**Министерство науки и высшего 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 студ. группы 24ВВВ4:

Агуреев А.П.

Невежин И.С.

Приняли:

Акифьев И.В.\_\_\_\_

Юрова О.В.\_\_\_\_

Пенза 2025

**Цель работы:**

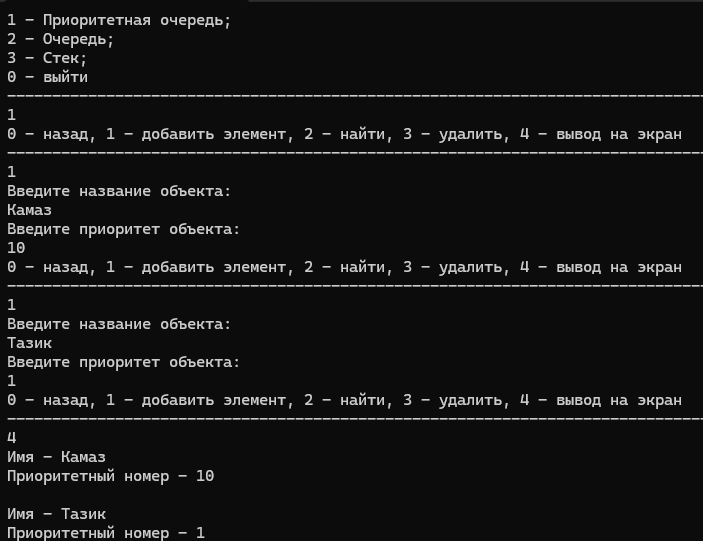
Вспомнить и закрепить на практике навыки работы с динамическими списками. Реализовать с их помощью такие структуры данных, как «Приоритетная очередь», «Очередь» и «Стек».

**Лабораторное задание:**

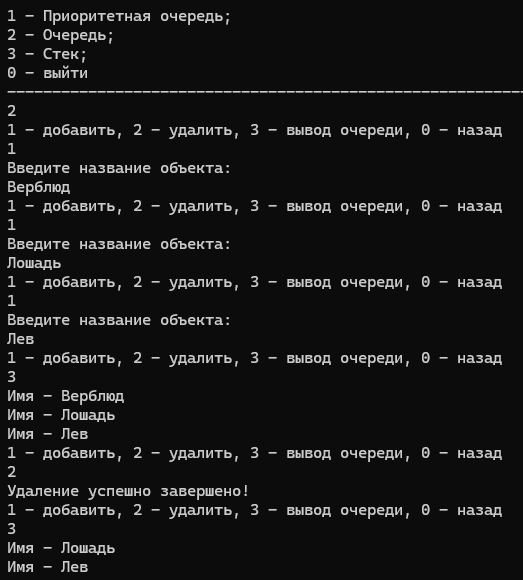
1. Реализовать приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта (т.е. объект с большим приоритетом становится перед объектом с меньшим приоритетом).
2. \* На основе приведенного кода реализуйте структуру данных *Очередь*.
3. \* На основе приведенного кода реализуйте структуру данных *Стек*.

**Результат работы программы:**

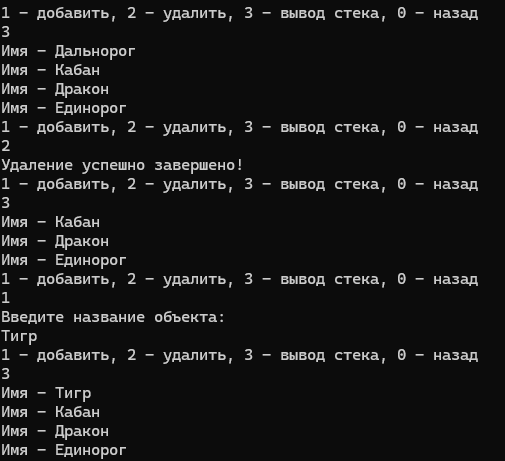
1. Приоритетная очередь



1. Очередь



1. Стек



**Вывод:**

Вспомнили и закрепить на практике навыки работы с динамическими списками. Реализовали с их помощью такие структуры данных, как «Приоритетная очередь», «Очередь» и «Стек».

**Листинг:**

# Файл Header.h

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <iostream>

#include <windows.h>

using namespace std;

struct node {

char inf[256];

int priority;

struct node\* next;

};

struct node\_queue {

char inf[256];

struct node\_queue\* next;

};

struct node\_stack {

char inf[256];

struct node\_stack\* next;

};

extern struct node\* head;

extern struct node\* last;

extern int leight;

extern struct node\_queue\* head\_queue;

extern struct node\_queue\* last\_queue;

// Функции - приоритетная очередь

struct node\* get\_struct(void);

void spstore(void);

void review(void);

struct node\* find(char\* name);

struct node\* find\_priority(int priority);

void del(char\* name);

void del\_prior(int priority);

// Функции - обычная очередь

struct node\_queue\* get\_struct\_queue(void);

void spstore\_queue(void);

void review\_queue(void);

void del\_queue(void);

// Функции - стек

struct node\_stack\* get\_struct\_stack(void);

void spstore\_stack(void);

void review\_stack(void);

void del\_stack(void);

# Файл Task\_1.cpp

#include "Header.h"

struct node\* head = NULL;

struct node\* last = NULL;

int leight = 0;

struct node\* get\_struct(void) {

struct node\* p = NULL;

char s[256];

int prior;

if ((p = (node\*)malloc(sizeof(struct node))) == NULL) {

printf("Ошибка при распределении памяти\n");

exit(1);

}

printf("Введите название объекта: \n");

scanf("%s", s);

if (\*s == 0) {

printf("Запись не была произведена\n");

return NULL;

}

cout << "Введите приоритет объекта:\n";

cin >> prior;

strcpy(p->inf, s);

p->priority = prior;

p->next = NULL;

return p;

}

void spstore(void) {

struct node\* p = NULL;

p = get\_struct();

if (head == NULL && p != NULL) {

head = p;

last = p;

}

else if (head != NULL && p != NULL) {

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

p->next = head;

head = p;

return;

}

struct node\* struc = head;

while (struc->next != NULL && struc->next->priority >= p->priority) {

struc = struc->next;

}

p->next = struc->next;

struc->next = p;

if (p->next == NULL) {

last = p;

}

}

}

void review(void) {

struct node\* struc = head;

if (head == NULL) {

printf("Список пуст\n");

}

while (struc) {

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

struc = struc->next;

}

}

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;

}

struct node\* find\_priority(int priority) {

struct node\* struc = head;

if (head == NULL) {

printf("Список пуст\n");

}

while (struc) {

if (struc->priority == priority) {

return struc;

}

struc = struc->next;

}

printf("Элемент не найден\n");

return NULL;

}

void del(char\* name) {

struct node\* struc = head;

struct node\* prev = NULL;

bool flag = 0;

if (head == NULL) {

printf("Список пуст\n");

return;

}

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

flag = 1;

head = struc->next;

free(struc);

struc = head;

}

else {

prev = struc;

struc = struc->next;

}

while (struc) {

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

flag = 1;

if (struc->next) {

prev->next = struc->next;

free(struc);

struc = prev->next;

}

else {

prev->next = NULL;

free(struc);

return;

}

}

else {

prev = struc;

struc = struc->next;

}

}

if (flag == 0) {

printf("Элемент не найден\n");

}

}

void del\_prior(int priority) {

struct node\* struc = head;

struct node\* prev = NULL;

bool flag = 0;

if (head == NULL) {

printf("Список пуст\n");

return;

}

if (priority == struc->priority) {

flag = 1;

head = struc->next;

free(struc);

struc = head;

}

else {

prev = struc;

struc = struc->next;

}

while (struc) {

if (priority == struc->priority) {

flag = 1;

if (struc->next) {

prev->next = struc->next;

free(struc);

struc = prev->next;

}

else {

prev->next = NULL;

free(struc);

return;

}

}

else {

prev = struc;

struc = struc->next;

}

}

if (flag == 0) {

printf("Элемент не найден\n");

}

}

# Файл Task\_2.cpp

#include "Header.h"

struct node\_queue\* head\_queue = NULL;

struct node\_queue\* last\_queue = NULL;

struct node\_queue\* get\_struct\_queue(void) {

struct node\_queue\* pointer = NULL;

char s[256];

if ((pointer = (node\_queue\*)malloc(sizeof(struct node\_queue))) == NULL) {

printf("Ошибка при распределении памяти\n");

exit(1);

}

printf("Введите название объекта: \n");

scanf("%s", s);

if (\*s == 0) {

printf("Запись не была произведена\n");

return NULL;

}

strcpy(pointer->inf, s);

pointer->next = NULL;

return pointer; // возвращаем указатель на созданный элемент

}

/\* Последовательное добавление в список элемента (в конец) \*/

void spstore\_queue(void) {

struct node\_queue\* pointer = NULL;

pointer = get\_struct\_queue();

if (head\_queue == NULL && pointer != NULL) { // если списка нет, то устанавливаем голову списка

head\_queue = pointer;

last\_queue = pointer;

}

else if (head\_queue != NULL && pointer != NULL) { // список уже есть, то вставляем в конец

last\_queue->next = pointer;

last\_queue = pointer;

}

return;

}

/\* Просмотр содержимого списка \*/

void review\_queue(void) {

struct node\_queue\* struc = head\_queue;

if (head\_queue == NULL) {

printf("Список пуст\n");

}

while (struc) {

printf("Имя - %s\n", struc->inf);

struc = struc->next;

}

}

/\* Удаление элемента с начала очереди \*/

void del\_queue() {

if (head\_queue == NULL) {

printf("Список пуст\n");

return;

}

struct node\_queue\* struc = head\_queue;

head\_queue = head\_queue->next;

free(struc);

cout << "Удаление успешно завершено!\n";

}

# Файл Task\_3.cpp

#include "Header.h"

struct node\_stack\* head\_stack = NULL;

struct node\_stack\* last\_stack = NULL;

struct node\_stack\* get\_struct\_second(void) {

struct node\_stack\* pointer = NULL;

char s[256];

if ((pointer = (node\_stack\*)malloc(sizeof(struct node\_stack))) == NULL) {

printf("Ошибка при распределении памяти\n");

exit(1);

}

printf("Введите название объекта: \n");

scanf("%s", s);

if (\*s == 0) {

printf("Запись не была произведена\n");

return NULL;

}

strcpy(pointer->inf, s);

pointer->next = NULL;

return pointer; // возвращаем указатель на созданный элемент

}

/\* Последовательное добавление в стек элемента (в начало) \*/

void spstore\_stack() {

struct node\_stack\* pointer = NULL;

pointer = get\_struct\_second();

if (head\_stack == NULL && pointer != NULL) { // если списка нет, то устанавливаем голову списка

head\_stack = pointer;

}

else if (head\_stack != NULL && pointer != NULL) { // список уже есть, то вставляем в конец

pointer->next = head\_stack;

head\_stack = pointer;

}

return;

}

/\* Просмотр содержимого списка \*/

void review\_stack(void) {

struct node\_stack\* struc = head\_stack;

if (head\_stack == NULL) {

printf("Список пуст\n");

}

while (struc) {

printf("Имя - %s\n", struc->inf);

struc = struc->next;

}

}

/\* Удаление элемента с начала стека\*/

void del\_stack() {

if (head\_stack == NULL) {

printf("Список пуст\n");

return;

}

struct node\_stack\* struc = head\_stack;

head\_stack = head\_stack->next;

free(struc);

cout << "Удаление успешно завершено!\n";

}

# Файл lb\_3.cpp

#include "Header.h"

int main()

{

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

short number\_task;

do {

cout << "1 - Приоритетная очередь;\n2 - Очередь;\n3 - Стек;\n0 - выйти\n";

cout << "------------------------------------------------------------------------------------------------------------------------\n";

cin >> number\_task;

switch (number\_task) {

case 1: {

int TmpFlag;

do {

cout << "0 - назад, 1 - добавить элемент, 2 - найти, 3 - удалить, 4 - вывод на экран\n";

cout << "------------------------------------------------------------------------------------------------------------------------\n";

cin >> TmpFlag;

switch (TmpFlag) {

case 0:

continue;

case 1:

spstore();

break;

case 2: {

char name[32];

int priority;

node\* Output;

bool tmpFlag;

cout << "0 - поиск по названию, 1 - по приоритетному номеру\n";

cout << "------------------------------------------------------------------------------------------------------------------------\n";

cin >> tmpFlag;

if (tmpFlag == 0) {

cout << "Введите название объекта:\n";

cin >> name;

Output = find(name);

if (Output)

cout << "Приоритетный номер найденного элемента - " << Output->priority << '\n';

}

else if (tmpFlag == 1) {

cout << "Введите приоритетный номер:\n";

cin >> priority;

Output = find\_priority(priority);

if (Output)

cout << "Название найденного элемента - " << Output->inf << '\n';

}

break;

}

case 3: {

char name\_2[32];

int priority\_2;

bool tmpFlag\_2;

cout << "удалить 0 - по названию, 1 - по приоритетному номеру\n";

cout << "------------------------------------------------------------------------------------------------------------------------\n";

cin >> tmpFlag\_2;

if (tmpFlag\_2 == 0) {

cout << "Введите название объекта, который хотите удалить:\n";

cin >> name\_2;

del(name\_2);

}

else if (tmpFlag\_2 == 1) {

cout << "Введите приоритетный номер объекта, который хотите удалить:\n";

cin >> priority\_2;

del\_prior(priority\_2);

}

break;

}

case 4:

review();

break;

}

} while (TmpFlag != 0);

break;

}

case 2: {

int FLAG;

do {

cout << "1 - добавить, 2 - удалить, 3 - вывод очереди, 0 - назад\n";

cin >> FLAG;

switch (FLAG) {

case 1:

spstore\_queue();

break;

case 2:

del\_queue();

break;

case 3:

review\_queue();

break;

case 0:

break;

default:

cout << "Неверная команда\n";

}

} while (FLAG != 0);

} break;

case 3:

int FLAG;

do {

cout << "1 - добавить, 2 - удалить, 3 - вывод стека, 0 - назад\n";

cin >> FLAG;

switch (FLAG) {

case 1:

spstore\_stack();

break;

case 2:

del\_stack();

break;

case 3:

review\_stack();

break;

case 0:

break;

default:

cout << "Неверная команда\n";

}

} while (FLAG != 0);

break;

case 0: break;

default: cout << "Неверный пункт меню\n";

}

} while (number\_task != 0);

return 0;

}