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»

Выполнили:

студенты группы 24ВВВ3

Кукушкин Антон

Азаров Максим

Приняли:

Юрова О.В.

Деев М.В.

Пенза 2025

**Цель работы** – **Освоить принципы работы с динамическими структурами данных на примере односвязных списков, включая их создание, манипулирование элементами и организацию памяти.**

**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*.

**Задание 1**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct node

{

char inf[256];

int priority;

struct node\* next;

};

struct node\* head = NULL, \* last = NULL, \* f = NULL;

int dlinna = 0;

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

char find\_el[256];

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

struct node\* get\_struct(void);

struct node\* get\_struct(void)

{

struct node\* p = NULL;

char s[256];

int prior;

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

{

printf("Oshibka pri raspredelenii pamyati\n");

exit(1);

}

printf("Vvedite nazvanie ob'ekta: \n");

scanf("%s", s);

if (\*s == 0)

{

printf("Zapis' ne byla proizvedena\n");

free(p);

return NULL;

}

strcpy(p->inf, s);

printf("Vvedite prioritet ob'ekta (chem bol'she chislo, tem vyshe prioritet): \n");

scanf("%d", &prior);

p->priority = prior;

p->next = NULL;

return p;

}

void spstore(void)

{

struct node\* p = NULL;

struct node\* current = NULL;

struct node\* prev = NULL;

p = get\_struct();

if (p == NULL) return;

if (head == NULL)

{

head = p;

last = p;

dlinna++;

return;

}

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

{

p->next = head;

head = p;

dlinna++;

return;

}

current = head;

prev = NULL;

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

{

prev = current;

current = current->next;

}

p->next = current;

if (prev != NULL)

{

prev->next = p;

}

if (current == NULL)

{

last = p;

}

dlinna++;

}

void review(void)

{

struct node\* struc = head;

if (head == NULL)

{

printf("Ochered' pusta\n");

return;

}

printf("Soderzhimoe prioritetnoy ocheredi (po ubyvaniyu prioriteta):\n");

while (struc)

{

printf("Imya: %s, Prioritet: %d\n", struc->inf, struc->priority);

struc = struc->next;

}

printf("Vsego elementov: %d\n\n", dlinna);

}

struct node\* find(char\* name)

{

struct node\* struc = head;

if (head == NULL)

{

printf("Ochered' pusta\n");

return NULL;

}

while (struc)

{

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

{

printf("Element '%s' nayden s prioritetom %d\n", name, struc->priority);

return struc;

}

struc = struc->next;

}

printf("Element ne nayden\n");

return NULL;

}

void del(char\* name)

{

struct node\* struc = head;

struct node\* prev = NULL;

int flag = 0;

if (head == NULL)

{

printf("Ochered' pusta\n");

return;

}

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

{

flag = 1;

head = struc->next;

if (head == NULL)

{

last = NULL;

}

free(struc);

dlinna--;

printf("Element '%s' udalen iz ocheredi\n", name);

return;

}

prev = struc;

struc = struc->next;

while (struc)

{

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

{

flag = 1;

prev->next = struc->next;

if (struc == last)

{

last = prev;

}

free(struc);

dlinna--;

printf("Element '%s' udalen iz ocheredi\n", name);

return;

}

prev = struc;

struc = struc->next;

}

if (flag == 0)

{

printf("Element ne nayden\n");

}

}

struct node\* pop(void)

{

struct node\* temp;

if (head == NULL)

{

printf("Ochered' pusta\n");

return NULL;

}

temp = head;

head = head->next;

if (head == NULL)

{

last = NULL;

}

dlinna--;

return temp;

}

void peek(void)

{

if (head == NULL)

{

printf("Ochered' pusta\n");

return;

}

printf("Element s naivysshim prioritetom: %s (prioritet: %d)\n",

head->inf, head->priority);

}

int main()

{

int choice;

char name[256];

struct node\* extracted;

while (1)

{

printf("\n=== PRIORITETNAYA OCHERED' ===\n");

printf("1. Dobavit' element\n");

printf("2. Posmotret' ochered'\n");

printf("3. Nayti element\n");

printf("4. Udalit' element\n");

printf("5. Izvlech' element s naivysshim prioritetom\n");

printf("6. Posmotret' element s naivysshim prioritetom\n");

printf("0. Vykhod\n");

printf("Vyberite deystvie: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

spstore();

break;

case 2:

review();

break;

case 3:

printf("Vvedite nazvanie dlya poiska: ");

scanf("%s", name);

find(name);

break;

case 4:

printf("Vvedite nazvanie dlya udaleniya: ");

scanf("%s", name);

del(name);

break;

case 5:

extracted = pop();

if (extracted != NULL)

{

printf("Izvlechen element: %s (prioritet: %d)\n",

extracted->inf, extracted->priority);

free(extracted);

}

break;

case 6:

peek();

break;

case 0:

printf("Vykhod iz programmy\n");

return 0;

default:

printf("Nevernyy vybor\n");

}

}

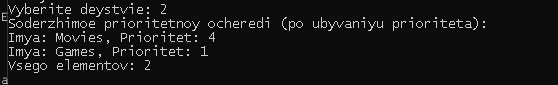
}

**Результат работы программы**

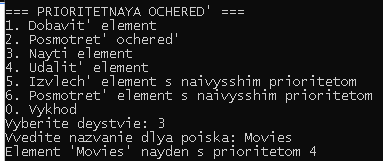
Результат работы программы показан на рисунках 1.1 – 1.5



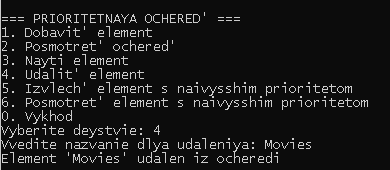
**Рисунок 1.1**



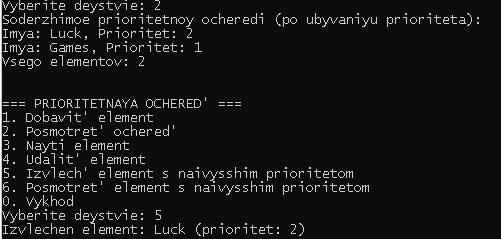
**Рисунок 1.2**

****

**Рисунок 1.3**

****

**Рисунок 1.4**

****

**Рисунок 1.5**

**Задание 2**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct node

{

char inf[256];

struct node\* next;

};

struct node\* front = NULL, \* rear = NULL, \* f = NULL;

int queue\_size = 0;

void enqueue(void), display\_queue(void), dequeue\_by\_name(char\* name);

int is\_empty(void);

void dequeue(void);

struct node\* peek(void);

char find\_el[256];

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

struct node\* get\_struct(void);

struct node\* get\_struct(void)

{

struct node\* p = NULL;

char s[256];

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

{

printf("Oshibka pri raspredelenii pamyati\n");

exit(1);

}

printf("Vvedite nazvanie ob'ekta: \n");

scanf("%s", s);

if (\*s == 0)

{

printf("Zapis' ne byla proizvedena\n");

free(p);

return NULL;

}

strcpy(p->inf, s);

p->next = NULL;

return p;

}

int is\_empty(void)

{

return (front == NULL);

}

void enqueue(void)

{

struct node\* p = NULL;

p = get\_struct();

if (p == NULL) return;

if (is\_empty())

{

front = p;

rear = p;

}

else

{

rear->next = p;

rear = p;

}

queue\_size++;

printf("Element '%s' dobavlen v ochered'\n", p->inf);

}

void dequeue(void)

{

struct node\* temp;

if (is\_empty())

{

printf("Ochered' pusta\n");

return;

}

temp = front;

printf("Element '%s' udalen iz ocheredi\n", temp->inf);

front = front->next;

if (front == NULL)

{

rear = NULL;

}

free(temp);

queue\_size--;

}

struct node\* peek(void)

{

if (is\_empty())

{

printf("Ochered' pusta\n");

return NULL;

}

printf("Pervyy element v ocheredi: %s\n", front->inf);

return front;

}

void display\_queue(void)

{

struct node\* struc = front;

if (is\_empty())

{

printf("Ochred' pusta\n");

return;

}

printf("Soderzhimoe ocheredi (ot nachala k kontsu):\n");

printf("Front -> ");

while (struc)

{

printf("%s", struc->inf);

if (struc->next != NULL)

printf(" -> ");

struc = struc->next;

}

printf(" <- Rear\n");

printf("Razmer ocheredi: %d\n\n", queue\_size);

}

struct node\* find(char\* name)

{

struct node\* struc = front;

int position = 1;

if (is\_empty())

{

printf("Ochered' pusta\n");

return NULL;

}

while (struc)

{

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

{

printf("Element '%s' nayden na pozitsii %d\n", name, position);

return struc;

}

struc = struc->next;

position++;

}

printf("Element ne nayden\n");

return NULL;

}

void dequeue\_by\_name(char\* name)

{

struct node\* struc = front;

struct node\* prev = NULL;

int flag = 0;

if (is\_empty())

{

printf("Ochered' pusta\n");

return;

}

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

{

flag = 1;

front = struc->next;

if (front == NULL)

{

rear = NULL;

}

free(struc);

queue\_size--;

printf("Element '%s' udalen iz ocheredi\n", name);

return;

}

prev = struc;

struc = struc->next;

while (struc)

{

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

{

flag = 1;

prev->next = struc->next;

if (struc == rear)

{

rear = prev;

}

free(struc);

queue\_size--;

printf("Element '%s' udalen iz ocheredi\n", name);

return;

}

prev = struc;

struc = struc->next;

}

if (flag == 0)

{

printf("Element ne nayden\n");

}

}

void clear\_queue(void)

{

while (!is\_empty())

{

dequeue();

}

printf("Ochered' polnost'yu ochishchena\n");

}

int main()

{

int choice;

char name[256];

while (1)

{

printf("\n=== OCHERED' ===\n");

printf("1. Dobavit' element v ochered' (enqueue)\n");

printf("2. Udalit' element iz ocheredi (dequeue)\n");

printf("3. Posmotret' pervyy element (peek)\n");

printf("4. Pokazat' ochered'\n");

printf("5. Nayti element\n");

printf("6. Udalit' element po nazvaniyu\n");

printf("7. Proverit' pustotu ocheredi\n");

printf("8. Ochitit' ochered'\n");

printf("0. Vykhod\n");

printf("Vyberite deystvie: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

enqueue();

break;

case 2:

dequeue();

break;

case 3:

peek();

break;

case 4:

display\_queue();

break;

case 5:

printf("Vvedite nazvanie dlya poiska: ");

scanf("%s", name);

find(name);

break;

case 6:

printf("Vvedite nazvanie dlya udaleniya: ");

scanf("%s", name);

dequeue\_by\_name(name);

break;

case 7:

if (is\_empty())

printf("Ochered' pusta\n");

else

printf("V ocheredi est' %d elementov\n", queue\_size);

break;

case 8:

clear\_queue();

break;

case 0:

clear\_queue();

printf("Vykhod iz programmy\n");

return 0;

default:

printf("Nevernyy vybor\n");

}

}

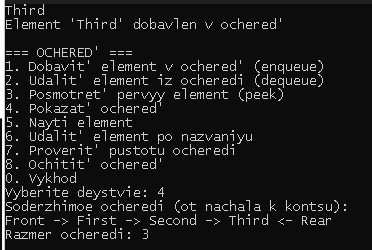
}

**Результат работы программы**

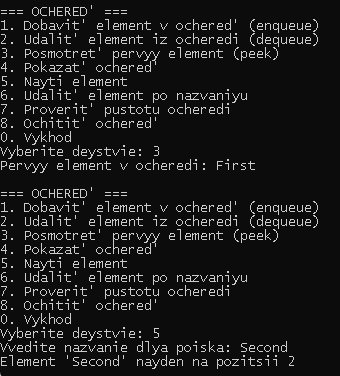
Результат работы программы показан на рисунках 2.1 – 2.4



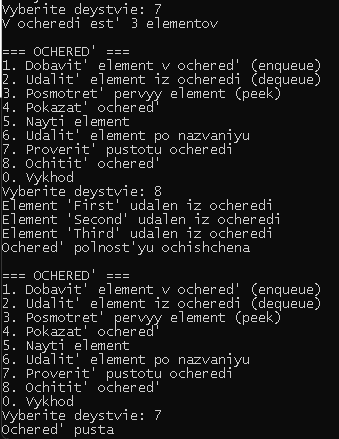
**Рисунок 2.1**

****

**Рисунок 2.2**

****

**Рисунок 2.3**

****

**Рисунок 2.4**

**Задание 3**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct node

{

char inf[256];

struct node\* next;

};

struct node\* top = NULL, \* f = NULL;

int stack\_size = 0;

void push(void), display\_stack(void), del\_by\_name(char\* name);

int is\_empty(void);

void pop(void);

struct node\* peek(void);

char find\_el[256];

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

struct node\* get\_struct(void);

struct node\* get\_struct(void)

{

struct node\* p = NULL;

char s[256];

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

{

printf("Oshibka pri raspredelenii pamyati\n");

exit(1);

}

printf("Vvedite nazvanie ob'ekta: \n");

scanf("%s", s);

if (\*s == 0)

{

printf("Zapis' ne byla proizvedena\n");

free(p);

return NULL;

}

strcpy(p->inf, s);

p->next = NULL;

return p;

}

int is\_empty(void)

{

return (top == NULL);

}

void push(void)

{

struct node\* p = NULL;

p = get\_struct();

if (p == NULL) return;

p->next = top;

top = p;

stack\_size++;

printf("Element '%s' dobavlen na vershinu steka\n", p->inf);

}

void pop(void)

{

struct node\* temp;

if (is\_empty())

{

printf("Stek pust (Stack Underflow)\n");

return;

}

temp = top;

printf("Element '%s' udalen s vershiny steka\n", temp->inf);

top = top->next;

free(temp);

stack\_size--;

}

struct node\* peek(void)

{

if (is\_empty())

{

printf("Stek pust\n");

return NULL;

}

printf("Verkhniy element steka: %s\n", top->inf);

return top;

}

void display\_stack(void)

{

struct node\* struc = top;

if (is\_empty())

{

printf("Stek pust\n");

return;

}

printf("Soderzhimoe steka (ot vershiny k osnovaniyu):\n");

printf("Top -> ");

while (struc)

{

printf("%s", struc->inf);

if (struc->next != NULL)

printf(" -> ");

struc = struc->next;

}

printf(" <- Bottom\n");

printf("Razmer steka: %d\n\n", stack\_size);

}

struct node\* find(char\* name)

{

struct node\* struc = top;

int position = 1;

if (is\_empty())

{

printf("Stek pust\n");

return NULL;

}

while (struc)

{

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

{

printf("Element '%s' nayden na pozitsii %d ot vershiny\n", name, position);

return struc;

}

struc = struc->next;

position++;

}

printf("Element ne nayden\n");

return NULL;

}

void del\_by\_name(char\* name)

{

struct node\* struc = top;

struct node\* prev = NULL;

int flag = 0;

if (is\_empty())

{

printf("Stek pust\n");

return;

}

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

{

flag = 1;

top = struc->next;

free(struc);

stack\_size--;

printf("Element '%s' udalen so steka\n", name);

return;

}

prev = struc;

struc = struc->next;

while (struc)

{

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

{

flag = 1;

prev->next = struc->next;

free(struc);

stack\_size--;

printf("Element '%s' udalen so steka\n", name);

return;

}

prev = struc;

struc = struc->next;

}

if (flag == 0)

{

printf("Element ne nayden\n");

}

}

void clear\_stack(void)

{

while (!is\_empty())

{

pop();

}

printf("Stek polnost'yu ochishchen\n");

}

int get\_size(void)

{

return stack\_size;

}

int main()

{

int choice;

char name[256];

while (1)

{

printf("\n=== STEK ===\n");

printf("1. Dobavit' element v stek (push)\n");

printf("2. Udalit' element so steka (pop)\n");

printf("3. Posmotret' verkhniy element (peek)\n");

printf("4. Pokazat' stek\n");

printf("5. Nayti element\n");

printf("6. Udalit' element po nazvaniyu\n");

printf("7. Proverit' pustotu steka\n");

printf("8. Poluchit' razmer steka\n");

printf("9. Ochitit' stek\n");

printf("0. Vykhod\n");

printf("Vyberite deystvie: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

push();

break;

case 2:

pop();

break;

case 3:

peek();

break;

case 4:

display\_stack();

break;

case 5:

printf("Vvedite nazvanie dlya poiska: ");

scanf("%s", name);

find(name);

break;

case 6:

printf("Vvedite nazvanie dlya udaleniya: ");

scanf("%s", name);

del\_by\_name(name);

break;

case 7:

if (is\_empty())

printf("Stek pust\n");

else

printf("V steke est' %d elementov\n", stack\_size);

break;

case 8:

printf("Razmer steka: %d\n", get\_size());

break;

case 9:

clear\_stack();

break;

case 0:

clear\_stack();

printf("Vykhod iz programmy\n");

return 0;

default:

printf("Nevernyy vybor\n");

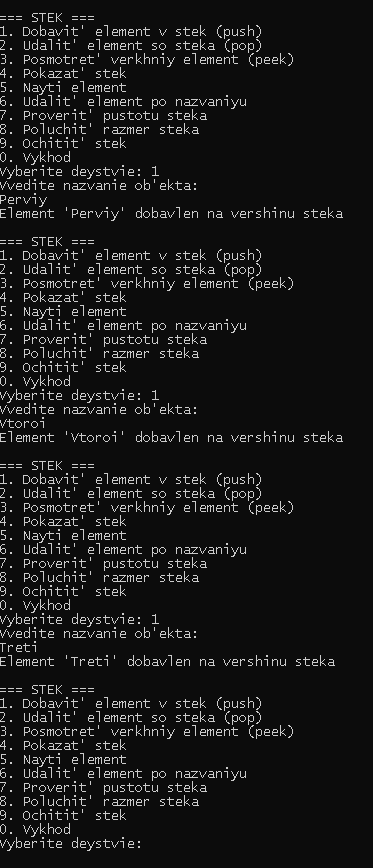
}

}

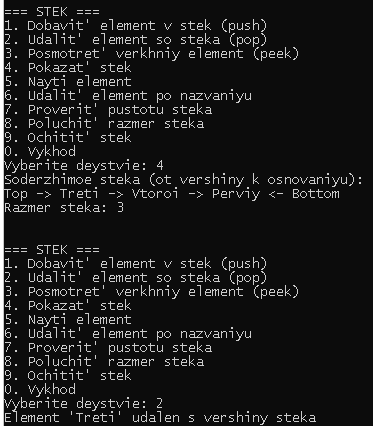
}

**Результат работы программы**

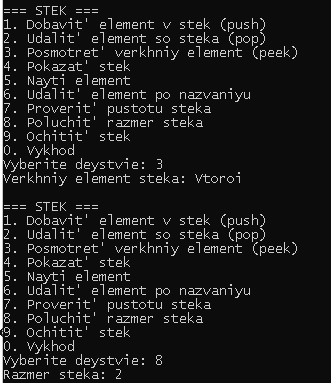
Результат работы программы показан на рисунках 3.1 – 3.3



**Рисунок 3.1**

****

**Рисунок 3.2**

****

**Рисунок 3.3**

**Вывод:** В ходе выполнения лабораторной работы были успешно освоены принципы работы с динамическими структурами данных на примере односвязных списков.