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

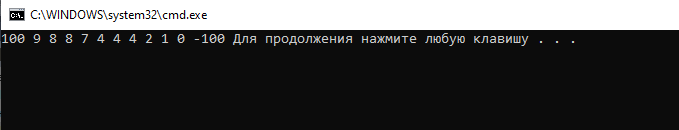
Кривенков И.В

Приняли: Юрова О.В.

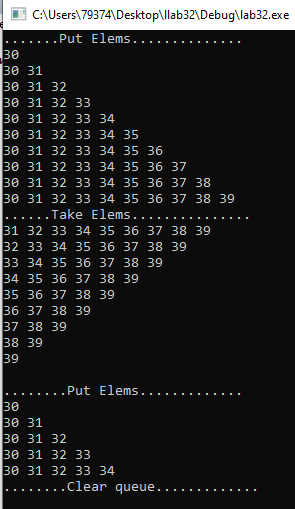
Акифьев И.В.

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 <malloc.h>

typedef struct \_node { // создание списка

struct \_node\* next; // это у нас ссылка на следующий указатель

int val;

} node;

typedef struct {

node\* lst; // это у нас ссылка на предыдущий указатель

} pqueue;

inline void pqueue\_init(pqueue\* pq) { pq->lst = NULL; } // встриваемая функция инициализации для вызова, пред знач pq становится пустым

inline int pqueue\_top(pqueue\* pq) { return pq->lst->val; } //встраиваемая функция возвращает

inline int pqueue\_empty(pqueue\* pq) { return (pq->lst == NULL); } // встраиваемая функция для вызова возвращает пред знач pq становится пустым

int pqueue\_push(pqueue\* pq, int val); // вызов функции

void pqueue\_pop(pqueue\* pq); // вызов функции

void pqueue\_clear(pqueue\* pq); // вызов функции

int main(void){

int i;

pqueue pq;

pqueue\_init(&pq);

for(i = 0; i < 10; ++i)

pqueue\_push(&pq, rand() % 10);

pqueue\_push(&pq, 100);

pqueue\_push(&pq, -100);

while(! pqueue\_empty(&pq)){

printf("%d ", pqueue\_top(&pq));

pqueue\_pop(&pq);

}

return 0;

}

//вставка

int pqueue\_push(pqueue\* pq, int val){

node\* p, \*i, \*\*q = &pq->lst;

for(i = pq->lst; (i != NULL) && (i->val > val); ){ //пока i не null и прошлое знач val больше val нынешнего выполняется

q = &i->next; // q присваивается значения указателя i с прошлого next

i = i->next; // i присваивается значение указателя i с прошлого next

}

// Создали указатель на переменную структуры списка и выделим немного памяти для нее

p = (node\*)malloc(sizeof(node));

if(p != NULL){ //если список p не пустой, то

p->val = val; // p с прошлым val становится новыйм val

p->next = i; // p следующее становится i

if(pq->lst == NULL) //если пред знач pq не пустая, то

pq->lst = p; // пред знач pq становится p

else

\*q = p; // указатель q становится p

}

return (p != NULL);

}

//вытолкнуть макс

void pqueue\_pop(pqueue\* pq){

node\* t;

if(pq->lst != NULL){ // если пред знач pq не пустое, то

t = pq->lst; // t становится пред знач pq

pq->lst = pq->lst->next; //пред знач pq становится след знач pq

free(t);

}

}

//удаление увсех

void pqueue\_clear(pqueue\* pq){

node\* t;

while(pq->lst != NULL){ // пока пред знач pq не пустое

t = pq->lst; // t становится пред значением pq

pq->lst = pq->lst->next; // пред знач pq становится след значением pq

free(t);

}

}

**Задание 2:**

#include <conio.h>

#include <stdio.h>

struct list // создание списка

{

int el;

list \*next;

};

struct queue // создание очереди

{

list \*beg, \*end;

};

void putToQueue(queue \*q, int iEl) //добавить в очередь

{

list \*tmp; // шаблон класса контейнеров последовательностей

tmp = new list;

tmp->next = NULL; // след элемент tmp пустой

tmp->el = iEl;

if(q->end != NULL) // если послед элем q не пустой, то

q->end->next = tmp; // q с указателем на next присваивается значение tmp

else

q->beg = tmp; // q с указателем на beg становится tmp

q->end = tmp; // q с указателем на end становится tmp

}

int takeFromQueue(queue \*q, int \*iEl) // взять из очереди

{

if(q->beg == NULL) // если q с указателем на beg пустое, то

return 0;

list \*tmp;

tmp = q->beg;

\*iEl = tmp->el;

q->beg = tmp->next;

delete tmp;

if(q->beg == NULL)

q->end = NULL;

return 1;

}

queue \*CreateQueue() //создать очередь

{

queue \*q;

q = new queue;

q->beg = NULL;

q->end = NULL;

return q;

}

int isQueueEmpty(queue \*q) //проверка очереди на пустоту

{

if(q->beg == NULL)

return 1;

return 0;

}

int ClearQueue(queue \*q) //очистка очереди

{

if(q->beg == NULL)

return 0;

list \*tmp, \*t;

tmp = q->beg;

while(tmp->next != NULL)

{

t = tmp;

tmp = t->next;

delete t;

}

q->beg = NULL;

q->end = NULL;

return 1;

}

void PrintQueue(queue \*q) //вывестии очередь на экран

{

printf("\n");

queue \*tmp = CreateQueue();

int iEl;

while(!isQueueEmpty(q))

{

takeFromQueue(q, &iEl);

printf("%d ", iEl);

putToQueue(tmp, iEl);

}

while(!isQueueEmpty(tmp))

{

takeFromQueue(tmp, &iEl);

putToQueue(q, iEl);

}

}

void main()

{

int i;

queue \*q = CreateQueue();

printf(".......Put Elems..............");

for(i=30; i<40; i++)

{

putToQueue(q, i);

PrintQueue(q);

}

printf("\n......Take Elems...............");

while(!isQueueEmpty(q))

{

takeFromQueue(q, &i);

PrintQueue(q);

}

printf("\n........Put Elems.............");

for(i=30; i<35; i++)

{

putToQueue(q, i);

PrintQueue(q);

}

printf("\n........Clear queue.............");

ClearQueue(q);

PrintQueue(q);

\_getch();

}