**Лабораторная работа N 3  
Структуры и связные списки**

**1. Цель работы**

Закрепление практических навыков в работе со структурами и указателями языка C, обеспечении функциональной модульности

**2. Темы для предварительного изучения**

* Указатели в языке C.
* Структуры.
* Функции и передача параметров.

**3. Постановка задачи**

**3. Постановка задачи**

Для заданной прикладной области разработать описание объектов этой области. Разработать процедуры, реализующие базовые операции над этими объектами, в том числе:

* текстовый ввод-вывод (консольный и файловый);
* присваивание;
* задание константных значений;
* сравнение (не менее 2-х типов).

Процедуры и описания данных должны составлять отдельный модуль (модуль типа данных).   
Подготовить на магнитном носителе файл исходных данных, содержащих не менее 10 значений конкретных объектов.   
Используя процедуры и описания модуля типа данных, разработать программу, обеспечивающую ввод исходных данных из первого файла данных в память и хранение их в памяти в виде связного списка, сортировку списка по алфавитному и по числовому параметру.

**3. Варианты индивидуальных заданий**

Для каждой области перечислены параметры объекта. Среди параметров обязательно есть ключевое алфавитное поле (например, фамилия), которое идентифицирует объект, у каждого объекта имеется также одно или несколько числовых полей, по которым вероятны обращения к объекту. Набор характеристик может быть расширен и усложнен по усмотрению исполнителя.

|  |  |  |
| --- | --- | --- |
| **Nпп** | **Прикладная область** | **Атрибуты информации** |
|  |  |  |
| 1 | Отдел кадров | фамилия сотрудника, имя, отчество, должность, стаж работы, оклад |
|  |  |  |
| 2 | Красная книга | вид животного, род, семейство, место обитания, численность популяции |
|  |  |  |
| 3 | Производство | обозначение изделия, группа к которой оно относится, год выпуска, объем выпуска, расход металла |
|  |  |  |
| 4 | Персональные ЭВМ | фирма-изготовитель, тип процессора, тактовая частота, емкость ОЗУ, емкость жесткого диска |
|  |  |  |
| 5 | Библиотека | автор книги, название, год издания, код УДК, цена, количество в библиотеке |
|  |  |  |
| 6 | Спутники планет | название, название планеты-хозяина, год открытия, диаметр, период обращения |
|  |  |  |
| 7 | Радиодетали | обозначение, тип, номинал, количество на схеме, обозначение возможного заменителя |
|  |  |  |
| 8 | Текстовые редакторы | наименование, фирма-изготовитель, количество окон, количество шрифтов, стоимость |
|  |  |  |
| 9 | Телефонная станция | номер абонента, фамилия, адрес, наличие блокиратора, задолженность |
|  |  |  |
| 10 | Быт студентов | фамилия студента, имя, отчество, факультет, размер стипендии, число членов семьи |
|  |  |  |
| 11 | Спортивные соревнования | фамилия спортсмена, имя, команда, вид спорта, зачетный результат, штрафные очки |
|  |  |  |
| 12 | Соревнование факультетов по успеваемости | факультет,количество студентов, средний балл по факультету, число отличников, число двоечников |
|  |  |  |
| 13 | С/х работы | фамилия студента, имя, отчество, факультет, вид работ, заработок |
|  |  |  |
| 14 | Сельхозработы | наименование с/х предприятия, вид собственности, число работающих, основной вид продукции, прибыль |
|  |  |  |
| 15 | Сведения о семье | фамилия студента, имя, отчество, факультет, специальность отца, специальность матери, количество братьев и сестер |
|  |  |  |
| 16 | Скотоводство | вид животных, количество особей в стаде в возрасте до 1 года, количество особей 1 - 3 лет, свыше 3 лет, смертность в каждой группе, рождаемость |
|  |  |  |
| 17 | Микросхемы памяти | обозначение, разрядность, емкость, время доступа, количество на схеме, стоимость |
|  |  |  |
| 18 | Описание изображения | тип фигуры (квадрат, окружность и т.п.), координаты на плоскости, числовые характеристики (длина строрны, радиус и т.п.). |
|  |  |  |
| 19 | Лесное хозяйство | наименование зеленого массива, площадь, основная порода, средний возраст, плотность деревьев на кв.км |
|  |  |  |
| 20 | Городской транспорт | вид транспорта, номер маршрута, начальная остановка, конечная остановка, время в пути |
| 21 | Университет | ФИО и должность преподавателя, названиепредмета, количество часов, тип контроля |
| 22 | Оптовая база | название товара, количество на складе, стоимость единицы, название поставщика, срок поставки |
| 23 | Сеть магазинов | номер, название, адрес, телефон магазина, ФИО, адрес, капитал владельцев магазина. |
| 24 | Авторемонтные мастерские | номер, марка, мощность и цвет автомобиля, ФИО и квалификация механика, тип работ |
| 253 | Зоопарк | вид животного, кличка, возраст, категория редкости, вес, суточный рацион мяса, овощей, молока |
| 26 | Договорная деятельность организации | шифр договора, наименование организации, наименование контрагента сроки выполнения, сумма договора, вид договора. |
| 27 | Поликлиника | ФИО и дата рождения пациента, ФИО, должность и специализация лечащего врача, диагноз |
| 28 | Домоуправление | номер квартиры, общая площадь, полезная площадь, количество комнат, фамилия квартиросъемщика, количество членов семьи, количество детей в семье, есть ли задолженность по квартплате |
| 29 | Аэропорт | номер рейса, пункт назначения, день рейса, тип самолета, время вылета, время в пути, является ли маршрут международным, |
| 30 | Шахматы | ФИО спортсмена, дата рождения, страна, спортивный разряд, участвовал ли в борьбе за звание чемпиона мира, рейтинг, |
| 31 | Ипподром | кличка лошади, масть, возраст, рейтинг, вид забега, фамилия наездника, занятое место |
| 32 | Малые планеты | Название, название планеты-хозяина (для спутников), дата открытия, диаметр, период обращения |
| 33 | Автотранспортное предприятие | номерной знак автомобиля, марка, техническое состояние, грузоподъемность, расход топлива, табельный номер и ФИО закрепленного водителя водителя |

**6. Пример решения задачи**

**6.1. Индивидуальное задание:**

Прикладная область - кафедра. Атрибуты:

* ФИО преподавателя;
* должность;
* ученое звание.

**6.2. Описание методов решения**

**6.2.1.** Представление в памяти

"База данных" в оперативной памяти представляется в виде однонаправленного линейного списка. Структура элемента списка содержит четыре поля:

struct \_emlp{

char name[25]; /\* Ф.И.О. \*/

int grade; /\* Должность \*/

int hight; /\* Звание \*/

struct \_emlp \*next; /\* Указатель на следующий элемент \*/

};

Для сокращения записи мы определяем текст struct \_emlp как \_emlp:

#define emlp struct \_emlp

**6.2.2.** Модульная структура программного изделия

Программное изделие выполняется в виде одного программного модуля, файла LAB3.C, в котором размещаются данные, функция main и вспомогательные функции.

**6.3. Описание логической структуры**

**6.3.1.** Общие переменные

Общими переменными программы являются:

struct \_emlp \*emlph=NULL; /\* указатель на начало списка \*/

char fname[]="D\_STRUCT.DA1"; /\* имя файла для хранения списка \*/

**6.3.2.**Функция main В функции main введена дополнительная структура, которая описывает меню. Ее полями являются:

* op - номер элемента меню;
* \*(opf) - адрес соответствующей функции обработки.

Основой функции main является цикл for в котором и осуществляется работа с базой данных. Сначала вертикально выводятся пункты меню, затем пользователю предлагается ввести код операции,которую он хочет проделать над базой. После чего производится вызов соответствующей функции обработки.

Рассмотрим подробнее эту операцию.В цикле перебираются все возможные значения номера меню (их всего 10) и идет сравнение ранее введенного номера с текущим,если они совпали, то по адресу, записанному в поле \*(opf) элемента массива структуры меню с текущим номером,вызывается функция обработки:

for (i=0; i<10;i++){

if (opcode==m[i].op) {

if (m[i].opf()==1) eoj=1;

break;

}

По завершению работы функции производится выход из внутреннего цикла for и вся последовательность действий повторяется заново. Выход из внешнего цикла и соответственно из программы осуществляется при выборе последнего пункта меню с помощью функции exit.

**6.3.3.**Функция печати списка

int f\_print(emlp \*);

Функция печати списка f\_print производит форматированный вывод всех элементов базы данных на экран.Если их количество превышает 20, то после вывода 20-ти элементов ( когда заполнен весь экран ) работа функции приостанавливается до нажатия любой клавиши.

**6.3.4.**Функция ввода списка

int f\_input(emlp \*);

Функция ввода списка f\_input осуществляет ввод элементов базы. Ввод производится с помощью функции добавления элемента f\_add. Конец ввода - при вводе вместо Ф.И.О. символа '\*'.

**6.3.5.**Функция добавления элемента в список

int f\_add(emlp \*);

Функция добавления элемента f\_add вносит новый элемент в базу.

**6.3.6.**Функция уничтожения элемента списка

int f\_delete(emlp \*);

Уничтожение элемента списка производится с помощью функции f\_delete, в которой происходит переприсваивание указателей и освобождение памяти, которую занимал удаленный элемент.

**6.3.7.**Функция изменения значения полей элемента списка

int f\_change(emlp \*);

Изменение значения полей элемента списка производит функция f\_change. При изменении выводятся старые значения полей и предлагается ввести новые на место старых.

**6.3.8.**Функция сортировки списка

int f\_sort(emlp \*);

Функция сортировки списка f\_sort реализована по методу "пузырька", который рассматривался в курсе "Структура и организация данных".

**6.3.9.**Функция сохранения списка на диске

int f\_save(emlp \*);

Функция сохранения списка на диске f\_save записывает все элементы базы данных в файл "D\_STRUCT.DAT".Запись осуществляется полями.

**6.3.10.**Перезапись списка из файла в оперативную память

int f\_restore(emlp \*);

Перезапись списка из файла в оперативную память производит функция f\_restore. Она читает из файла "D\_STRUCT.DAT" поля записей в поля элементов списка.

**6.4. Текст программы**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Файл LAB3.C \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Для сокращения записи типа структуры введем следующую константу \*/

#define emlp struct \_emlp

/\* Функция печати списка \*/

int f\_print();

/\* Функция ввода списка \*/

int f\_input();

/\* Добавление элемента в список \*/

int f\_add();

/\* Уничтожение элемента списка \*/

int f\_delete();

/\* Изменение значения полей элемента списка \*/

int f\_change() ;

/\* Функция сортировки списка \*/

int f\_sort();

/\* Функция сохранения списка на диске \*/

int f\_save();

/\* Перезапись списка из файла в динамическую память \*/

int f\_restore();

#include <stdio.h>

#include <alloc.h>

/\* Описание структуры \*/

emlp{

char name[25]; /\* Ф.И.О. \*/

int grade; /\* Должность \*/

int hight; /\* Звание \*/

emlp \*next; /\* Указатель на следующий элемент \*/

};

emlp \*emlph=NULL; /\* Начало списка \*/

char fname[]="D\_STRUCT.DA1"; /\* Файл для хранения списка \*/

main() {

char eoj; /\* Флаг окончания работы \*/

int i; /\* Вспомогательная переменная \*/

/\* Структура меню \*/

struct {

int op; /\* Номер операции \*/

(\*opf)(); /\* Функция обработки \*/

} m[9] = {

{'1',f\_print},{'2',f\_input},{'3',f\_add},

{'4',f\_delete},{'5',f\_change},{'6',f\_sort},

{'7',f\_save},{'8',f\_restore},{'0',}

};

int opcode; /\* Код операции \*/

for ( ; ; ) { /\* Пока не конец работы \*/

clrscr(); /\* Очистка экрана \*/

printf("1. Print\n"); /\* Вывод пунктов меню на экран \*/

printf("2. Input\n");

printf("3. Add\n");

printf("4. Delete\n");

printf("5. Change\n");

printf("6. Sort\n");

printf("7. Save\n");

printf("8. Restore\n");

printf("0. Quit\n");

printf("Enter operation code > "); /\* Запрос на ввод номера

пункта для выполнения \*/

opcode=getche(); /\* Ввод номера пункта \*/

putchar('\n');

if (opcode!='0') { /\* выход из программы,

если выбран QUIT \*/

printf("Press any key...");

getch();

exit(0);

}

for (i=0; i<10;i++){ /\* Запуск соответствующей функции

обработки \*/

if (opcode==m[i].op) {

if (m[i].opf()==1) exit(0);

break;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Функция вывода списка на экран \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

f\_print() {

emlp \*a; /\* Указатель на структуру \*/

int i, j;

/\* Если списка нет в памяти,то вывод соотвтствуюшего

сообщения \*/

/\* Иначе - вывод всего списка на экран \*/

if (emlph==NULL) printf("List empty\n");

else {

for (a=emlph,i=1,j=1; a!=NULL; a=a->next,j++,i++) {

printf("#%-2d %-10s %-4d %-4d\n",

i,a->name, a->grade,a->hight);

if (j==20){

printf("Press any key for continue...\n");

getch();

j=1;

}

}

printf("======= end of list ========\n");

}

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\* Функция ввода элементов списка \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

f\_input() {

int cc;

printf("Enter name=\* for end of stream\n");

/\* Конец ввода - при вводе '\*' вместо имени \*/

while (!(cc=f\_add())); /\* Вызов функции добавления \*/

return cc;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\* Добавление элемента в список \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int f\_add() {

emlp \*a, \*b;

char ss[40];

int i=1;

/\* Если список существует,осуществляем вставку элемента \*/

if (emlph!=NULL)

for (i++,a=emlph; a->next!=NULL; a=a->next,i++);

/\* Приглашение к вводу \*/

printf("Line #%d. Enter: name grade hight >",i);

scanf("%s",ss);

if (ss[0]=='\*') return 2;

/\* Выделение памяти под новый элемент \*/

b=(emlp \*)malloc(sizeof(emlp));

strcpy(b->name,ss);

scanf("%d %d",&(b->grade),&(b->hight));

b->next=NULL;

/\* Элемент вставляется после головы списка или в начало,

если список пустой \*/

if (emlph==NULL) emlph=b;

else a->next=b;

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\* Функция сохранения списка на диске \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

f\_save() {

FILE \*dat;

emlp \*a;

dat=fopen(fname,"w"); /\* Открытие файла на запись \*/

/\* Запись в файл осуществляется полями \*/

for (a=emlph; a!=NULL; a=a->next)

fprintf(dat,"%s %d %d\n",a->name,a->grade,a->hight);

/\* В конце файла - спецкод '\*\*\*' \*/

fprintf(dat,"\*\*\*\n");

fclose(dat); /\* Закрытие файла \*/

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\* Перезапись списка из файла в динамическую память \*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

f\_restore() {

FILE \*dat;

char ss[40];

emlp \*a, \*b;

/\* Открытие файла для чтения,если файл не найден-вывод

соответствующего сообщения \*/

if ((dat=fopen(fname,"r"))==NULL) {

printf("File not found : %s\n",fname);

return 1;

}

else {

emlph=NULL;

do {

/\* Чтение из файла по полям пока не дошли до

спецкода '\* '\*/

fscanf(dat,"%s",ss);

if (ss[0]!='\*') {

/\* Выделение памяти под новый элемент \*/

b=(emlp \*)malloc(sizeof(emlp));

if (emlph==NULL) emlph=b;

else a->next=b;

strcpy(b->name,ss);

fscanf(dat,"%d %d\n",&(b->grade),&(b->hight));

b->next=NULL;

a=b;

}

} while (ss[0]!='\*');

fclose(dat); /\* Закрытие файла \*/

}

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Функция сортировки списка \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

f\_sort() {

int n;

emlp \*a, \*b, \*c;

/\* Если список пустой или в нем один элемент,

то выход из функции \*/

if ((emlph==NULL)||(emlph->next==NULL)) return 0;

/\* Сортировка списка методом "пузырька" \*/

for (n=1; n; ) {

n=0;

for (a=emlph, b=emlph->next; b!=NULL; )

if (strcmp(a->name,b->name)>0) {

a->next=b->next; b->next=a;

if (a==emlph) emlph=b;

else c->next=b;

c=b; b=a->next;

n=1;

}

else {

c=a; a=b; b=b->next;

}

}

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\* Ввод номера элемента \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int get\_ln () {

int ln;

printf("Enter line number >");

do {

/\* Ввод номера элемента и проверка его(если он меньше единицы-

выдается сообщение об ошибке \*/

scanf("%d",&ln);

if (ln<1) {

printf("Illegial line number. Try again >");

ln=0;

}

} while (!ln);

return ln;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\* Уничтожение элемента списка \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

f\_delete () {

int ln;

emlp \*a, \*b;

/\* Если списка нет в памяти,то вывод соотвтствуюшего

сообщения \*/

if (emlph==NULL) printf("List empty\n");

/\* Иначе-ввод номера элемента с помощью функции GET\_LN \*/

else {

ln=get\_ln()-1;

if (!ln) {

/\* Если номер корректен - переприсваивание указателей

и освобождение памяти \*/

a=emlph; emlph=a->next; free(a);

}

else {

/\* Иначе- ??????? \*/

for(ln--, a=emlph; ln&&(a!=NULL); a=a->next,ln--);

if (a!=NULL)

if ((b=a->next)!=NULL) {

a->next=b->next; free(b);

}

}

}

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\* Изменение значения полей элемента списка \*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

f\_change() {

char ss[40];

int ln;

emlp \*a;

ln=get\_ln()-1; /\* Ввод номера элемента \*/

for (a=emlph; ln && a!=NULL; ln--, a=a->next);

if (ln) return 0;

/\* Вывод старых и ввод новых значений \*/

/\* Запись новых значений в список \*/

printf("Old name = %s New name >",a->name);

gets(ss);

gets(ss);

if (\*ss) strcpy(a->name,ss);

printf("Old grade = %d New grade >",a->grade);

gets(ss);

if (\*ss) sscanf(ss,"%d",&(a->grade));

printf("Old hight = %d New hight >",a->hight);

gets(ss);

if (\*ss) sscanf(ss,"%d",&(a->hight));

return 0;

}