Федеральное государственное бюджетное образовательное учреждение высшего образования «Пензенский Государственный университет»

Кафедра «Вычислительная техника»

ОТЧЕТ

**По лабораторной работе №3**

по курсу «Логика и основы алгоритмизации в ИЗ»

на тему «Динамические списки»

Выполнили: ст. гр. 21ВВ1

Сурков А.А

Хорошильцев Е.А

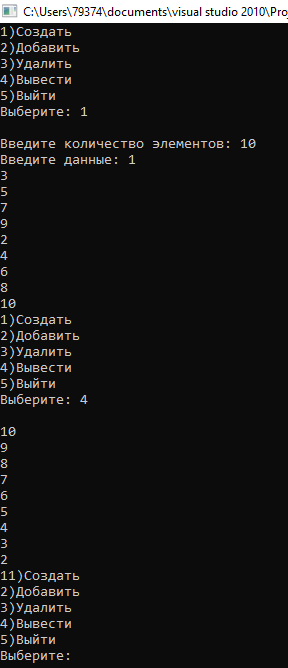
Кривенков И.В

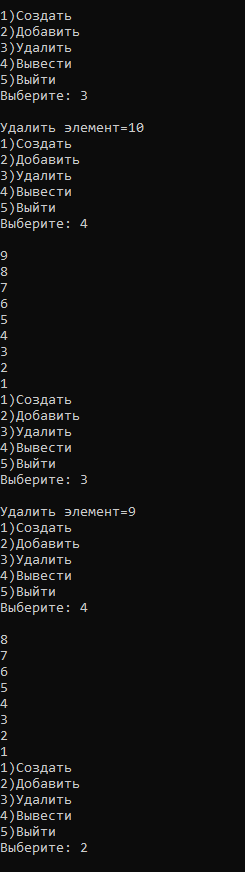
Приняли: Юрова О.В.

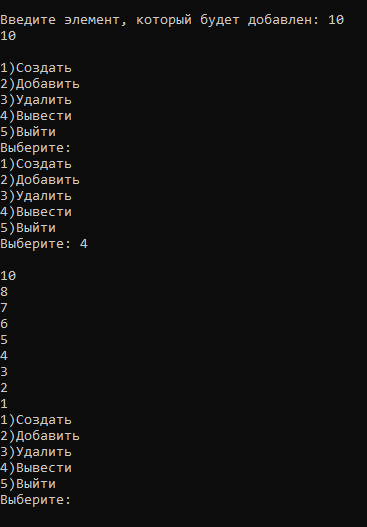
Акифьев И.В.

2022

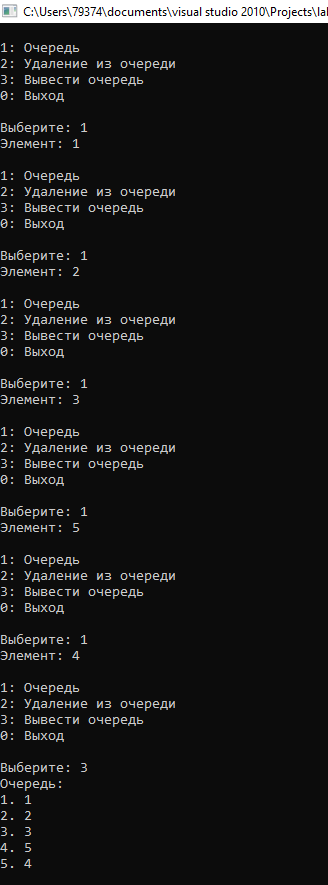
**Задание 1**: Реализовать приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта.

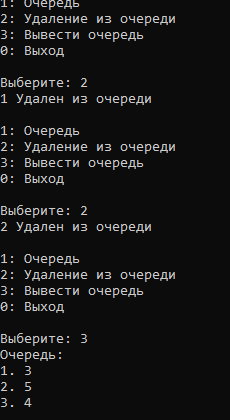






**Задание 2:** Реализовать структуру данных “Очередь”.





**Вывод:** в ходе выполнения данной лабораторной работы мы вспомнили работу со списками, а также научились реализовывать очередь и приоритетную очередь.

**Листинг:**

**Задание 1:**

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#define MAX 30

typedef struct pqueue

{

int data[MAX];

int rear,front;

}pqueue;

void initialize(pqueue \*p);

int empty(pqueue \*p);

int full(pqueue \*p);

void enqueue(pqueue \*p, int x);

int dequeue(pqueue \*p);

void print(pqueue \*p);

void main()

{

setlocale(LC\_ALL, "Rus");

int x,op,n,i;

pqueue q;

initialize(&q);

do

{

printf("\n1)Создать \n2)Добавить \n3)Удалить \n4)Вывести \n5)Выйти");

printf("\nВыберите: ");

scanf("%d",&op);

switch (op) {

case 1: printf("\nВведите количество элементов: ");

scanf("%d",&n );

initialize(&q);

printf("Введите данные: ");

for(i=0; i<n; i++)

{

scanf("%d",&x);

if(full(&q))

{

printf("\nОчередь полная..");

exit(0);

}

enqueue(&q,x);

}

break;

case 2: printf("\nВведите элемент, который будет добавлен: ");

scanf("%d\n",&x);

if(full(&q))

{

printf("\nОчередь полная");

exit(0);

}

enqueue(&q,x);

break;

case 3: if(empty(&q))

{

printf("\nОчередь пуста..");

exit(0);

}

x=dequeue(&q);

printf("\nУдалить элемент=%d",x);

break;

case 4: print(&q);

break;

default: break;

}

}while (op!=5);

}

void initialize(pqueue \*p)

{

p->rear=-1;

p->front=-1;

}

int empty(pqueue \*p)

{

if(p->rear==-1)

return(1);

return(0);

}

int full(pqueue \*p)

{

if((p->rear+1)%MAX==p->front)

return(1);

return(0);

}

void enqueue(pqueue \*p, int x)

{

setlocale(LC\_ALL, "Rus");

int i;

if(full(p))

printf("\nПереполнение");

else

{

if(empty(p))

{

p->rear=p->front=0;

p->data[0]=x;

}

else

{

i=p->rear;

while(x>p->data[i])

{

p->data[(i+1)%MAX]=p->data[i];

i=(i-1+MAX)%MAX; //движение против часовой стрелки внутри очереди

if((i+1)%MAX==p->front)

break;

}

//добавить x

i=(i+1)%MAX;

p->data[i]=x;

//re-adjust rear

p->rear=(p->rear+1)%MAX;

}

}

}

int dequeue(pqueue \*p)

{

setlocale(LC\_ALL, "Rus");

int x;

if(empty(p))

{

printf("\nПотеря значимости..");

}

else

{

x=p->data[p->front];

if(p->rear==p->front) //удалить последний элемент

initialize(p);

else

p->front=(p->front +1)%MAX;

}

return(x);

}

void print(pqueue \*p)

{

setlocale(LC\_ALL, "Rus");

int i,x;

if(empty(p))

{

printf("\nОчередь пуста..");

}

else

{

i=p->front;

while(i!=p->rear)

{

x=p->data[i];

printf("\n%d",x);

i=(i+1)%MAX;

}

//вывести последний элемент

x=p->data[i];

printf("\n%d",x);

}

}

**Задание 2:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

#define BUFSIZE 256

struct element {

char \*name;

struct element \*next;

};

struct element \*tail;

void init\_queue (void);

void enqueue (char \*name);

int dequeue (char \*name);

void print\_queue (void);

void error (char \*msg);

int main (int argc, char \*\*argv)

{

setlocale(LC\_ALL, "Rus");

char buf [BUFSIZE];

init\_queue ();

while (1) {

printf ("\n1: Очередь\n2: Удаление из очереди\n3: Вывести очередь\n0: Выход\n\nВыберите: ");

if (fgets (buf, BUFSIZE, stdin) == NULL)

break;

if (buf [0] == '1') {

// Очередь

// Ввести элемент

printf ("Элемент: ");

if (fgets (buf, BUFSIZE, stdin) == NULL)

break;

int len = strlen (buf);

if (buf [len - 1] == '\n')

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

enqueue (buf);

} else if (buf [0] == '2') {

// Удаление элемента

if (dequeue (buf) != -1)

printf ("%s Удален из очереди\n", buf);

} else if (buf [0] == '3')

print\_queue ();

else if (buf [0] == '0')

break;

else

fprintf (stderr, "Ошибка ввода\n");

}

exit (0);

}

void init\_queue (void)

{

tail = NULL;

}

void enqueue (char \*name)

{

struct element \*ptr;

char \*cp;

if ((ptr = (struct element \*) malloc (sizeof (struct element))) == NULL)

error ("malloc");

if ((cp = (char \*) malloc (strlen (name) + 1)) == NULL)

error ("malloc");

strcpy (cp, name);

ptr -> name = cp;

if (tail == NULL) {

ptr -> next = ptr;

}

else

{

ptr -> next = tail -> next;

tail -> next = ptr;

}

tail = ptr;

}

int dequeue (char \*name) // возвращает -1 при ошибке

{

setlocale(LC\_ALL, "Rus");

struct element \*ptr;

char \*cp;

if (!tail) {

fprintf (stderr, "Очередь пустая\n");

return -1;

}

// get the head

ptr = tail -> next;

cp = ptr -> name;

if (ptr == tail)

tail = NULL;

else

tail -> next = ptr -> next;

free (ptr);

strcpy (name, cp);

free (cp);

return 0;

}

void print\_queue (void)

{

setlocale(LC\_ALL, "Rus");

struct element \*ptr, \*head;

if (!tail) {

fprintf (stderr, "Очередь пустая\n");

return;

}

printf ("Очередь: \n");

// get the head

head = ptr = tail -> next;

int i = 1;

do {

printf ("%d. %s\n", i, ptr -> name);

ptr = ptr -> next;

i++;

} while (ptr != head);

}

void error (char \*msg)

{

perror (msg);

exit (1);

}