Министерство науки и высшего образования Российской Федерации

Пензенской государственный университет

Кафедра "Вычислительная техника"

**Отчёт**

по лабораторной работе №3

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Динамические списки»

Выполнили: ст. гр.24ВВВ1

Черенков Д.А.

Соколов О.С.

Принял:

к.т.н, доцент Юрова О. В.

к.т.н., доцент Деев М.В.

Пенза 2025

**Цель работы:** Изучение и практическая реализация динамических структур данных, включая освоение принципов работы с указателями, динамическим выделением памяти и организацией связей между элементами структур.

1. Реализовали приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объектом.

Код:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

#include <ctype.h>

#ifdef \_WIN32

#include <windows.h>

#endif

struct node

{

char inf[256];

int priority;

struct node\* next;

};

struct node\* head = NULL;

// Функция для очистки буфера ввода

void clear\_input\_buffer() {

int c;

while ((c = getchar()) != '\n' && c != EOF);

}

// Функция для безопасного ввода строки с пробелами

void safe\_input\_string(char\* buffer, int size) {

if (fgets(buffer, size, stdin) != NULL) {

// Удаляем символ новой строки

buffer[strcspn(buffer, "\n")] = 0;

}

}

// Функция для безопасного ввода числа

int safe\_input\_int() {

char input[100];

int number;

while (1) {

if (fgets(input, sizeof(input), stdin) != NULL) {

// Пытаемся преобразовать в число

if (sscanf(input, "%d", &number) == 1) {

return number;

}

printf("Ошибка! Введите целое число: ");

}

}

}

struct node\* get\_struct(void)

{

struct node\* p = NULL;

char s[256];

int prio;

if ((p = (struct node\*)malloc(sizeof(struct node))) == NULL)

{

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

exit(1);

}

printf("Введите название объекта: ");

safe\_input\_string(s, sizeof(s));

printf("Введите приоритет объекта: ");

prio = safe\_input\_int();

if (strlen(s) == 0)

{

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

free(p);

return NULL;

}

strcpy(p->inf, s);

p->priority = prio;

p->next = NULL;

return p;

}

void spstore(void)

{

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

if (p == NULL) return;

if (head == NULL || p->priority > head->priority)

{

p->next = head;

head = p;

return;

}

struct node\* current = head;

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

{

current = current->next;

}

p->next = current->next;

current->next = p;

}

void review(void)

{

struct node\* struc = head;

if (head == NULL)

{

printf("Список задач пуст\n");

return;

}

printf("\nЗадачи\n");

int count = 1;

while (struc)

{

printf("%d. Имя: %s, Приоритет: %d\n", count, struc->inf, struc->priority);

struc = struc->next;

count++;

}

}

void free\_list(void)

{

struct node\* current = head;

struct node\* next;

while (current != NULL)

{

next = current->next;

free(current);

current = next;

}

head = NULL;

}

void menu(void)

{

int choice;

while (1)

{

printf("\nПРИОРИТЕТНАЯ ОЧЕРЕДЬ\n");

printf("1. Добавить элемент\n");

printf("2. Просмотреть очередь\n");

printf("3. Выход\n");

printf("Выберите действие: ");

// Безопасный ввод выбора меню

char input[100];

if (fgets(input, sizeof(input), stdin) != NULL) {

// Проверяем, что введено одно число

if (sscanf(input, "%d", &choice) == 1) {

if (choice >= 1 && choice <= 3) {

switch (choice)

{

case 1:

spstore();

break;

case 2:

review();

break;

case 3:

free\_list();

printf("Вы вышли из программы\n");

return;

}

}

else {

printf("Ошибка! Введите число от 1 до 3.\n");

}

}

else {

printf("Ошибка! Введите число от 1 до 3.\n");

}

}

}

}

int main(void)

{

#ifdef \_WIN32

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

#endif

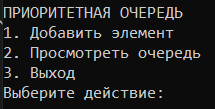
setlocale(LC\_ALL, "Russian");

menu();

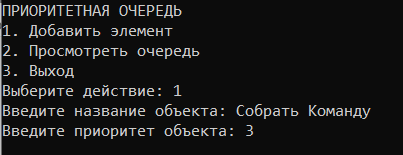
return 0;

}

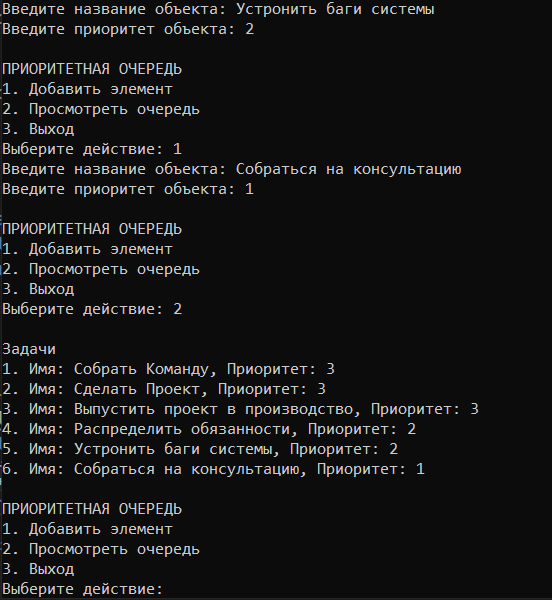
Меню программы.



Добавление элемента в приоритетную очередь.



Вывод объектов приоритеной очереди программы.



1. \* На основе приведенного кода реализовали структуру данных *Очередь*.

Код:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

#ifdef \_WIN32

#include <windows.h>

#endif

struct node

{

char inf[256];

struct node\* next;

};

struct node\* front = NULL; // начало очереди

struct node\* rear = NULL; // конец очереди

void clear\_input\_buffer() {

int c;

while ((c = getchar()) != '\n' && c != EOF);

}

void safe\_input\_string(char\* buffer, int size) {

if (fgets(buffer, size, stdin) != NULL) {

buffer[strcspn(buffer, "\n")] = 0;

}

}

int safe\_input\_int() {

char input[100];

int number;

while (1) {

if (fgets(input, sizeof(input), stdin) != NULL) {

if (sscanf(input, "%d", &number) == 1) {

return number;

}

printf("Ошибка! Введите целое число: ");

}

}

}

struct node\* get\_struct(void)

{

struct node\* p = NULL;

char s[256];

if ((p = (struct node\*)malloc(sizeof(struct node))) == NULL)

{

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

exit(1);

}

printf("Введите данные элемента: ");

safe\_input\_string(s, sizeof(s));

if (strlen(s) == 0)

{

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

free(p);

return NULL;

}

strcpy(p->inf, s);

p->next = NULL;

return p;

}

void enqueue(void)

{

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

if (p == NULL) return;

if (rear == NULL)

{

front = p;

rear = p;

}

else

{

rear->next = p;

rear = p;

}

printf("Элемент '%s' добавлен в очередь!\n", p->inf);

}

void dequeue(void)

{

if (front == NULL)

{

printf("Очередь пуста!\n");

return;

}

struct node\* temp = front;

printf("Элемент '%s' удален из очереди!\n", front->inf);

front = front->next;

if (front == NULL)

{

rear = NULL;

}

free(temp);

}

void display\_queue(void)

{

struct node\* current = front;

if (front == NULL)

{

printf("Очередь пуста\n");

return;

}

printf("\nСодержимое очереди\n");

int count = 1;

while (current)

{

printf("%d. %s\n", count, current->inf);

current = current->next;

count++;

}

}

void free\_queue(void)

{

struct node\* current = front;

struct node\* next;

while (current != NULL)

{

next = current->next;

free(current);

current = next;

}

front = NULL;

rear = NULL;

}

void menu(void)

{

int choice;

while (1)

{

printf("\nСТРУКТУРА ДАННЫХ 'ОЧЕРЕДЬ'\n");

printf("1. Добавить элемент\n");

printf("2. Удалить элемент\n");

printf("3. Просмотреть очередь\n");

printf("4. Выход\n");

printf("Выберите действие: ");

char input[100];

if (fgets(input, sizeof(input), stdin) != NULL) {

if (sscanf(input, "%d", &choice) == 1) {

if (choice >= 1 && choice <= 4) {

switch (choice)

{

case 1:

enqueue();

break;

case 2:

dequeue();

break;

case 3:

display\_queue();

break;

case 4:

free\_queue();

printf("Выход из программы. Память очищена.\n");

return;

}

}

else {

printf("Ошибка! Введите число от 1 до 4.\n");

}

}

else {

printf("Ошибка! Введите число от 1 до 4.\n");

}

}

}

}

int main(void)

{

#ifdef \_WIN32

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

#endif

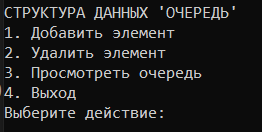
setlocale(LC\_ALL, "Russian");

menu();

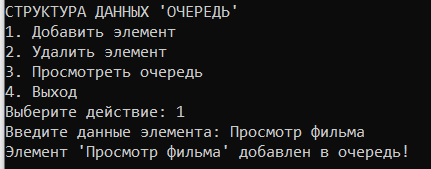
return 0;

}

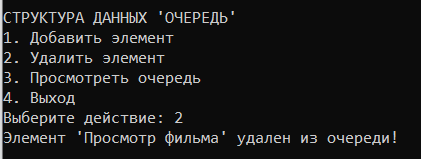
Меню программы.



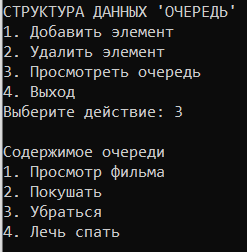
Добавление элемента в очередь.



Удаление элемента очереди



Вывод объектов приоритетной очереди программы.



1. \* На основе приведенного кода реализовали структуру данных *Стек*.

Код:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

#include <windows.h>

struct node

{

char inf[256];

struct node\* next;

};

struct node\* head = NULL;

void set\_console\_russian()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Russian");

}

void clear\_input\_buffer()

{

int c;

while ((c = getchar()) != '\n' && c != EOF);

}

struct node\* get\_struct(void)

{

struct node\* p = NULL;

char s[256];

if ((p = (struct node\*)malloc(sizeof(struct node))) == NULL)

{

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

exit(1);

}

printf("Введите название объекта: ");

if (fgets(s, sizeof(s), stdin) == NULL) {

printf("Ошибка ввода\n");

free(p);

return NULL;

}

size\_t len = strlen(s);

if (len > 0 && s[len - 1] == '\n') {

s[len - 1] = '\0';

}

if (strlen(s) == 0)

{

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

free(p);

return NULL;

}

strcpy(p->inf, s);

p->next = NULL;

return p;

}

void spstore(void)

{

struct node\* p = NULL;

p = get\_struct();

if (p != NULL)

{

p->next = head;

head = p;

printf("Элемент '%s' добавлен в стек\n", p->inf);

}

}

void review(void)

{

struct node\* struc = head;

if (head == NULL)

{

printf("Стек пуст\n");

return;

}

printf("Содержимое стека (от вершины к основанию):\n");

while (struc)

{

printf("Имя - %s\n", struc->inf);

struc = struc->next;

}

}

void del\_stack(void)

{

if (head == NULL)

{

printf("Стек пуст\n");

return;

}

struct node\* temp = head;

printf("Удален элемент: %s\n", temp->inf);

head = head->next;

free(temp);

}

void find\_element(void)

{

char search\_name[256];

if (head == NULL)

{

printf("Стек пуст\n");

return;

}

printf("Введите имя для поиска: ");

if (fgets(search\_name, sizeof(search\_name), stdin) == NULL) {

printf("Ошибка ввода\n");

return;

}

size\_t len = strlen(search\_name);

if (len > 0 && search\_name[len - 1] == '\n') {

search\_name[len - 1] = '\0';

}

struct node\* struc = head;

int found = 0;

while (struc)

{

if (strcmp(search\_name, struc->inf) == 0)

{

printf("Элемент найден: %s\n", struc->inf);

found = 1;

break;

}

struc = struc->next;

}

if (!found)

{

printf("Элемент не найден\n");

}

}

int main()

{

set\_console\_russian();

int choice;

do {

printf("\nМеню операций со стеком:\n");

printf("1 - Добавить элемент\n");

printf("2 - Просмотреть стек\n");

printf("3 - Удалить элемент из стека\n");

printf("4 - Поиск элемента\n");

printf("5 - Выход\n");

printf("Выберите действие: ");

if (scanf("%d", &choice) != 1) {

printf("Ошибка ввода!\n");

clear\_input\_buffer();

continue;

}

clear\_input\_buffer();

switch (choice) {

case 1:

spstore();

break;

case 2:

review();

break;

case 3:

del\_stack();

break;

case 4:

find\_element();

break;

case 5:

printf("Выход из программы\n");

break;

default:

printf("Неверный выбор! Попробуйте снова.\n");

}

} while (choice != 5);

// Освобождение памяти перед выходом

while (head != NULL)

{

struct node\* temp = head;

head = head->next;

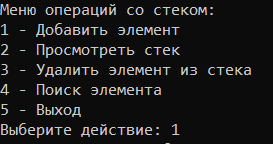
free(temp);

}

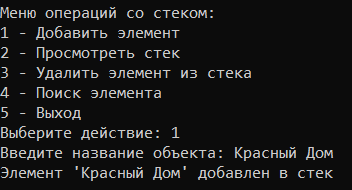
return 0;

}

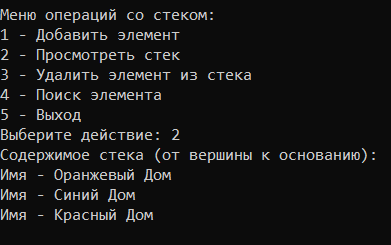
Меню программы.



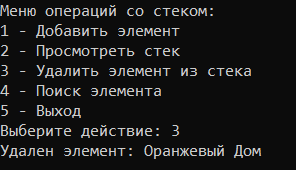
Добавление элемента в стек.



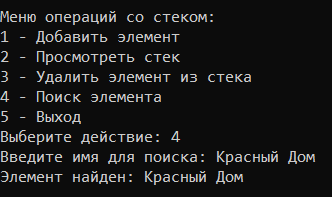
Вывод содержимого стека.



Удаление элемента из стека.



Поиск элемента в стеке.



**Вывод:** В ходе выполнения лабораторной работы были успешно реализованы и исследованы три основные динамические структуры данных: Стек, Очередь, Приоритетная очередь.