МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ

И.С. ТУРГЕНЕВА»

#### Отчет по лабораторной работе №4

по дисциплине «Императивное программирование»

|  |
| --- |
| Работу выполнил(а) студент(ка)  **Тупикин М.Н.**  группа **91-ИТ**  **г. Орел** |
| Работу проверил(а)  **Амелина О.В.** |

Отметка о зачете дата «\_\_\_\_» \_\_\_\_\_\_\_\_\_20 г.

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**г. Орел, 20 г.**

Лабораторная работа №4

Тема: «Односвязный список»

#include <iostream>

#include <clocale>

using namespace std;

typedef struct Node {

int value;

struct Node \*next;

} Node;

void pushFirst(Node \*\*head, int data) {

Node \*tmp = (Node\*) malloc(sizeof(Node));

tmp->value = data;

tmp->next = (\*head);

(\*head) = tmp;

}

int popFirst(Node \*\*head) {

Node\* prev = NULL;

int val;

if (head == NULL) {

exit(-1);

}

prev = (\*head);

val = prev->value;

(\*head) = (\*head)->next;

free(prev);

return val;

}

Node\* getNth(Node\* head, int n) {

int counter = 0;

while (counter < n && head) {

head = head->next;

counter++;

}

return head;

}

Node\* getLast(Node \*head) {

if (head == NULL) {

return NULL;

}

while (head->next) {

head = head->next;

}

return head;

}

void pushBack(Node \*head, int value) {

Node \*last = getLast(head);

Node \*tmp = (Node\*) malloc(sizeof(Node));

tmp->value = value;

tmp->next = NULL;

last->next = tmp;

}

Node\* predLast(Node\* head) {

if (head == NULL) {

exit(-2);

}

if (head->next == NULL) {

return NULL;

}

while (head->next->next) {

head = head->next;

}

return head;

}

int popBack(Node \*\*head) {

Node \*lastbn = NULL;

int val;

if (!head) {

exit(-1);

}

if (!(\*head)) {

exit(-1);

}

lastbn = predLast(\*head);

val = lastbn->next->value; //следующий за предыдущим

if (lastbn == NULL) {

free(\*head);

\*head = NULL;

} else {

free(lastbn->next);

lastbn->next = NULL;

}

return val;

}

void PushNPosition(Node \*head, unsigned n, int val) {

unsigned i = 0;

Node \*tmp = NULL;

while (i < n && head->next) {

head = head->next;

i++;

}

tmp = (Node\*) malloc(sizeof(Node));

tmp->value = val;

if (head->next) {

tmp->next = head->next;

} else {

tmp->next = NULL;

}

head->next = tmp;

}

int popNPosition(Node \*\*head, int n) {

if (n == 0) {

return popFirst(head);

} else {

Node \*prev = getNth(\*head, n-1);

Node \*elm = prev->next;

int val = elm->value;

prev->next = elm->next;

free(elm);

return val;

}

}

void deleteList(Node \*\*head) {

while ((\*head)->next) {

popFirst(head);

\*head = (\*head)->next;

}

free(\*head);

}

void print(const Node \*head) {

while (head) {

printf("%d ", head->value);

head = head->next;

}

printf("\n");

}

void massivInList(Node \*\*head, int \*arr, size\_t size) {

size\_t i = size - 1;

if (arr == NULL || size == 0) {

return;

}

do {

pushFirst(head, arr[i]);

} while(i--!=0);

}

int main (){

setlocale(0, "");

Node\* head = NULL;

int arr[] = {1,2,3,4,5,6,7,8,9,10};

massivInList(&head, arr, sizeof(arr) / sizeof(arr[0]));

cout<<"Элементы в списке:\n";

print(head);

int com;

while (true){

cout<<"Что хотите сделать?\n";

cout<<"1 - Добавить элемент в начало списка\n";

cout<<"2 - Добавить элемент в конец списка\n";

cout<<"3 - Добавить элемент в определенное место списка\n";

cout<<"4 - Удалить первый элемент в списке\n";

cout<<"5 - Удалить последний элемент в списке\n";

cout<<"6 - Удалить определенный элемент в списке\n";

cin>>com;

int a = 0;

switch(com){

case 1:

cout<<"Введите элемент: \n";

cin>>a;

pushFirst(&head, a);

cout<<"Новый список:\n";

print(head);

break;

case 2:

cout<<"Введите элемент: \n";

cin>>a;

pushBack(head, a);

cout<<"Новый список:\n";

print(head);

break;

case 3:

cout<<"Список:\n";

print(head);

int b;

cout<<"Введите позицию элемента после которого хотите добавить элемент:\n";

cin>>b;

b=b-1;

cout<<"Введите элемент: \n";

cin>>a;

PushNPosition(head, b, a);

cout<<"Новый список:\n";

print(head);

break;

case 4:

cout<<"Элемент = "<<popFirst(&head)<<" удален\n";

cout<<"Новый список:\n";

print(head);

break;

case 5:

cout<<"Элемент = "<<popBack(&head)<<" удален\n";

cout<<"Новый список:\n";

print(head);

break;

case 6:

cout<<"Введите позицию элемента которого хотите удалить:\n";

cin>>a;

a = a-1;

cout<<"Элемент = "<<popNPosition(&head, a)<<" удален\n";

cout<<"Новый список:\n";

print(head);

break;

default:

cout<<"Команда не найдена\n";

break;

}

}

deleteList(&head);

return 0;

}













