СибГУТИ
Кафедра ВС
Программирование на языке высокого уровня (ПЯВУ), семестр 2
2009-2010 учебный год
Поляков А. Ю.

Лабораторная работа №8. Обработка таблиц. Файловый ввод/вывод.

Цель работы: Разработать программное обеспечение, выполняющее обработку табличных данных. Содержимое таблиц должно сохраняться в файле перед выходом из программы и восстанавливаться при запуске.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Представление табличных данных в ЭВМ

Для хранения табличной информации в памяти обычно используются структуры (или записи). Структурой называется набор из одной или более переменных, возможно различных типов, сгруппированных под одним именем для удобства обработки.

Описание структуры, состоящее из заключенного в фигурные скобки списка описаний, начинается с ключевого слова struct. За словом struct может следовать необязательное имя, называемое ярлыком (тэгом) структуры. Такой ярлык именует структуры этого вида и может использоваться в дальнейшем как сокращенная запись подробного описания. Элементы структуры называются ее полями. Формат определения следующий:

```
struct [<тэг>] {
<тип данных1> <имя поля1>;
<тип данных2> <имя поля2>;
.....
<тип данныхN> <имя поляN>;
} [ <имя переменной1> [,<имя переменной2> [, ...]] ];
```

Листинг 1

Например, информация о человеке (Фамилия, имя, возраст, пол, стаж работы, ...) может быть представлена в виде структуры следующим образом:

```
struct people_info{
char sname[64];   // Фамилия (максимум 63 символа)
char name[64];    //  Имя (максимум 63 символа)
unsigned int age; // Возраст – целое число, например 21 год
char pol;         // Пол – один символ: 'm' или 'w'.
float exp;        // Опыт (стаж) работы, например 2,5 года
} a;
....
struct people_info b;
```

Листинг 2

В данном примере объявлено 2 переменные (*a* и *b*), причем *a* объявлена одновременно с описанием структуры, а *b* — в произвольном месте программы, где допускается объявление переменных. При описании переменной *b* (как и при описании обычной переменной) сначала задается тип данного (*struct people_info*) — он состоит из служебного слова *struct* и тэга структуры *people_info*. Далее указывается имя переменной, содержащей структуру. Для именования структур используются те же правила что и для именования переменных базовых типов (*int*, *float*, *char* и т.д.).

К переменной типа структура допустимо применение следующих операций:

1. присваивание переменных-структур переменным того же типа;
2. взятие адреса структуры;

3. обращение к элементам (полям) структуры;
4. применение операции sizeof для определения ее размера.

Обращение к полю структуры

Для обращения к полю структуры в зависимости от контекста используется одна из следующих операций: "." (точка) или "->" (стрелочка — минус и знак больше). Если необходимо получить доступ к полю переменной типа структура, используется операция *точка* ("."). Если необходимо получить доступ к полю структуры через указатель, то используется операция *стрелочка* ("->"). Например:

```
struct people_info variable, *pointer = variable;
variable.age = 21;      // Присвоить полю "возраст" значение 21 (год)
pointer->exp = 3.5;     // Присвоить полю "стаж" значение 3,5 (года)
(*pointer).pol = 'm';   // Присвоить полю "пол" значение 'm' – мужчина
.......
int var_for_age = variable.age;     // К полям структуры можно обращаться как для
char current_pol = pointer->pol;    // записи так и для чтения
```

Листинг 3

Обратите внимание что в приведенном выше примере все операции присваивания применяются к полям одной и той же структуры, соответствующей переменной *variable*, поскольку указатель *pointer* настроен (указывает) на *variable.*

Обращение через указатель возможно двумя способами (как показано в примере), которые являются эквивалентными. В первом случае используется специализированная операция, а во втором — операция разыменования, которая выполняет обращение к структуре в целом (*pointer эквивалентно variable).

Инициализация структуры

При определении переменной типа структура возможна ее *инициализация*. Например:

```
struct people_info variable = { "Ivanov", "Ivan", 30, 'm', 10.5 };
```

При этом каждому полю ставится в соответвие инициализируемое значение. Если необходимо инициализировать значения только нескольких полей используется следующая запись:

```
struct people_info variable = { .name="Ivan", .pol= 'm' };
```

Массивы структур

Из структур, как и из базовых типов, можно формировать *массивы*. Такая структура данных позволяет удобно представлять таблицы в памяти компьютера. Например, таблица, содержащую информацию о человеке:

Имя (name)	Фамилия (sname)	Возраст (age)	Пол (pol)	Стаж (exp)
Иван	Иванов	21	м	1,5
Ирина	Петрова	25	ж	7,3
.....				
Александр	Сидоров	40	м	

Таблица 1

Таблица 1 может быть представлена в программе с помощью массива, где элементами будут структуры типа *struct people_info*, описанные в Листинге 2:

```
struct people_info table1[100];
```

Максимальное количество записей (строк с информацией) в такой таблице будет равно 100. При этом объявление структуры, показанное на Листинге 2 можно рассматривать как заголовок ("шапку") таблицы, не способный хранить какой-либо полезной информации, но задающий структуру таблицы. И действительно простое определение структуры без объявления переменных не приводит к выделению памяти. Память выделяется **только** под переменные!

Вложенность структур

Элементом (или полем) структуры может быть не только базовый тип, но и другая структура. В этом случае говорят о вложенности структур. Например описанная выше структура *struct people_info* может быть использована как поле структуры сотрудник (*struct employee*) для описания личной информации о человеке:

```
struct employee {
      people_info pers_info;  // Личная информация – вложенная структура
      char duties[64];        // Должность, строка из 63 символов
      float salaty;           // зарплата – вещественное поле
} a, *b = &a;
```

Для обращения к полям вложенной структуры используются обычные операции обращения ('.' и '->'). Например:

```
a.pers_info.pol = 'm';
b->pers_info.pol = 'm';
```

Файловый ввод/вывод

Поток

По историческим причинам структура данных языка Си, описывающая поток данных называется FILE (а не STREAM (англ.) поток). Эта структура объявлена в файле stdio.h. FILE содержит внутреннюю информацию о состоянии соединения с ассоциированным файлом, включая информацию о позиции курсора, информацию о буферизации, индикатор конца файла и ошибки (они могут быть получены с помощью функций feof и ferror). Выделение памяти и управление данной структурой выполняется библиотечными функциями ввода/вывода . Программист не должен работать с FILE напрямую, для написания переносимого кода необходимо работать с ней ТОЛЬКО через соответствующие функции.

Алгоритм работы с файлом выглядит следующим образом:

1. открытие потока;
2. операции чтения/записи/смещения курсора;
3. закрытие потока.

Открытие потока

Открытие файла выполняется с помощью функции *fopen*, которая создает новый поток ассоциированный с открываемым файлом. Данное действие может привести к созданию нового файла.

```
FILE * fopen (const char *filename, const char *opentype)
```

Функция *fopen* открывает поток для ввода/вывода в/из файла *filename* и возвращает указатель на поток. Агрумент *opentype* — это строка, управляющая режимом открытия файла. Она должна начинаться со следующих символов:

opentype	Значение аргумента
"r"	Открытие существующего файла filename только для чтения
"w"	Открытие файла только для записи. Если файл уже существует его содержи-

	мое очищается. В противном случае будет создан новый файл с именем filename.
"a"	Открытие файла для добавления, т.е. только запись в конец файла. Если файл уже существует его содержимое не изменяется, а новые данные дописываются в конец. В противном случае создается новый файл.
"r+"	Открытие существующего файла как для записи так и для чтения. Содержимое файла не изменяется, позиция чтения/записи установлена в начало файла. При записи старое содержимое будет заменяться на новое.
"w+"	Открытие существующего файла как для записи так и для чтения. Если файл уже существует его содержимое очищается. В противном случае будет создан новый файл с именем filename.
"a+"	Открытие файла как для чтения так и для добавления. Если файл существует — его содержимое не изменяется, позиция чтения установлена на начало файла, добавление выполняется в конец файла./записи установлена в начало файла. Если файла не существует — он создается.

'+' запрашивает поток в котором разрешены операции как ввода так и вывода. Стандарт ANSI требует вызова функции fflush или вызова позиционирующей функции (например fseek) при переключении между чтением и записью и наоборот. В противном случае встроенные буферы могут быть очищены некорректно.

В библиотеке GNU также используется символ `x', означающий что open выполнится успешно ТОЛЬКО в том случае если файл не существовал на момент открытия.

Символ 'b' означает что открывается бинарный файл (не текстовый). В POSIX системах (включая GNU системы) – различий между бинарными и текстовыми файлами не делается.

Любые другие символы в аргументе *opentype* игнорируются.

В случае неуспеха fopen возвращает NULL.

Закрытие потока

Поток закрывается с помощью функции *fclose*, которая разрывает соединение между потоком и файлом. Буферизованный вывод записывается в файл, а любой буферизованный ввод отбрасывается. *fclose* возвращает 0 в случае успеха и EOF в случае ошибки.

```
int fclose (FILE *stream);
```

Очень важно проверять не произошла ли ошибка, поскольку это достаточно обычная ситуация. Например ситуация когда носитель переполнен.

В случае завершения функции *main* или вызова *exit* – все открытые потоки *автоматически* и *корректно* закрываются. В случае же аварийного завершения возможно некорректное закрытие и потеря данных в файле.

Бинарные файлы

Файлы могут быть разделены на текстовые и бинарные. Текстовые содержат только символы, распознаваемые текстовыми редакторами. Двоичный (бинарный) файл — в широком смысле: последовательность произвольных байтов. В узком смысле слова двоичные файлы противопоставляются текстовым файлам. При этом с точки зрения технической реализации на уровне аппаратуры, текстовые файлы являются частным случаем двоичных файлов, и, таким образом, в широком значении слова под определение «двоичный файл» подходит любой файл.

В данной лабораторной работы будут рассмотрено использование бинарных файлов. Для работы с ними могут быть использованы различные функции, но в рамках данной лабораторной работы будут рассмотрены только две из них: *fread* и *fwrite*. Эти функции позволяют читать и записывать блоки данных любого типа. Портотипы этих функций следующие:

```
size_t fread (void * buffer, size_t size, size_t count, FILE * fp);
size_t fwrite (const void * buffer, size_t size, size_t count, FILE * fp);
```

Буфер (*buffer*)– указатель на область памяти, в которую будут прочитаны данные из файла. Счетчик — *count* — определяет, сколько считывается и записывается элементов данных, причем длина каждого элемента в байтах равна *size*. Функция fread возвращает количество прочитанных элементов, если достигнут конец файла или произошла ошибка, то возвращаемое значение может быть меньше, чем счетчик. Функция *fwrite* возвращает количество записанных элементов. Если ошибка не произошла, то возвращаемый результат будет равен значению счетчика. Одним из самых полезных применений функций fread() и fwrite() является чтение и запись данных пользовательских типов, особенно структур. Например:

```
. . . . . .
FILE * fp;
char filename[] = "some_file";
struct some_struct buff[64]; // массив структур из 64-х элементов
fp =fopen(filename, "wb")
if( fp == NULL ){
      printf("Error opening file %s\n",filename);
}else{
      fwrite(buff, sizeof(struct some_struct), 64, fp);
      fclose(fp);
}
. . . . . .
fp =fopen(filename, "rb")
if( fp == NULL ){
      printf("Error opening file %s\n",filename);
}else{
      fread(buff, sizeof(struct some_struct), 64, fp);
      fclose(fp);
}
```

ЗАДАНИЯ НА ЛАБОРАТОРНУЮ РАБОТУ

Реализовать работу с таблицами в соответствии с вариантом задания. Вариант соответствует порядковому номеру студента в журнале (уточнить у преподавателя).

1. Работа с таблицей реализуется в виде интерактивного меню. Должны быть предусмотрены стандартные пункты: добавить запись, удалить запись, сохранить таблицу в файл, прочитать таблицу из файла. Также должны быть реализованы функции предусмотренные конкретным вариантом.
2. Количество записей в любой таблице < 1024. Для этого необходимо создать массив структур из 1024 элементов. Также необходимо создать переменную для хранения фактического количества занятых элементов.
3. При сдаче лабораторной работы в таблице должно находиться не менее 30 записей.

Варианты заданий

1. Реализовать таблицу для хранения записной книжки. Таблица должна содержать следующие **поля**: *фамилия, телефон, улица, дом, квартира, год рождения*. Необходимо реализовать следующие **функции по обработке таблицы**:

- найти количество людей, живущих на заданной улице;
- вывести статистику количества людей, родившихся водном десятилетии;

2. Реализовать таблицу для учета продажи фильмов в интернет магазине. Таблица должна содержать следующие **столбцы**: *название, год, продолжительность, стоимость, отзыв*

(положит./отр.), объем в байтах. Необходимо реализовать следующие **функции по обработке таблицы**:

- найти 10 самых покупаемых фильмов;
- найти средний объем по фильмам;

3. Реализовать таблицу хранящую информацию о скачках с файлового сервера. Таблица должна содержать следующие **столбцы**: *имя файла, тип файла (видео, аудио, документ, ...), объем в байтах, скорость (на которой файл был скачан), IP адрес получателя*. Необходимо реализовать следующие **функции по обработке таблицы**:

- вывести статистику средней скорости скачивания по IP адресам получателей;
- вывести статистику скачивания по типам файлов;

4. Реализовать таблицу для хранения авиа-вылетов из Новосибирска. Таблица должна содержать следующие **столбцы**: *авиакомпания, город прилета, тип судна, время вылета, время в полете, количество пассажиров*. Необходимо реализовать следующие **функции по обработке таблицы**:

- вывести статистику вылетов по времени суток (ночь, утро, день, вечер);
- определить среднее время полета;

5. Реализовать таблицу для хранения информации о странах мира. Таблица должна содержать следующие **столбцы**: *название страны, столица, государственный язык, численность населения, материк, площадь*. Необходимо реализовать следующие **функции по обработке таблицы**:

- определить среднюю плотность населения;
- вывести статистику количества стран по материкам.

6. Реализовать таблицу для хранения детализации счета абонента. Таблица должна содержать следующие **столбцы**: *тип (входящий/исходящий), номер собеседника, телефонный оператор собеседника, время вызова, продолжительность разговора, стоимость минуты разговора*. Необходимо реализовать следующие **функции по обработке таблицы**:

- определить продолжительность исходящих звонков;
- вывести статистику исходящих звонков по сотовому оператору.

7. Реализовать таблицу хранящую информацию о выдача книг в библиотеке. Таблица должна содержать следующие **столбцы**: *фамилия читателя, название книги, автор, год издания, издательство, количество страниц*. Необходимо реализовать следующие **функции по обработке таблицы**:

- определить количество книг, находящихся у указанного читателя;
- вывести статистику количества книг изданных в одном десятилетии.

8. Реализовать таблицу продаж товаров в продуктовом магазине. Таблица должна содержать следующие **столбцы**: *наименование товара, стоимость, производитель, время продажи, срок годности, время хранения товара на складе на момент продажи (дней)*. Необходимо реализовать следующие **функции по обработке таблицы**:

- определить среднюю стоимость товаров магазина;
- статистику товаров по производителю.

9. Реализовать таблицу хранящую информацию об успеваемости оп дисциплине "Программирование на языке высокого уровня". Таблица должна содержать следующие **столбцы**: *фамилия студента, год обучения, количество сданных лабораторных работ, оценка за курсовой, оценка за экзамен*. Необходимо реализовать следующие **функции по обработке таблицы**:

- определить среднюю успеваемость по курсовому проекту;
- определить лучший, по успеваемости, год.

10. Реализовать таблицу для хранения информацию о прокате фильмов. Таблица должна содержать следующие **столбцы**: *фамилия клиента, название фильма, тип носителя, жанр, стоимость, количество дней проката*. Необходимо реализовать следующие **функции по обработке таблицы**:

- определить среднюю успеваемость по курсовому проекту;
- определить лучший, по успеваемости, год.

11. Реализовать таблицу хранящую информацию о приеме пациентов в городских поликлиниках. Таблица должна содержать следующие **столбцы**: *фамилия врача, специальность врача, фамилия пациента, время приема, номер поликлиники, год обращения*. Необходимо реализовать следующие **функции по обработке таблицы**:

- вывести статистику посещений по поликлиникам;
- определить количество пациентов для указанного врача.

12. Реализовать таблицу о продажах автомобилей в автосалоне. Таблица должна содержать следующие **столбцы**: *производитель, марка, год выпуска, объем двигателя, стоимость, цвет*. Необходимо реализовать следующие **функции по обработке таблицы**:

- определить средний объем двигателя для проданных автомобилей;
- вывести статистику количества проданных автомобилей по производителю.

13. Реализовать таблицу хранящую информацию о пассажирский потоке через некоторую остановку. Таблица должна содержать следующие **столбцы**: *номер маршрута, время прибытия, тип транспорта, количество зашедших пассажиров, количество вышедших пассажиров, стоимость проезда*. Необходимо реализовать следующие **функции по обработке таблицы**:

- определить среднее число приезжающих пассажиров;
- вывести статистику количества уезжающих пассажиров по маршрутам.

14. Реализовать таблицу для хранения информации о перевозках пассажиров на такси. Таблица должна содержать следующие **столбцы**: *время отправления, адрес отправления, время прибытия, адрес прибытия, пробег, стоимость*. Необходимо реализовать следующие **функции по обработке таблицы**:

- определить наиболее востребованное время суток;
- вывести статистику перевозок по улицам.

15. Реализовать таблицу для хранения информации о ремонте автомобилей в СТО. Таблица должна содержать следующие столбцы: *фамилия клиента, тип автомобиля (грузовой, джип, минивен, легковой), сложность ремонта (A,B,C,D), сезон ,стоимость, время ремонта (дней)*. Необходимо реализовать следующие функции по обработке таблицы:

- вывести статистику ремонтов по типу автомобиля;
- определить 5 самых активных клиентов.

16. Реализовать таблицу записей дорожных нарушений. Таблица должна содержать следующие **столбцы**: *фамилия нарушителя, тип нарушения, район, улица , размер штрафа, марка автомобиля*. Необходимо реализовать следующие **функции по обработке таблицы**:

- определить 10 наиболее активных нарушителей;
- вывести статистику количества нарушений по маркам автомобилей.

17. Реализовать таблицу для хранения электронной переписки. Таблица должна содержать следующие **столбцы**: *тип письма (входящее/исходящее), адрес, объем, время получения/отправки, количество прикрепленных файлов, спам/не спам*. Необходимо реализовать следующие **функции по обработке таблицы**:

- определить трех наиболее популярных адресатов исходящих сообщений;
- вывести статистику среднего объема сообщений по адресатам.

18. Реализовать таблицу продаж персональных компьютеров. Таблица должна содержать следующие **столбцы**: *фамилия продавца, производитель процессора, частота процессора, объем ОЗУ, объем жесткого диска, год продажи*. Необходимо реализовать следующие **функции по обработке таблицы**:
- определить средний объем жесткого диска;
- вывести статистику продаж по частоте процессора.

19. Реализовать таблицу для хранения информацию о трудоустройстве через кадровое агентство. Таблица должна содержать следующие **столбцы**: *фамилия клиента, должность, стаж, зарплата, сфера деятельности, год трудоустройства*. Необходимо реализовать следующие **функции по обработке таблицы**:
- вывести статистику устройства на работу по годам;
- определить среднюю зарплату сотрудников в указанной сфере деятельности.

20. Реализовать таблицу, описывающую работу праздничного агентства. Таблица должна содержать следующие **столбцы**: *фамилия заказчика, месяц обращения, вид услуги (банкет, свадьба, день рождения, ...), стоимость, количество человек, район проведения мероприятия*. Необходимо реализовать следующие **функции по обработке таблицы**:
- определить наиболее популярное мероприятие;
- вывести статистику количества мероприятий по районам.

21. Реализовать таблицу для хранения информации о выступлениях сотрудников НИИ на конференциях. Таблица должна содержать следующие **столбцы**: *фамилия докладчика, тема доклада, год, количество слайдов, время выступления, город*. Необходимо реализовать следующие **функции по обработке таблицы**:
- определить самого активного сотрудника;
- вывести статистику выступлений по городам.