Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учереждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Направление подготовки: Разработка информационных систем (РИС)

**Лабораторная работа N11**

Выполнил студент гр. РИС-24-3б

Караваев Артем Андреевич

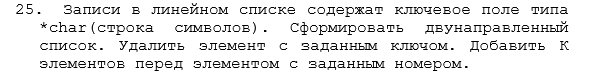
Проверил:

Доц. каф. ИТАС

Ольга Андреевна Полякова

г. Пермь, 2024

**Постановка задачи:**

****

**Код программы:**

#include <iostream>

#include <cstring>

using namespace std;

//стек:

struct nodestack {

char data[50];

nodestack\* nextptr;

};

struct Stack {

int size = 0;

nodestack\* head = nullptr;

};

void dob(Stack& st, char data[50]) {

nodestack\* new\_dinam = new nodestack;

for (int i = 0; i < 50; i++) {

new\_dinam->data[i] = data[i];

}

new\_dinam->nextptr = st.head;

st.head = new\_dinam;

st.size++;

}

void del(Stack& st, char data[50], Stack& st1) {

nodestack\* new\_dinam = st.head;

bool flag = true;

while (new\_dinam != nullptr) {

if ((strcmp(new\_dinam->data, data) == 0) && flag) {

new\_dinam = new\_dinam->nextptr;

flag = false;

st.size--;

}

else if (new\_dinam != nullptr) {

dob(st1, new\_dinam->data);

new\_dinam = new\_dinam->nextptr;

}

}

}

void dobk(Stack& st, int k, int number, Stack& st1) {

nodestack\* new\_dinam = st.head;

char m[50];

int size = st.size;

bool flag = true;

while (flag) {

if (size == number - 1) {

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

cout << "Элемент " << k - i << ": ";

cin >> m;

nodestack\* new\_dinam2 = new nodestack;

for (int i = 0; i < 50; i++) {

new\_dinam2->data[i] = m[i];

}

dob(st1, new\_dinam2->data);

}

}

if (new\_dinam != nullptr) {

size--;

dob(st1, new\_dinam->data);

new\_dinam = new\_dinam->nextptr;

}

else {

flag = false;

}

}

}

void showStack(const Stack& st) {

nodestack\* current = st.head;

while (current != nullptr) {

cout << current->data << " ";

current = current->nextptr;

}

cout << endl;

}

void clear(Stack& st)

{

nodestack\* current = st.head;

nodestack\* next;

while (current != nullptr) {

next = current->nextptr;

delete current;

current = next;

}

st.head = nullptr;

}

//однонаправленный список

struct nodelist {

char data[50];

nodelist\* nextptr;

};

struct List {

int size = 0;

nodelist\* head = nullptr;

nodelist\* tail = nullptr;

};

void dob(List& list, char data[50]) {

nodelist\* new\_dinam = new nodelist;

for (int i = 0; i < 50; i++) {

new\_dinam->data[i] = data[i];

}

new\_dinam->nextptr = nullptr;

if (list.head == nullptr) list.head = new\_dinam;

if (list.tail != nullptr) list.tail->nextptr = new\_dinam;

list.tail = new\_dinam;

list.size++;

}

void del(List& list, char data[50]) {

nodelist\* new\_dinam = list.head;

if (strcmp(new\_dinam->data, data) == 0) {

nodelist\* temp = new\_dinam;

list.head = new\_dinam->nextptr;

delete temp;

list.size--;

}

else {

while (new\_dinam->nextptr != nullptr && !(strcmp(new\_dinam->nextptr->data, data) == 0)) {

new\_dinam = new\_dinam->nextptr;

}

if (new\_dinam->nextptr != nullptr && strcmp(new\_dinam->nextptr->data, data) == 0) {

nodelist\* temp = new\_dinam->nextptr;

new\_dinam->nextptr = new\_dinam->nextptr->nextptr;

delete temp;

list.size--;

}

}

}

void dobk(List& list, int k, int number) {

char m[50];

if (number == 1) {

nodelist\* new\_dinam1 = list.head;

cout << "Элемент 1" << ": ";

cin >> m;

nodelist\* new\_dinam = new nodelist;

list.head = new\_dinam;

for (int i = 0; i < 50; i++) {

new\_dinam->data[i] = m[i];

}

for (int i = 1; i < k; i++) {

cout << "Элемент " << i + 1 << ": ";

cin >> m;

nodelist\* new\_dinam2 = new nodelist;

for (int i = 0; i < 50; i++) {

new\_dinam2->data[i] = m[i];

}

new\_dinam->nextptr = new\_dinam2;

new\_dinam = new\_dinam2;

list.size++;

}

new\_dinam->nextptr = new\_dinam1;

}

else {

int l = 1;

nodelist\* new\_dinam = list.head;

while (l <= number - 2) {

new\_dinam = new\_dinam->nextptr;

l++;

}

nodelist\* new\_dinam1 = new\_dinam->nextptr;

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

cout << "Элемент " << i + 1 << ": ";

cin >> m;

nodelist\* new\_dinam2 = new nodelist;

for (int i = 0; i < 50; i++) {

new\_dinam2->data[i] = m[i];

}

new\_dinam->nextptr = new\_dinam2;

new\_dinam = new\_dinam2;

list.size++;

}

new\_dinam->nextptr = new\_dinam1;

}

}

void clear(List& list)

{

nodelist\* current = list.head;

nodelist\* next;

while (current != nullptr) {

next = current->nextptr;

delete current;

current = next;

}

list.head = nullptr;

list.tail = nullptr;

}

void showList(const List& list) {

nodelist\* current = list.head;

while (current != nullptr) {

cout << current->data << " ";

current = current->nextptr;

}

cout << endl;

}

//двунаправленный список

struct nodelist1 {

char data[50];

nodelist1\* nextptr;

nodelist1\* prevptr;

};

struct List1 {

int size = 0;

nodelist1\* head = nullptr;

nodelist1\* tail = nullptr;

};

void dob(List1& list, char data[50]) {

nodelist1\* new\_dinam = new nodelist1;

for (int i = 0; i < 50; i++) {

new\_dinam->data[i] = data[i];

}

new\_dinam->nextptr = nullptr;

new\_dinam->prevptr = list.tail;

if (list.head == nullptr) list.head = new\_dinam;

if (list.tail != nullptr) list.tail->nextptr = new\_dinam;

list.tail = new\_dinam;

list.size++;

}

void del(List1& list, char data[50]) {

nodelist1\* new\_dinam = list.head;

if (strcmp(new\_dinam->data, data) == 0) {

nodelist1\* temp = new\_dinam;

list.head = new\_dinam->nextptr;

list.head->prevptr = nullptr;

delete temp;

list.size--;

}

else {

while (new\_dinam->nextptr != nullptr && !(strcmp(new\_dinam->nextptr->data, data) == 0)) {

new\_dinam = new\_dinam->nextptr;

}

if (new\_dinam->nextptr != nullptr && strcmp(new\_dinam->nextptr->data, data) == 0) {

nodelist1\* temp = new\_dinam->nextptr;

new\_dinam->nextptr = new\_dinam->nextptr->nextptr;

new\_dinam->nextptr->prevptr = new\_dinam;

delete temp;

list.size--;

}

}

}

void dobk(List1& list, int k, int number) {

char m[50];

if (number == 1) {

nodelist1\* new\_dinam1 = list.head;

cout << "Элемент 1" << ": ";

cin >> m;

nodelist1\* new\_dinam = new nodelist1;

list.head = new\_dinam;

for (int i = 0; i < 50; i++) {

new\_dinam->data[i] = m[i];

}

for (int i = 1; i < k; i++) {

cout << "Элемент " << i + 1 << ": ";

cin >> m;

nodelist1\* new\_dinam2 = new nodelist1;

for (int i = 0; i < 50; i++) {

new\_dinam2->data[i] = m[i];

}

new\_dinam->nextptr = new\_dinam2;

new\_dinam2->prevptr = new\_dinam;

new\_dinam = new\_dinam2;

list.size++;

}

new\_dinam->nextptr = new\_dinam1;

new\_dinam1->prevptr = new\_dinam;

}

else if (list.size / 2 >= number) {

int l = 1;

nodelist1\* new\_dinam = list.head;

while (l <= number - 2) {

new\_dinam = new\_dinam->nextptr;

l++;

}

nodelist1\* new\_dinam1 = new\_dinam->nextptr;

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

cout << "Элемент " << i << ": ";

cin >> m;

nodelist1\* new\_dinam2 = new nodelist1;

for (int i = 0; i < 50; i++) {

new\_dinam2->data[i] = m[i];

}

new\_dinam->nextptr = new\_dinam2;

new\_dinam2->prevptr = new\_dinam;

new\_dinam = new\_dinam2;

list.size++;

}

new\_dinam->nextptr = new\_dinam1;

new\_dinam1->prevptr = new\_dinam;

}

else {

int l = list.size;

nodelist1\* new\_dinam = list.tail;

while (number + 1 <= l) {

new\_dinam = new\_dinam->prevptr;

l--;

}

nodelist1\* new\_dinam1 = new\_dinam->prevptr;

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

cout << "Элемент " << k - i << ": ";

cin >> m;

nodelist1\* new\_dinam2 = new nodelist1;

for (int i = 0; i < 50; i++) {

new\_dinam2->data[i] = m[i];

}

new\_dinam->prevptr = new\_dinam2;

new\_dinam2->nextptr = new\_dinam;

new\_dinam = new\_dinam2;

list.size++;

}

new\_dinam->prevptr = new\_dinam1;

new\_dinam1->nextptr = new\_dinam;

}

}

void showList1(const List1& list) {

nodelist1\* current = list.head;

while (current != nullptr) {

cout << current->data << " ";

current = current->nextptr;

}

cout << endl;

}

void clear(List1& list)

{

nodelist1\* current = list.head;

nodelist1\* next;

while (current != nullptr) {

next = current->nextptr;

delete current;

current = next;

}

list.head = nullptr;

list.tail = nullptr;

}

//очередь

struct nodeq {

char data[50];

nodeq\* nextptr = nullptr;

};

struct Queue {

int size = 0;

nodeq\* head = nullptr;

nodeq\* tail = nullptr;

};

void dob(Queue& q, char data[50]) {

nodeq\* new\_dinam = new nodeq;

for (int i = 0; i < 50; i++) {

new\_dinam->data[i] = data[i];

}

if (q.head == nullptr) q.head = new\_dinam;

if (q.tail != nullptr) q.tail->nextptr = new\_dinam;

q.tail = new\_dinam;

q.size++;

}

void del(Queue& q, char data[50]) {

bool flag = true;

int size = q.size;

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

if (strcmp(q.head->data, data) == 0 && flag) {

nodeq\* temp = q.head;

q.head = q.head->nextptr;

delete temp;

flag = false;

q.size--;

}

else {

dob(q, q.head->data);

nodeq\* temp = q.head;

q.head = q.head->nextptr;

delete temp;

}

}

}

void dobk(Queue& q, int k, int number) {

bool flag = true;

int size = q.size;

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

if (i + 1 == number && flag) {

char m[50];

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

cout << "Элемент " << j + 1 << ": ";

cin >> m;

dob(q, m);

flag = false;

}

i--;

}

else {

dob(q, q.head->data);

nodeq\* temp = q.head;

q.head = q.head->nextptr;

delete temp;

}

}

}

void showQueue(const Queue& q) {

nodeq\* current = q.head;

while (current != nullptr) {

cout << current->data << " ";

current = current->nextptr;

}

cout << endl;

}

void clear(Queue& q)

{

nodeq\* current = q.head;

nodeq\* next;

while (current != nullptr) {

next = current->nextptr;

delete current;

current = next;

}

q.head = nullptr;

q.tail = nullptr;

}

int main() {

Stack st;

Stack st1;

List list;

List1 list1;

Queue q;

for(int i=0;i<5;i++){

char m[50];

cin>>m;

dob(st,m);

}

char del1[50]{"del"};

del(st,del1,st1);

showStack(st1);

for (int i = 0; i < 5; i++) {

char m[50];

cin >> m;

dob(st, m);

}

showStack(st);;

dobk(st, 2, 3, st1);

showStack(st1);

for(int i=0;i<5;i++){

char m[50];

cin>>m;

dob(list,m);

}

showList(list);

char del1[50]{"del"};

del(list,del1);

showList(list);

for(int i=0;i<5;i++){

char m[50];

cin>>m;

dob(list,m);

}

showList(list);

dobk(list,2,1);

showList(list);

for(int i=0;i<5;i++){

char m[50];

cin>>m;

dob(list1,m);

}

showList1(list1);

dobk(list1,3,4);

showList1(list1);

for(int i=0;i<5;i++){

char m[50];

cin>>m;

dob(q,m);

}

showQueue(q);

char del1[50]{"del"};

del(q,del1);

showQueue(q);

for (int i = 0; i < 5; i++) {

char m[50];

cin >> m;

dob(q, m);

}

showQueue(q);

dobk(q, 1, 5);

showQueue(q);

clear(q);

clear(list);

clear(list1);

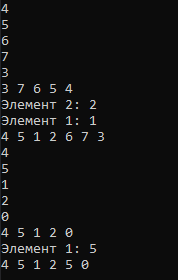
clear(st);

clear(st1);

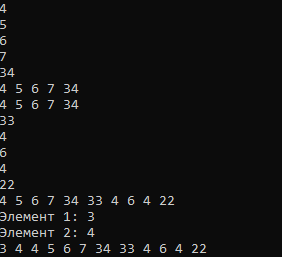
return 0;

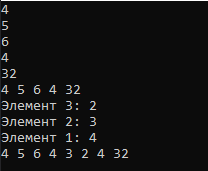
}

**Работа программы:**

****

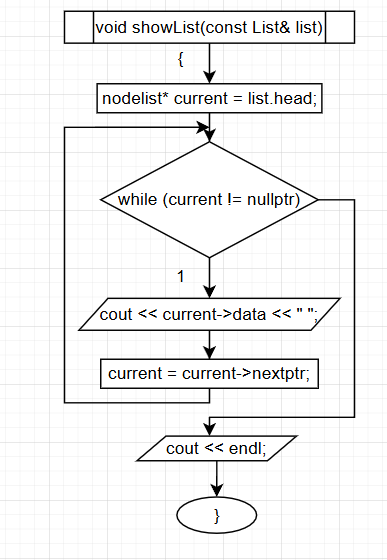
Стек + очередь

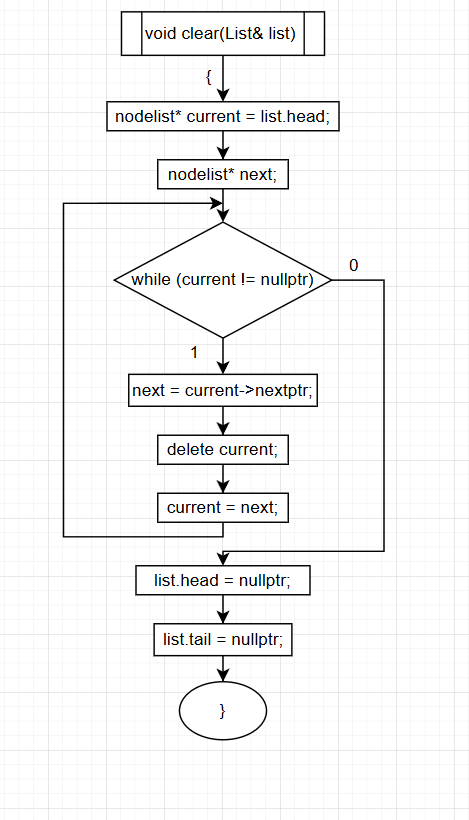


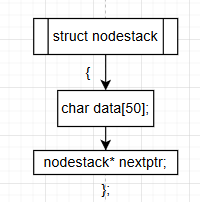


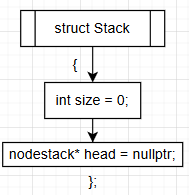
Оба списка

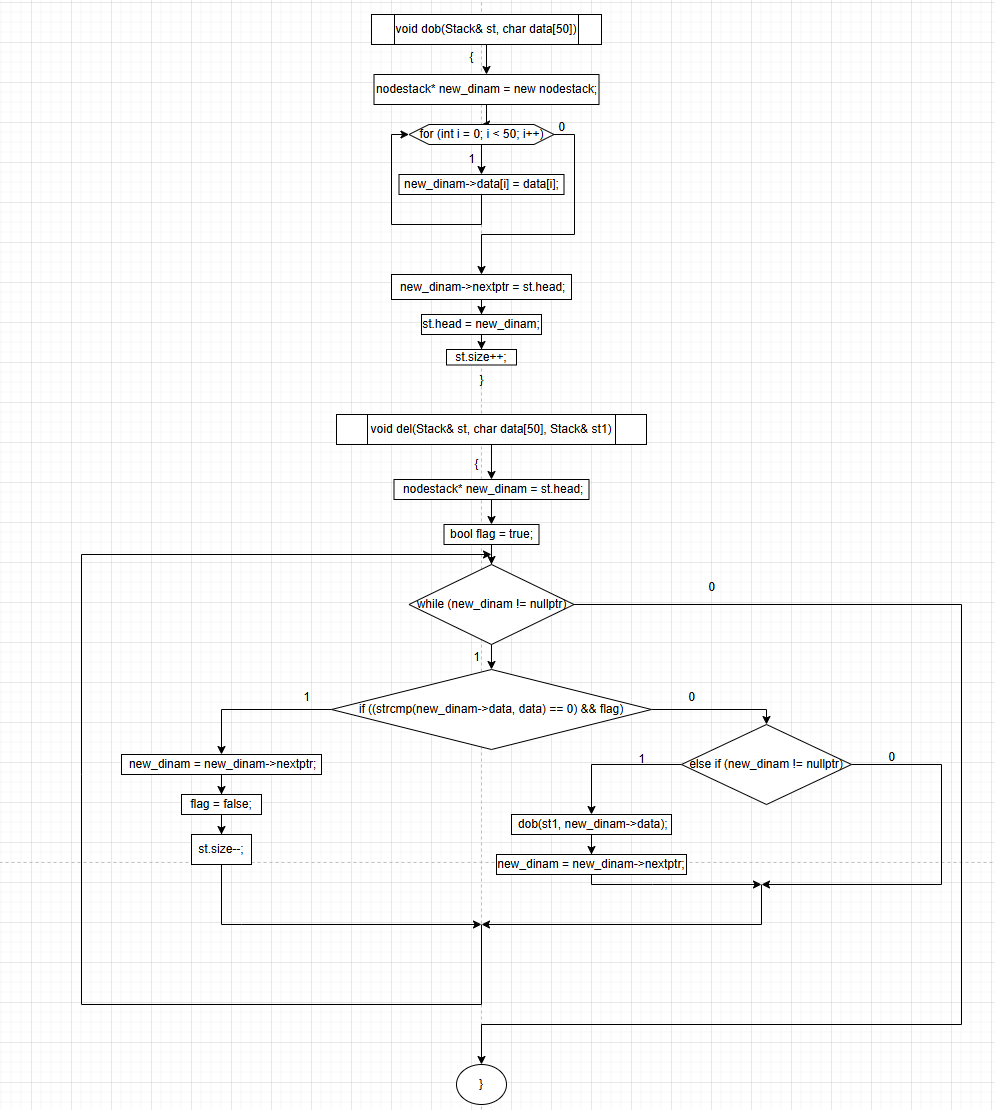
**Блок – схема:**

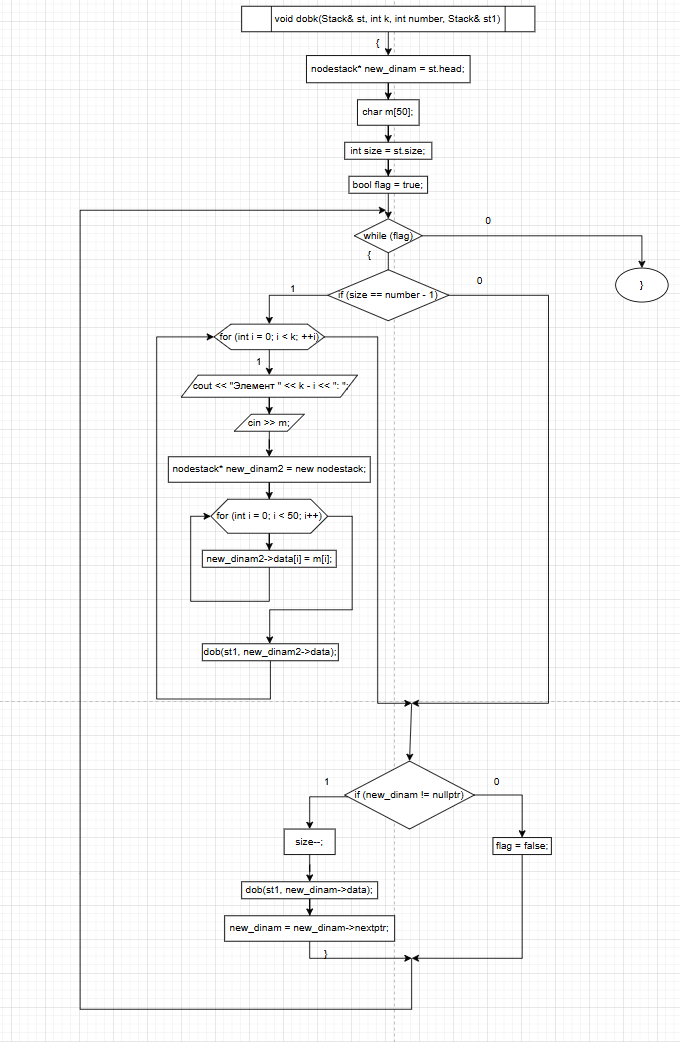
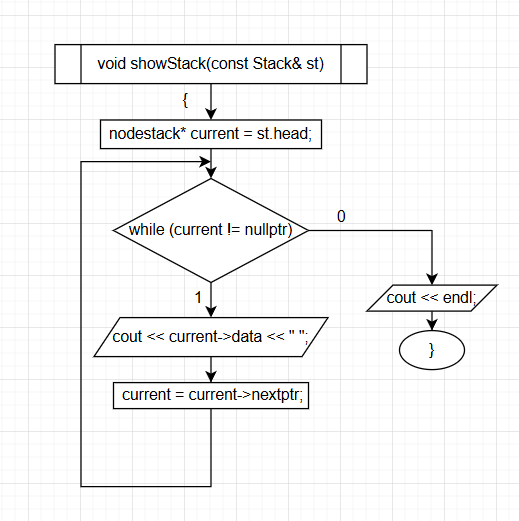
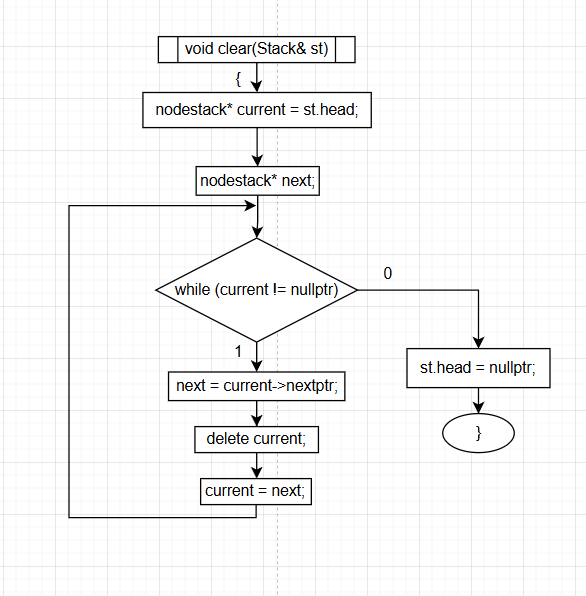
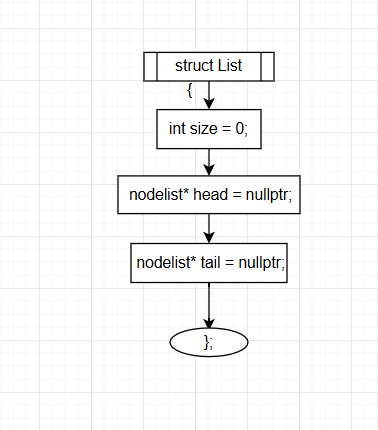
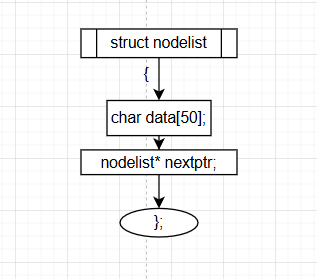
****

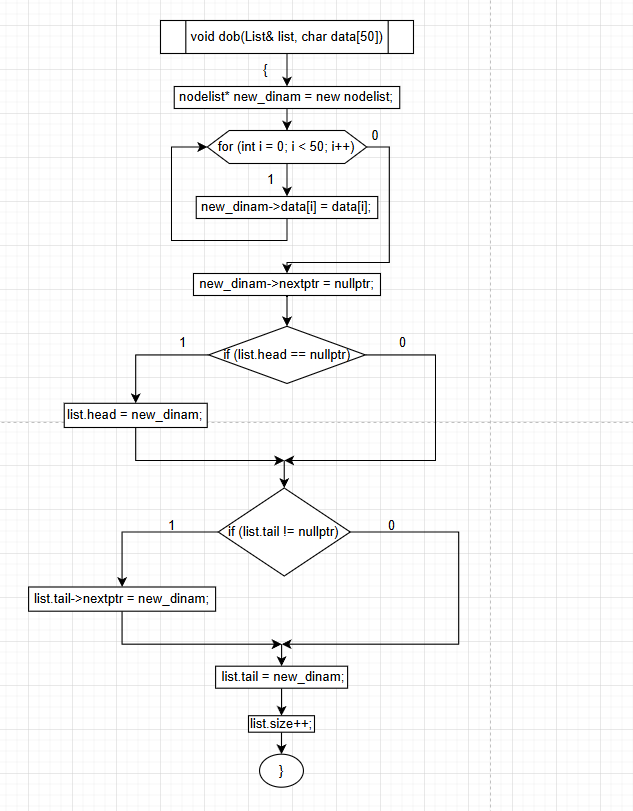


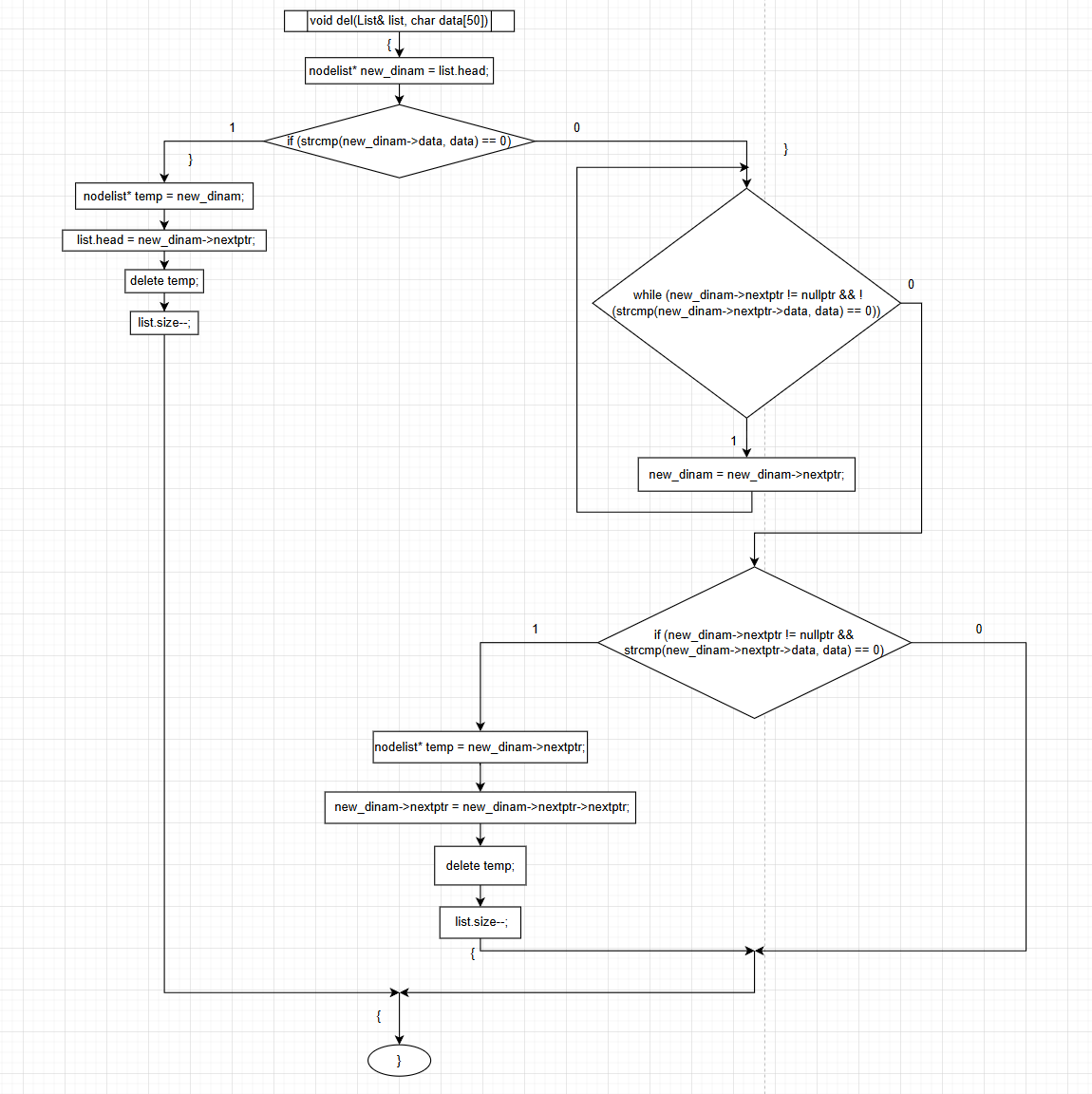


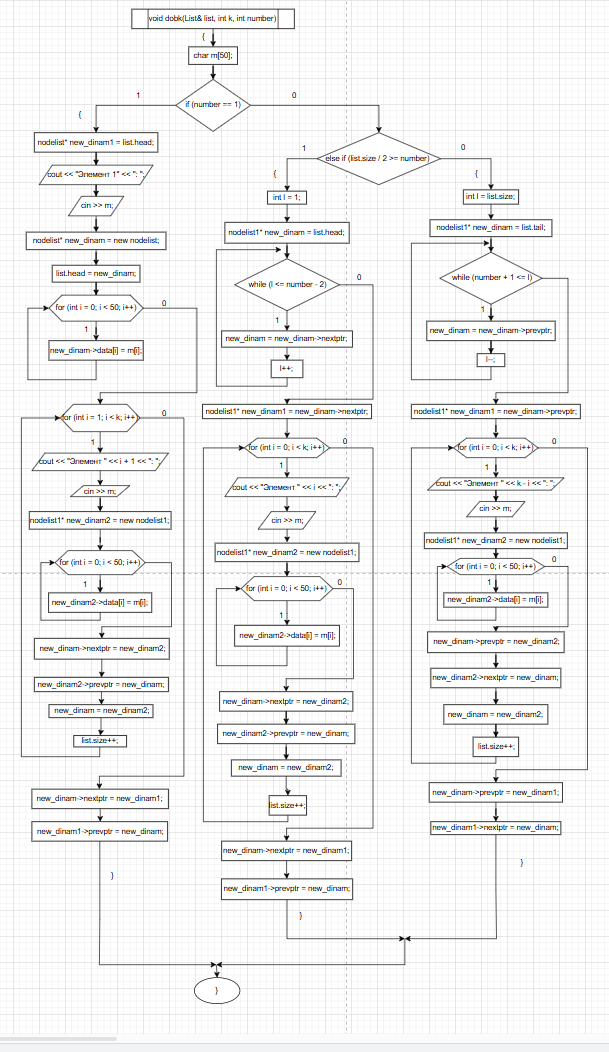


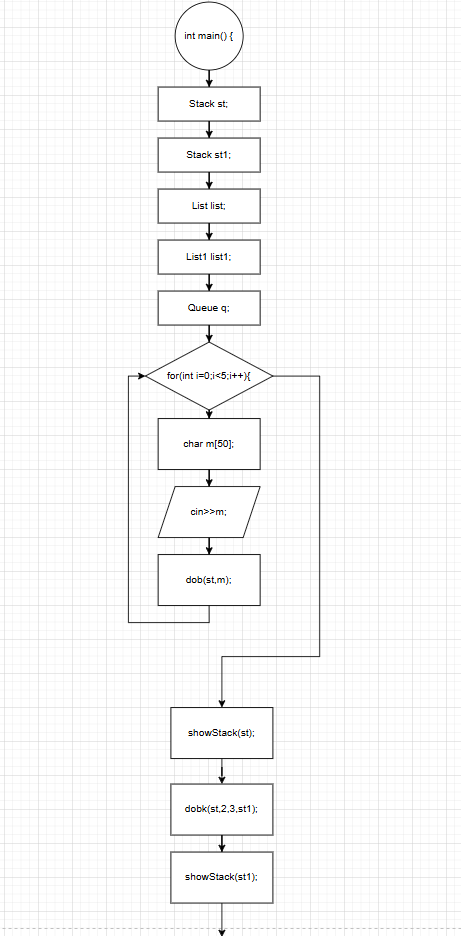


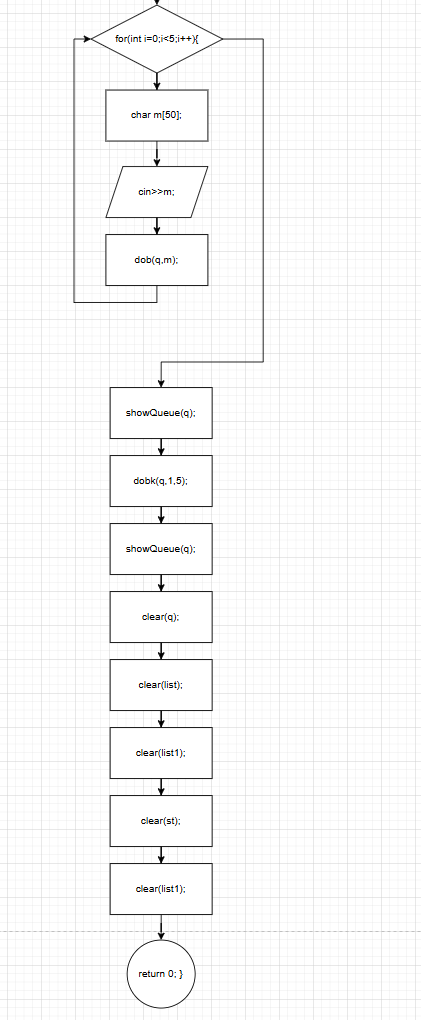
    

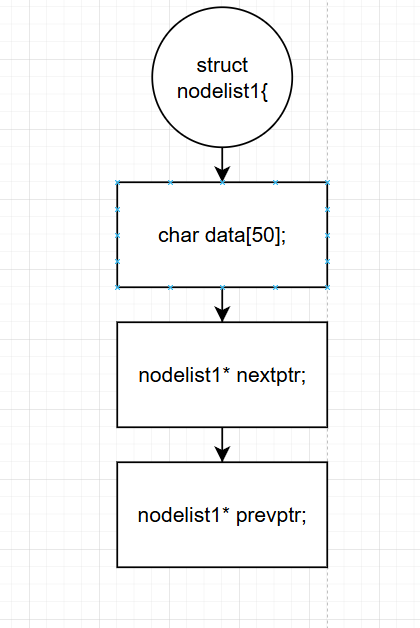


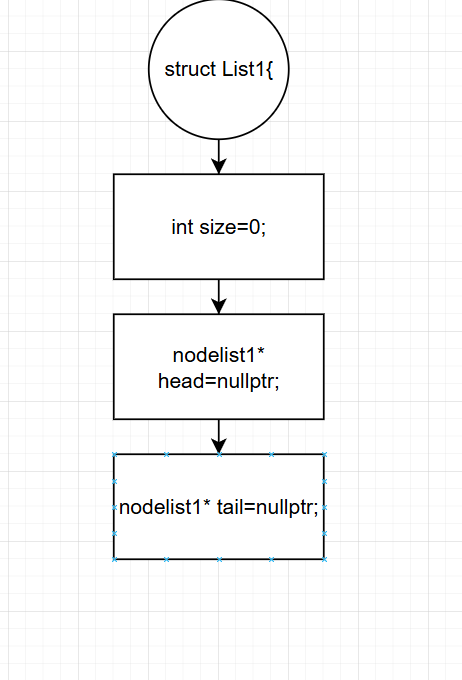
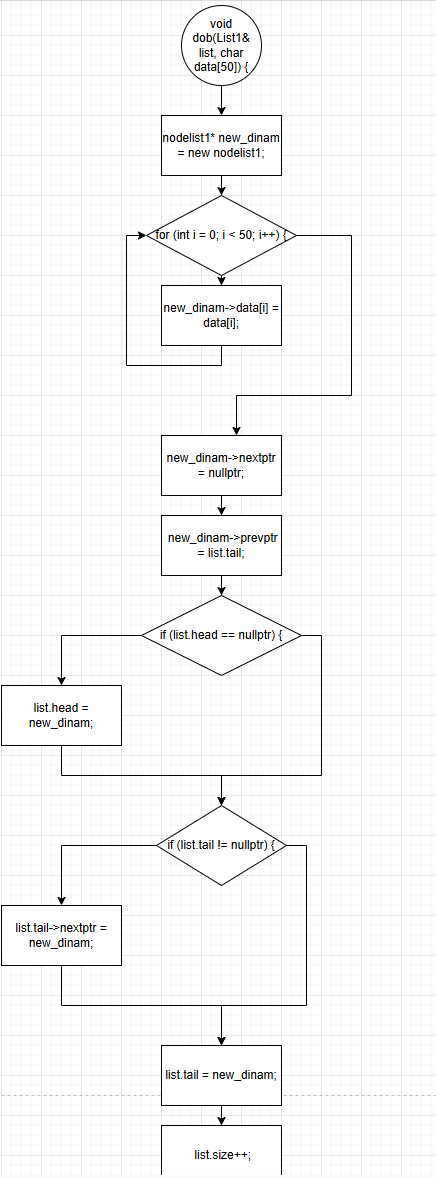
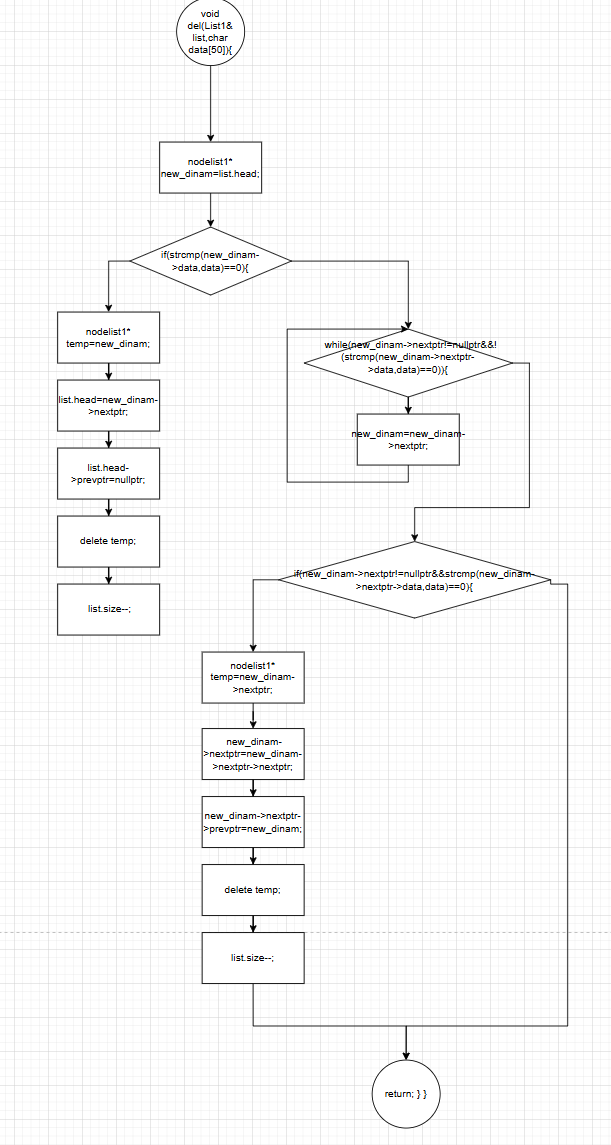
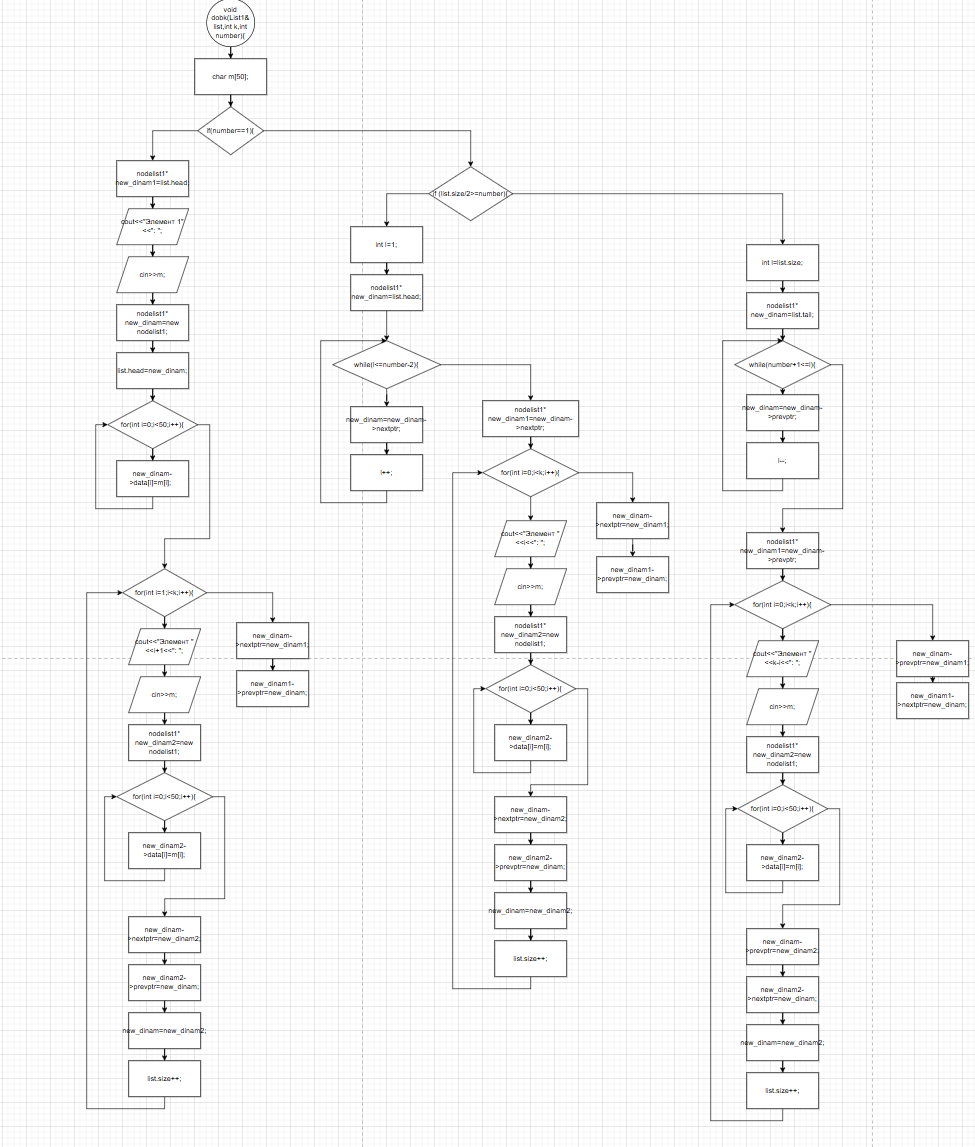
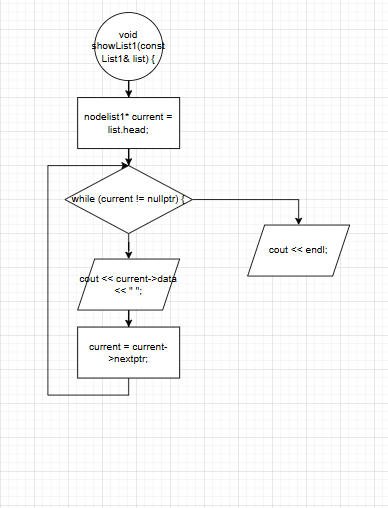
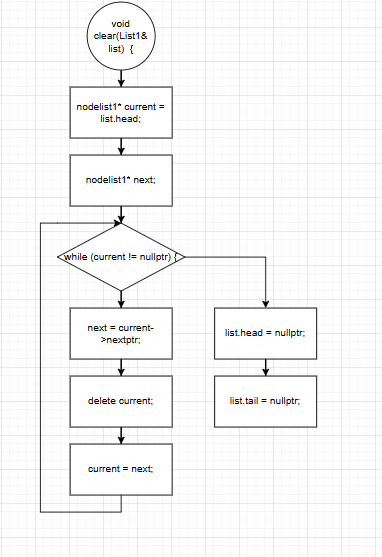
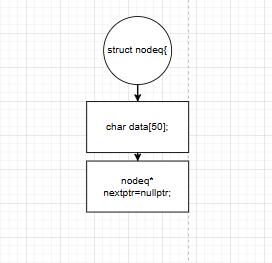
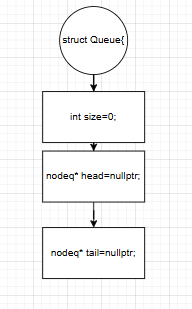
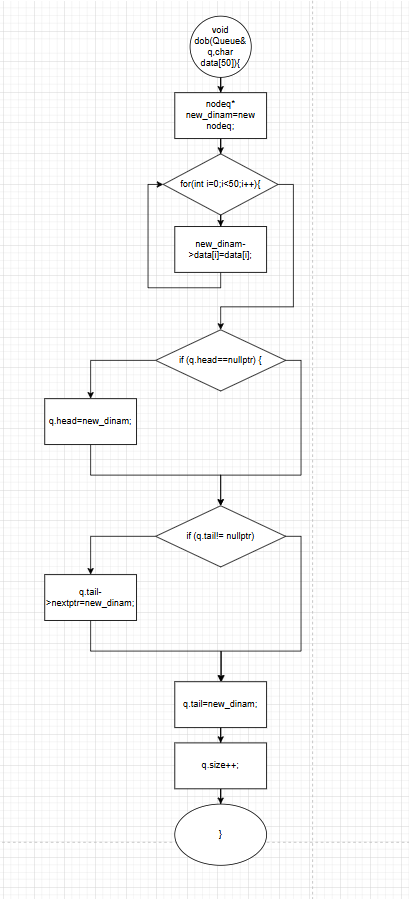
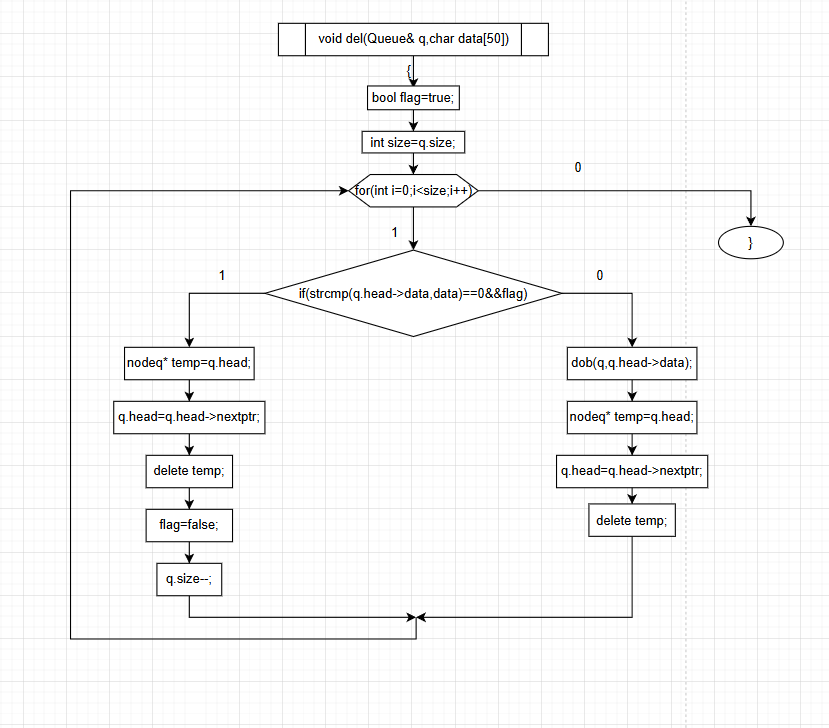
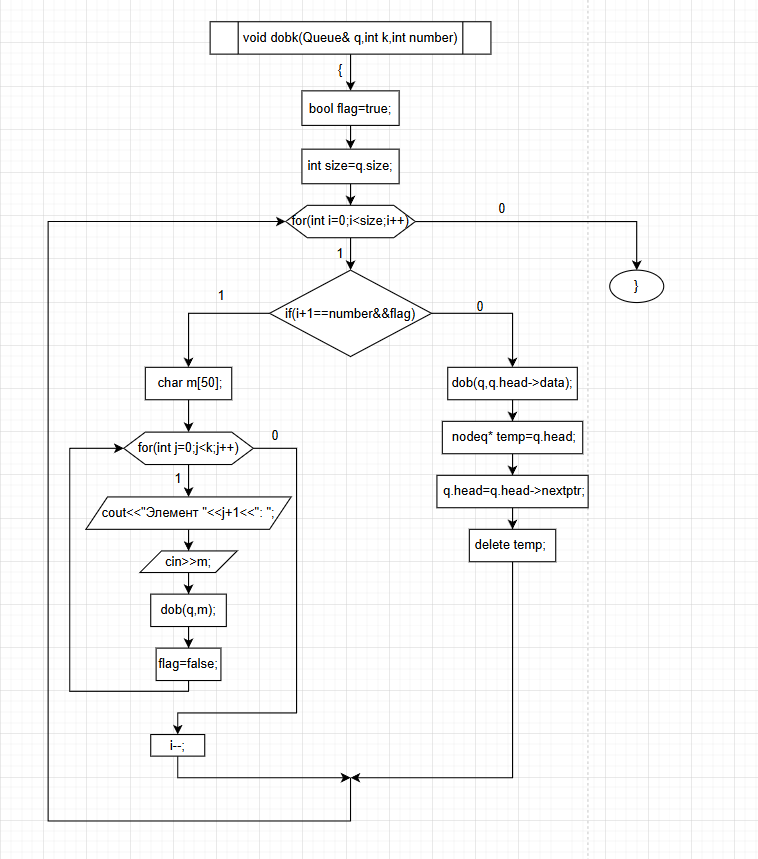
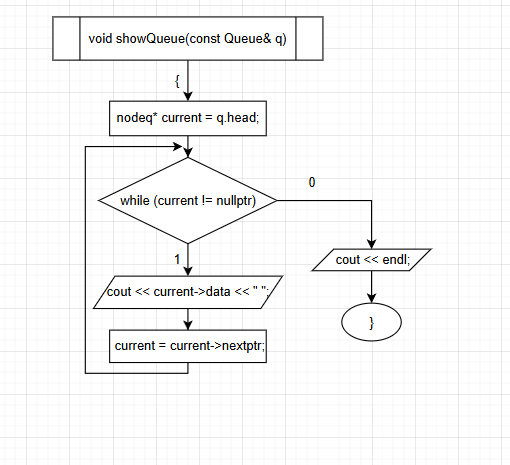
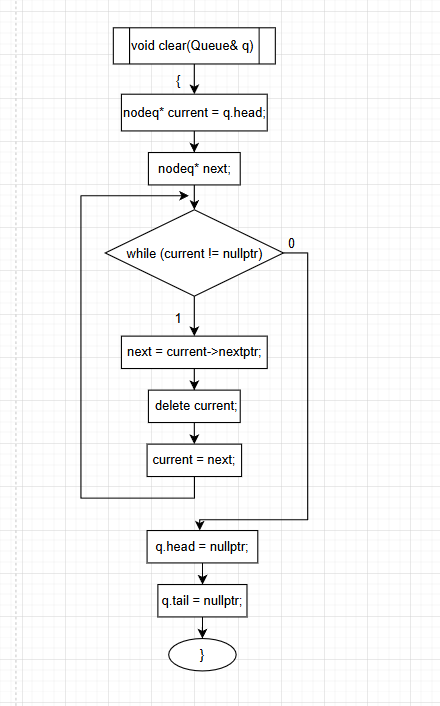










https://github.com/Prefix008/lab