Министерство науки и образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ОТЧЕТ

по лабораторной работе №3

по курсу «Логика и основы алгоритмизации в инженерных задачах»

Выполнил:

студент группы 21ВВ2

Аракчеев А.М.

Приняла:

Юрова О.В.

Пенза 2022

**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).

// laba3.cpp: определяет точку входа для консольного приложения.

//

#include <stdio.h> // ДОБАВЛЕНО

#include <stdlib.h> // ДОБАВЛЕНО

#include <string.h> // ДОБАВЛЕНО

#include <locale.h>

struct node

{

char inf[256]; // полезная информация

int priority; // приоритет

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

};

// Обращение к списку и его элементам осуществляется посредством указателей:

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

int dlinna = 0;

// Для списка реализованы функции создания, добавления, удаления элемента, просмотра списка, нахождения нужного элемента списка:

// 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

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

scanf("%s", s);

if (\*s == 0)

{

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

return NULL;

}

strcpy(p->inf, s);

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

scanf("%s", s);

if (\*s == 0)

{

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

return NULL;

}

p->priority = atoi(s);

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

/\* Последовательное добавление в список элемента \*/

// ИЗМЕНЕНО

void spstore(void)

{

struct node \*p = 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, то ищем куда вставить

{

node\* current = head;

node\* prev = NULL;

while (current != NULL && current->priority >= p->priority) {

prev = current;

current = current->next; // переходим на соседний справа элемент

}

if (prev == NULL) { // новый элемент - с самым большим весом, вставляем в начало

// Если приоритет нового элемента совпадает с первым, то вставляем элемент между первым и следующим

if (p->priority == head->priority) {

p->next = head->next;

head->next = p;

} else { // Значит приоритет нового - самый высокий, вставляем в голову

p->next = head;

head = p;

}

} else if (current == NULL) { // новый элемент - с самым маленьким весом, вставляем в конец

prev->next = p; // prev = last, last->next = p

last = p;

} else { // вставляем элемент между prev и current

p->next = prev->next;

prev->next = p;

}

}

return;

}

/\* Просмотр содержимого списка. \*/

void review(void)

{

struct node \*struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

printf("Имя - %s, вес = %i \n", struc->inf, struc->priority);

struc = struc->next;

}

return;

}

/\* Получение элемента очереди из головы \*/

// ДОБАВЛЕНО

node\* dequeue() {

node\* struc = head;

if (head != NULL) {

struc = head;

head = head->next;

}

return struc;

}

/\* Поиск элемента по содержимому. \*/

void find()

{

char name[256];

printf("Введите имя для поиска: ");

getchar();

gets(name);

struct node \*struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

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

{

printf("Искомый элемент: Имя - %s, вес = %i \n", struc->inf, struc->priority);

return;

}

struc = struc->next;

}

printf("Элемент не найден\n");

}

/\* Удаление элемента по содержимому. \*/

void del(char \*name)

{

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

struct node \*prev;// 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

}

else

{

prev = struc;

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

}

else // если найденный элемент последний в списке

{

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

free(struc); // удаляем элемент

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;

}

}

// ДОБАВЛЕНО - основная точка запуска программы

int main() {

setlocale(LC\_ALL, "Russian");

system("chcp 1251");

// Заполнение очереди

for (int cx=1; cx <= 3; cx++){

spstore();

}

find();

// Вывод очереди на экран

node\* struc;

while ((struc = dequeue()) != NULL) {

printf("Имя - %s, вес - %i \n", struc->inf, struc->priority);

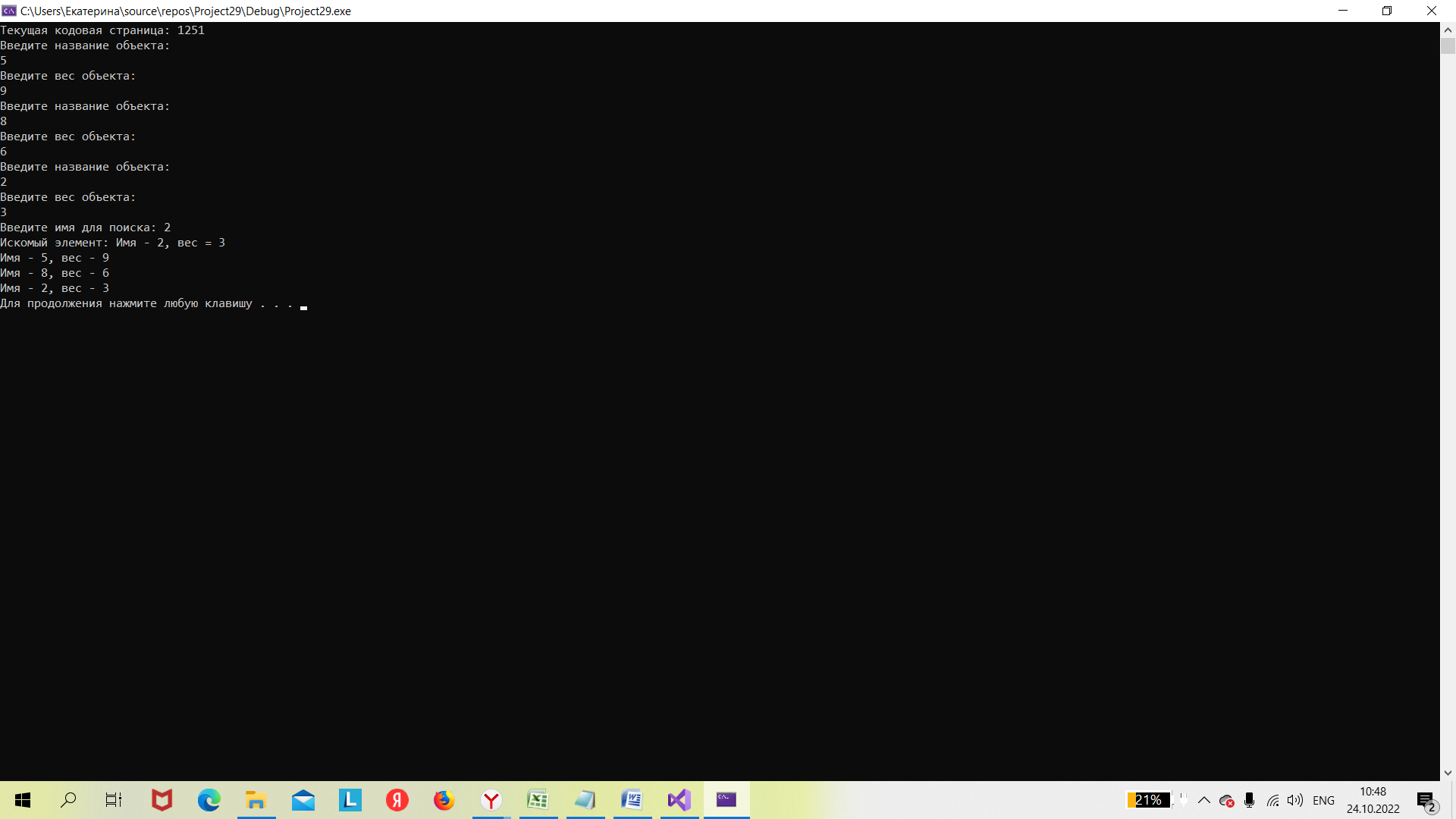
free(struc);

}

system("pause");

return 0;

}



**Задание 2**:

\* На основе приведенного кода реализуйте структуру данных **Очередь**.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h> // ДОБАВЛЕНО

#include <stdlib.h> // ДОБАВЛЕНО

#include <string.h> // ДОБАВЛЕНО

#include <locale.h>

struct node

{

char inf[256]; // полезная информация

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

};

// Обращение к списку и его элементам осуществляется посредством указателей:

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

int dlinna = 0;

// Для списка реализованы функции создания, добавления, удаления элемента, просмотра списка, нахождения нужного элемента списка:

// 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

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

scanf("%s", s);

if (\*s == 0)

{

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

return NULL;

}

strcpy(p->inf, s);

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

/\* Последовательное добавление в список элемента (в конец)\*/

void spstore(void)

{

struct node \*p = 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, то вставляем в конец

{

last->next = p;

last = p;

}

return;

}

/\* Получение текущего элемента \*/

// ДОБАВЛЕНО

node\* del() {

node\* struc = head;

// Так как добавляли в конец списка, то ранее вставленнное находится со стороны головы

if (head != NULL) {

struc = head;

head = head->next;

}

return struc;

}

/\* 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)

{

struct node \*struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

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

{

return struc;

}

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;// 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

}

else

{

prev = struc;

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

}

else // если найденный элемент последний в списке

{

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

free(struc); // удаляем элемент

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;

}

}

// ДОБАВЛЕНО - основная точка запуска программы

int main() {

setlocale(LC\_ALL, "Russian");

// Заполнение очереди

for (int cx=1; cx <= 3; cx++) {

spstore(); // запрашиваем ввод значения с клавиатуры

}

del();

// Вывод элементов очереди на экран

node\* struc;

while ((struc = del()) != NULL) {

printf("Имя - %s, \n", struc->inf);

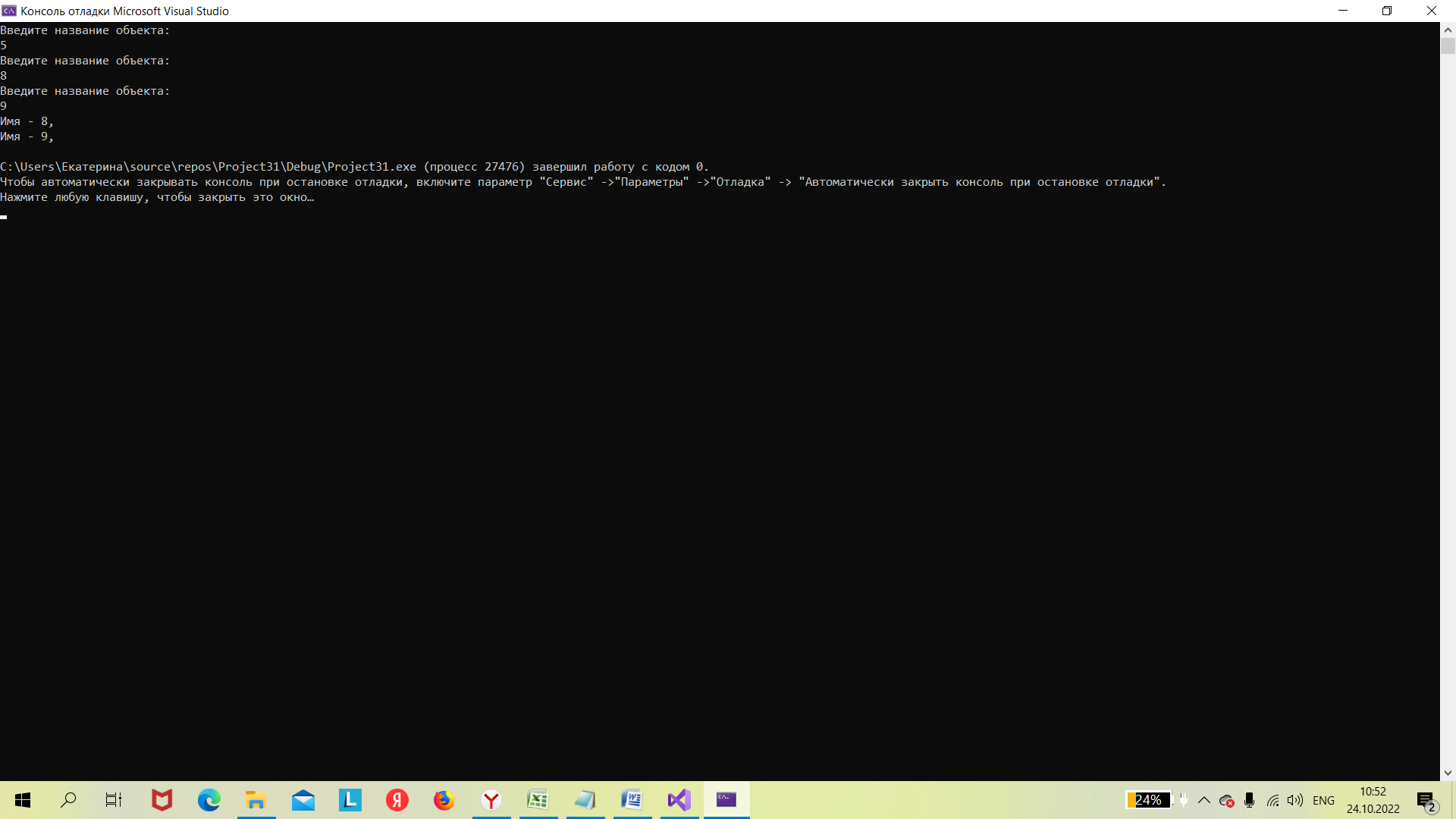
free(struc);

}

// review();

return 0;

}



**Задание 3**:

\* На основе приведенного кода реализуйте структуру данных **Стек**.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <locale.h>

#include <stdio.h> // ДОБАВЛЕНО

#include <stdlib.h> // ДОБАВЛЕНО

#include <string.h> // ДОБАВЛЕНО

struct node

{

char inf[256]; // полезная информация

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

};

// Обращение к списку и его элементам осуществляется посредством указателей:

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

int dlinna = 0;

// Для списка реализованы функции создания, добавления, удаления элемента, просмотра списка, нахождения нужного элемента списка:

// 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

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

scanf("%s", s);

if (\*s == 0)

{

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

return NULL;

}

strcpy(p->inf, s);

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

/\* Последовательное добавление в список элемента (в начало)\*/

// ИЗМЕНЕНО

void spstore(void)

{

struct node\* p = 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, то вставляем в начало

{

p->next = head;

head = p;

}

return;

}

/\* Удаляем элемент из головы списка (с вершины стека) \*/

// ДОБАВЛЕНО

node\* del() {

node\* struc = head;

if (struc != NULL) {

head = struc->next;

}

return struc;

}

/\* 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)

{

struct node\* struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

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

{

return struc;

}

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;// 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

}

else

{

prev = struc;

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

{

free(struc); // удаляем элемент

}

else // если найденный элемент последний в списке

{

free(struc); // удаляем элемент

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;

}

}

// ДОБАВЛЕНО - основная точка запуска программы

int main() {

setlocale(LC\_ALL, "Russian");

// Заполнение стека

for (int cx = 1; cx <= 3; cx++) {

spstore();

}

del();

// Вывод элементов стека на экран

node\* struc;

while ((struc = del()) != NULL) {

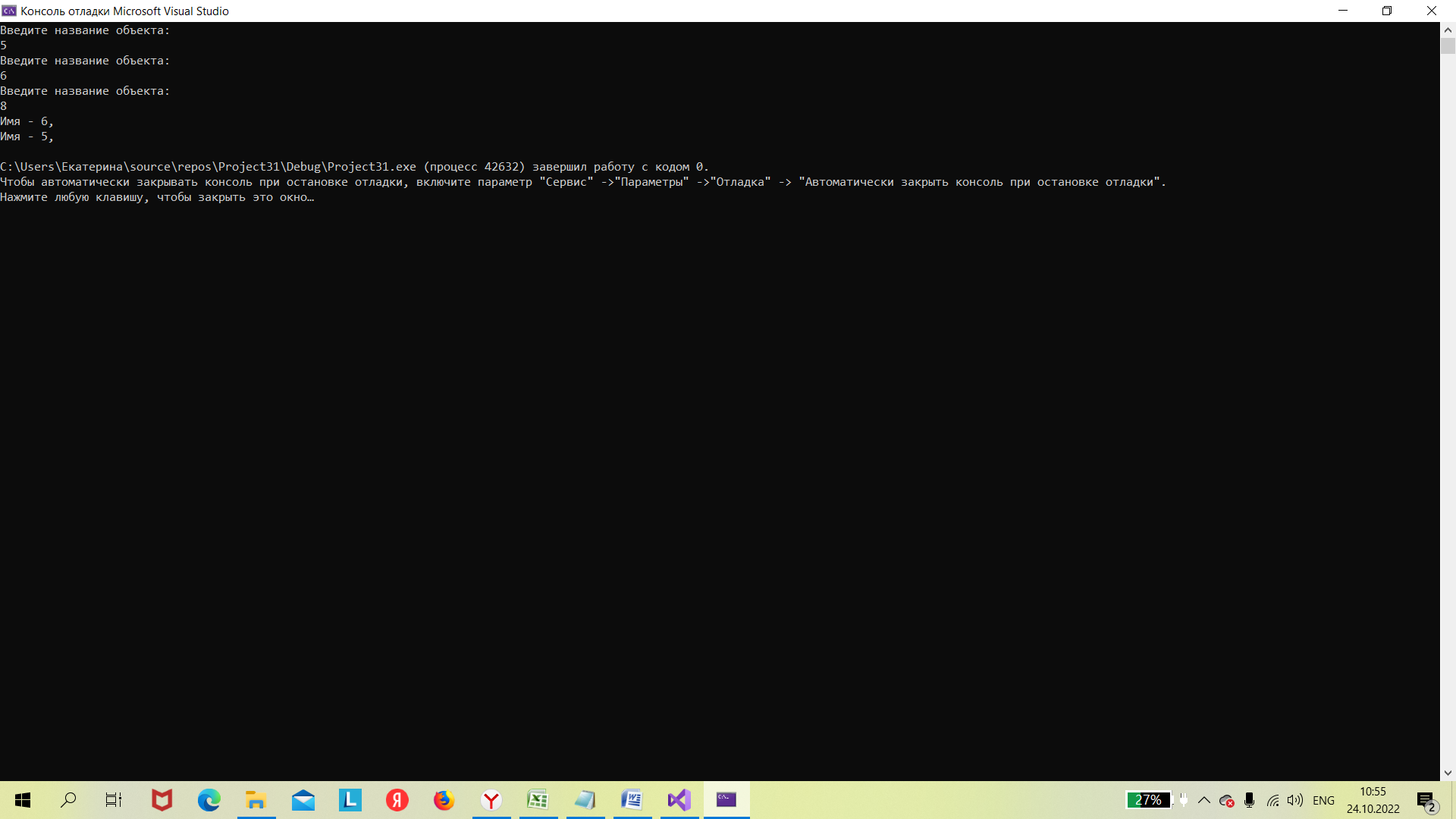
printf("Имя - %s, \n", struc->inf);

free(struc);

}

return 0;

}



**Вывод:** в ходе данной лабораторной работы я научился разрабатывать программу, реализующую очередь с приоритетом, очереди и стека.