Министерство 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»

Выполнил

студент группы 21ВВП2:

Ефимушкин А. Н.

Приняли:

Митрохин М. А.

Юрова О. В.

Пенза 2023

**Цель работы**

Реализация различных видов динамических списков посредством массивов структур.

**Лабораторное задание**

**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).

**Теоретический материал**

*Динамические списки относятся к динамическим структурам и используются, когда размер данных заранее неизвестен. Созданием динамических данных должна заниматься сама программа во время своего исполнения, этим достигается эффективное распределение памяти, но снижается эффективность доступа к элементам.*

*Для обращения к динамическим данным применяют указатели.*

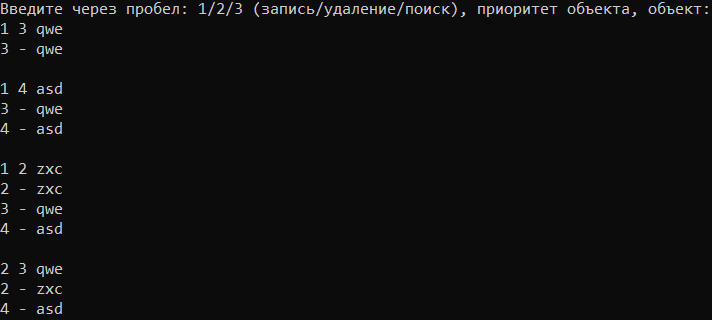
*Набор операций над списком будет включать добавление и удаление элементов, поиск элементов списка.*

**Описание метода решения задачи**

1. *Выводим памятку по использованию программы*
2. *Вводим номер команды (запись/удаление/поиск) и остальную важную информацию*
3. *Выводим результат выполнения команды*

**Результаты работ программ**

**Задание 1:**

****

**Листинг 1**#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h> //указание прототипов

#include <stdlib.h> //библиотечных функций

#include <windows.h>

#include <malloc.h>

#include <string.h>

struct node

{

int inf; // приоритет

char name[200]; // полезная информация

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

};

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

int t;

char ss[200];

struct node\* get\_struct(void){

struct node\* p = NULL;

if ((p = (node\*)malloc(sizeof(struct node))) == NULL) // выделяем память под новый элемент списка

{

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

exit(1);

}

p->inf = t;

strcpy(p->name, ss);

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

void spstore(void){

struct node \*p = NULL, \*f = 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

f = head;

if (f->inf > p->inf) {

p->next = head;

head = p;

}

if (last->inf <= p->inf) {

last->next = p;

last = p;

}

if (last->inf > p->inf && f->inf <= p->inf){

while (f->next->inf <= p->inf) {

f = f->next;

}

p->next = f->next;

f->next = p;

}

}

return;

}

void del(int eee, char\* name)

{

struct node\* struc = head; // указатель, проходящий по списку установлен на начало списка

struct node\* prev = NULL;// 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->name) == 0 && eee == struc->inf) // 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->name) == 0 && eee == struc->inf) // 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 // если не нашли, то

{

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 find(int eee, char\* name)

{

struct node\* struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

int iii = 1, ii = 0;

while (struc)

{

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

{

printf("Порядковый номер элемента: %d\n", iii);

break;

}

struc = struc->next;

iii += 1;

}

if (ii = 0) {

printf("Элемент не найден\n");

}

}

/\* Просмотр содержимого списка. \*/

void review(void){

struct node\* struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

printf("%d - %s \n", struc->inf, struc->name);

struc = struc->next;

}

return;

}

void main(){

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

struct node\* p = NULL;

int e;

printf("Введите через пробел: 1/2/3 (запись/удаление/поиск), приоритет объекта, объект: \n"); // вводим данные

while(1){

scanf("%d%d%s", &e, &t, &ss);

if (e == 1) {

spstore();

}

if (e == 2) {

del(t ,ss);

}

if (e == 3) {

find(t, ss);

}

review();

printf("\n");

}

}

**Вывод**

В ходе выполнения данной лабораторной работы мы научились реализовывать динамические списки с приоритетом.