Министерство науки и высшего образования РФ

Пензенский государственный университет

Кафедра «Вычислительная кафедра»

**ОТЧЕТ**

по лабораторной работе №3

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Динамические списки»

Выполнила:

студентка группы 22ВВП1

Расторгуева К.В.

Приняли:

Акифьев И.В.

Митрохин М.А.

Пенза 2023

**Название**

Динамические списки.

**Цель работы**

Повторение динамических структур данных: динамических списков.

**Лабораторное задание**

1. Реализовать приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта (т.е. объект с большим приоритетом становится перед объектом с меньшим приоритетом).
2. \* На основе приведенного кода реализуйте структуру данных *Очередь*.
3. \* На основе приведенного кода реализуйте структуру данных *Стек*.

**Пояснительный текст к программе**

Для каждого задания программа содержит основные функции работы со списком: создание, добавление, удаление элемента, просмотр списка, нахождение нужного элемента списка.

Задание 1:

Создаётся односвязный список, т.е. каждый элемент содержит только одну ссылку на следующий элемент. Реализация приоритетной очереди происходит следующим образом. Сначала проверяется наличие списка. Если списка нет, то устанавливается голова списка:

if (head == NULL && p != NULL) {

head = p;

last = p;

}

Если список уже сформирован, приоритет каждого узла сравнивается с приоритетом добавляемого узла. Отдельно обрабатывается случай добавления узла перед головой текущего списка:

else if (head != NULL && p != NULL) {

do {

if (struc->priority > p->priority) {

if (prev==NULL) {

p->next = head;

head = p;

}

else {

prev->next = p;

p->next = struc;

}

return;

}

prev = struc;

struc = struc->next;

} while (struc != NULL);

last = prev;

last->next = p;

last = p;

}

return;

Задания 2 и 3:

Создаётся двусвязный список, в котором каждый из элементов имеет содержит ссылку как на следующий элемент (*next*), так и на предыдущий (*prev*). Функции создания, добавления, удаления элемента, просмотра списка, нахождения нужного элемента списка написаны с учетом особенностей структур данных Стек и Очередь.

Назначение отдельных операторов программы указывается в виде комментариев. Программа завершает свою работу после нажатия на любую клавишу на клавиатуре.

**Результат выполнения программы**

Результат работы программы для задания 1 представлены на рис. 1-2.

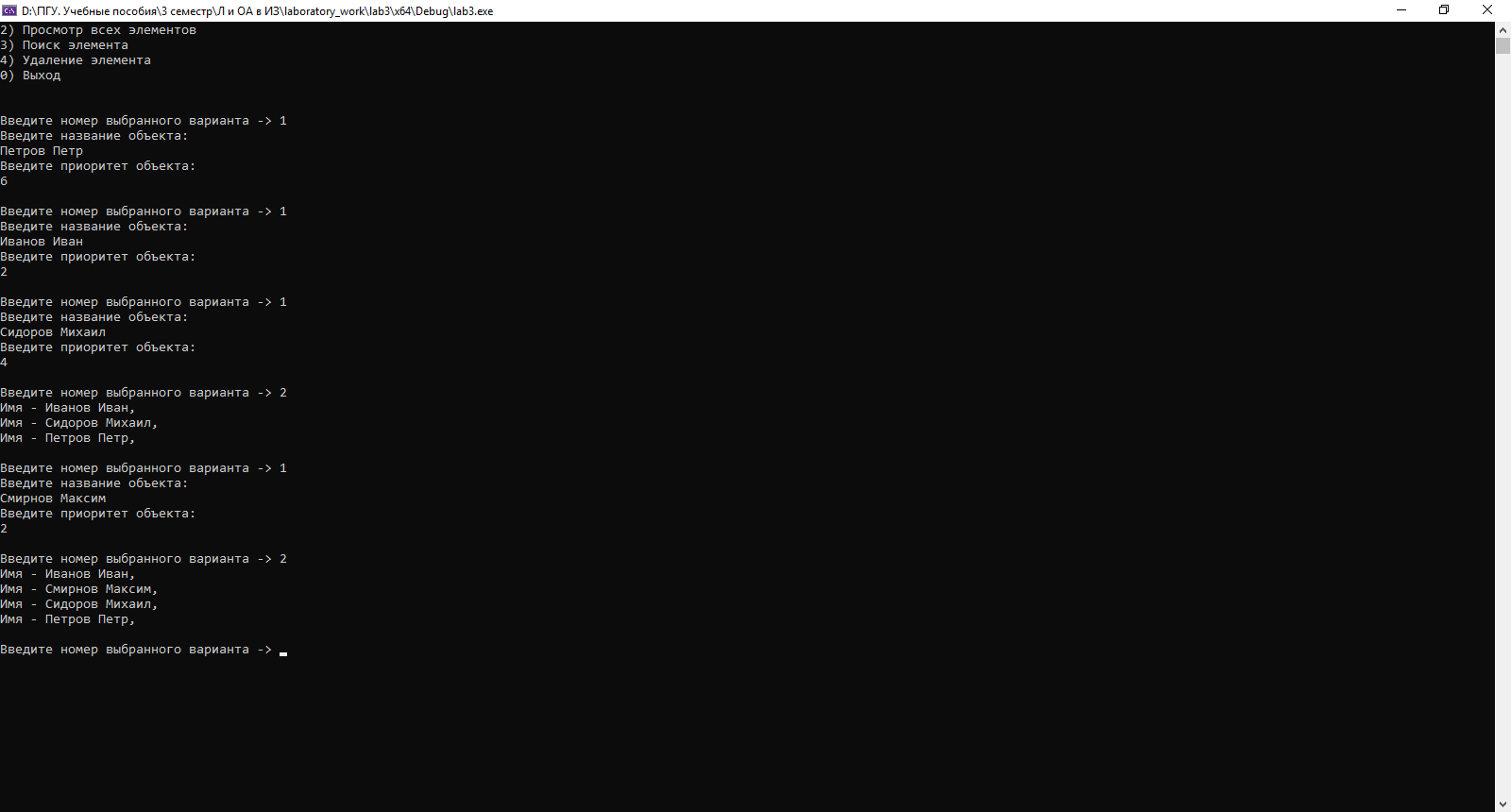


Рисунок 1 – очередь с приоритетом

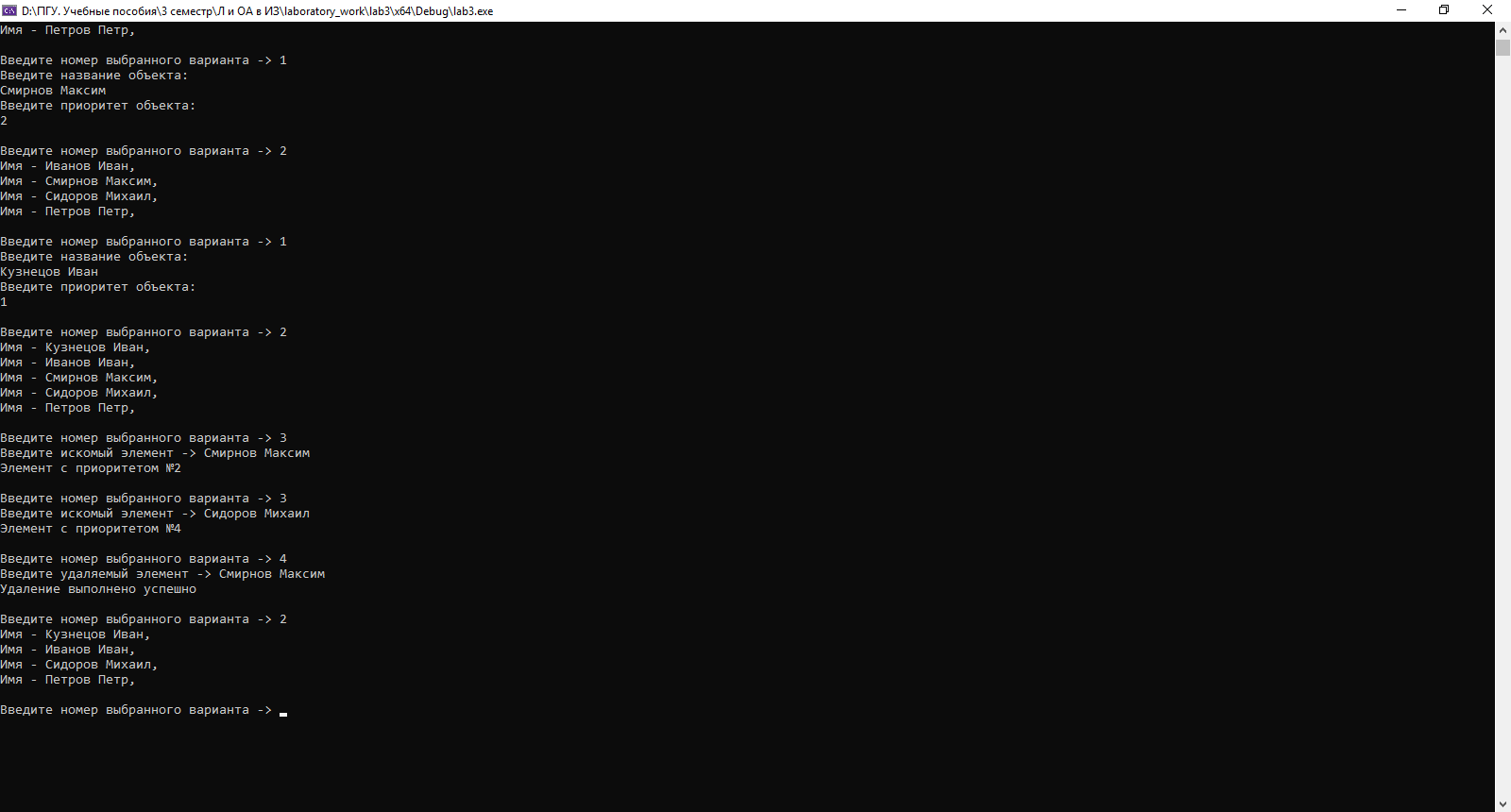


Рисунок 2 – удаление и поиск элементов

Результат работы программы для задания 2 представлены на рис. 3.

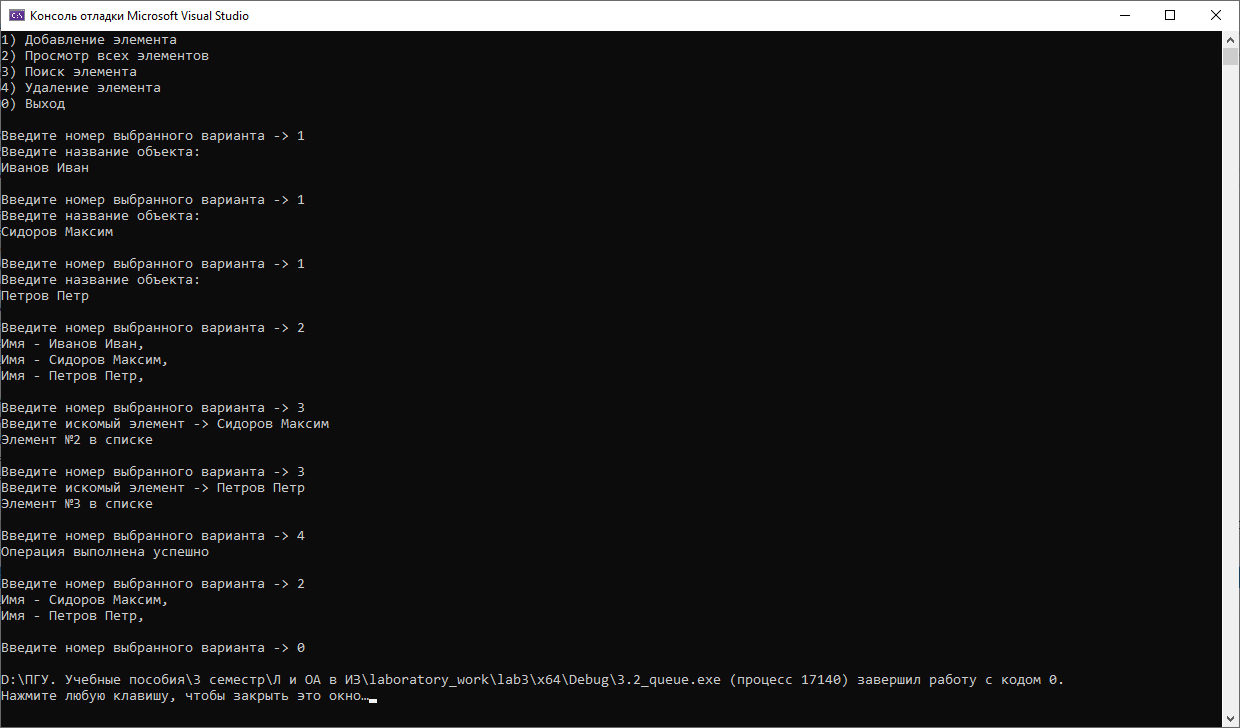


Рисунок 3 – работа со структурой данных Очередь

Результат работы программы для задания 3 представлены на рис. 4.

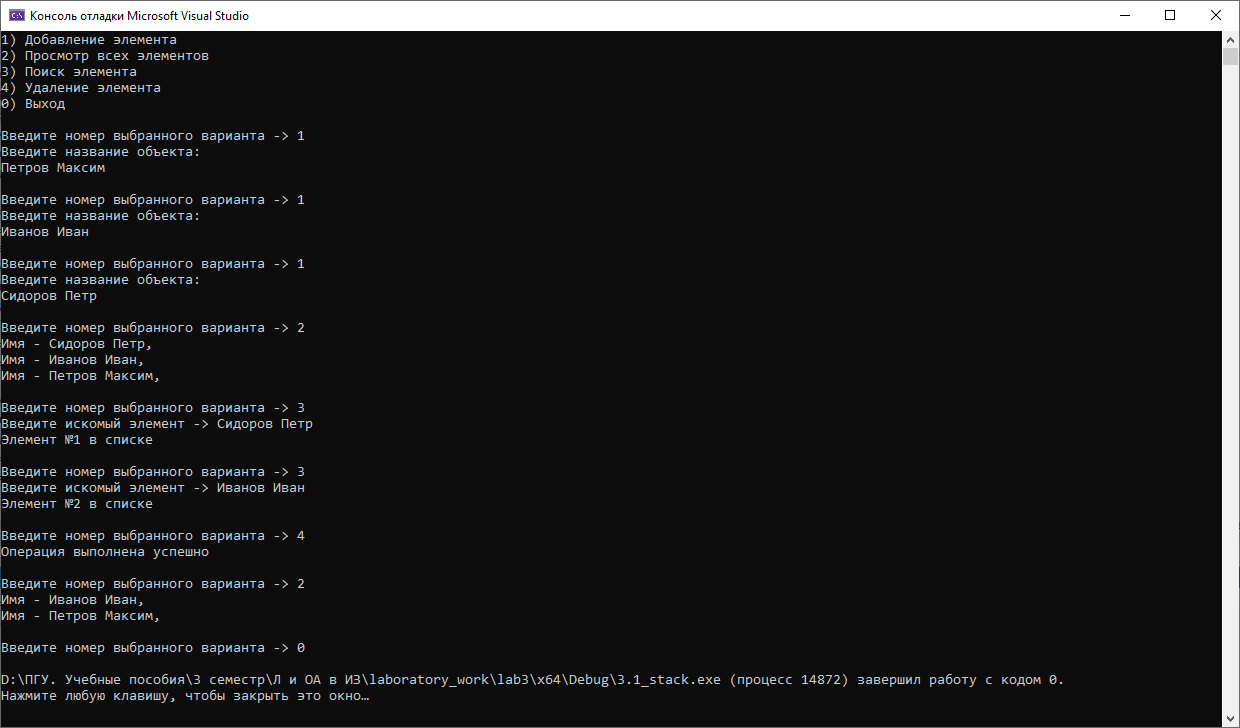


Рисунок 4 – работа со структурой данных Стек

Результаты вычисления вручную совпали с вычислениями программы.

**Вывод**

В ходе выполнения лабораторной работы была разработана на языке Си программа, осуществляющая работу с динамическими списками:

– реализована приоритетная очередь;

– реализована структура данных *Очередь*;

– реализована структура данных *Стек*;

Результаты работы программы совпали с результатами расчета вручную.

**Приложение**

**Программа**

Задание 1.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <string.h>

struct node{

int priority;

char inf[256]; // полезная информация

struct node\* next; // ссылка на следующий элемент

};

struct node\* head = NULL, \* last = NULL, \* f = NULL; // указатели на первый и последний элементы списка

int dlinna = 0;

// Функции добавления элемента, просмотра списка

void spstore(void), review(void), del(char\* name);

char find\_el[256];

struct node\* find(char\* name); // функция нахождения элемента

struct node\* get\_struct(void); // функция создания элемента

void crop(char\* str) {

int str\_num = 0;

while (str[str\_num] != '\0')

{

if (str[str\_num] == char(10))

{

str[str\_num] = char(0);

};

str\_num++;

};

}

struct node\* get\_struct(void)

{

struct node\* p = NULL;

char s[256];

if ((p = (node\*)malloc(sizeof(struct node))) == NULL) // выделяем память под новый элемент списка

{

printf("Ошибка при распределении памяти\n");

exit(1);

}

printf("Введите название объекта: \n"); // вводим данные

getchar();

fgets(s, 50, stdin);

crop(s);

if (\*s == 0)

{

printf("Запись не была произведена\n");

return NULL;

}

strcpy(p->inf, s);

printf("Введите приоритет объекта: \n"); // вводим данные

scanf("%d", &(p->priority));

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

/\* Добавление в список элемента\*/

void spstore(void){

struct node\* p = NULL;

struct node\* struc = head;

struct node\* prev = NULL;

p = get\_struct();

if (head == NULL && p != NULL) { // если списка нет, то устанавливаем голову списка

head = p;

last = p;

}

else if (head != NULL && p != NULL) { // список уже есть

do {

if (struc->priority > p->priority) { //приоритет выше существующего

if (prev==NULL) { //приоритет добавляется на первое место

p->next = head;

head = p;

}

else { //приоритет добавляется после головы

prev->next = p;

p->next = struc;

}

return;

}

prev = struc;

struc = struc->next;

} while (struc != NULL);

last = prev; //приоритет добавляется в конец списка

last->next = p;

last = p;

}

return;

}

/\* Просмотр содержимого списка. \*/

void review(void){

struct node\* struc = head;

if (head == NULL){

printf("Список пуст\n");

}

while (struc){

printf("Имя - %s, \n", struc->inf);

struc = struc->next;

}

return;

}

/\* Поиск элемента по содержимому. \*/

struct node\* find(char\* name){

int i = 1;

struct node\* struc = head;

if (head == NULL){

printf("Список пуст\n");

}

while (struc){

if (strcmp(name, struc->inf) == 0){

printf("Элемент с приоритетом №%d \n", struc->priority);

return struc;

}

i++;

struc = struc->next;

}

printf("Элемент не найден\n");

return NULL;

}

/\* Удаление элемента по содержимому. \*/

void del(char\* name){

struct node\* struc = head; // указатель, проходящий по списку установлен на начало списка

struct node\* prev = struc; // указатель на предшествующий удаляемому элемент

int flag = 0; // индикатор отсутствия удаляемого элемента в списке

if (head == NULL) { // если голова списка равна NULL, то список пуст

printf("Список пуст\n");

return;

}

if (strcmp(name, struc->inf) == 0) { // если удаляемый элемент - первый

flag = 1;

head = struc->next; // установливаем голову на следующий элемент

free(struc); // удаляем первый элемент

struc = head; // устанавливаем указатель для продолжения поиска

printf("Удаление выполнено успешно\n");

}

else

{

struc = struc->next;

}

while (struc) // проход по списку и поиск удаляемого элемента

{

if (strcmp(name, struc->inf) == 0) // если нашли, то

{

flag = 1; // выставляем индикатор

if (struc->next) // если найденный элемент не последний в списке

{

prev->next = struc->next; // меняем указатели

free(struc); // удаляем элемент

struc = prev->next; // устанавливаем указатель для продолжения поиска

printf("Удаление выполнено успешно\n");

}

else // если найденный элемент последний в списке

{

prev->next = NULL; // обнуляем указатель предшествующего элемента

free(struc); // удаляем элемент

printf("Удаление выполнено успешно\n");

return;

}

}

else // если не нашли, то

{

prev = struc; // устанавливаем указатели для продолжения поиска

struc = struc->next;

}

}

if (flag == 0) // если флаг = 0, значит нужный элемент не найден

{

printf("Элемент не найден\n");

return;

}

}

int answer;

char name[50];

int main() {

setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

printf("1) Добавление элемента\n");

printf("2) Просмотр всех элементов\n");

printf("3) Поиск элемента\n");

printf("4) Удаление элемента\n");

printf("0) Выход\n\n");

do {

printf("\nВведите номер выбранного варианта -> ");

scanf("%d", &answer);

switch (answer) {

case (1):{

spstore();

break;

}

case (2): {

review();

break;

}

case (3): {

printf("Введите искомый элемент -> ");

getchar();

fgets(name, 50, stdin);

crop(name);

if (\*name == 0) {

printf("Запись не была произведена\n");

break;

}

find(name);

break;

}

case (4): {

printf("Введите удаляемый элемент -> ");

getchar();

fgets(name, 50, stdin);

crop(name);

if (\*name == 0) {

printf("Запись не была произведена\n");

break;

}

del(name);

break;

}

}

} while (answer != 0);

return 0;

}

Задание2.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <string.h>

struct node {

char inf[256]; // полезная информация

struct node\* next, \* prev; // ссылка на соседние элементы

};

typedef node\* pnode; // тип "указатели на узел"

struct stack {

pnode head, tail;

};

void crop(char\* str) {

int str\_num = 0;

while (str[str\_num] != '\0')

{

if (str[str\_num] == char(10))

{

str[str\_num] = char(0);

};

str\_num++;

};

}

/\* Последовательное добавление в список элемента\*/

stack\* push(stack& S, char& name) {

pnode newnode;

newnode = new node;

strcpy(newnode->inf, &name);

if (S.head == NULL) {

newnode->next = NULL;

newnode->prev = NULL;

S.tail = newnode;

}

else {

newnode->next = S.head;

newnode->prev = NULL;

}

if (S.head) S.head->prev = newnode;

S.head = newnode;

return (&S);

}

/\* Просмотр содержимого списка. \*/

void review(stack& S) {

struct stack S\_tmp = S;

if (S.tail == NULL) {

printf("Список пуст\n");

return;

}

while (1) {

printf("Имя - %s, \n", S\_tmp.tail->inf);

if (S\_tmp.tail->prev == NULL) {

return;

}

else {

S\_tmp.tail = S\_tmp.tail->prev;

}

}

}

/\* Поиск элемента по содержимому. \*/

void find(stack& S, char& name) {

int i = 1;

struct stack S\_tmp = S;

if (S.tail == NULL) {

printf("Список пуст\n");

}

do {

if (strcmp(&name, S\_tmp.tail->inf) == 0) {

printf("Элемент №%d в списке\n", i);

return;

}

i++;

S\_tmp.tail = S\_tmp.tail->prev;

} while (S\_tmp.tail != NULL);

printf("Элемент не найден\n");

return;

}

/\* Удаление элемента по содержимому. \*/

stack\* pop(stack& S) {

if (S.tail == NULL) {

printf("Список пуст\n");

}

pnode EndNode = S.tail;

S.tail = EndNode->prev;

if (S.tail) S.tail->next = NULL;

else S.head = NULL;

delete EndNode;

printf("Операция выполнена успешно\n");

return(&S);

}

int answer;

char name[50];

int flag\_first = 1;

int main() {

setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

printf("1) Добавление элемента\n");

printf("2) Просмотр всех элементов\n");

printf("3) Поиск элемента\n");

printf("4) Удаление элемента\n");

printf("0) Выход\n");

stack\* S = (stack\*)malloc(sizeof(stack));

do {

if (flag\_first == 1) {

S->head = NULL;

S->tail = NULL;

flag\_first = 0;

}

printf("\nВведите номер выбранного варианта -> ");

scanf("%d", &answer);

switch (answer) {

case (1): {

printf("Введите название объекта: \n"); // вводим данные

getchar();

fgets(name, 50, stdin);

crop(name);

if (\*name == 0) {

printf("Запись не была произведена\n");

break;

}

S = push(\*S, \*name);

break;

}

case (2): {

review(\*S);

break;

}

case (3): {

printf("Введите искомый элемент -> ");

getchar();

fgets(name, 50, stdin);

crop(name);

if (\*name == 0) {

printf("Запись не была произведена\n");

break;

}

find(\*S, \*name);

break;

}

case (4): {

S = pop(\*S);

break;

}

}

} while (answer != 0);

return 0;

}

Задание 3.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <string.h>

struct node {

char inf[256]; // полезная информация

struct node\* next, \*prev; // ссылка на соседние элементы

};

typedef node\* pnode; // тип "указатели на узел"

struct stack {

pnode head, tail;

};

void crop(char\* str) {

int str\_num = 0;

while (str[str\_num] != '\0')

{

if (str[str\_num] == char(10))

{

str[str\_num] = char(0);

};

str\_num++;

};

}

/\* Последовательное добавление в список элемента\*/

stack\* push(stack &S, char &name) {

pnode newnode;

newnode = new node;

strcpy(newnode->inf, &name);

if (S.head == NULL) {

newnode->next = NULL;

newnode->prev = NULL;

S.tail = newnode;

}

else {

newnode->next = S.head;

newnode->prev = NULL;

}

if (S.head) S.head->prev = newnode;

S.head = newnode;

return (&S);

}

/\* Просмотр содержимого списка. \*/

void review(stack &S) {

struct stack S\_tmp = S;

if (S.tail == NULL) {

printf("Список пуст\n");

return;

}

while(1){

printf("Имя - %s, \n", S\_tmp.head->inf);

if (S\_tmp.head->next == NULL) {

return;

}

else {

S\_tmp.head = S\_tmp.head->next;

}

}

}

/\* Поиск элемента по содержимому. \*/

void find(stack &S, char &name) {

int i = 1;

struct stack S\_tmp = S;

if (S.tail == NULL) {

printf("Список пуст\n");

}

do {

if (strcmp(&name, S\_tmp.head->inf) == 0) {

printf("Элемент №%d в списке\n", i);

return;

}

i++;

S\_tmp.head = S\_tmp.head->next;

} while (S\_tmp.head != NULL);

printf("Элемент не найден\n");

return;

}

/\* Удаление элемента по содержимому. \*/

stack\* pop(stack &S) {

if (S.tail == NULL) {

printf("Список пуст\n");

}

pnode TopNode = S.head;

S.head = TopNode->next;

if (S.head) S.head->prev = NULL;

else S.tail = NULL;

delete TopNode;

printf("Операция выполнена успешно\n");

return(&S);

}

int answer;

char name[50];

int flag\_first = 1;

int main() {

setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

printf("1) Добавление элемента\n");

printf("2) Просмотр всех элементов\n");

printf("3) Поиск элемента\n");

printf("4) Удаление элемента\n");

printf("0) Выход\n");

stack\* S = (stack\*)malloc(sizeof(stack));

do {

if (flag\_first == 1){

S->head = NULL;

S->tail = NULL;

flag\_first = 0;

}

printf("\nВведите номер выбранного варианта -> ");

scanf("%d", &answer);

switch (answer) {

case (1): {

printf("Введите название объекта: \n"); // вводим данные

getchar();

fgets(name, 50, stdin);

crop(name);

if (\*name == 0) {

printf("Запись не была произведена\n");

break;

}

S = push(\*S, \*name);

break;

}

case (2): {

review(\*S);

break;

}

case (3): {

printf("Введите искомый элемент -> ");

getchar();

fgets(name, 50, stdin);

crop(name);

if (\*name == 0) {

printf("Запись не была произведена\n");

break;

}

find(\*S, \*name);

break;

}

case (4): {

S = pop(\*S);

break;

}

}

} while (answer != 0);

return 0;

}