МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Вычислительной техники»

**ОТЧЁТ**

по лабораторной работе №3

по курсу «Л и ОА в ИЗ»

на тему «Динамические списки»

Выполнили:

Студенты группы 22ВВС1

Костин Максим

Макеева Дарья

Приняли:

Акифьев И.В.

Юрова О.В.

Пенза 2023

***Лабораторное задание:***

**Задание 1:**

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. Реализовать стек.
3. Реализовать очередь.

**Листинг:**

**Задание 1:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <windows.h>

struct node

{

char inf[256]; // полезная информация

int prioritet;

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);

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];

int prior = 0;

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);

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

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

void spstore(void)

{

int prior = 0;

int k = 0;

struct node\* p = NULL;

p = get\_struct();

printf("Введите приоритет объекта(не меньше нуля): \n"); // вводим данные

scanf("%d", &prior);

if (prior < 0)

{

while (prior < 0) {

printf("Запись не была произведена, повторите ввод\n");

scanf("%d", &prior);

}

}

p->prioritet = prior;

struct node\* struc = head;

struct node\* prev = NULL;

if (head != NULL) {

dlinna = 0;

while (struc) {

if ((prior < struc->prioritet) && (struc == head)) {

prev = struc;

head = p;

p->next = prev;

return;

}

if ((prior == struc->prioritet) && (struc == head)) {

if (struc->next == NULL) {

last->next = p;

last = p;

return;

}

p->next = struc->next;

struc->next = p;

return;

}

if (prior >= struc->prioritet && struc->next == NULL) {

last->next = p;

last = p;

return;

}

if (prior >= struc->prioritet && struc->next != NULL) {

dlinna++;

}

else

break;

struc = struc->next;

}

if (dlinna != 0) {

k = 0;

struc = head;

while (struc) {

k++;

if (dlinna == k && struc->next != NULL) {

p->next = struc->next;

struc->next = p;

return;

}

struc = struc->next;

}

}

}

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

{

head = p;

last = p;

}

else if (head != NULL && p != NULL && head == last) // 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, приоритет - %d\n", struc->inf, struc->prioritet);

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 = 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->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() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n = 0;

char dele[256];

while (1) {

printf("\nВведите номер операции:\n1)Добавить эл. в очередь 2)Удалить элемент из очереди 3)Просмотреть очередь 4)Выйти\n");

scanf("%d", &n);

switch (n) {

case 1:

spstore();

break;

case 2:

printf("Введите содержимое удаляемого элемента:");

scanf("%s", dele);

del(dele);

break;

case 3:

review();

break;

case 4:

return 0;

}

}

\_getch();

# }

Очередь:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <windows.h>

struct node

{

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;

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);

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

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

void spstore(void)

{

struct node\* p = NULL;

struct node\* struc = head;

struct node\* prev = NULL;

p = get\_struct();

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

head = p;

last = p;

}

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

prev = struc;

head = p;

p->next = prev;

return;

}

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;

}

/\* Удаление элемента по содержимому. \*/

void del()

{

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;

}

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

free(struc); // удаляем первый элемент

struc = head; // устанавливаем указатель для продолжения поиска

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n = 0;

char dele[256];

while (1) {

printf("\nВведите номер операции:\n1)Добавить эл. в стек 2)Удалить элемент из стека 3)Просмотреть стек 4)Выйти\n");

scanf("%d", &n);

switch (n) {

case 1:

spstore();

break;

case 2:

del();

break;

case 3:

review();

break;

case 4:

return 0;

}

}

\_getch();

}

Стек:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <windows.h>

struct node

{

char inf[256]; // полезная информация

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

struct node\* prev;

};

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

int dlinna = 0;

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);

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

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

void spstore(void)

{

struct node\* p = NULL;

struct node\* struc = head;

struct node\* prev = NULL;

p = get\_struct();

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

head = p;

last = p;

}

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

struc = last;

last->next = p;

last = p;

p->prev = struc;

}

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;

}

/\* Удаление элемента по содержимому. \*/

void del()

{

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;

}

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

free(struc); // удаляем первый элемент

struc = head; // устанавливаем указатель для продолжения поиска

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n = 0;

while (1) {

printf("\nВведите номер операции:\n1)Добавить эл. в очередь 2)Удалить элемент из очереди 3)Просмотреть очередь 4)Выйти\n");

scanf("%d", &n);

switch (n) {

case 1:

spstore();

break;

case 2:

del();

break;

case 3:

review();

break;

case 4:

return 0;

}

}

\_getch();

}

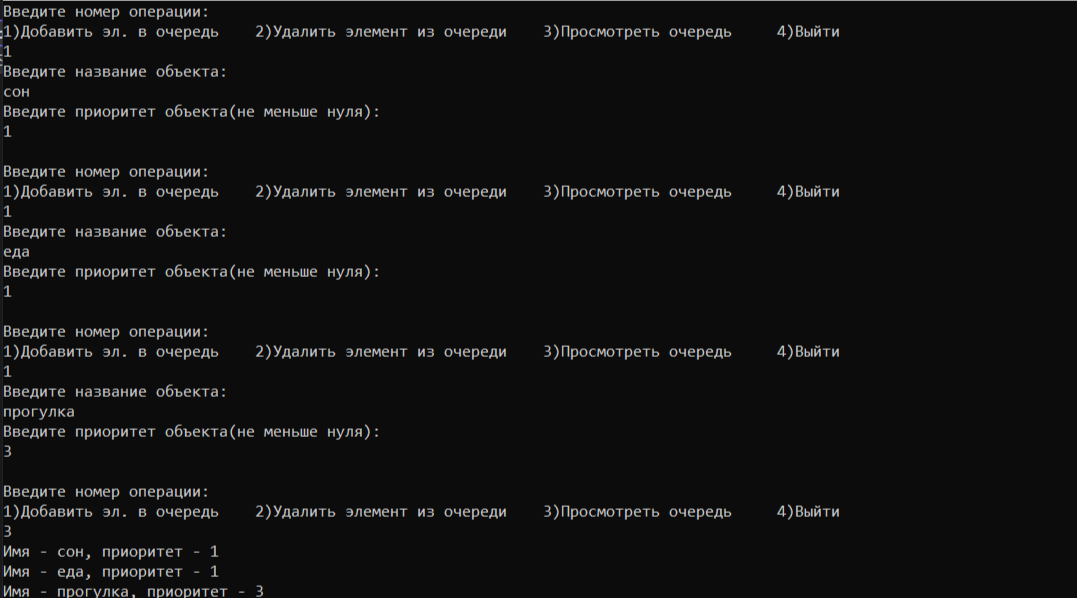
# Пояснительный текст к программе:

**Задание 1:**

p = (node\*)malloc(sizeof(struct node)) – возвращает указатель р на начало области динамической памяти размером (sizeof(struct node)

**Результат работы программы**

**Задание 1:**

****

**Вывод**

В ходе выполнения лабораторной работы была реализована приоритетная очередь путём добавления элемента в список в соответствии с приоритетом объекта (т.е. объект с большим приоритетом становится перед объектом с меньшим приоритетом).