Пензенский государственный университет

Кафедра «Вычислительная техника»

**Отчёт**

По лабораторной работе №3

По дисциплине: «[Логика и основы алгоритмизации в инженерных задачах](http://moodle.pnzgu.ru/course/view.php?id=48560)»

на тему: «Динамические списки»

*Выполнил студент группы 21ВВ2:*

*Зайкин С.А.*

*Приняли:*

*Митрохин М.А.*

*Юрова О.А.*

Пенза 2022

**Цель работы:**

Изучение организации списковых структур и построение реальных структур данных на базе списков.

**Лабораторная работа:**

**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).

**Листинг:**

#include "stdafx.h"

#include <iostream>

#include <string>

using namespace std;

//очередь c приоритетом

//реализация списка(двухсвязный)

typedef struct \_Node {

void\* value;

int prio;

struct \_Node\* next; //хранит адрес следующего узла

struct \_Node\* prev; //хранит адрес предыдущего узла

} Node;

// структура элемента

typedef struct \_DblLinkedList {

size\_t size;

Node\* head;

Node\* tail;

//В случае, когда в списке нет элементов, оба они равны нулю

} DblLinkedList;

//оздаёт экземпляр структуры DblLinkedList

DblLinkedList\* createDblLinkedList() {

DblLinkedList\* tmp = (DblLinkedList\*)malloc(sizeof(DblLinkedList));

tmp->size = 0;

tmp->head = tmp->tail = NULL;

return tmp;

}

//удаление списка

void deleteDblLinkedList(DblLinkedList\*\* list) {

Node\* tmp = (\*list)->head;

Node\* next = NULL;

while (tmp) {

next = tmp->next;

free(tmp);

tmp = next;

}

free(\*list);

(\*list) = NULL;

}

//добавление в конец

void pushBack(DblLinkedList\* list, void\* Value, int value, int prio) {

Node\* tmp = list->head;

Node\* tmp\_prev = tmp;

int index = 0;

while (tmp != NULL) {

if (prio <= tmp->prio) {

tmp\_prev = tmp;

tmp = tmp->next;

index++;

}

else

break;

}

if (index == 0) {

Node\* ins = (Node\*)malloc(sizeof(Node));

if (ins == NULL) {

exit(1);

}

ins->value = Value;

ins->prio = prio;

ins->next = list->head;

ins->prev = NULL;

if (list->head) {

list->head->prev = ins;

}

list->head = ins;

if (list->tail == NULL) {

list->tail = ins;

}

list->size++;

}

else if (tmp == NULL) {

Node\* ins = (Node\*)malloc(sizeof(Node));

if (ins == NULL) {

exit(3);

}

ins->value = Value;

ins->prio = prio;

ins->next = NULL;

ins->prev = list->tail;

if (list->tail) {

list->tail->next = ins;

}

list->tail = ins;

if (list->head == NULL) {

list->head = ins;

}

list->size++;

}

else {

Node\* ins = NULL;

if (tmp\_prev == NULL) {

exit(5);

}

ins = (Node\*)malloc(sizeof(Node));

ins->value = Value;

ins->prio = prio;

ins->prev = tmp\_prev;

ins->next = tmp\_prev->next;

if (tmp\_prev->next) {

tmp\_prev->next->prev = ins;

}

tmp\_prev->next = ins;

if (!tmp\_prev->prev) {

list->head = tmp\_prev;

}

if (!tmp\_prev->next) {

list->tail = tmp\_prev;

}

list->size++;

}

}

//удаление первого элемента

void\* popFront(DblLinkedList\* list) {

Node\* prev;

void\* tmp;

if (list->head == NULL) {

exit(2);

}

prev = list->head;

list->head = list->head->next;

if (list->head) {

list->head->prev = NULL;

}

if (prev == list->tail) {

list->tail = NULL;

}

tmp = prev->value;

free(prev);

list->size--;

return tmp;

}

//вывод первого элемента

Node\* getN(DblLinkedList\* list) {

Node\* tmp = list->head;

return tmp;

}

//вывод списка

void printDblLinkedList(DblLinkedList\* list) {

Node\* tmp = list->head;

while (tmp) {

printf("%d ", \*((int\*)tmp->value));

tmp = tmp->next;

}

printf("\n");

}

int main()

{

setlocale(0, "");

DblLinkedList\* list = createDblLinkedList();

int arr[40];

int i=0;

while (true){

printf("Для завершения введите \* \n");

printf("введите элемент и приоритет \n");

string ss,s;

//scanf("%s",s);

//scanf("%s",ss);

cin>>s>>ss;

if (ss == "\*" || s == "\*")

break;

else{

arr[i]=atoi(s.c\_str()), arr[i+1]=atoi(ss.c\_str());

pushBack(list,&arr[i],arr[i],arr[i+1]);

i+=2;

}

}

//вывод очереди

printDblLinkedList(list);

printf("Первый элемент %d\n", \*((int\*)(popFront(list))));

printf("Первый элемент %d\n", \*((int\*)(getN(list))->value));

popFront(list);

printf("Первый элемент %d\n", \*((int\*)(list->head->value)));

popFront(list);

while (true){

printf("Для завершения введите \* \n");

printf("введите элемент и приоритет \n");

string ss,s;

//scanf("%s",s);

//scanf("%s",ss);

cin>>s>>ss;

if (ss == "\*" || s == "\*")

break;

else{

arr[i]=atoi(s.c\_str()), arr[i+1]=atoi(ss.c\_str());

pushBack(list,&arr[i],arr[i],arr[i+1]);

i+=2;

}

}

printDblLinkedList(list);

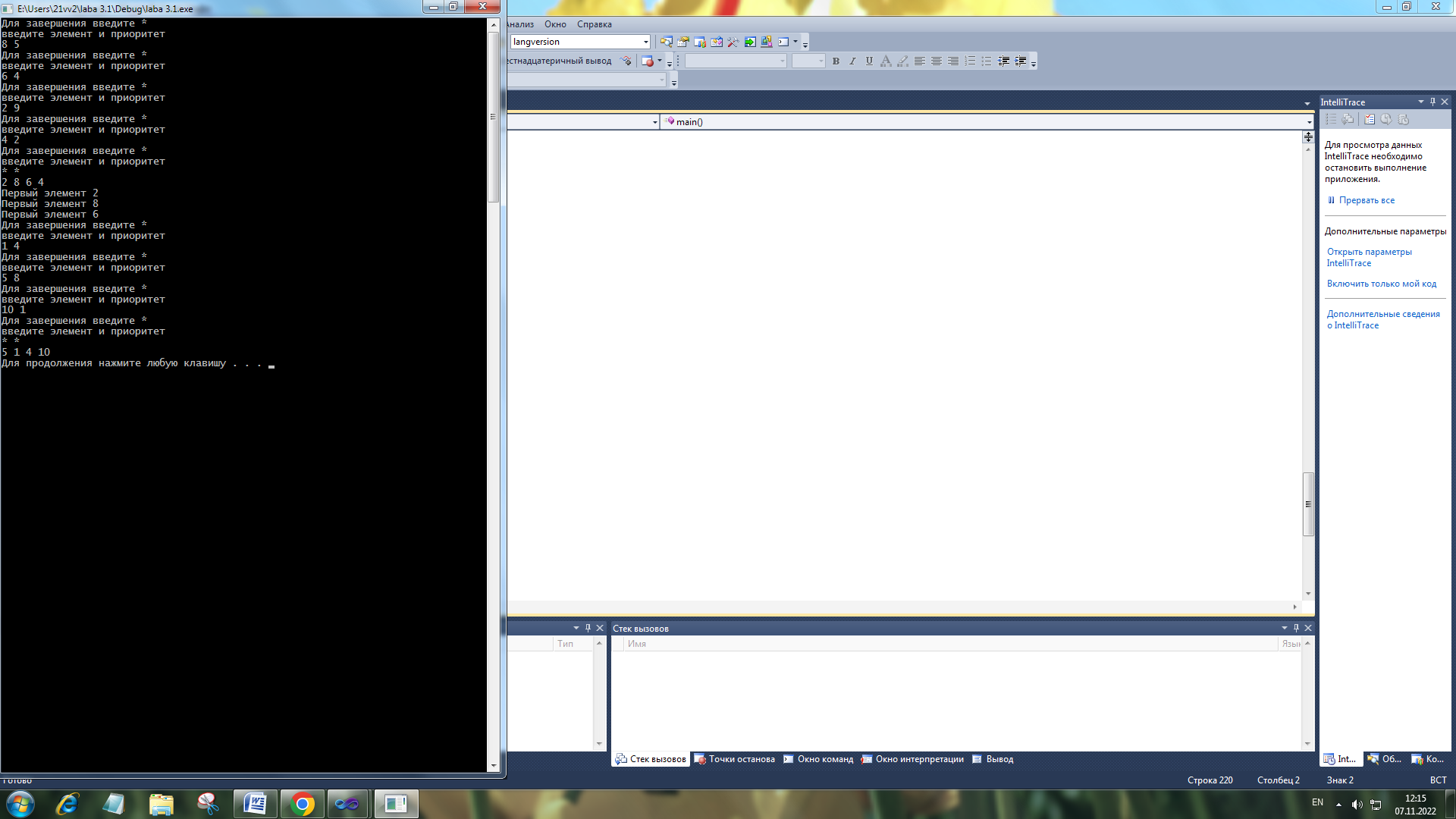
deleteDblLinkedList(&list);

system("pause");

return 0;

}

**Результат работы программы**



**Вывод:** изучил организацию списковых структур и построение реальных структур данных на базе списков.