Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №3

по курсу «Логика и основы алгоритмизации в инженерных задачах»

## на тему «Динамические списки»

**Выполнили студенты группы 21вв1:**

Кирьянов В.Е.

Аляев А.О.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**Задания:**

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*.

**Листинг:**

**1 задание:**

#include "StdAfx.h"

#include <iostream>

using namespace std;

struct node

{

char inf[256]; // полезная информация

int priority;

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

};

int size=0;

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

int dlinna = 0;

void spstore(), 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

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{

struct node \*struc = head;

struct node \*prev = NULL;

while(struc)

{

if (struc != NULL && p != NULL) // список уже есть, то вставляем в конец

{

if(p->priority > struc->priority){

if(prev !=NULL)

prev->next = p;

if(head==struc)

head=p;

p->next = struc;

return;

}

else{

prev = struc;

struc=struc->next;

}

}

}

prev->next=p;

p->next=NULL;

}

return;

}

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 %d", s, &p->priority);

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)\*/

/\* 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[20];

scanf("%s",name);

struct node \*struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

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

{

printf("Элемент найден!\n");

return struc;

}

struc = struc->next;

}

printf("Элемент не найден\n");

return NULL;

}

/\* Удаление элемента по содержимому. \*/

void del()

{

char name[20];

scanf("%s",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; // устанавливаем 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;

}

printf("Элемент удалён\n");

}

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;

}

return;

}

void remove(){

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

struct node \*prev;// указатель на предшествующий удаляемому элемент

if (head->next != NULL) // если найденный элемент последний в списке

{

head=head->next; // обнуляем указатель предшествующего элемента

free(struc); // удаляем элемент

return;

}

else

{

head = NULL;

free(struc);

}

return;

}

int main() {

setlocale(LC\_ALL, "Russian");

int k = 1;

node \*Begin, \*End;

node \*el = new node;

el->next = NULL;

Begin = End = el;

while (k) {

cout << "1 - добавление элемента в конец очереди\n2 - удаление элемента по содержимому\n3 - поиск элемента по содержимому\n4 - функция вывода элементов очереди на экран\n5 - удаление последнего элемента\n" << "0 - выход" << endl;

cin >> k;

if (k == 5){

remove();

}

if (k == 0)

return 0;

if (k == 1)

spstore();

if (k == 2){

cout << "Введите элемент поиска: ";

del();

}

if (k == 3){

cout << "Введите элемент поиска: ";

\*find();

}

if (k == 4)

review();

}

cout << endl;

system("pause");

return 0;

}

**Задание 2:**

#include "StdAfx.h"

#include <iostream>

using namespace std;

struct node

{

char inf[256]; // полезная информация

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

};

int size=0;

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

int dlinna = 0;

void spstore(), 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

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;

}

size++;

return;

}

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)\*/

/\* 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[20];

scanf("%s",name);

struct node \*struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

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

{

printf("Элемент найден!\n");

return struc;

}

struc = struc->next;

}

printf("Элемент не найден\n");

return NULL;

}

/\* Удаление элемента по содержимому. \*/

void del()

{

char name[20];

scanf("%s",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; // устанавливаем 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;

}

printf("Элемент удалён\n");

}

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;

}

return;

}

void remove(){

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

struct node \*prev;// указатель на предшествующий удаляемому элемент

if (head->next != NULL) // если найденный элемент последний в списке

{

head=head->next; // обнуляем указатель предшествующего элемента

free(struc); // удаляем элемент

return;

}

else

{

head = NULL;

free(struc);

}

return;

}

int main() {

setlocale(LC\_ALL, "Russian");

int k = 1;

node \*Begin, \*End;

node \*el = new node;

el->next = NULL;

Begin = End = el;

while (k) {

cout << "1 - добавление элемента в конец очереди\n2 - удаление элемента по содержимому\n3 - поиск элемента по содержимому\n4 - функция вывода элементов очереди на экран\n5 - удаление последнего элемента\n" << "0 - выход" << endl;

cin >> k;

if (k == 5){

remove();

}

if (k == 0)

return 0;

if (k == 1)

spstore();

if (k == 2){

cout << "Введите элемент поиска: ";

del();

}

if (k == 3){

cout << "Введите элемент поиска: ";

\*find();

}

if (k == 4)

review();

}

cout << endl;

system("pause");

return 0;

}

**Задание 3:**

#include "StdAfx.h"

#include <iostream>

using namespace std;

struct node

{

char inf[256]; // полезная информация

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

};

int size=0;

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

int dlinna = 0;

void spstore(), 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

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

{

p->next = head;

head = p;

}

size++;

return;

}

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)\*/

/\* 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[20];

scanf("%s",name);

struct node \*struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

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

{

printf("Элемент найден!\n");

return struc;

}

struc = struc->next;

}

printf("Элемент не найден\n");

return NULL;

}

/\* Удаление элемента по содержимому. \*/

void del()

{

char name[20];

scanf("%s",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; // устанавливаем 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;

}

printf("Элемент удалён\n");

}

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;

}

return;

}

void remove(){

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

struct node \*prev;// указатель на предшествующий удаляемому элемент

if (head->next != NULL) // если найденный элемент последний в списке

{

head=head->next; // обнуляем указатель предшествующего элемента

free(struc); // удаляем элемент

return;

}

else

{

head = NULL;

free(struc);

}

return;

}

int main() {

setlocale(LC\_ALL, "Russian");

int k = 1;

node \*Begin, \*End;

node \*el = new node;

el->next = NULL;

Begin = End = el;

while (k) {

cout << "1 - добавление элемента в конец очереди\n2 - удаление элемента по содержимому\n3 - поиск элемента по содержимому\n4 - функция вывода элементов очереди на экран\n5 - удаление последнего элемента\n" << "0 - выход" << endl;

cin >> k;

if (k == 5){

remove();

}

if (k == 0)

return 0;

if (k == 1)

spstore();

if (k == 2){

cout << "Введите элемент поиска: ";

del();

}

if (k == 3){

cout << "Введите элемент поиска: ";

\*find();

}

if (k == 4)

review();

}

cout << endl;

system("pause");

return 0;

}

**Результаты работы программ:**

**Задание 1:**

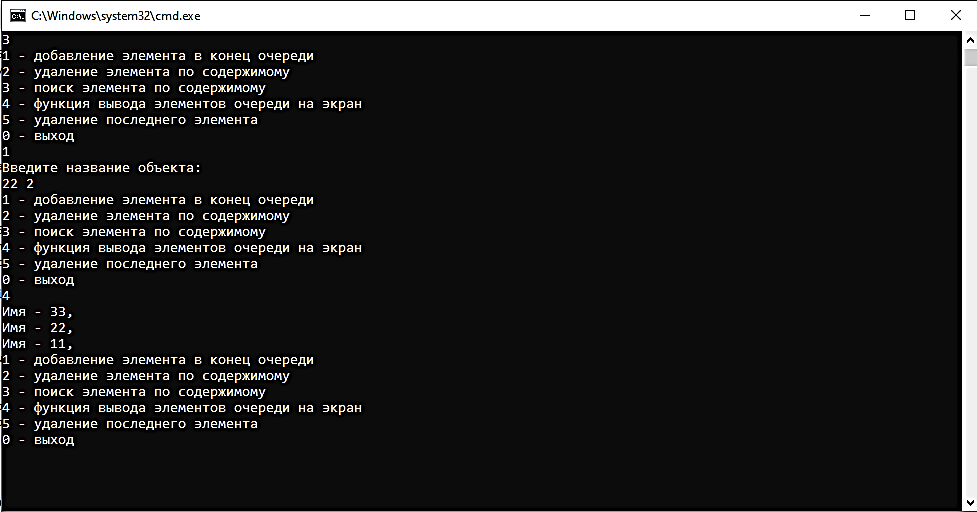
****

Рисунок 1 – результат работы программы.

Добавление элементов 11, с приоритетом 1; 22 с приоритетом 2; 33 с приоритетом 3.

**Задание 2:**

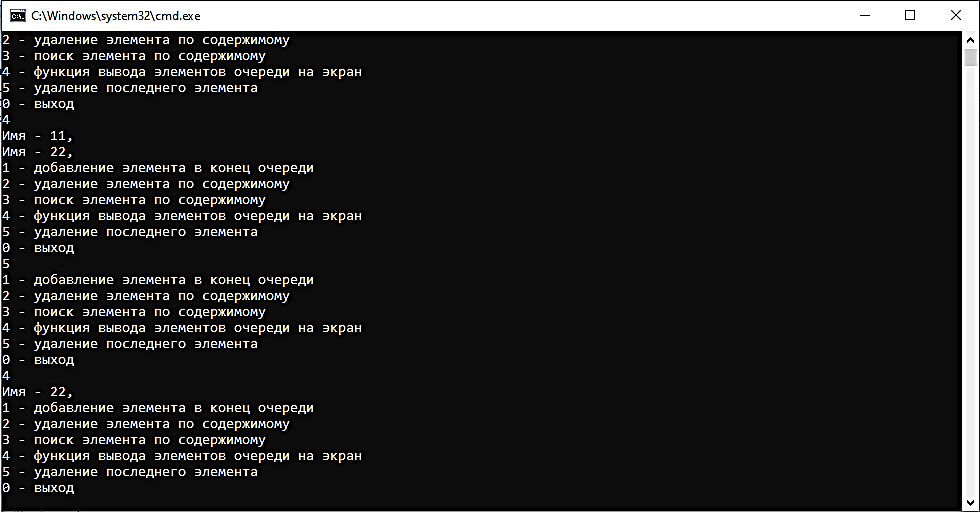


Рисунок 2 – результат работы программы.

Добавление элементов 11; 22, удаление последнего элемента очереди – 11.

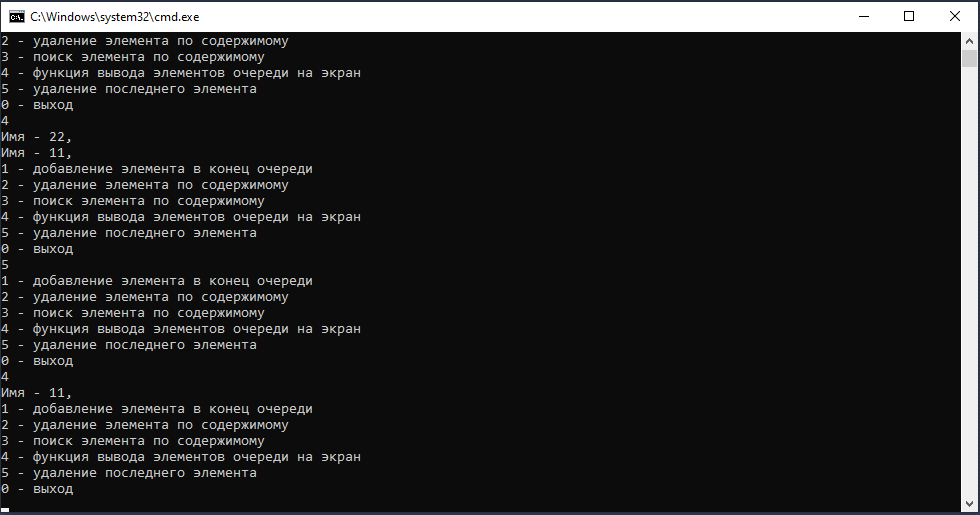
**Задание 3:**

Рисунок 2 – результат работы программы.

Добавление элементов 11; 22, удаление последнего элемента стека – 22.

**Вывод:** в ходе выполнения лабораторной работы были разработаны программы, выполняющие работу с очередью и стеком.

Получили опыт в создании проектов в среде Microsoft Visual Studio, научились писать и отлаживать программы с применение очередей и стеков на языке Си.