

## Глава 6

# Компьютер ведет учет

---

В повседневной жизни мы уже привыкли, что компьютер широко используется в различных учреждениях и фирмах для учета товаров, пассажиров различных видов транспорта, правонарушителей и т.д., и т.п. Для этих целей используются специальные среды программирования различных баз данных, с помощью которых в специализированных фирмах опытными программистами создаются довольно сложные специальные программы, предназначенные для всевозможных видов учета.

Но очень часто в повседневной жизни рядового пользователя возникает необходимость создать свою, небольшую программу учета, не прибегая к изучению сложного программирования баз данных. В этой главе на небольшом примере будет показано то, как можно простыми средствами с использованием среды программирования Borland Turbo C/C++ создать свою программу для учета имеющихся у вас собственности компакт-дисков. Среда программирования Turbo C/C++ предназначена для работы под управлением операционной системы MS DOS, следовательно, и созданная в этой среде программа также будет предназначена для работы под управлением операционной системы MS DOS.

Владельцев современных компьютеров, которые имеют смутные понятия о возможностях этой, как бы «старой» операционной системы, хочу успокоить — практически любую из программ MS DOS можно запустить в работу на любом современном компьютере.

Следует знать, что операционная система Microsoft Windows, которая сейчас установлена у большинства пользователей, сначала задумывалась для приобщения к компьютеру домохозяек и других малоквалифицированных пользователей. Затем, по мере совершенствования, эта ОС прочно заняла первое место в сфере обслуживания офисов — подготовка текстов, презентаций, просмотр рисунков и фильмов, прослушивание музыкальных программ.

В то же время все «более серьезные» программы, как правило, разрабатываются и работают в операционных системах UNIX, Linux, MS DOS и других.

О том, как работать с программами MS DOS на современном компьютере смотрите в приложении 1.

## Учет компакт-дисков

Почему нас должен интересовать учет компакт-дисков?

Не волнуйтесь! Любого человека, в том числе и компьютерного пользователя, всегда интересовал, и всегда будет интересовать учет самых разных предметов. В данном случае в качестве примера, создаваемого с учебными целями, автору захотелось взять именно компакт-диски. Далее, в конце этой главы, вы прочитаете о том, что по образцу и подобию программы для учета компакт-дисков можно сде-

лать программу для учета радиосвязей, проведенных на любительской радиостанции и много других аналогичных программ.

Приведенные ниже тексты исходных кодов программы `CD_uchet` разрабатывались в среде программирования Borland Turbo C/C++ v.3. Программа создана для демонстрационных и учебных целей, но все входящие в ее состав функции можно с успехом использовать в своих собственных разработках. Особенно полезной программа может быть для начинающих программировать на C/C++, т.к. в ней приведены практически все основные команды, операторы и функции, применяющиеся при создании компьютерных программ на языке программирования C/C++.

Для учета дисков в данном случае используется система учета с применением учетных карточек, на которых располагается вся необходимая информация о каждом из компакт-дисков. Подобная система учета с использованием специальных карточек и сейчас еще применяется, например, для учета больных в районных поликлиниках.

В листинге 6.1 приведено начало главного файла программы, который создан на языке СИ с некоторыми элементами СИ++.

**Листинг 6.1.** Начало файла `cdapp2.c`

---

```
//-----
/* CDapp.c — файл программы CD_uchet*/
#include <stdio.h>
#include <conio.h>
#define FILENAME "cdfile"      // файл с записями карточек
#define TEMPFILE "temp"       // Вспомогательный файл
#define MAX 20                 // Наибольшее количество дисков
FILE *fp, *tp, *printer;
struct CD
{
    char name[20];              // Название карточки (имя диска)
    char description[40];       // Описание особенностей диска
    char category[12];          // Категория
    float cost;                 // Стоимость диска
    int number;                 // Порядковый номер карточки
} disc;
int slots[MAX];               // Максимальное число карточек в контейнере
int count;
char select;
//-----
main()
{
    textbackground(15);
    textcolor(0);
    getslots();
    do
    {
```

```

clrscr();
puts("\n\n\t\t\t\t\tМоя коллекция дисков");
puts("\t *****\n");
puts("\t1 - добавить карточку");
puts("\t2 - удалить карточку");
puts("\t3 - редактировать содерж. карточки");
puts("\t4 - изменить номер ячейки");
puts("\t5 - сортировка карточек");
puts("\t6 - найти карточку");
puts("\t7 - вывести на печать");
puts("\t8 - выйти из программы\n\n");
printf(" \t Сделайте Ваш выбор: ");
select = getchar();
putchar('\n');
switch(select)
{
case '1' : add_cd(); break;
case '2' : del_cd(); break;
case '3' : ch_cd(); break;
case '4' : ch_loc(); break;
case '5' : sort(); break;
case '6' : locate(); break;
case '7' : pr_list(); break;
case '8' : break;
default:
puts(" Ошибка, повторите ввод\n\n");
}
}
while(select != '8');
return(0);
}
//-----

```

---

## Функция main()

Все основные задачи в программе выполняются созданными собственными функциями. Функция `main()` позволяет выбрать задачу, которую предстоит выполнить, и вызвать для выполнения соответствующую функцию. Наиболее удобным способом выбора задачи является указание одного из пунктов представленного на экране меню. Таким образом, функция `main()` должна выполнять следующие операции:

- ◆ предлагать меню до тех пор, пока пользователь не выберет пункт, прекращающей работу программы;
- ◆ выводить на экран список задач;

- ♦ принимать вводимое с клавиатуры значение выбранного пункта меню;
- ♦ вызывать функцию, выполняющую выбранную задачу;
- ♦ завершать программу, если пользователь сделал соответствующий выбор.

Для получения выбора из меню в функции `main()` нужно определить специально предназначенную для этого локальную переменную. При выборе одного из пунктов меню с одной из кнопок функция `main()` будет присваивать соответствующее значение этой переменной и в дальнейшем использовать его для определения функции, которую следует вызвать.

Далее, прежде чем предоставить пользователю меню, мы должны знать, какие ячейки в нашем виртуальном контейнере уже заняты. Чтобы определить это, вызываем функцию `getslots()`. Эта функция открывает файл с записями, читает все записи и присваивает каждое считанное значение переменной `disc.number` элементам массива `slots[]`, а также подсчитывает общее количество записей, внесенных в файл.

Вывод меню на экран необходимо повторять каждый раз после выполнения любой функции, вызываемой из меню. Если этого не сделать, выполнение программы завершится после выполнения одной функции. В этом случае, если вы захотите выполнить две задачи, например, добавить новую «карточку» и вывести на печать обновленный список, вам придется запустить программу дважды. Так как вам не известно точное количество повторов меню, мы используем цикл `do`.

Меню будет иметь восемь пунктов, по одному на каждую основную задачу, плюс восьмой для завершения программы. Для вывода текстовых строк меню на экран используются функции `puts()` или `printf()`, а ввод ответа, означающего выбор одного из пунктов, осуществляется в помощью функции `getchar()`. Использование `getchar()` позволяет избежать необходимости нажатия клавиши <Enter> при выборе каждой задачи. Если для выбора использовать цифры, а не буквы, мы тем самым избавляем себя от необходимости учитывать регистр вводимого символа.

После того как выбор пункта меню сделан, `main()` должна вызвать соответствующую функцию. Восемь вариантов выбора требуют использования семи инструкций `if`. Для того чтобы работу программы было легче контролировать, а текст ее было легче читать, имеет смысл использовать переключатель `switch`, который будет проверять введенное значение переменной. Помните о необходимости включить ветвь `default` на случай, если пользователь введет цифру или букву, не предусмотренную в меню.

Цикл `do...while` повторяет меню (в данном случае инструкции `switch`) до тех пор, пока в ответ на предложение сделать выбор пользователь не введет цифру 8, завершающую работу программы. После того, как пользователь сделает выбор и выбранная задача будет завершена, меню снова появится на экране и можно будет попросить программу выполнить другую задачу.

На рис. 6.1 приведено рабочее окно программы, на котором изображено меню.

В листинге 6.2 приведен текст функции `add_cd()`, которая служит для заполнения учетной карточки необходимыми данными.

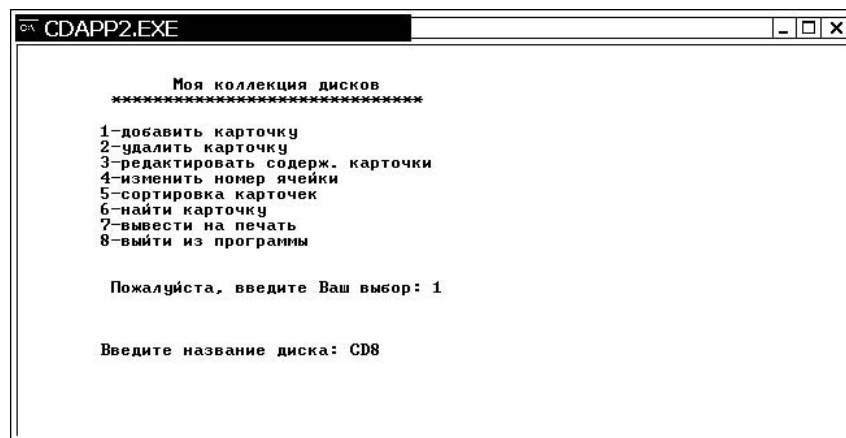


Рис. 6.1. Рабочее окно программы

**Листинг 6.2.** Продолжение 1 файла *cdapp2.c*

```
//-----
add_cd()
{
    if(count >= MAX)
    {
        puts("Свободных ячеек в контейнере нет!\n");
        getchar(); // Ожидание нажатия любой клавиши
        return;
    }
    if((fp = fopen( FILENAME, "a")) == NULL)
    {
        printf("Невозможно открыть файл %s\n", FILENAME);
        getchar(); // Ожидание нажатия любой клавиши
        exit();
    }
    puts("\n");
    printf("\t Введите название диска: ");
    gets(disc.name);
    printf("\t Введите описание особенностей: ");
    gets(disc.description);
    printf("\t Введите категорию: ");
    gets(disc.category);
    printf("\t Введите стоимость диска: ");
    scanf(" %f ", &disc.cost);
    delay(100); // Задержка длительностью 100 мс
    printf("\n\t Последняя занятая ячейка - %d", disc.number);
    getslot(); // Вводится номер карточки
    fwrite(&disc, sizeof(disc), 1, fp);
}
```

```

    fclose(fp); // Закрыть файл с данными
    getslots(); // Установить новый номер карточки
    return;
}
//-----

```

## Функция add\_cd()

Функция `add_cd()`, текст которой расположен в предыдущем листинге, добавляет в контейнер информацию о компакт-диске. Функция построена по шаблону, который обычно используется для добавления данных в файл.

В нашей программе пользователь не может добавить информацию о новом диске, если в контейнере нет места, куда бы можно было поместить карточку с данными об этом диске. Функция `getslots()` подсчитывает количество дисков, уже имеющихся в контейнере. Когда их общее число достигает максимального значения, на экране появляется соответствующее сообщение и программа возвращается в меню.

Далее, в нашей программе файл открывается с режимом доступа «а», поэтому файл будет создан даже в том случае, если он не существовал к моменту запуска программы.

Если файл успешно открыт или вновь создан, данные будут добавлены в его конец.

Следующая функция просит пользователя ввести номер ячейки от 1 до 20. Так как ввод этой информации необходим при выполнении нескольких задач в программе, мы выделили его в отдельную функцию, которая называется `getslots()` и может быть вызвана при необходимости.

Функция `getslots()` не позволяет вводить номер уже занятой ячейки. После записи в файл информации о новом компакт-диске происходит вызов функции `getslots()`, которая обновляет массив номеров ячеек. Текст этих двух функций располагается ниже. На рис. 6.2 приведено рабочее окно программы, в котором показана процедура добавления нового диска с названием CD8.

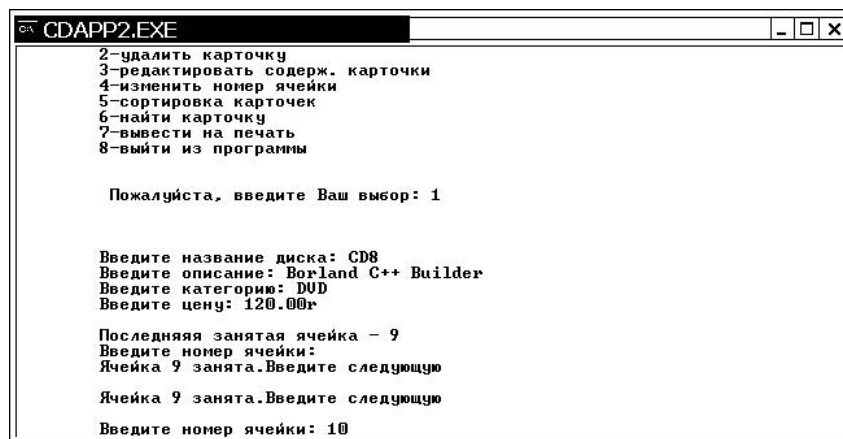


Рис. 6.2. Ввод записи о новом диске

В листинге 6.3 располагается текст функции `del_cd()`, которая служит для удаления учетной карточки.

**Листинг 6.3.** Продолжение 2 файла *cdapp.c*

---

```
//-----
del_cd()
{
    char delname[20];
    char fflag;
    fflag='n';
    openrw();
    puts("\tУдаление информации о диске\n");
    gets(delname);
    printf("\tВведите название диска: ");
    gets(delname);
    while(fread(&disc, sizeof(disc), 1, fp)==1)
    {
        if(strcmp(disc.name, delname) != 0)
            fwrite(&disc, sizeof(disc), 1, tp);
        else
            fflag='y';
    }
    fclose(fp); // Закрыть файл с данными
    fclose(tp); // Закрыть вспомогательный файл
    if(fflag=='n')
        nofind(); // Сообщить, что диск не найден
    else
    {
        openwr();
        while(fread(&disc, sizeof(disc), 1, tp)==1)
            fwrite(&disc, sizeof(disc), 1, fp);
        fclose(fp);
        fclose(tp);
    }
    getslots();
    return;
}
//-----
```

---

## Функция `del_cd()`

Функция, удаляющая информацию о диске, использует стандартный алгоритм для работы с последовательными файлами. Используя последовательный доступ,

мы не можем непосредственно перейти к интересующему нас месту в файле, чтобы изменить содержащуюся там запись. Если открыть файл с режимом доступа «w», его содержимое будет уничтожено. При использовании режима доступа «a» мы сможем только добавить данные в конец файла.

Решение этой проблемы лежит в использовании двух файлов. Функция `del_cd()` открывает файл с данными при режиме доступа «r», и временный файл с режимом доступа «w».

После этого вводим название диска, который следует удалить.

Цикл `while` осуществляет чтение каждой структуры (записи) из исходного файла.

Если запись не является той, которую пользователь хочет удалить, функция `del_cd()` заносит ее во временный файл.

В нашей программе функция `strcmp()` используется в нескольких местах для определения того, прочитана ли из файла нужная запись. Функция `strcmp()` не входит в стандарт Си, но она есть во всех библиотеках ANSI Си и Си++.

Встретив нужную запись, функция `del_cd()` не копирует ее во временный файл, а вместо этого устанавливает флаг, чтобы впоследствии узнать, была ли структура удалена.

Цикл `while` продолжает копирование из одного файла в другой до тех пор, пока не достигнет конца файла, после чего оба файла закрываются.

Если функция `del_cd()` не нашла записи, которую пользователь хочет удалить, она выведет на экран соответствующее сообщение, после чего завершает работу.

Произведя удаление записи, вы получите два разных файла. Исходный файл содержит все данные, включая и информацию, которую вы хотите удалить. Временный файл содержит только данные, которые требовалось сохранить, так как функция `del_cd()` не записала в него структуру, подлежащую удалению. Но мы не можем оставить данные в таком виде, программа способна вносить дополнительные данные и выполнять другие функции лишь с исходным файлом, поэтому мы должны привести его содержимое в соответствие с полученным результатом.

Одни компиляторы Си++ поставляются с библиотечными функциями, позволяющими изменять имена файлов, но в других компиляторах они отсутствуют. В общем, мы должны снова открыть оба файла, теперь уже с обратным порядком доступа, т. е. исходный файл открывается с режимом доступа «w», после чего данные из него будут удалены, а временный файл открывается с режимом доступа «r». Для этого используется функция `openwr()`, вызываемая функцией `del_cd()` и другими функциями.

Когда файлы открыты, функция `del_cd()` считывает записи из временного файла и переносит их в исходный файл. После закрытия файлов, исходный файл содержит уже исправленную информацию, не включающую данные, которые мы хотели удалить.

Перед завершением функции `del_cd()` программа вызывает функцию `getslots()`, обновляющую массив номеров ячеек.

Итак, теперь у нас есть два файла, которые содержат одинаковые данные. Единственный недостаток этой процедуры состоит в том, что диск компьютера должен иметь достаточно свободного места для хранения двух файлов с данными. Временный файл можно рассматривать как резервную копию. Если с исходным файлом



произойдет что-то нежелательное, остается временный файл, содержащий данные в том виде, в каком они были до выполнения последней операции. Если вы по каким-либо причинам не хотите хранить временный файл на диске, то можете открыть его с режимом доступа «w», а потом немедленно закрыть.

В результате файл останется на диске, но совершенно пустой, так что он займет очень мало дискового пространства.

Можно несколько усилить процедуру, если выводить на экран указанную запись и просить пользователя дать подтверждение, что эту запись действительно необходимо удалить. Если пользователь решит не удалять запись, программа должна скопировать ее во временный файл.

На рис. 6.3 показан момент удаления диска CD5.

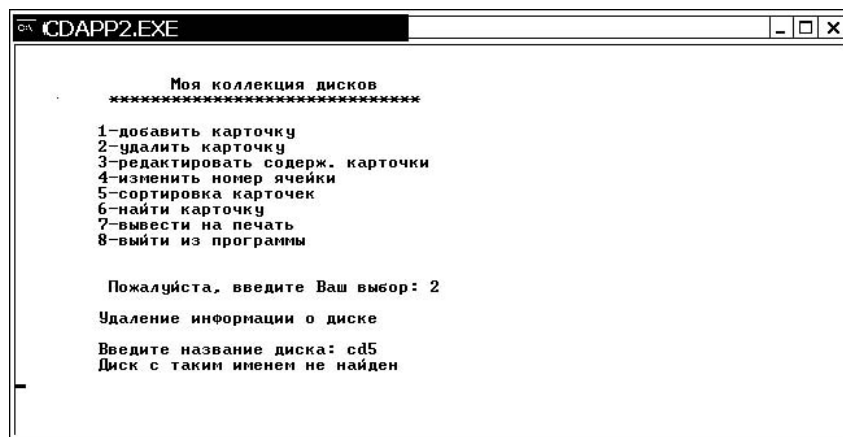


Рис. 6.3. Пример с удалением диска

В листинге 6.4 приведено продолжение файла `cdapp.c`, в котором представлены исходные коды функции `ch_cd()`.

Листинг 6.4. Продолжение 3 файла `cdapp.c`

```
//-----
ch_cd()
{
    char chname[20];
    char fflag;
    fflag='n';
    openrw();
    puts("\tРедактирование сведений о диске\n");
    gets(chname);
    printf("\tВведите название диска : ");
    gets(chname);
    while(fread(&disc, sizeof(disc), 1, fp)==1)
    {
        if(strcmp(disc.name, chname) != 0)
```

```
        fwrite(&disc, sizeof(disc), 1, tp);
    else
    {
        fflag='y';
        puts("\tТекущая информация\n");
        showdisc();
        puts("\tНовая информация\n");
        printf("\tВведите название диска : ");
        gets(disc.name);
        printf("\tВведите описание: ");
        gets(disc.description);
        printf("\tВведите категорию: ");
        gets(disc.category);
        printf("\tВведите цену: ");
        scanf(" %f ", &disc.cost);
        if(count>=MAX)
        {
            puts("\tСвободных мест в контейнере нет\n");
            getchar();
        }
        else
        {
            getslot();
        }
        fwrite(&disc, sizeof(disc), 1, tp);
    }
}
fclose(fp);
fclose(tp);
if(fflag=='n')
    nofind();
else
{
    openwr();
    while(fread(&disc, sizeof(disc), 1, tp)==1)
        fwrite(&disc, sizeof(disc), 1, fp);
    fclose(fp);
    fclose(tp);
}
getslots();
return;
}
//-----
```

## Функция `ch_cd()`

Для внесения изменений в картотеку используется тот же основной алгоритм, что и для удаления записи: вся обновленная информация записывается во временный файл, а затем копируется обратно в файл данных. В этом случае, вместо того, чтобы игнорировать записи, которые мы хотим изменить, в них записывается новая информация, а уже затем они сохраняются во временном файле.

Прежде всего, функция `ch_cd()` запрашивает название компакт-диска, в который вы хотите внести изменения, затем выполняет цикл `while`, в котором считывается каждая структура.

Затем, если очередную структуру редактировать не следует, функция записывает ее во временный файл.

Когда функция встречает искомую запись, она выводит на дисплей текущие данные, а потом дает подсказку для ввода новой информации.

Инструкция `if (count>=MAX)` не позволяет пользователю ввести новый номер ячейки, если файл содержит максимально допустимое количество записей. В этом случае функция `getslot()` не выполнится и исходный номер ячейки будет прочитан с диска и занесен в конец заново, даже если он захочет изменить всего один пункт. Вы можете самостоятельно отредактировать информацию более удобным для пользователя способом, изменив ее так, чтобы он смог ограничиться простым нажатием клавиши `<Enter>` в том случае, если необходимо сохранить текущее содержимое какого-нибудь пункта.

Если пользователь вводит новое название, оно присваивается переменной `disc.name`, после чего заносится в файл вместе с соответствующей записью. Если пользователь нажимает клавишу `<Enter>`, не печатая нового названия, содержимое `disc.name` остается без изменений и в записи сохраняется прежнее имя диска.

Поскольку нам необходимо выводить информацию на экран несколько раз во время работы программы, процедуру вывода можно выделить в самостоятельную функцию, названную нами `showdisc()`, и вызывать ее по мере надобности.

После чтения новых данных функция `ch_cd()` записывает структуру во временный файл.

После завершения чтения файла `fp` функция `ch_cd()` закрывает оба файла. Если ни одна запись не была отредактирована (например, потому что не был найден указанный диск), функция выводит на экран соответствующее сообщение и останавливается.

Если изменения были внесены, функция `ch_cd()` снова открывает файлы с обратным порядком доступа, переписывает данные в исходный файл данных и обновляет массив номеров ячеек.

В листинге 6.5 приведены исходные коды следующей функции из файла `cdapp.c`. Это функция изменения кода ячейки `ch_loc()`.

**Листинг 6.5.** Продолжение 4 файла `cdapp.c`

---

```
//-----
ch_loc()
{
    char chname[20];
```

```
char fflag;
fflag='n';
if(count>=MAX)
{
    puts("\Свободных ячеек в контейнере нет\n");
    getchar();
    return;
}
openrw();
puts("\tИзменение номера ячейки\n");
gets(chname);
printf("\tВведите название диска : ");
gets(chname);
while(fread(&disc, sizeof(disc), 1, fp)==1)
{
    if(strcmp(disc.name, chname)!=0)
        fwrite(&disc, sizeof(disc), 1, tp);
    else
    {
        fflag='y';
        puts("\tТекущая информация\n");
        showdisc();
        puts("\n\tНовый номер ячейки\n");
        getslot();
        fwrite(&disc, sizeof(disc), 1, tp);
    }
}
fclose(fp);
fclose(tp);
if(fflag=='n')
    nofind();
else
{
    openwr();
    while(fread(&disc, sizeof(disc), 1, tp)==1)
        fwrite(&disc, sizeof(disc), 1, fp);
    fclose(fp);
    fclose(tp);
}
getslots();
return;
}
```

//-----

## Функция `ch_loc()`

Функция `ch_loc()`, которая используется для изменения номера ячейки в карточке компакт-диска, в сущности, не отличается от функции редактирования записи, за исключением процедуры ввода номера ячейки.

Однако в начало функции добавлена инструкция `if`, чтобы избежать изменения номера ячейки в том случае, если в контейнере нет свободных ячеек.

В листинге 6.6 приведены исходные коды функции `locate()`.

**Листинг 6.6.** Продолжение 5 файла *cdapp.c*

---

```
//-----
locate()
{
    char name[20];
    char fflag;
    fflag='n';
    if((fp= fopen(FILENAME, "r")) == NULL)
    {
        printf("Невозможно открыть файл %s\n", FILENAME);
        exit();
    }
    puts("\tПоиск диска \n");
    gets(name);
    printf("\tВведите название диска : ");
    gets(name);
    while(fread(&disc, sizeof(disc), 1, fp)==1)
    {
        if(strcmp(disc.name, name)==0)
        {
            fflag='y';
            showdisc();
            printf("\tДля продолжения нажмите клавишу <Enter>");
            getchar();
            putchar('\n');
        }
    }
    fclose(fp);
    fclose(tp);
    if(fflag=='n')
        nofind();
    return;
}
//-----
```

## Функция locate()

Для того чтобы вывести на экран определенную запись, следует ввести название диска, открыть файл для чтения и считывать записи до тех пор, пока не будет найдена нужная.

Функция `locate()` вызывает функцию `showdisc()`, выводящую информацию на экран, затем временно останавливает выполнение программы с тем, чтобы дать пользователю время прочитать представленные данные.

Функция `showdisc()` читает файл целиком, поэтому на экран будут выведены все записи с одинаковыми наименованиями.

На рис. 6.4 показан рабочий экран программы в случае поиска информации о диске CD2.

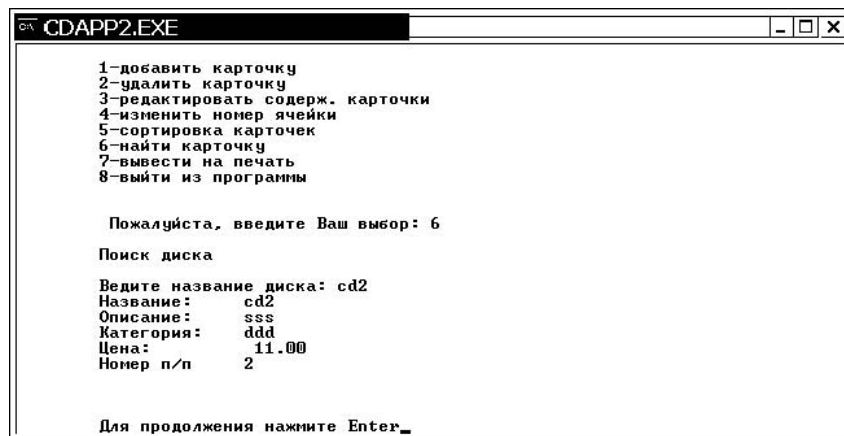


Рис. 6.4. Информация о диске CD2

В листинге 6.7 представлены исходные коды функции `pr_list()`.

### Листинг 6.7. Продолжение 6 файла *cdapp.c*

```
//-----
pr_list()
{
    if((fp = fopen(FILENAME, "r")) == NULL)
    {
        printf("\tНевозможно открыть файл %s\n", FILENAME);
        exit();
    }

    if((printer = fopen("prn", "w")) == NULL)
    {
        printf("\tПечатающее устройство не готово к работе\n");
        fclose(fp);
        exit();
    }
}
```

```

while(fread(&disc,sizeof(disc), 1, fp)==1)
{
    fprintf(printer, "\tНазвание:   %s\n", disc.name);
    fprintf(printer, "\tОписание:  %s\n", disc.description);
    fprintf(printer, "\tКатегория: %s\n", disc.category);
    fprintf(printer, "\tЦена :      %6.2f\n", disc.cost);
    fprintf(printer, "\tНомер п/п: %d\n", disc.number);
    fprintf(printer, "\n\n");
}
fclose(printer);
fclose(fp);
return;
}
//-----

```

---

## Функция pr\_list()

Для того чтобы вывести на печать каталог коллекции компакт-дисков, программа должна открыть два файла: файл данных должен быть открыт для чтения, а файл принтера, для которого используется стандартное имя системы MS DOS «prn», должен быть открыт для записи.

Функция `pr_list()` читает каждую запись и затем печатает данные с помощью функции `fprintf()`.

В листинге 6.8 приведены исходные коды функции `sort()`.

### Листинг 6.8. Продолжение 7 файла *cdapp.c*

```

//-----
sort()
{
    struct CD temp[MAX];
    int index, loop1, loop2, endloop;
    loop1=0;
    loop2=0;
    endloop=0;
    index = 0;
    if((fp = fopen(FILENAME, "r")) == NULL)
    {
        printf("\tНевозможно открыть файл %s\n", FILENAME);
        exit();
    }
    while(fread(&disc, sizeof(disc), 1, fp)==1)
    {

```

```
        temp[index]=disc;
        index++;
    }
    fclose(fp);
    if((fp = fopen(FILENAME,"w")) == NULL)
    {
        printf("\tНевозможно открыть файл %s\n",FILENAME);
        exit();
    }
    for(loop1=1;loop1<MAX+1;loop1++)
    {
        for(loop2=0;loop2<count;loop2++)
            if(temp[loop2].number==loop1)
            {
                fwrite(&temp[loop2],sizeof(temp[loop2]), 1, fp);
                endloop++;
            }
        if(endloop==count)
            break;
    }
    fclose(fp);
    return;
}

//-----
```

---

## Функция sort()

Добавляя сведения о новом диске в картотеку, вы можете поместить его в ячейку, которая имеет номер не обязательно, следующий по порядку за номером ячейки предыдущего компакт-диска. Может случиться, что вы захотите оставить несколько ячеек пустыми, предназначая их для определенных дисков, которые еще не куплены. Иными словами, добавляя записи в файл, вы можете вводить их в порядке, который не соответствует их расположению в контейнере.

При выводе на принтер карточки дисков будут напечатаны в том порядке, в котором вы их вводили в файл, и для того чтобы найти диск по его номеру ячейки, придется просмотреть весь список.

Сортировка записей в данном случае означает, что карточки дисков будут переписаны таким образом, чтобы они хранились в порядке возрастания номеров ячеек. Диск, помещенный в первую ячейку, станет первым в списке, диск из второй ячейки — вторым и т. д.

Существуют буквально десятки способов сортировки содержимого файла. Некоторые алгоритмы включают чтение данных в массив, сортировку элементов массива в памяти и запись массива обратно на диск. Другие алгоритмы используют два



или более файлов, в которые данные заносятся блоками, а потом опять переписываются в один файл, но уже в другом порядке.

В приведенной здесь функции `sort()` применен алгоритм, который требует участия только одного файла, но зато использует массив структур. Для работы функции необходимо определить массив структур и несколько переменных.

Функция `sort()` открывает файл данных для чтения, считывает данные в массив и закрывает файл.

Переменная `index` используется в качестве индекса массива.

Далее, функция `sort()` снова открывает файл, уже для записи, и заносит в него данные, используя для этого два цикла `for`.

Внешний цикл `for` увеличивает значение переменной `loop1` от 1 до максимального допустимого количества дисков. Во время первого прохождения внешнего цикла, внутренний цикл `for` просматривает массив в поисках диска, занимающего ячейку с номером 1. Как только этот диск найден, описывающая его структура записывается в файл и внутренний цикл завершается. После этого повторяется выполнение внешнего цикла, и внутренний цикл снова просматривает массив, теперь уже в поисках диска в ячейке с номером 2. Как только соответствующий диск находится, данные о нем тут же записываются в файл. Этот процесс повторяется для всех имеющихся в наличии номеров ячеек.

Как только данные обо всех дисках оказываются записанными обратно в файл, необходимость дальнейшего поиска номеров ячеек отпадает. Мы использовали переменную `endloop` для запоминания количества структур, записанных в файл. После записи очередной структуры, функция `sort()` увеличивает значение `endloop` на единицу. Когда значение переменной `endloop` становится равным максимальному допустимому количеству компакт-дисков в коллекции, это означает, что все структуры уже записаны в файл и повторение циклов может быть прекращено.

После чего файл закрывается и выполнение функции `sort()` завершается.

В листинге 6.9 приведены исходные коды функций `showdisk()`, `nofind()`, `openrw()` и `openwr()`.

#### Листинг 6.9. Продолжение 8 файла *cdapp.c*

```
//-----
showdisc()
{
    printf("\tНазвание:      %s\n", disc.name);
    printf("\tОписание:       %s\n", disc.description);
    printf("\tКатегория:        %s\n", disc.category);
    printf("\tЦена :             %6.2f\n", disc.cost);
    printf("\tНомер п/п         %d\n", disc.number);
    puts("\n\n");
    return;
}
//-----
nofind()
```

```
    {
// char pause;
    puts("\tДиск с таким именем не найден");
    getchar();
    return;
    }
//-----
openrw()
{
    if((fp = fopen(FILENAME, "r")) == NULL)
    {
        printf("\tНевозможно открыть файл %s\n", FILENAME);
        exit();
    }
    if((tp = fopen(TEMPFILE, "w")) == NULL)
    {
        printf("\tНевозможно открыть файл %s\n", TEMPFILE);
        fclose(fp);
        exit();
    }
    return;
}
//-----
openwr()
{
    if((fp = fopen(FILENAME, "w")) == NULL)
    {
        printf("\tНевозможно открыть файл %s\n", FILENAME);
        exit();
    }
    if((tp = fopen(TEMPFILE, "r")) == NULL)
    {
        printf("\tНевозможно открыть файл %s\n", TEMPFILE);
        fclose(fp);
        exit();
    }
    return;
}
//-----
```

---

Функция `showdisc()` выводит на экран данные о выбранном диске.

Функция `nofind()` выводит на экран соответствующее сообщение, если указанная пользователем запись не найдена.

Функция `openwr()` открывает файл с данными для чтения, а временный файл для записи.

Функция `openrw()` открывает файл с данными для записи, а временный файл для чтения.

В листинге 6.10 приведены исходные коды двух последних функций `getslot()` и `getslots()` из файла `cdapp.c`.

---

**Листинг 6.10.** Продолжение 9 файла *cdapp.c*

---

```
//-----
getslot()
{
    int index, flag;
    do
    {
        flag=0;
        printf("\n\tВведите номер ячейки: ");
        scanf(" %d", &disc.number);
        getchar();
        for(index=0;index<count;index++)
        {
            if(slots[index]==disc.number)
            {
                printf("\n\tЯчейка %d занята. Попробуйте другую\n", disc.number);
                flag=1;
            }
        }
    }
    while(disc.number<1 || disc.number>MAX || flag==1);
    count++;
    slots[count]=disc.number;
    return;
}

//-----
getslots()
{
    int index;
    index=0;
    count=0;
    if((fp = fopen(FILENAME, "r")) != NULL)
    {
        while(fread(&disc, sizeof(disc), 1, fp)==1)
        {
            slots[index]=disc.number;

```

```

        index++;
        count++;
    }
    fclose(fp);
}
}
//-----

```

Функция `getslot()` не позволяет вводить номер уже занятой ячейки. После записи в файл информации о новом компакт-диске происходит вызов функции `getslots()`, которая обновляет массив номеров ячейки.

## Учет радиосвязей на компьютере

Выше уже сообщалось о том, что на базе описанной ранее в этой главе учебной программы можно с успехом создать свою собственную программу, осуществляющую учет и хранение данных об одинаковых предметах или явлениях.

В 1996 году была разработана программа `QSO_log`, предназначенная для учета радиосвязей, проведенных на любительской радиостанции за какой-то определенный период времени.

В настоящее время существует огромное количество самых разнообразных аналогичных компьютерных программ, практически все они очень хорошо работают и достойны похвалы. Но большинство из них созданы зарубежными радиолюбителями-программистами и имеют документацию не на русском языке, что может вызвать затруднение у большого числа радиолюбителей в правильной эксплуатации этой программы. Программа большая и хорошая, но может использоваться только на 5–10% от своих возможностей. Кроме того, многие наши российские разработчики программ делают на английском языке документацию для своих программ, хотя заранее известно, что зарубежному пользователю такая программа совершенно не нужна.

Программа `QSO_log` небольшая по размерам и простая в обращении, вся документация к ней выполнена на русском языке. Особенно удобной эта программа может быть для радиолюбителей, которые часто участвуют в различных региональных соревнованиях по радиосвязи, таких как областные, межобластные или зональные соревнования. После проведения соревнований с помощью этой программы можно создать текстовый файл с перечнем всех проведенных радиосвязей, выполненный по требованиям судейской коллегии, т.е. в так называемом формате CABRILLA.

Не буду приводить здесь исходные коды этой программы, вы их можете найти на прилагаемом к книге компакт-диске или в Internet на сайте `r3xb.nm.ru`.

Ниже приведено описание основных свойств программы `QSO_log`.

## О программе `QSO_log`

Программа предназначена для хранения перечня наиболее важных радиосвязей на простых IBM PC или совместимых компьютерах под управлением MS DOS, ко-

торые имеют небольшие размеры жесткого диска, или на которых используются только гибкие дискеты. Сравнивать эту программу с мощными программами под управлением операционных систем Microsoft Windows или MS DOS, имеющими мегабайтные размеры, не стоит.

Однако, несмотря на свои исключительно малые размеры, программа обеспечивает некоторый сервис, зачастую достаточный для радиолюбителя, и работает надежно. Разработана она на языке программирования Turbo C++ v.3.

Программа распространяется свободно среди радиолюбителей-коротковолновиков по принципу «как есть», т.е. без каких либо гарантий со стороны автора-разработчика и никаких претензий к автору.

В комплект программы входят следующие файлы:

- ♦ `qso_log2.exe` — собственно файл программы (запускаемый);
- ♦ `qso_log2.dat` — файл хранения базы данных.

Программа позволяет:

- ♦ сохранять в файле `qso_log2.dat` записи данных по проведенным радиосвязям (базу данных);
- ♦ выводить на экран перечень всех записанных эфиров радиосвязей при некотором сокращении выведенных данных — выводятся только основные параметры;
- ♦ распечатывать на принтере основные данные по всем записям;
- ♦ осуществлять поиск записи радиосвязи с определенным корреспондентом, при этом на запрос программы позывного можно вводить только префиксную часть позывного или даже только одну первую букву искомого позывного;
- ♦ просматривать полные данные по любому позывному;
- ♦ создавать файл отчета по соревнованиям в формате CABRILLA, и т.д.

Все файлы комплекта должны находиться в одном каталоге. При этом никаких особенностей при эксплуатации программы не выявлено. Максимальное число записей в одном файле базы данных равно 400. Для сохранения следующих записей следует старый файл данных переименовать или хранить в другой директории, а для работы создать новый файл под именем `qso_log2.dat`.

Программа `QSO_log` может обрабатывать файлы с записями учитываемых радиосвязей, выполненных программами `CW_qso` и `RTTY_qso`, предназначенными для проведения любительских радиосвязей телеграфом и телетайпом. Эти программы вы также сможете найти в Internet на указанном выше сайте.