Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра "Вычислительная техника"

Отчёт

по лабораторной работе №3

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «динамические списки»

Вариант 1

Выполнили:

Студенты группы 24ВВВ1

Захаров А.В.

Гурин А.Н.

Приняли:

к.т.н., доцент Юрова О.В.

к.т.н., доцент Деев М.В.

Пенза 2025

**Лабораторное задание**

1. Реализовать приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта.

2. На основе приведенного кода реализуйте структуру данных Очередь.

3. На основе приведенного кода реализуйте структуру данных Стек.

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS #include <stdio.h> #include <stdlib.h> #include <string.h> #include <locale.h>

struct node { char data[256]; int priority; struct node\* next; };

//глобальные указатели на структуры struct node\* priority\_head = NULL; struct node\* queue\_front = NULL, \* queue\_rear = NULL; struct node\* stack\_top = NULL;

//создаем новый элемент (без запроса ввода) struct node\* create\_node\_silent(int with\_priority, const char\* name, int prio) { struct node\* p = (struct node\*)malloc(sizeof(struct node)); if (p == NULL) { printf("Ошибка памяти\n"); return NULL; }

strcpy(p->data, name);  
p->priority = with\_priority ? prio : 0;  
p->next = NULL;  
return p;

}

//очистка списка void clear\_list(struct node\*\* head) { struct node\* current = *head; while (current != NULL) { struct node* next = current->next; free(current); current = next; } \*head = NULL; }

//просмотр списка void print\_list(struct node\* head, const char\* title) { printf("\n%s:\n", title); if (head == NULL) { printf("Пусто\n"); return; }

struct node\* current = head;  
while (current != NULL) {  
 if (current->priority > 0) {  
 printf("- %s (приоритет: %d)\n", current->data, current->priority);  
 }  
 else {  
 printf("- %s\n", current->data);  
 }  
 current = current->next;  
}

}

//ПРИОРИТЕТНАЯ ОЧЕРЕДЬ

void priority\_add\_multiple() { char name[256]; int priority;

printf("\n--- Добавление элементов (для выхода введите '0' как название) ---\n");  
  
while (1) {  
 printf("Введите название (0 для выхода): ");  
 scanf("%s", name);  
  
 if (strcmp(name, "0") == 0) {  
 break;  
 }  
  
 printf("Введите приоритет: ");  
 scanf("%d", &priority);  
  
 struct node\* new\_node = create\_node\_silent(1, name, priority);  
 if (new\_node == NULL) continue;  
  
 //если список пуст или новый элемент имеет высший приоритет  
 if (priority\_head == NULL || new\_node->priority > priority\_head->priority) {  
 new\_node->next = priority\_head;  
 priority\_head = new\_node;  
 }  
 else {  
 //поиск места для вставки  
 struct node\* current = priority\_head;  
 while (current->next != NULL && current->next->priority >= new\_node->priority) {  
 current = current->next;  
 }  
 new\_node->next = current->next;  
 current->next = new\_node;  
 }  
 printf("Добавлен: %s (приоритет: %d)\n", name, priority);  
}

}

void priority\_remove() { if (priority\_head == NULL) { printf("Очередь пуста\n"); return; }

struct node\* temp = priority\_head;  
printf("Удален: %s\n", temp->data);  
priority\_head = priority\_head->next;  
free(temp);

}

void priority\_menu() { int choice; do { printf("\n--- ПРИОРИТЕТНАЯ ОЧЕРЕДЬ ---\n"); printf("1. Добавить элементы\n"); printf("2. Удалить элемент\n"); printf("3. Показать очередь\n"); printf("0. Назад\n"); printf("Выбор: "); scanf("%d", &choice);

switch (choice) {  
 case 1:  
 priority\_add\_multiple();  
 break;  
 case 2:  
 priority\_remove();  
 break;  
 case 3:  
 print\_list(priority\_head, "Приоритетная очередь");  
 break;  
 case 0:  
 printf("Возврат в главное меню\n");  
 break;  
 default:  
 printf("Неверный выбор\n");  
 break;  
 }  
} while (choice != 0);

}

//ОЧЕРЕДЬ (FIFO)

void queue\_add\_multiple() { char name[256];

printf("\n--- Добавление элементов (для выхода введите '0' как название) ---\n");

while (1) {  
 printf("Введите название (0 для выхода): ");  
 scanf("%s", name);  
  
 if (strcmp(name, "0") == 0) {  
 break;  
 }  
  
 struct node\* new\_node = create\_node\_silent(0, name, 0);  
 if (new\_node == NULL) continue;  
  
 if (queue\_rear == NULL) {  
 queue\_front = new\_node;  
 queue\_rear = new\_node;  
 }  
 else {  
 queue\_rear->next = new\_node;  
 queue\_rear = new\_node;  
 }  
 printf("Добавлен в очередь: %s\n", name);  
}

}

void queue\_remove() { if (queue\_front == NULL) { printf("Очередь пуста\n"); return; }

struct node\* temp = queue\_front;  
printf("Удален из очереди: %s\n", temp->data);  
queue\_front = queue\_front->next;  
  
if (queue\_front == NULL) {  
 queue\_rear = NULL;  
}  
  
free(temp);

}

void queue\_menu() { int choice; do { printf("\n--- ОЧЕРЕДЬ (FIFO) ---\n"); printf("1. Добавить элементы\n"); printf("2. Удалить элемент\n"); printf("3. Показать очередь\n"); printf("0. Назад\n"); printf("Выбор: "); scanf("%d", &choice);

switch (choice) {  
 case 1:  
 queue\_add\_multiple();  
 break;  
 case 2:  
 queue\_remove();  
 break;  
 case 3:  
 print\_list(queue\_front, "Очередь");  
 break;  
 case 0:  
 printf("Возврат в главное меню\n");  
 break;  
 default:  
 printf("Неверный выбор\n");  
 break;  
 }  
} while (choice != 0);

}

//СТЕК (LIFO)

void stack\_push\_multiple() { char name[256];

printf("\n--- Добавление элементов (для выхода введите '0' как название) ---\n");  
  
while (1) {  
 printf("Введите название (0 для выхода): ");  
 scanf("%s", name);  
  
 if (strcmp(name, "0") == 0) {  
 break;  
 }  
  
 struct node\* new\_node = create\_node\_silent(0, name, 0);  
 if (new\_node == NULL) continue;  
  
 new\_node->next = stack\_top;  
 stack\_top = new\_node;  
 printf("Добавлен в стек: %s\n", name);  
}

}

void stack\_pop() { if (stack\_top == NULL) { printf("Стек пуст\n"); return; }

struct node\* temp = stack\_top;  
printf("Удален из стека: %s\n", temp->data);  
stack\_top = stack\_top->next;  
free(temp);

}

void stack\_menu() { int choice; do { printf("\n--- СТЕК (LIFO) ---\n"); printf("1. Добавить элементы\n"); printf("2. Удалить элемент\n"); printf("3. Показать стек\n"); printf("0. Назад\n"); printf("Выбор: "); scanf("%d", &choice);

switch (choice) {  
 case 1:  
 stack\_push\_multiple();  
 break;  
 case 2:  
 stack\_pop();  
 break;  
 case 3:  
 print\_list(stack\_top, "Стек");  
 break;  
 case 0:  
 printf("Возврат в главное меню\n");  
 break;  
 default:  
 printf("Неверный выбор\n");  
 break;  
 }  
} while (choice != 0);

}

//ГЛАВНОЕ МЕНЮ

int main() { setlocale(LC\_ALL, "RU"); int choice;printf("=== СТРУКТУРЫ ДАННЫХ ===\n");  
do {  
 printf("\nГлавное меню:\n");  
 printf("1. Приоритетная очередь\n");  
 printf("2. Очередь (FIFO)\n");  
 printf("3. Стек (LIFO)\n");  
 printf("4. Очистить все\n");  
 printf("0. Выход\n");  
 printf("Выбор: ");  
 scanf("%d", &choice);  
  
 switch (choice) {  
 case 1:  
 priority\_menu();  
 break;  
 case 2:  
 queue\_menu();  
 break;  
 case 3:  
 stack\_menu();  
 break;  
 case 4:  
 clear\_list(&priority\_head);  
 clear\_list(&queue\_front);  
 queue\_rear = NULL;  
 clear\_list(&stack\_top);  
 printf("Все структуры очищены\n");  
 break;  
 case 0:  
 printf("Выход из программы\n");  
 break;  
 default:  
 printf("Неверный выбор\n");  
 break;  
 }  
} while (choice != 0);  
  
// Очистка памяти перед выходом  
clear\_list(&priority\_head);  
clear\_list(&queue\_front);  
clear\_list(&stack\_top);  
  
return 0;

}

**Результат работы программ**

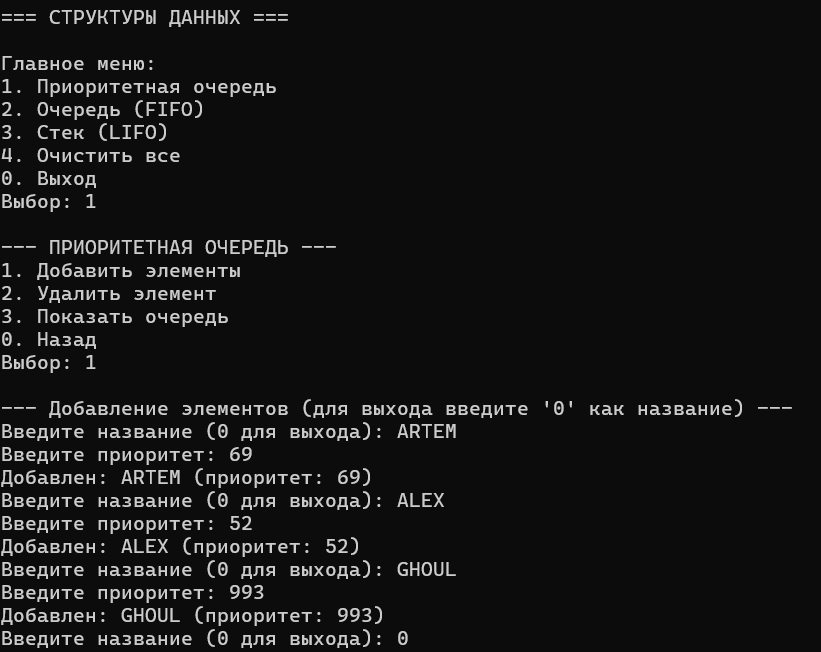
Результат работы программы представлен на рисунках 1 - 8.

Приоритетная очередь рисунки 1 и 2.

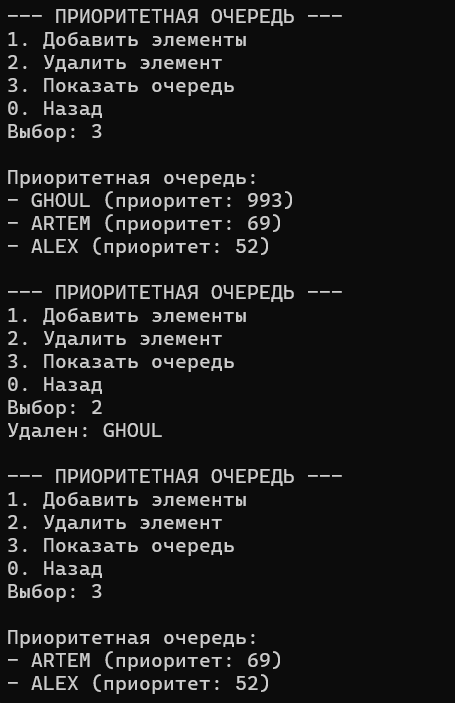
Очередь (FIFO) рисунки 3 и 4.

Очередь (LIFO) рисунки 5 и 6.

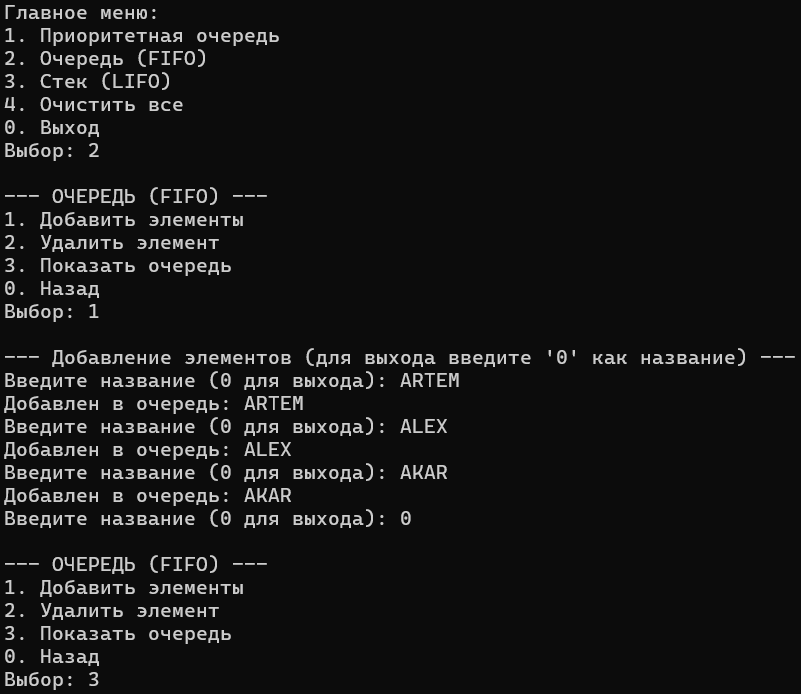
Протокол трассировки рисунки 7 и 8.



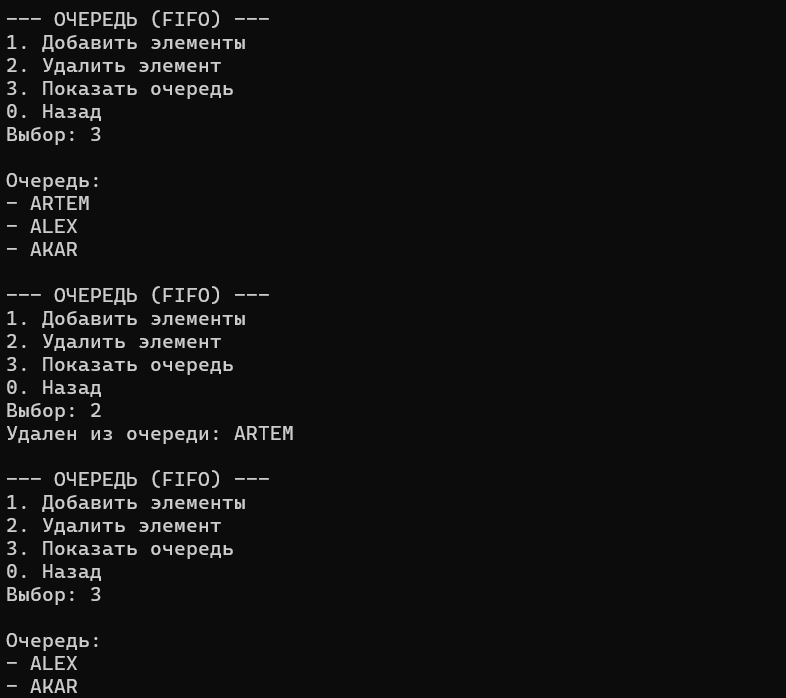
*Рисунок 1 - результат приоритетной очереди*



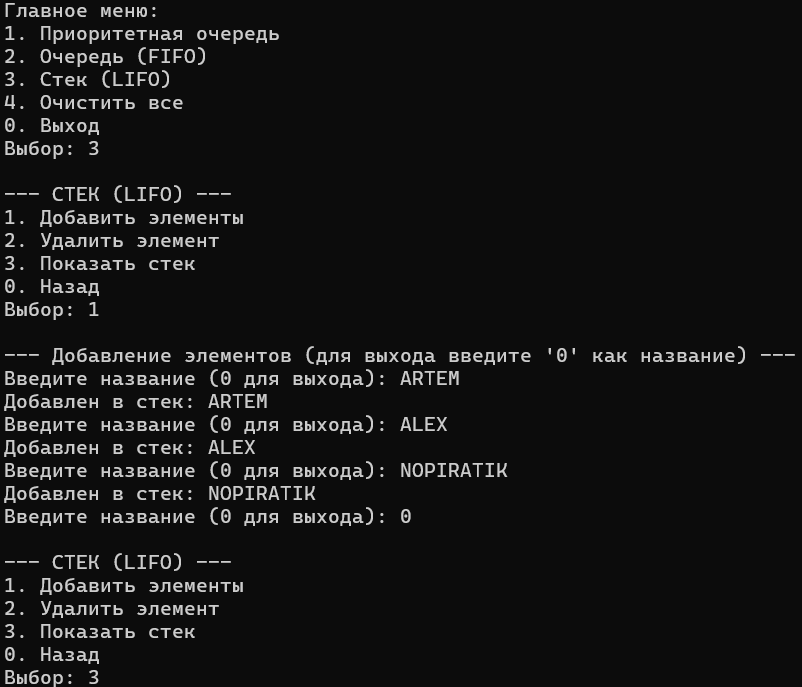
*Рисунок 2 - результат приоритетной очереди*



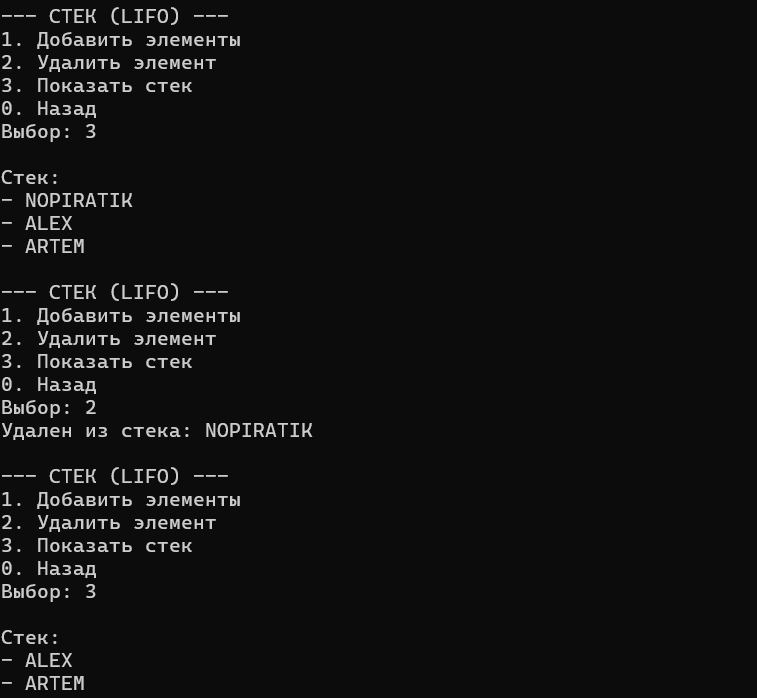
*Рисунок 3 – результат очереди (FIFO)*



*Рисунок 4 - результат очереди (FIFO)*

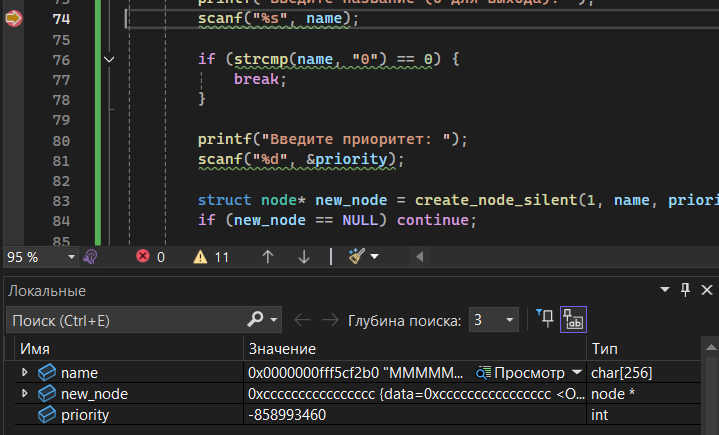


*Рисунок 5 - результат очереди (LIFO)*

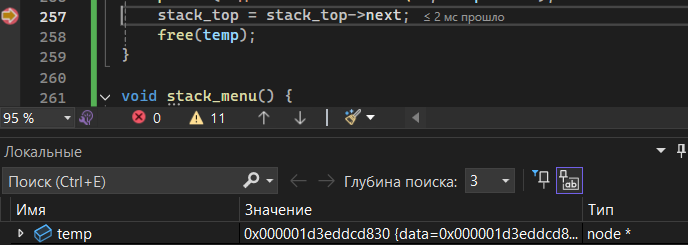


*Рисунок 6 - результат очереди (LIFO)*

**Протокол трассировки программы**



*Рисунок 7 – добавление элемента*



*Рисунок 8 – удаление элемента*

**Вывод**

В ходе выполнения лабораторной работы была разработана программа,

с реализованной приоритетной очередью, путём добавления элемента в список в соответствии с приоритетом объекта, структурой данных Очередь, структурой данных Стек.

Результаты работы программы совпали с результатами трассировки, программа работает без ошибок. Получили опыт в создании проектов в среде Microsoft Visual Studio, научились писать и отлаживать операторы условия и разветвляющихся алгоритмов на языке Си.

