**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“

Выполнили студенты группы 23ВВВ1 :

Жиганов Н.Д.

Видяев А.А.

Приняли:

д.т.н. доцент Митрохин М.А.

к.т.н. доцент Юрова О.В.

Пенза 2024

**Цель работы.**

Вспомнить работу с динамическими списками в Си и закрепить на практике.

**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).

**Задание 2**:

 На основе приведенного кода реализуйте структуру данных *Очередь*.

**Задание 3**:

На основе приведенного кода реализуйте структуру данных *Стек*.

**Ход работы:**

**1)**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#include <locale.h>

struct node

{

char inf[256]; // полезная информация

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

int pr;

};

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

int dlinna = 0;

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

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)

{

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);

if (\*s == 0)

{

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

return NULL;

}

strcpy(p->inf, s);

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

scanf\_s("%i", &p->pr);

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

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

void spstore(void)

{

struct node\* p = NULL, \* temp = head;

p = get\_struct();

if (head == NULL && p != NULL) // если списка нет, то устанавливаем голову

{

head = p;

}

else if (head != NULL && p != NULL) // список уже есть, то вставляем в конец

{

if (p->pr > temp->pr) {

p->next = temp;

head = p;

return;

}

while (true) {

if (temp->next == NULL || p->pr > temp->next->pr) {

p->next = temp->next;

temp->next = p;

return;

}

temp = temp->next;

}

}

return;

}

/\* Просмотр содержимого списка. \*/

void review(void)

{

struct node\* struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

printf("Имя - %s, \n", struc->inf);

struc = struc->next;

}

return;

}

/\* 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 = struc;// 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;

}

}

int main()

{

setlocale(LC\_ALL, "RUS");

spstore();

spstore();

spstore();

review();

return 0;

}

**2)**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#include <locale.h>

struct node

{

char inf[256]; // полезная информация

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

int pr;

};

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

int dlinna = 0;

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

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)

{

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);

if (\*s == 0)

{

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

return NULL;

}

strcpy(p->inf, s);

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

scanf\_s("%i", &p->pr);

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)

{

printf("Имя - %s, \n", struc->inf);

struc = struc->next;

}

return;

}

/\* 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(void)

{

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

struct node\* prev = struc;// указатель на предшествующий удаляемому элемент

if (head == NULL) // если голова списка равна NULL, то список пуст

{

printf("Список пуст\n");

return;

}

head = struc->next; // установливаем голову на следующий элемент

free(struc); // удаляем первый элемент

struc = head; // устанавливаем указатель для продолжения поиска

}

int main()

{

setlocale(LC\_ALL, "RUS");

spstore();

spstore();

spstore();

review();

while (1)

{

int nu;

printf("\nВыберите вариант дальнейшего действия:\n1. убрать 1 элемент\n2. добавить элемент\n3. вывод");

scanf\_s("%i", &nu);

if (nu == 1)

{

del();

}

if (nu == 2)

{

spstore();

}

if (nu == 3)

{

review();

}

}

return 0;

}

**3)**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#include <locale.h>

struct node

{

char inf[256]; // полезная информация

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

int pr;

};

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

int dlinna = 0;

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

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)

{

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);

if (\*s == 0)

{

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

return NULL;

}

strcpy(p->inf, s);

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

scanf\_s("%i", &p->pr);

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;

predlast = last;

last = p;

}

return;

}

/\* Просмотр содержимого списка. \*/

void review(void)

{

struct node\* struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

printf("Имя - %s, \n", struc->inf);

struc = struc->next;

}

return;

}

/\* 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(void)

{

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

struct node\* prev = NULL; // указатель на предшествующий удаляемому элемент

if (head == NULL) // если голова списка равна NULL, то список пуст

{

printf("Список пуст\n");

return;

}

if (head->next == NULL) // если в списке только один элемент

{

free(head);

head = NULL;

return;

}

while (struc->next != NULL) // проходим по списку до последнего элемента

{

prev = struc;

struc = struc->next;

}

prev->next = NULL; // отсекаем последний элемент

last = prev;

free(struc); // освобождаем память для последнего элемента

}

int main()

{

setlocale(LC\_ALL, "RUS");

spstore();

spstore();

spstore();

review();

while (1)

{

int nu;

printf("\nВыберите вариант дальнейшего действия:\n1. убрать 1 элемент\n2. добавить элемент\n3. вывод\n");

scanf\_s("%i", &nu);

if (nu == 1)

{

del();

}

if (nu == 2)

{

spstore();

}

if (nu == 3)

{

review();

}

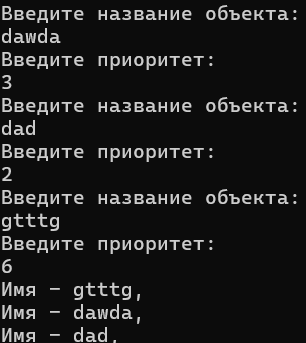
}

return 0;

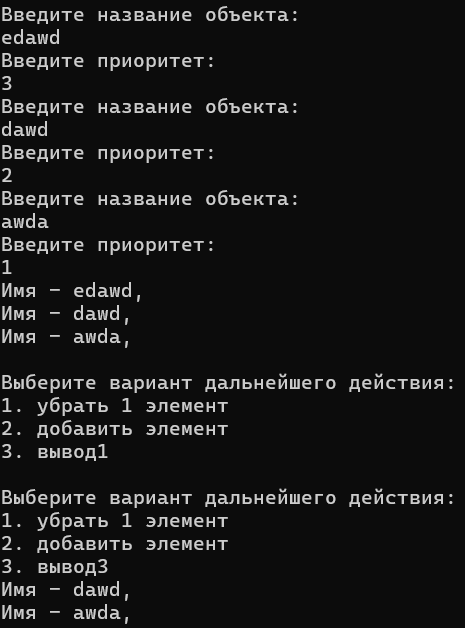
}

**Результат:**

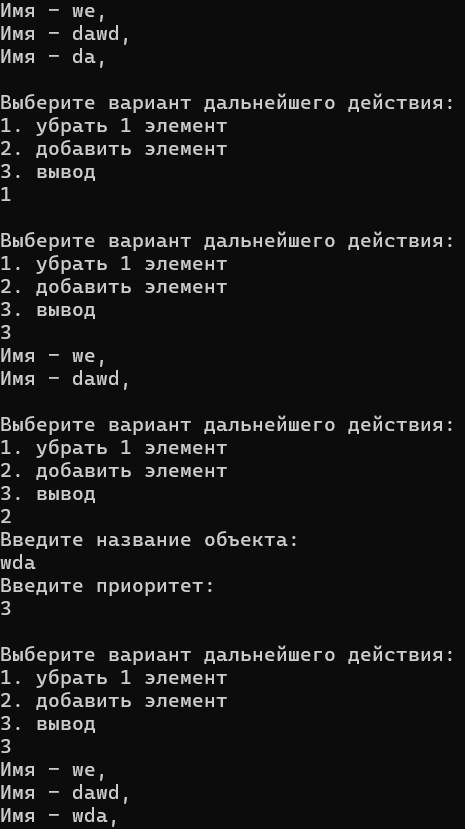
**1)**



**2)**

****

**3)**

****

**Вывод:**

В ходе лабораторной работы мы вспомнили работу с динамическими списками в Си и закрепили на практике.