МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
Кафедра «Вычислительной техники»

**ОТЧЁТ**

по лабораторной работе №3  
по курсу «Логика и основы алгоритмизации в инженерных задачах»  
на тему «Оценка времени выполнения программ»

Выполнил:

Студент группы 23ВВВ2

Чупраков С. В.

Приняли:

Митрохин М. А.  
Юрова О.В.

Пенза 2024

**Цель работы**

Разработка и реализация структур данных, таких как список с приоритетом элементов, очередь и стек, с использованием базовых принципов программирования.

**Задание**

**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).

**Задание 2**:

На основе приведенного кода реализуйте структуру данных *Очередь*.

**Задание 3**:

На основе приведенного кода реализуйте структуру данных *Стек*.

**Ход работы**

Задание 1.

С использованием заранее приведённых функций внесли изменения для создания рабочей программы. Использовали int prior для приоритетного расположения элементов в списке.

struct node

{

char inf[256]; // полезная информация

int prior;

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

};

Добавили сортировку по приоритету при добавлении элемента.

if((struc->prior < p->prior)&&(n == NULL)){

struc->next = p;

last = p;

break;

}

else if ((struc->prior > p->prior)) {

p->next = struc;

if (x!=NULL)

{

x->next = p;

}

if (struc == head) {

head = p;

}

break;

}

else if((struc->prior <= p->prior)){

x = struc;

struc = struc->next;

}

**Задание 2.**

Структура данных *Очередь* была реализована на базе выполненной программы, с изменением функции удаления и записи элементов.

Удаляется только первый элемент списка.

void del()

{

struct node\* struc = head;

struct node\* prev=NULL;

int flag = 0

if (head == NULL)

{

printf("Список пуст\n");

return;

}

prev = struc->next;

head = prev;

free(struc);

}

Добавляется элемент же только в конец.

**Задание 3.**

Добавление и удаление осуществляется только в начале списка.

struct node\* p = NULL;

struct node\* struc = head;

p = get\_struct();

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

{

head = p;

last = p;

}

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

{

p->next=head;

head = p;

}

printf("next? (+ or -)\n");

scanf(" %c", &specchar);

getchar();

if (specchar == '+') {

g = 1;

}

else {

g = 0;

}

### Результаты работы программы

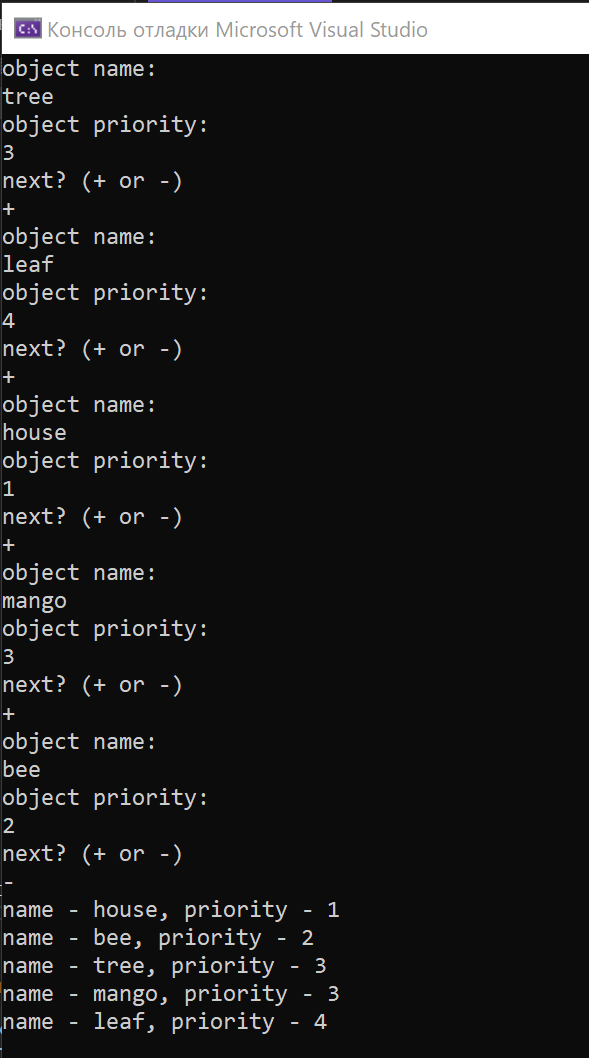


Рисунок 1 — Результаты работы программы пункта №1

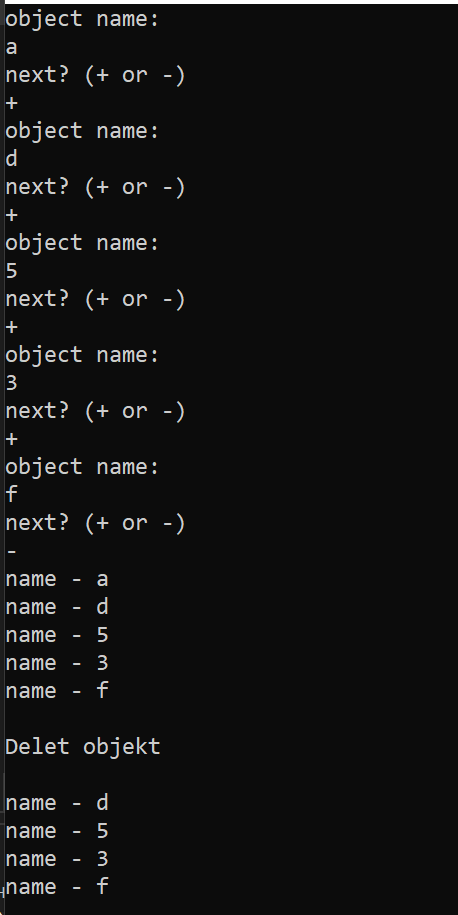


Рисунок 2 — Результаты работы программы пункта №2

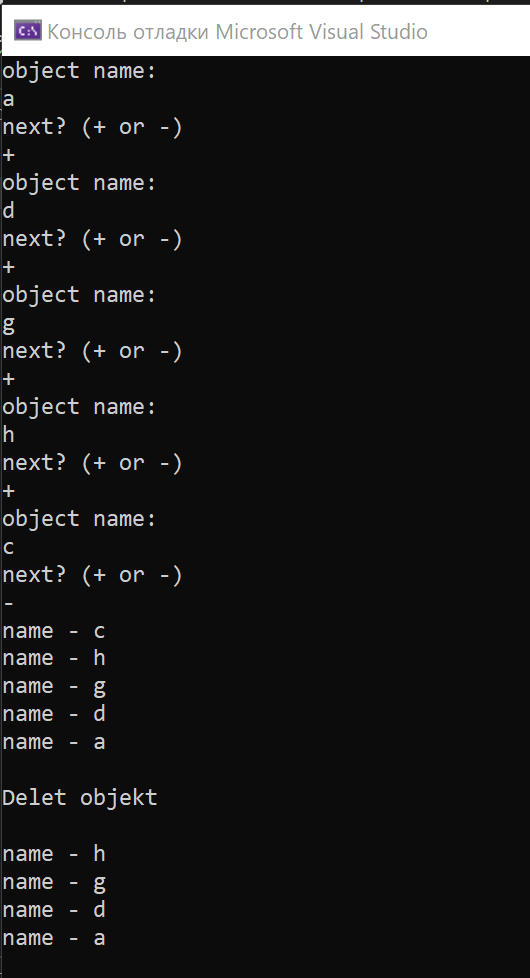


Рисунок 3 — Результаты работы программы пункта №3

### Вывод

В результате выполнения работы были успешно разработаны и реализованы три ключевые структуры данных: приоритетная очередь, очередь и стек. Каждая из этих структур имеет свои уникальные свойства и области применения, что делает их важными инструментами в арсенале программиста. Освоение этих структур данных помогает лучше понимать алгоритмы и оптимизировать решения для различных задач.

**Листинг**

**Задание 1**

**﻿#define \_CRT\_SECURE\_NO\_WARNINGS**

**#define \_CRT\_SECURE\_NO\_DEPRECATE**

**#define \_CRT\_NONSTDC\_NO\_DEPRECATE**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**struct node**

**{**

**char inf[256]; // полезная информация**

**int prior;**

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

**};**

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

**char specchar = '+';**

**int g = 0;**

**int 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];**

**int num=0;**

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

**{**

**printf("Memory allocation error\n");**

**exit(1);**

**}**

**printf("object name: \n"); // вводим данные**

**scanf("%s", s);**

**fflush(stdin);**

**if (\*s == 0)**

**{**

**printf("No recording was made\n");**

**return NULL;**

**}**

**strcpy(p->inf, s);**

**printf("object priority: \n"); // вводим данные**

**scanf("%d", &num);**

**fflush(stdin);**

**p->prior = num;**

**p->next = NULL;**

**return p; // возвращаем указатель на созданный элемент**

**}**

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

**int spstore()**

**{**

**struct node\* p = NULL;**

**struct node\* x = NULL;**

**struct node\* n = NULL;**

**struct node\* struc = head;**

**p = get\_struct();**

**fflush(stdin);**

**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**

**{**

**while (struc)**

**{**

**n = struc->next;**

**if((struc->prior < p->prior)&&(n == NULL)){**

**struc->next = p;**

**last = p;**

**break;**

**}**

**else if ((struc->prior > p->prior)) {**

**p->next = struc;**

**if (x!=NULL)**

**{**

**x->next = p;**

**}**

**if (struc == head) {**

**head = p;**

**}**

**break;**

**}**

**else if((struc->prior <= p->prior)){**

**x = struc;**

**struc = struc->next;**

**}**

**}**

**}**

**printf("next? (+ or -)\n");**

**scanf(" %c", &specchar);**

**getchar();**

**if (specchar == '+') {**

**g = 1;**

**}**

**else {**

**g = 0;**

**}**

**return (g);**

**}**

**/\* Просмотр содержимого списка. \*/**

**void review(void)**

**{**

**struct node\* struc = head;**

**if (head == NULL)**

**{**

**printf("The list is empty\n");**

**}**

**while (struc)**

**{**

**printf("name - %s, priority - %d \n", struc->inf, struc->prior);**

**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=NULL;// 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;**

**}**

**}**

**int main(){**

**spstore();**

**while (g==1) {**

**g=spstore();**

**}**

**review();**

**}**

**Задание 2**

**﻿#define \_CRT\_SECURE\_NO\_WARNINGS**

**#define \_CRT\_SECURE\_NO\_DEPRECATE**

**#define \_CRT\_NONSTDC\_NO\_DEPRECATE**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**struct node**

**{**

**char inf[256]; // полезная информация**

**/\*int prior;\*/**

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

**};**

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

**char specchar = '+';**

**int g = 0;**

**int 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];**

**int num=0;**

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

**{**

**printf("Memory allocation error\n");**

**exit(1);**

**}**

**printf("object name: \n"); // вводим данные**

**scanf("%s", s);**

**fflush(stdin);**

**if (\*s == 0)**

**{**

**printf("No recording was made\n");**

**return NULL;**

**}**

**strcpy(p->inf, s);**

**//printf("object priority: \n"); // вводим данные**

**//scanf("%d", &num);**

**//fflush(stdin);**

**//p->prior = num;**

**p->next = NULL;**

**return p; // возвращаем указатель на созданный элемент**

**}**

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

**int spstore()**

**{**

**struct node\* p = NULL;**

**struct node\* struc = head;**

**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;**

**}**

**printf("next? (+ or -)\n");**

**scanf(" %c", &specchar);**

**getchar();**

**if (specchar == '+') {**

**g = 1;**

**}**

**else {**

**g = 0;**

**}**

**return (g);**

**}**

**/\* Просмотр содержимого списка. \*/**

**void review(void)**

**{**

**struct node\* struc = head;**

**if (head == NULL)**

**{**

**printf("The list is empty\n");**

**}**

**while (struc)**

**{**

**printf("name - %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()**

**{**

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

**struct node\* prev=NULL;**

**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;**

**}**

**prev = struc->next;**

**head = prev;**

**free(struc);**

**}**

**int main(){**

**spstore();**

**while (g==1) {**

**g=spstore();**

**}**

**review();**

**del();**

**printf("\nDelet objekt\n\n");**

**review();**

**}**

**Задание 3**

**﻿#define \_CRT\_SECURE\_NO\_WARNINGS**

**#define \_CRT\_SECURE\_NO\_DEPRECATE**

**#define \_CRT\_NONSTDC\_NO\_DEPRECATE**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**struct node**

**{**

**char inf[256]; // полезная информация**

**/\*int prior;\*/**

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

**};**

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

**char specchar = '+';**

**int g = 0;**

**int 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];**

**int num=0;**

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

**{**

**printf("Memory allocation error\n");**

**exit(1);**

**}**

**printf("object name: \n"); // вводим данные**

**scanf("%s", s);**

**fflush(stdin);**

**if (\*s == 0)**

**{**

**printf("No recording was made\n");**

**return NULL;**

**}**

**strcpy(p->inf, s);**

**//printf("object priority: \n"); // вводим данные**

**//scanf("%d", &num);**

**//fflush(stdin);**

**//p->prior = num;**

**p->next = NULL;**

**return p; // возвращаем указатель на созданный элемент**

**}**

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

**int spstore()**

**{**

**struct node\* p = NULL;**

**struct node\* struc = head;**

**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**

**{**

**p->next=head;**

**head = p;**

**}**

**printf("next? (+ or -)\n");**

**scanf(" %c", &specchar);**

**getchar();**

**if (specchar == '+') {**

**g = 1;**

**}**

**else {**

**g = 0;**

**}**

**return (g);**

**}**

**/\* Просмотр содержимого списка. \*/**

**void review(void)**

**{**

**struct node\* struc = head;**

**if (head == NULL)**

**{**

**printf("The list is empty\n");**

**}**

**while (struc)**

**{**

**printf("name - %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()**

**{**

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

**struct node\* prev=NULL;**

**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;**

**}**

**prev = struc->next;**

**head = prev;**

**free(struc);**

**}**

**int main(){**

**spstore();**

**while (g==1) {**

**g=spstore();**

**}**

**review();**

**del();**

**printf("\nDelet objekt\n\n");**

**review();**

**}**