Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительной техники»

**ОТЧЕТ**

по лабораторной работе №3

по курсу «Логика и основы алгоритмизации в инженерных задачах.»

на тему: «Динамические списки.»

Выполнили**:**

студенты группы 21ВВ4

Куряев Ю.Р.

Купцов Т.А.

Проверили:

Акифьев И.В.

Юрова О.В,

Пенза

2022

**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 <iostream>

struct node

{

int priority;

char inf[256]; // полезная информация

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

};

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

int dlinna = 0;

// Функции добавления элемента, просмотра списка

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

void WorkWithPriorityQueue();

char find\_el[256];

struct node \*find(char \*name); // функция нахождения элемента

struct node \*get\_struct(void); // функция создания элемента

struct node \*get\_struct(void)

{

int priority;

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);

printf("Введите приоритет \n"); // вводим данные

scanf("%d", &priority);

p->priority = 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)\*/

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;

}

return;

}

/\* Просмотр содержимого списка. \*/

void review(void)

{

struct node \*struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc != NULL)

{

printf("Имя - %s, Приоритет - %d \n", struc->inf, struc->priority);

struc = struc->next;

}

}

/\* Поиск элемента по содержимому. \*/

struct node \*find(char \*name)

{

struct node \*struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

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

{

return struc;

}

struc = struc->next;

}

printf("Элемент не найден\n");

return NULL;

}

/\* Удаление элемента по содержимому. \*/

void del(char \*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; // установливаем 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;

}

}

else

{

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;

}

}

void AddElementToQueuePriority(node\* Element){

node\* item = head;

node\* previuseElem = NULL;

while (item != NULL) {

if (item->priority < Element->priority) {

if (previuseElem == NULL) {

Element->next = item;

head = Element;

}

else {

struct node\* temp = previuseElem->next;

previuseElem->next = Element;

Element->next = item;

}

return;

}

else{

previuseElem = item;

item = item->next;

}

}

if (head == NULL) {

head = Element;

}

else {

previuseElem->next = Element;

}

}

void WorkWithPriorityQueue() {

int UserInput = 1;

printf("Работа с приоритетной очередью\n");

while (UserInput > 0) {

node\* InputElement = get\_struct();

AddElementToQueuePriority(InputElement);

printf("Введите 0, чтобы закончить \t или 1 , чтобы продолжить\n");

scanf("%d", &UserInput);

}

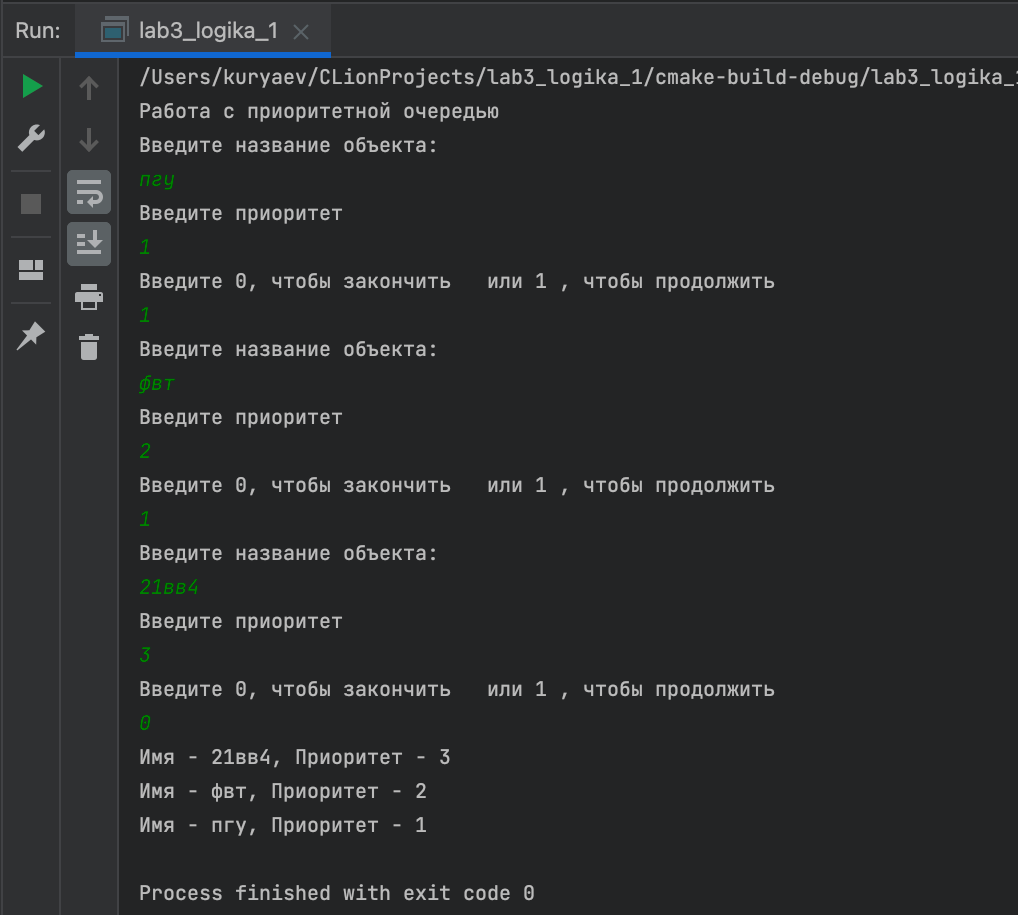
review();

}

int main(void) {

WorkWithPriorityQueue();

}

**Результат:**

**Вывод:** в ходе выполнения лабораторной работы мы научились реализовывать приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта.