Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЁТ**

по лабораторной работе №3

по курсу «Логика и основы алгоритмизации в ИЗ»

на тему «Динамические списки»

Выполнил:

студент групп 22ВВВ2

Дасаев Т.З.

Приняли:

д.т.н. профессор Митрохин М.А.

к.э.н. доцент Акифьев И.В.

Пенза 2023

**Название**

Динамические списки

**Цель работы**

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).

**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).
2. \* На основе приведенного кода реализуйте структуру данных *Очередь*.
3. \* На основе приведенного кода реализуйте структуру данных *Стек*.

**Листинг**

1)

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <string.h>

// Функции добавления элемента, просмотра списка

void spstore(void), review(void), del(void);

void priority(void); // Функция нахождения максимального значения приоритетной очереди

char find\_el[256];

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

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

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

int dlinna = 0;

int prior = 0; // для приоритетной очереди

struct node

{

char surname[256]; // фамилия студента

int scores; // баллы ЕГЭ

int numb; // для приоритетной очереди

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

};

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 фамилию студента: \n"); // вводим данные

scanf("%s", &s);

if (\*s == 0)

{

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

return NULL;

}

strcpy(p->surname, s);

printf("Введите сумму баллов:\n");

scanf("%d", &p->scores);

printf("Введите номер приоритета:\n");

scanf("%d", &p->numb);

printf("Номер записи: %d\n", p->numb);

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("Фамилия - %s, \n", struc->surname);

printf("Сумма баллов - %d, \n", struc->scores);

printf("Номер записи - %d, \n", struc->numb);

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->surname) == 0)

{

printf("Фамилия - %s, \n", struc->surname);

printf("Сумма баллов - %d, \n", struc->scores);

printf("Номер записи - %d, \n", struc->numb);

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;// 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 (prior == struc->numb) // 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 (prior == struc->numb) // 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;

}

}

/\* Вычисление максимального приоритета \*/

void priority()

{

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

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

prior = 0;

while (struc) // проход по списку и поиск удаляемого элемента

{

if (struc->numb > prior)

prior = struc->numb;

prev = struc; // устанавливаем указатели для продолжения поиска

struc = struc->next;

}

}

int main ()

{

setlocale(LC\_ALL, "rus");

int menu = 0;

char m[256];

char \*name;

printf("Определить максимальный приоритет: 0\n");

printf("Для добавления элемента нажмите: 1\n");

printf("Для удаления элемента нажмите: 2\n");

printf("Для поиска элемента нажмите: 3\n");

printf("Для просмотра элемента нажмите: 4\n");

printf("Для выхода нажмите: 5\n");

while (menu != 5)

{

scanf("%d", &menu);

switch (menu) {

case 0:

priority();

printf("Максимальный приоритет: %d\n", prior);

break;

case 1:

spstore();

break;

case 2:

priority();

del();

break;

case 3:

printf("Введите фамилию студента для поиска данных: \n");

scanf("%s", m);

name = m;

find(name);

break;

case 4:

review();

break;

}

}

return 0;

}

2)

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <string.h>

// Функции добавления элемента, просмотра списка

void spstore(void), review(void), del(void);

char find\_el[256];

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

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

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

int dlinna = 0;

int number;

struct node

{

char surname[256]; // фамилия студента

int scores; // баллы ЕГЭ

int numb; // для приоритетной очереди

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

};

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 фамилию студента: \n"); // вводим данные

scanf("%s", &s);

if (\*s == 0)

{

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

return NULL;

}

strcpy(p->surname, s);

printf("Введите сумму баллов:\n");

scanf("%d", &p->scores);

p->numb = number;

printf("Номер записи: %d\n", p->numb);

number += 1;

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("Фамилия - %s, \n", struc->surname);

printf("Сумма баллов - %d, \n", struc->scores);

printf("Номер записи - %d, \n", struc->numb);

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->surname) == 0)

{

printf("Фамилия - %s, \n", struc->surname);

printf("Сумма баллов - %d, \n", struc->scores);

printf("Номер записи - %d, \n", struc->numb);

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;// 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;

}

flag = 1;

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

free(struc); // удаляем первый элемент

struc = head; // устанавливаем указатель для продолжения поиска

if (flag == 0) // если флаг = 0, значит нужный элемент не найден

{

printf("Элемент не найден\n");

return;

}

}

int main ()

{

setlocale(LC\_ALL, "rus");

int menu = 0;

char m[256];

char \*name;

number = 1;

printf("Для добавления элемента нажмите: 1\n");

printf("Для удаления элемента нажмите: 2\n");

printf("Для поиска элемента нажмите: 3\n");

printf("Для просмотра элемента нажмите: 4\n");

printf("Для выхода нажмите: 5\n");

while (menu != 5)

{

scanf("%d", &menu);

switch (menu) {

case 1:

spstore();

break;

case 2:

del();

break;

case 3:

printf("Введите фамилию студента для поиска данных: \n");

scanf("%s", m);

name = m;

find(name);

break;

case 4:

review();

break;

}

}

return 0;

}

3)

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <string.h>

// Функции добавления элемента, просмотра списка

void spstore(void), review(void), del(void);

void priority(void); // Функция нахождения максимального значения приоритетной очереди

char find\_el[256];

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

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

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

int dlinna = 0;

int number = 1;

int prior;

struct node

{

char surname[256]; // фамилия студента

int scores; // баллы ЕГЭ

int numb; // для приоритетной очереди

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

};

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 фамилию студента: \n"); // вводим данные

scanf("%s", &s);

if (\*s == 0)

{

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

return NULL;

}

strcpy(p->surname, s);

printf("Введите сумму баллов:\n");

scanf("%d", &p->scores);

p->numb = number;

printf("Номер записи: %d\n", p->numb);

number += 1;

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("Фамилия - %s, \n", struc->surname);

printf("Сумма баллов - %d, \n", struc->scores);

printf("Номер записи - %d, \n", struc->numb);

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->surname) == 0)

{

printf("Фамилия - %s, \n", struc->surname);

printf("Сумма баллов - %d, \n", struc->scores);

printf("Номер записи - %d, \n", struc->numb);

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;// 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 (prior == struc->numb) // 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 (prior == struc->numb) // 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

last->next = prev;

last = prev;

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;

}

}

/\* Вычисление максимального приоритета \*/

void priority(void)

{

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

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

prior = 0;

while (struc) // проход по списку и поиск удаляемого элемента

{

if (struc->numb > prior)

prior = struc->numb;

prev = struc; // устанавливаем указатели для продолжения поиска

struc = struc->next;

}

}

int main ()

{

setlocale(LC\_ALL, "rus");

int menu = 0;

char m[256];

char \*name;

printf("Для добавления элемента нажмите: 1\n");

printf("Для удаления элемента нажмите: 2\n");

printf("Для поиска элемента нажмите: 3\n");

printf("Для просмотра элемента нажмите: 4\n");

printf("Для выхода нажмите: 5\n");

while (menu != 5)

{

scanf("%d", &menu);

switch (menu)

{

case 1:

spstore();

break;

case 2:

priority();

del();

break;

case 3:

printf("Введите фамилию студента для поиска данных: \n");

scanf("%s", m);

name = m;

find(name);

break;

case 4:

review();

break;

}

}

return 0;

}

**Результат работы программы**

Результат работы программы показан на рисунках.

1)приоритетная очередь

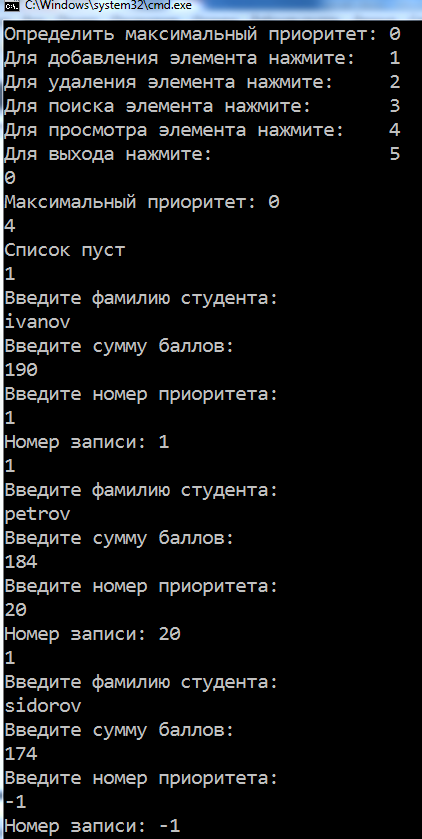
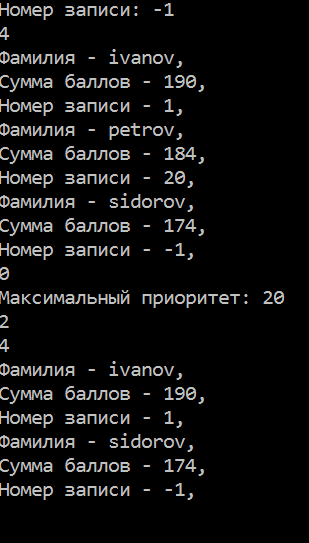
 

Рис. 1 и 2

2)очередь

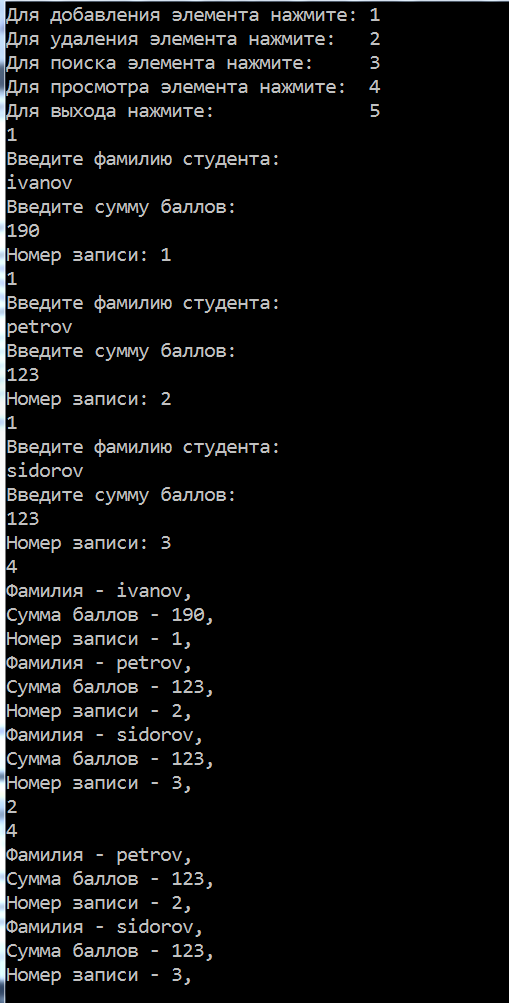


Рис. 3

3)стек

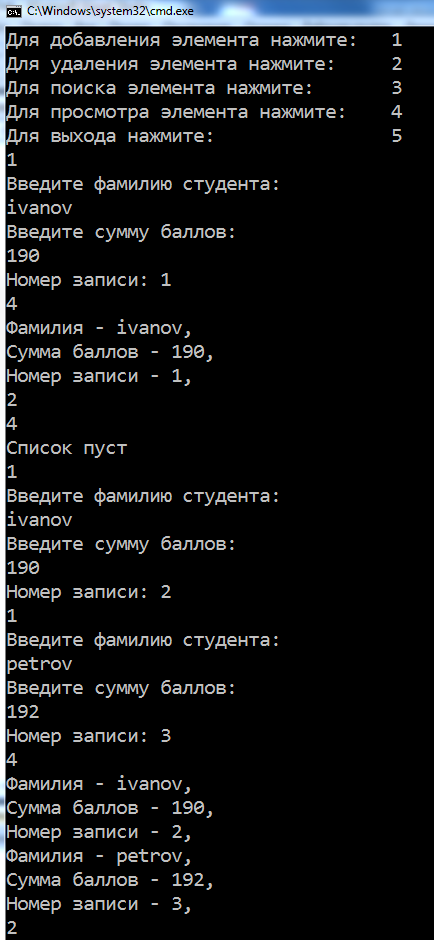
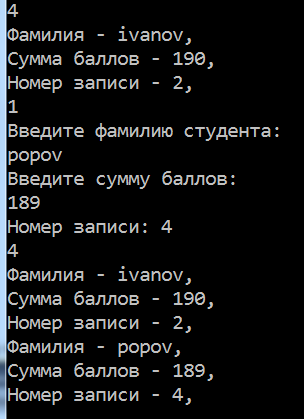
 

Рис. 4 и 5

**Вывод:** в ходе выполнения лаб. работы мы научились создавать приоритетные очереди, стек и простую очередь. Научились работать с динамическими списками.

Ссылка на репозиторий: https://github.com/t1muurr/LiOA\_l3.git