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

ПНИПУ

**Лабораторная работа №11**

**"Однонаправленный список”**

Выполнил:   
студент группы РИС-23-1б   
Агзамов Артур Альферович

Проверила:   
доцент кафедры ИТАС   
О.А. Полякова

2024 г.

**Задача:**

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

**Анализ задачи:**

1. Задача заключается в работе с линейным однонаправленным списком, содержащим записи с ключевым полем типа char.

2. Используется структура Node для хранения данных и указателя на следующий элемент в списке.

3. Для управления списком создана структура List, содержащая размер списка и указатель на его начало.

4. Функция add добавляет новый элемент в конец списка.

5. Функция out выводит все элементы списка.

6. Функция put вставляет новый элемент по указанному индексу в списке.

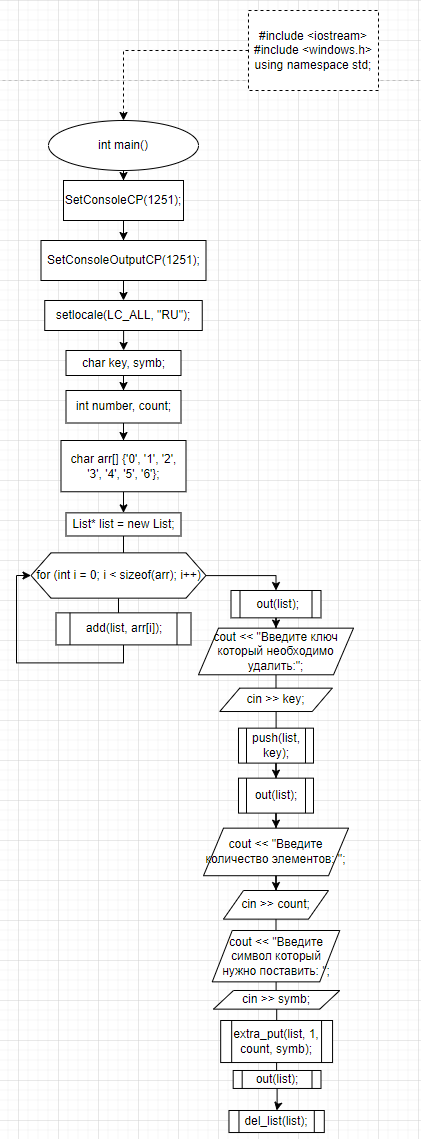
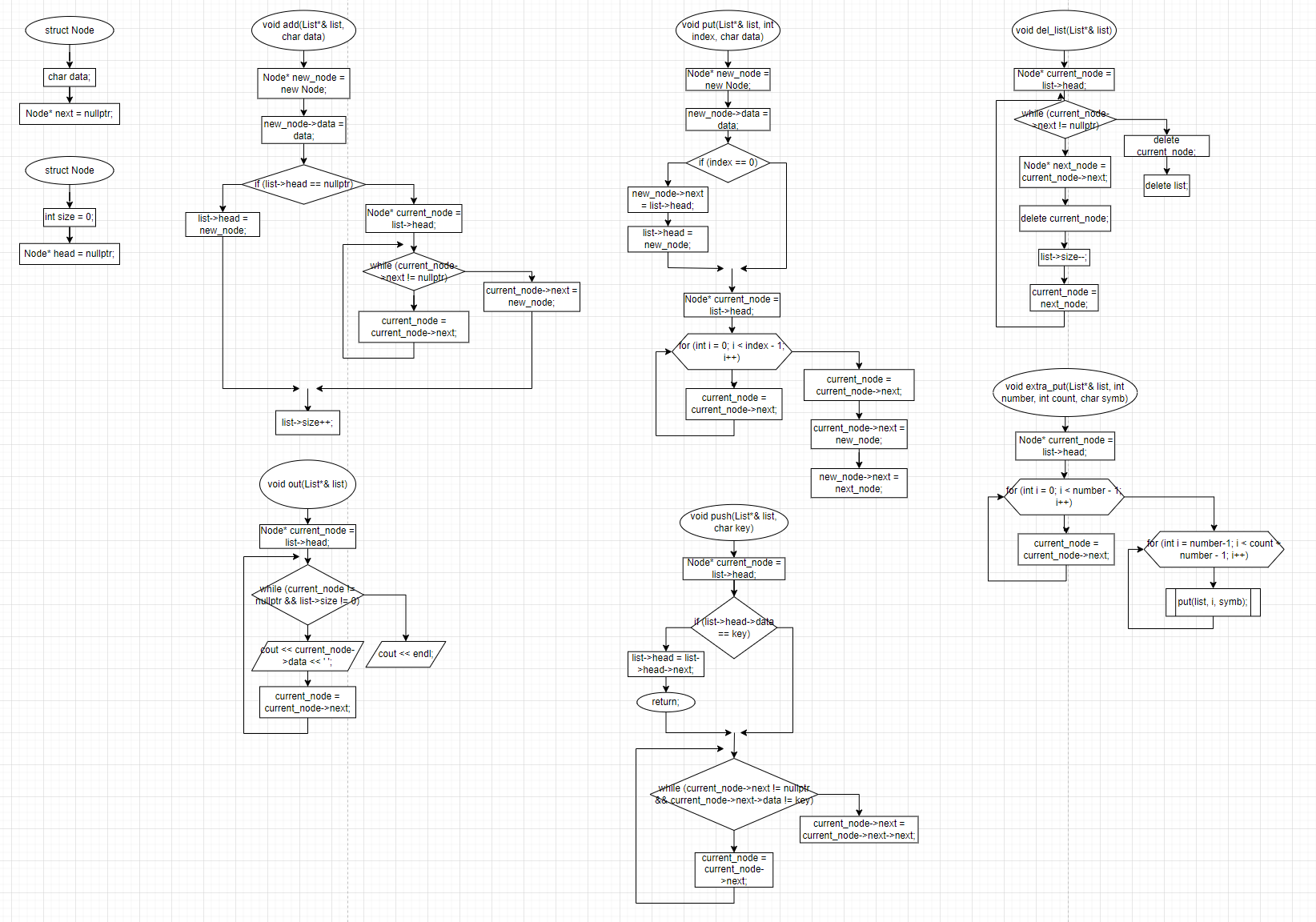
7. Функция push удаляет элемент с заданным ключом из списка.

8. Функция dellist освобождает память, выделенную под список, предотвращая утечку памяти.

9. Функция extraput добавляет K элементов со значением symb в начало списка, начиная с элемента с номером number.

10. В функции main происходит создание списка, добавление элементов из массива, вывод на экран, удаление элемента с заданным ключом, добавление новых элементов в начало списка и вывод обновленного списка, затем освобождение памяти.

**Блок схема**:



**Код:**

#include <iostream>

#include <windows.h>

using namespace std;

struct Node

{

char data;

Node\* next = nullptr;

};

struct List

{

int size = 0;

Node\* head = nullptr;

};

void add(List\*& list, char data)

{

Node\* new\_node = new Node;

new\_node->data = data;

if (list->head == nullptr)

{

list->head = new\_node;

}

else

{

Node\* current\_node = list->head;

while (current\_node->next != nullptr)

{

current\_node = current\_node->next;

}

current\_node->next = new\_node;

}

list->size++;

}

void out(List\*& list)

{

Node\* current\_node = list->head;

while (current\_node != nullptr && list->size != 0)

{

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

current\_node = current\_node->next;

}

cout << endl;

}

void put(List\*& list, int index, char data)

{

Node\* new\_node = new Node;

new\_node->data = data;

if (index == 0)

{

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

list->head = new\_node;

}

Node\* current\_node = list->head;

for (int i = 0; i < index - 1; i++)

{

current\_node = current\_node->next;

}

Node\* next\_node = current\_node->next;

current\_node->next = new\_node;

new\_node->next = next\_node;

}

void push(List\*& list, char key)

{

Node\* current\_node = list->head;

if (list->head->data == key)

{

list->head = list->head->next;

return;

}

while (current\_node->next != nullptr && current\_node->next->data != key)

{

current\_node = current\_node->next;

}

current\_node->next = current\_node->next->next;

}

void del\_list(List\*& list)

{

Node\* current\_node = list->head;

while (current\_node != nullptr)

{

Node\* next\_node = current\_node->next;

delete current\_node;

list->size--;

current\_node = next\_node;

}

delete current\_node;

delete list;

}

void extra\_put(List\*& list, int number, int count, char symb)

{

Node\* current\_node = list->head;

for (int i = 0; i < number - 1; i++)

{

current\_node = current\_node->next;

}

for (int i = number-1; i < count + number - 1; i++)

{

put(list, i, symb);

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "RU");

char key, symb;

int number, count;

char arr[] {'0', '1', '2', '3', '4', '5', '6'};

List\* list = new List;

for (int i = 0; i < sizeof(arr); i++)

{

add(list, arr[i]);

}

out(list);

cout << "Введите ключ который необходимо удалить:";

cin >> key;

push(list, key);

out(list);

cout << "Введите количество элементов: ";

cin >> count;

cout << "Введите символ который нужно поставить: ";

cin >> symb;

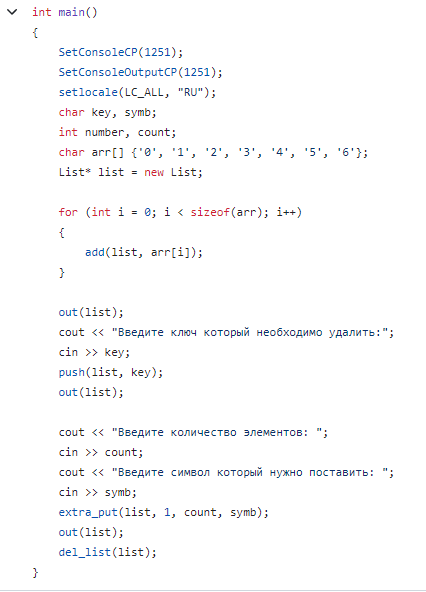
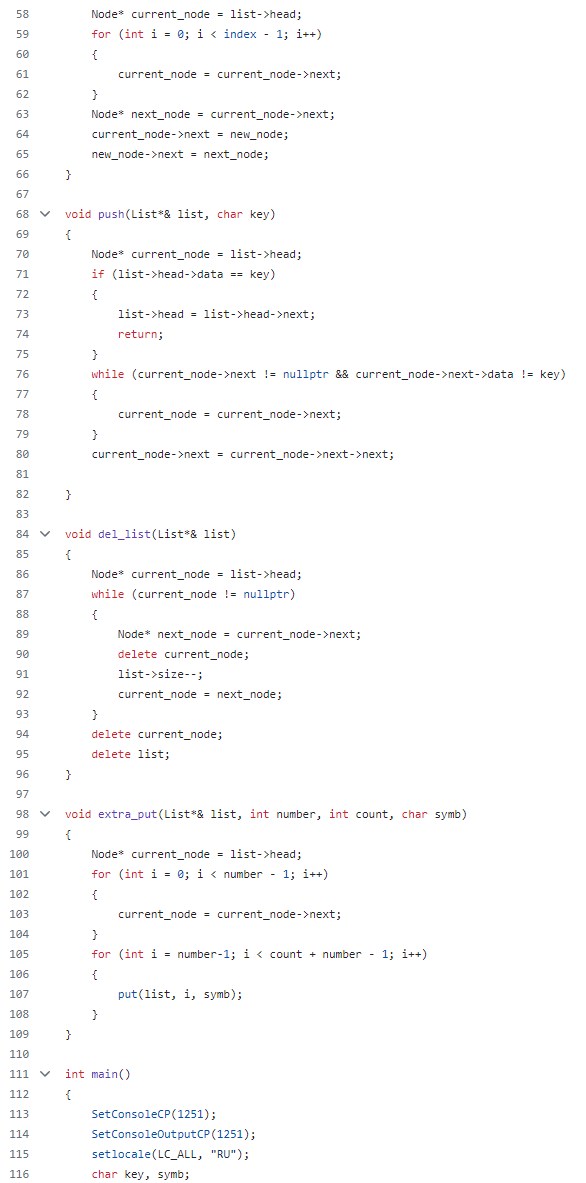
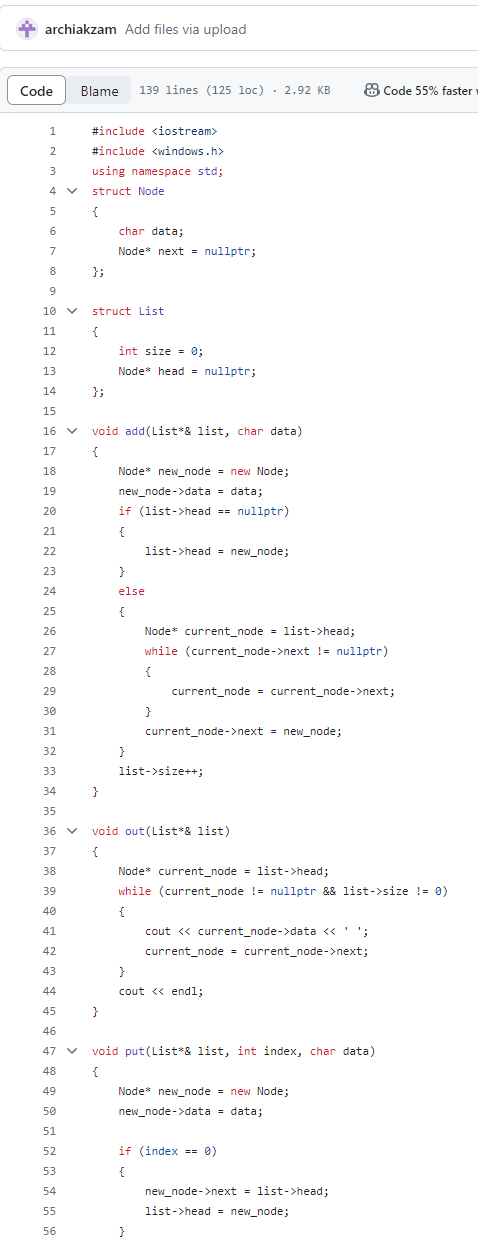
extra\_put(list, 1, count, symb);

out(list);

del\_list(list);

}

Скрины из git:



**Выводы:** программа сработала корректно и вывела все возможные решения.