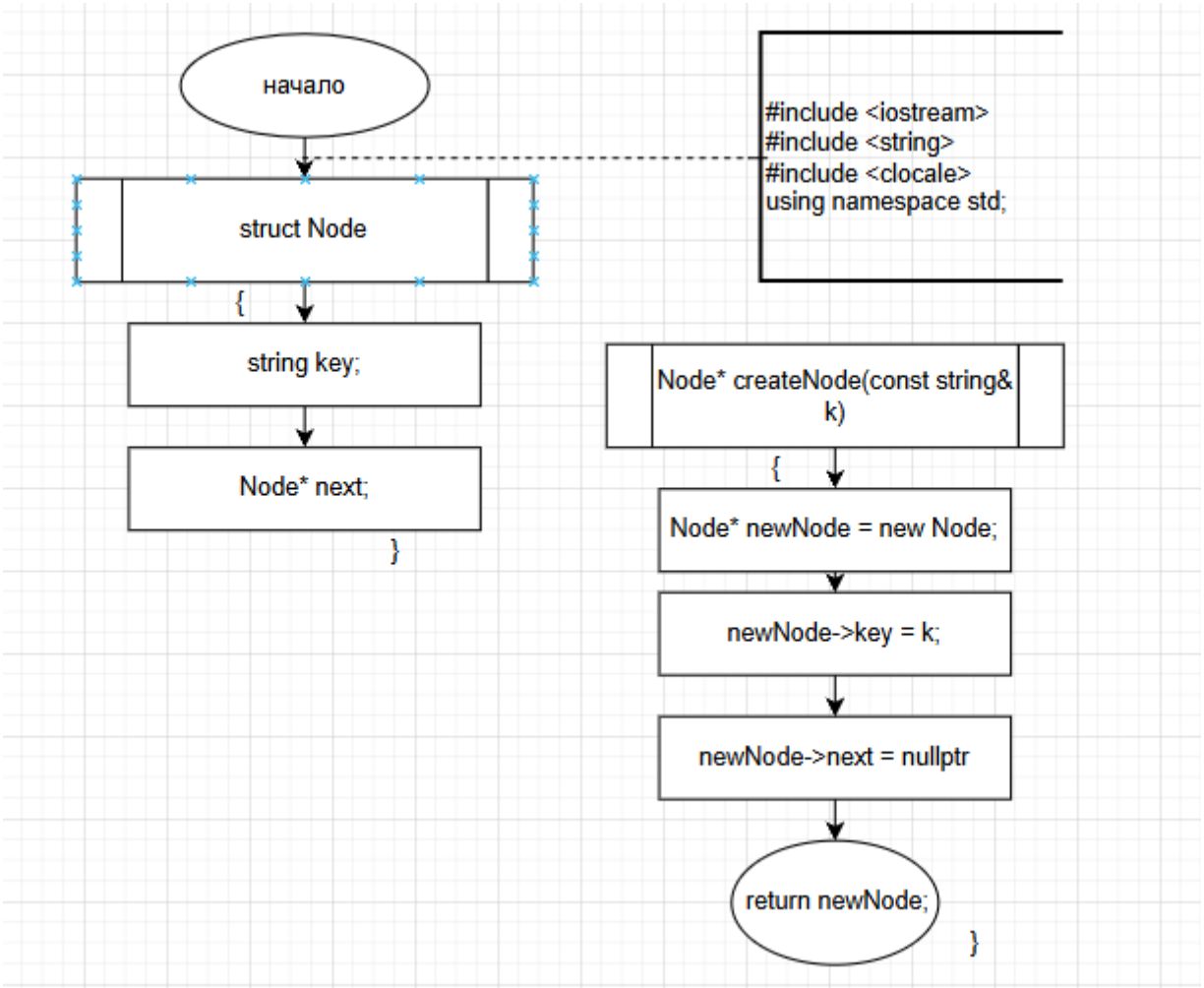
```cpp
void printList(Node* head) {
    Node* curr = head;
    while (curr != nullptr) {
        cout << curr->key << " ";
        curr = curr->next;
    }
    cout << endl;
}

// Функция для удаления элементов с одинаковыми ключевыми полями
void removeDuplicates(Node*& head) {
    if (head == nullptr) {
        return;
    }

    Node* curr = head;
    Node* prev = nullptr;

    while (curr != nullptr) {
        Node* runnerPrev = curr;
        Node* runner = curr->next;

        while (runner != nullptr) {
            if (curr->key == runner->key) {
                // Нашли дубликат

                // Исключаем runner из списка
                runnerPrev->next = runner->next;

                // Если удаляем голову списка
                if (runner == head) {
                    head = runner->next;
                }

                Node* temp = runner;
                runner = runner->next;
                delete temp;
            }
            else {
                runnerPrev = runner;
                runner = runner->next;
            }
        }
        curr = curr->next;
    }
}

```

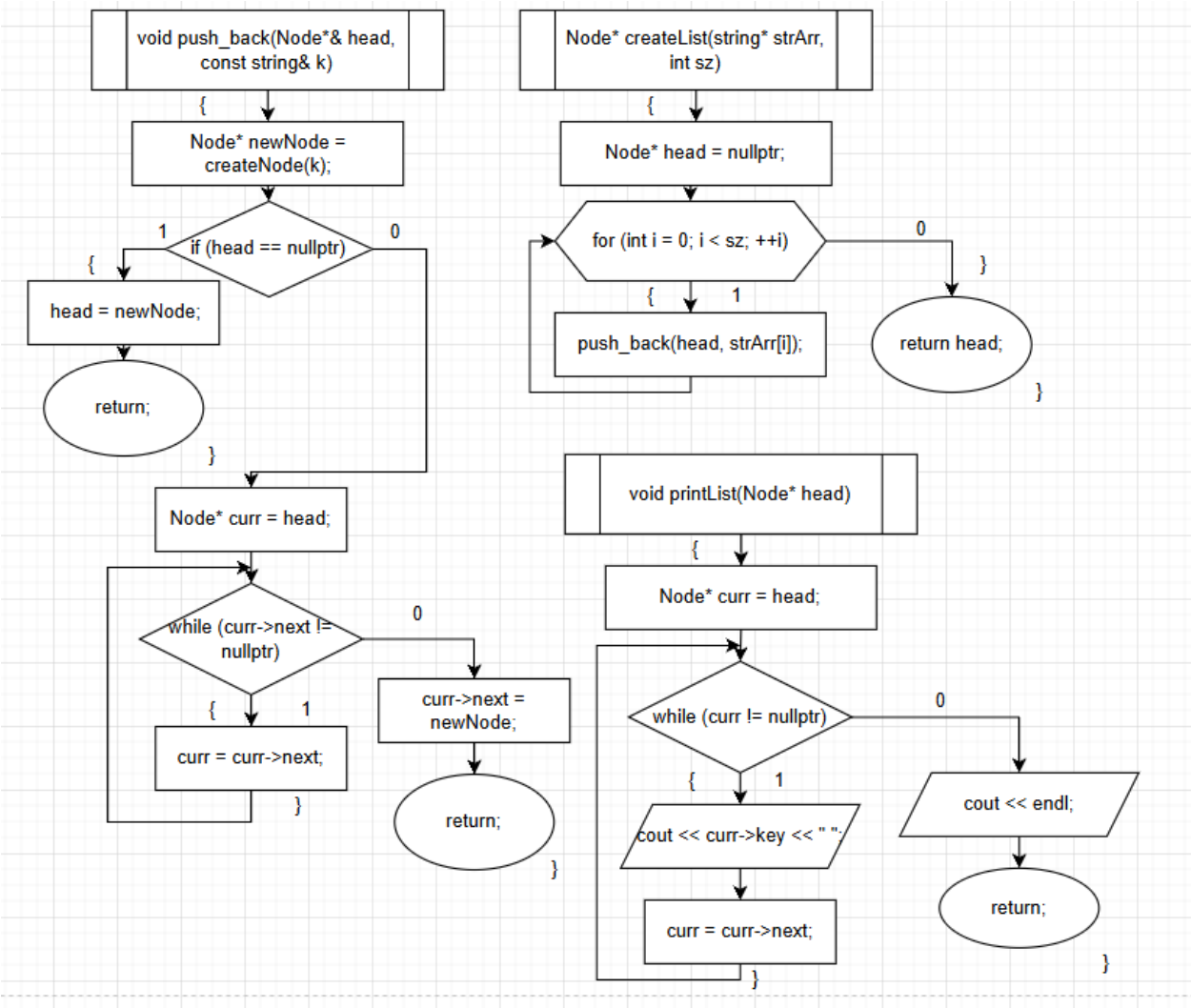
```
95
96  // Функция для добавления элемента после элемента с заданным ключевым полем
97  void addAfter(Node* head, const string& targetKey, const string& newKey) {
98      if (head == nullptr) {
99          return;
100     }
101
102     Node* curr = head;    while (curr != nullptr) {
103         if (curr->key == targetKey) {
104             // Нашли элемент после которого нужно добавить
105             Node* newNode = createNode(newKey); // Используем createNode вместо конструктора
106             newNode->next = curr->next;
107             curr->next = newNode;
108             return; // Завершаем функцию после добавления элемента
109         }
110         curr = curr->next;
111     }
112
113     // Если targetKey не найден, ничего не делаем (можно вывести сообщение об ошибке)
114     cout << "Элемент с ключом '" << targetKey << "' не найден." << endl;
115 }
116
117 // Функция для освобождения памяти, занимаемой списком
118 void deleteList(Node* head) {
119     Node* curr = head;
120     while (curr != nullptr) {
121         Node* next = curr->next;
122         delete curr;
123         curr = next;
124     }
125 }
126
127 int main() {
128     // Пример использования
129     setlocale(LC_ALL, "ru");
130     string strArr[] = { "bmw", "garry", "read", "bmw", "date", "garry", "fak" };
131     int sz = sizeof(strArr) / sizeof(strArr[0]);
132
133     cout << "Исходный линейный список: ";
134     for (int i = 0; i < sz; ++i) {
135         cout << strArr[i] << " ";
136     }
137     cout << endl;
138
139     // 1. Формирование однонаправленного списка
140     Node* head = createList(strArr, sz);
141     cout << "Однонаправленный список: ";
142     printList(head);
143
144     // 2. Удаление элементов с одинаковыми ключевыми полями
145     removeDuplicates(head);
146     cout << "Однонаправленный список после удаления дубликатов: ";
147     printList(head);
148
149     // 3. Добавление элемента после элемента с заданным ключевым полем
150     addAfter(head, "read", "gop");
151     cout <<"Однонаправленный список после добавления элемента: ";
152     printList(head);
153
154     // Освобождение памяти
155     deleteList(head);
156
157     return 0;
158 }
```

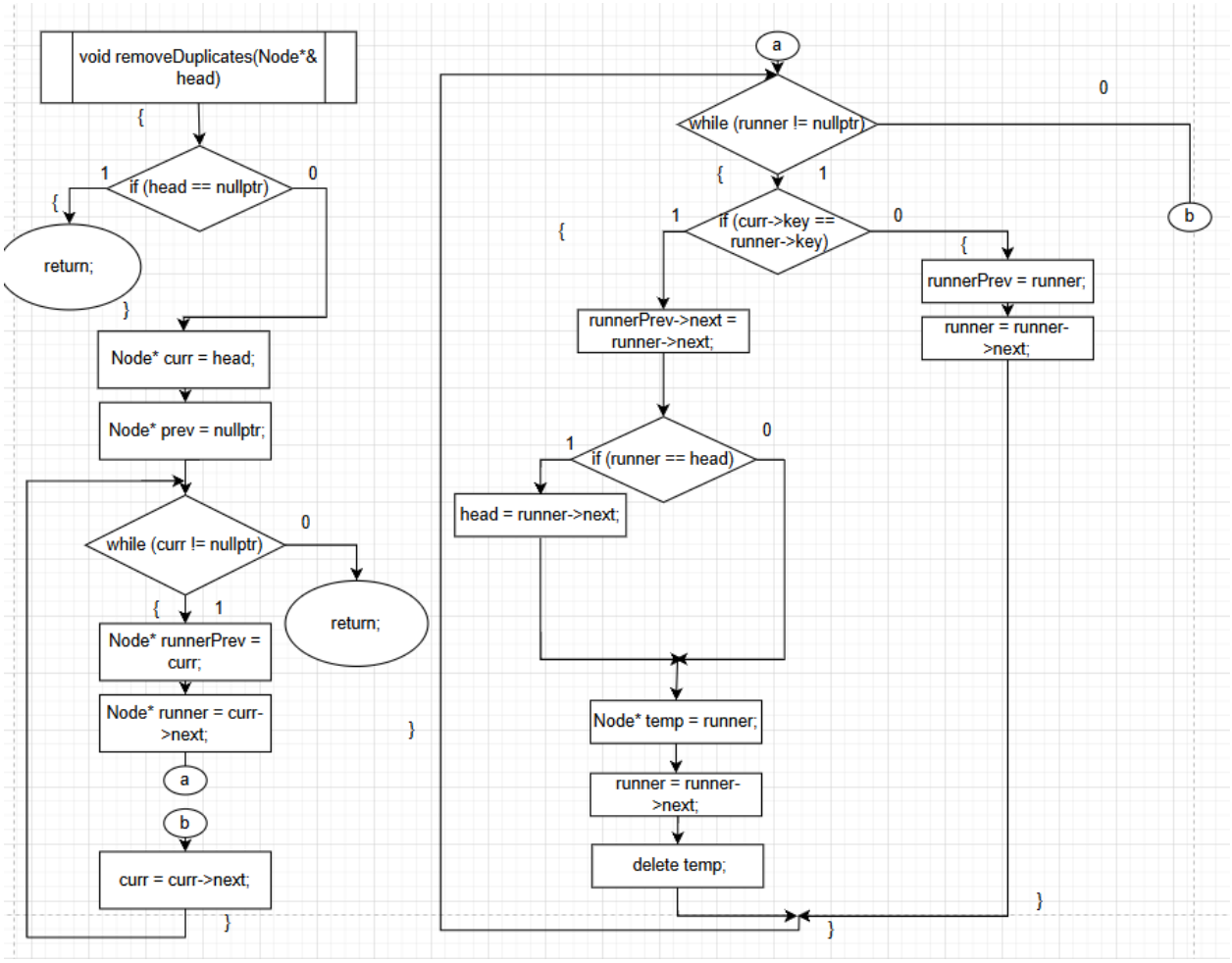
3 Результат работы программы

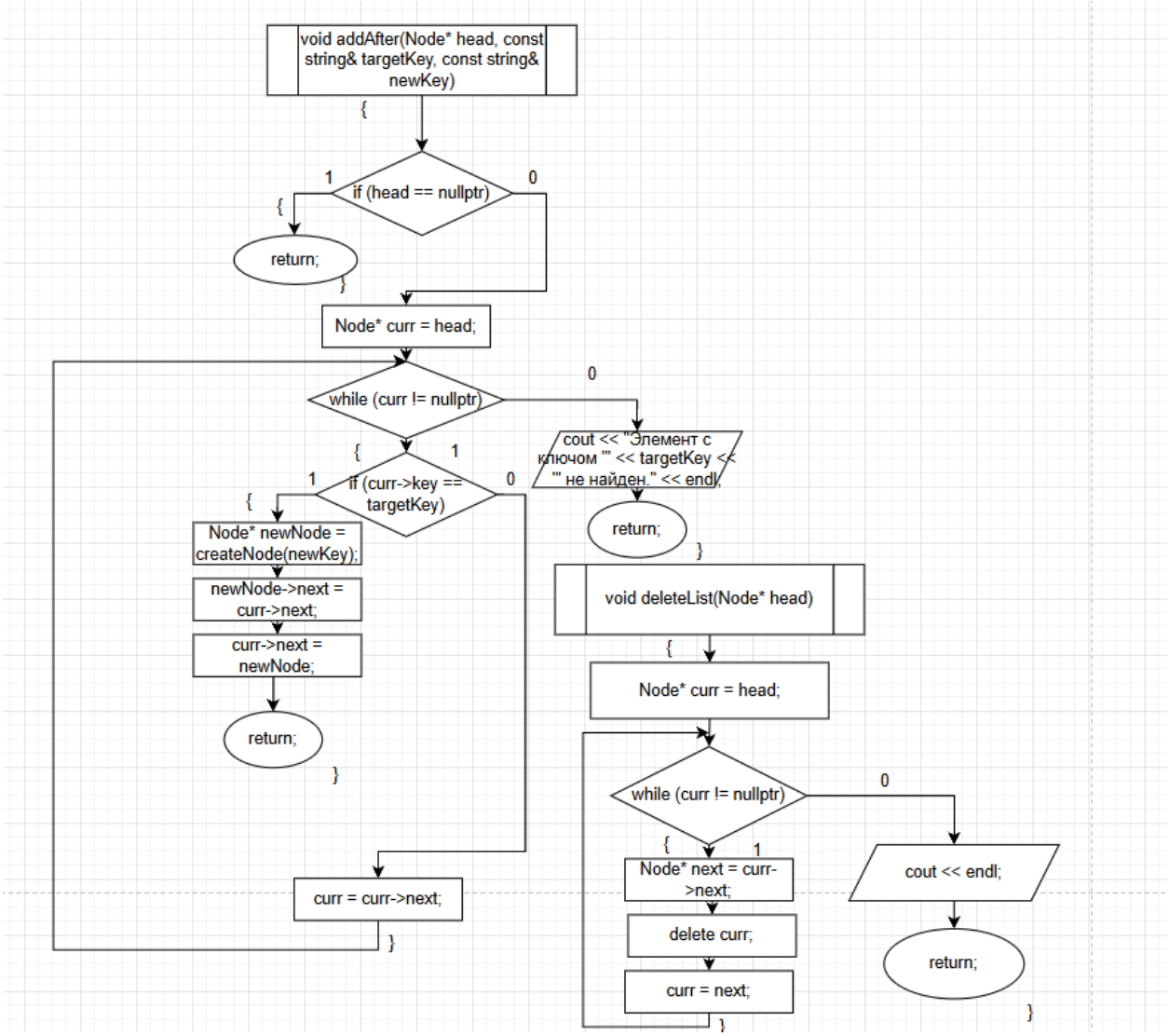
```
Исходный линейный список: bmw garry read bmw date garry fak
Однонаправленный список: bmw garry read bmw date garry fak
Однонаправленный список после удаления дубликатов: bmw garry read date fak
Однонаправленный список после добавления элемента: bmw garry read gop date fak
```

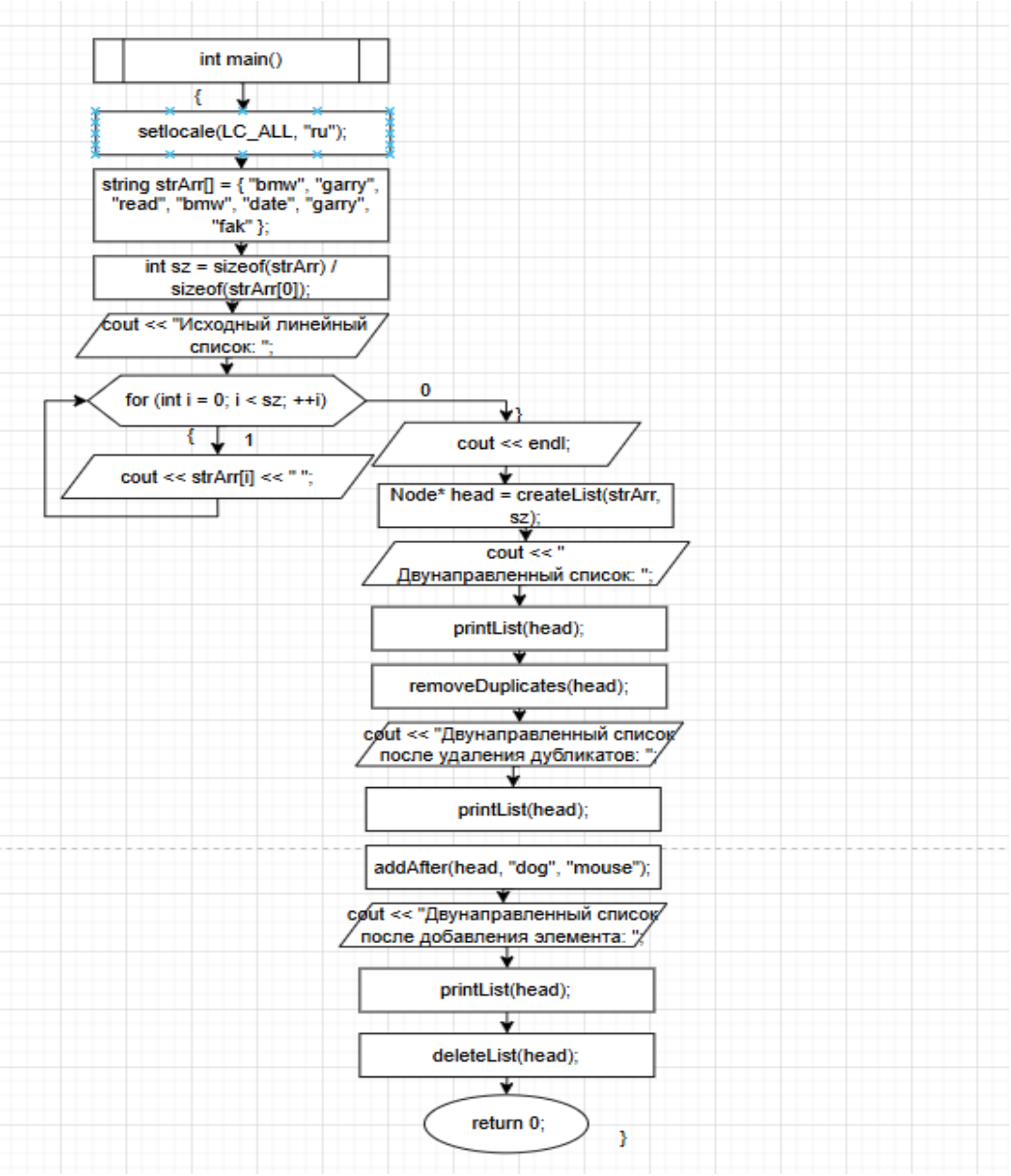
4 Блок-схема




void push_back(Node*& head, const string& k)
{
Node* newNode = createNode(k);
if (head == nullptr)
1
0
{
head = newNode;
return;
}
Node* curr = head;
while (curr->next != nullptr)
0
{
1
curr = curr->next;
}
curr->next = newNode;
return;
}
Node* createList(string* strArr, int sz)
{
Node* head = nullptr;
for (int i = 0; i < sz; ++i)
0
}
{
1
push_back(head, strArr[i]);
return head;
}
void printList(Node* head)
{
Node* curr = head;
while (curr != nullptr)
0
{
1
cout << curr->key << " ";
curr = curr->next;
}
cout << endl;
return;
}


void removeDuplicates(Node*& head)
{
if (head == nullptr)
1
0
{
return;
}
Node* curr = head;
Node* prev = nullptr;
while (curr != nullptr)
0
{
1
Node* runnerPrev = curr;
Node* runner = curr->next;
a
b
curr = curr->next;
}
return;
}
a
while (runner != nullptr)
0
{
1
if (curr->key == runner->key)
1
0
{
runnerPrev->next = runner->next;
if (runner == head)
1
0
head = runner->next;
Node* temp = runner;
runner = runner->next;
delete temp;
{
runnerPrev = runner;
runner = runner->next;
}
}
b


void addAfter(Node* head, const string& targetKey, const string& newKey)
{
if (head == nullptr)
1
0
{
return;
}
Node* curr = head;
while (curr != nullptr)
0
{
1
if (curr->key == targetKey)
1
0
{
Node* newNode = createNode(newKey);
newNode->next = curr->next;
curr->next = newNode;
return;
}
curr = curr->next;
}
cout << "Элемент с ключом "" << targetKey << "" не найден." << endl;
return;
}
void deleteList(Node* head)
{
Node* curr = head;
while (curr != nullptr)
0
{
1
Node* next = curr->next;
delete curr;
curr = next;
}
cout << endl;
return;
}



5 Ссылка на github

ссылка на github - https://github.com/MAKSPOWERO/mas1