Лабораторная работа №7
Тема: «Структуры и файлы»

Цель работы: сформировать навыки и умения обработки структурированных типов данных, организованных в виде структур и файлов.

Время выполнения: 4 часа.

Теоретические сведения

Чтение и запись структур в файл

Хотя функции getc()/putc() позволяют вносить в файл отдельные символы, но фактически мы имеем дело с бинарными файлами. Если мы записываем в файл строку, то в принципе мы даже можем открыть записанный файл любом текстовом редакторе и понять, что там было записано. Но не всегда данные могут представлять строки. И чтобы более наглядно разобраться с работой с бинарными файлами, рассмотрим еще одни пример - с записью-чтением структуры из файла.

```
#include <stdio.h>
#include <stdlib.h>

struct person
{
   char name[16];
   int age;
};

int save(char * filename, struct person *p);
int load(char * filename);

int main(void)
{
   char * filename = "person.dat";
   struct person tom = { "Tom Smith", 21 };

   save(filename, &tom);
   load(filename);

   return 0;
}

// запись структуры в файл
```

```
int save(char * filename, struct person *p)
{
   FILE * fp;
   char *c;
   int size = sizeof(struct person); // количество записываемых байтов

   fp = fopen(filename, "wb"); //открываем файл для бинарной записи
   if (!fp)    // если не удалось открыть файл
   {
      printf("Error occured while opening file \n");
      return 1;
   }
   // устанавливаем указатель на начало структуры
   c = (char *)p;
   // посимвольно записываем в файл структуру
   for (int i = 0; i < size; i++)
   {
      putc(*c++, fp);
   }
   fclose(fp);
   return 0;
}

// загрузка из файла структуры
int load(char * filename)
{
   FILE * fp;
   char *c;
   int i; // для считывания одного символа
   // количество считываемых байтов
   int size = sizeof(struct person);
   // выделяем память для считываемой структуры
   struct person * ptr = malloc(size);
   fp = fopen(filename, "rb");     // открываем файл для бинарного чтения
   if (!fp)
   {
      printf("Error occured while opening file \n");
      return 1;
   }
```

```
    // устанавливаем указатель на начало блока выделенной памяти
    c = (char *)ptr;
    // считываем посимвольно из файла
    while ((i = getc(fp))!=EOF)
    {
        *c = i;
        c++;
    }

    fclose(fp);
    // вывод на консоль загруженной структуры
    printf("%-20s %5d \n", ptr->name, ptr->age);
    free(ptr);
    return 0;
}
```

В данном случае запись и чтение структуры выделены в отдельные функции: save() и load() соответственно.

Для записи в функции save() через параметр struct person *p получаем указатель на сохраняемую структур. Фактически его значением является начальный адрес блока памяти, где располагается структура.

Функция putc записывает отдельный символ в файл, однако нам надо записать структуру. Для этого мы создаем указатель на символ (который по сути представляет один байт) и устанавливаем этот указатель на начало блока памяти, выделенного для структуры.

```
c = (char *)p;
```

То есть в данном случае мы получаем адрес в памяти первого байта из блока памяти, которая выделена для структуры. И затем мы можем пройтись по всему этому блоку и получить отдельные байты и занести их в файл:

```
for (int i = 0; i < size; i++)
{
    putc(*c++, fp);
}
```

И в данном случае нам не важно, какие поля имеет структура, какой она имеет размер. Мы работаем с ней как с набором байт и заносим эти байты в файл. После занесения каждого отдельного байта в файл указатель с в блоке памяти перемещается на один байт вперед.

При чтении файла в функции load() используется похожий принцип только в обратную сторону.

Во-первых, для считывания структуры из файла мы выделяем блок динамической памяти для хранения прочитанных данных:

```
struct person * ptr = (struct person *) malloc(size);
```

После этого указатель ptr будет указывать на первый адрес блока из 20 байт (наша структура занимает 20 байт = 16 символов и 4 байта для числа int).

Затем так как при прочтении мы получаем символы, устанавливаем указатель на первый байт выделенного блока и в цикле считываем данные из файла в этот блок:

```
c = (char *)ptr;
// считываем посимвольно из файла
while ((i = getc(fp))!=EOF)
{
   *c = i;
   c++;
}
```

Здесь стоит обратить внимание на то, что в данном случае на самом деле считываем даже не символ, а числовой код символа в переменную типа int и только потом передаем значение указателю c. Это сделано для корректной обработки окончания файла EOF. Это значение может представлять любое отрицательное число. И если бы мы сохранили отрицательное число (например, возраст пользователя был бы отрицательным), то оно было бы некорректно интерпретировано при чтении как конец файла, и итоговый результа был бы неопределенным. Поэтому более правильно считывать именно числовой код символа в переменную int, а затем числовой код передавать в char.

Запись и чтение массива структур

Выше приведен пример по работе с одной структурой. Но, как правило, при работе с файлами мы оперируем не одной структурой, а каким-то набором структур. Поэтому усложним задачу и сохраним и считаем из файла массив структур:

```
#include <stdio.h>
#include <stdlib.h>

struct person
{
```

```
    char name[20];
    int age;
};

int save(char * filename, struct person *st, int n);
int load(char * filename);

int main(void)
{
    char * filename = "people.dat";
    struct person people[] = { {"Tom", 23}, {"Alice", 27}, {"Bob", 31},
{"Kate", 29 }};
    int n = sizeof(people) / sizeof(people[0]);

    save(filename, people, n);
    load(filename);
    return 0;
}

// запись в файл массива структур
int save(char * filename, struct person * st, int n)
{
    char *c;    // для записи отдельных символов
    // число записываемых байтов
    int size = n * sizeof(struct person);
    FILE * fp = fopen(filename, "wb");
    if (!fp)
    {
        printf("Error occured while opening file\n");
        return -1;
    }
    // записываем количество структур
    c = (char *)&n;
    for (int i = 0; i < sizeof(n); i++)
    {
        putc(*c++, fp);
    }

    // посимвольно записываем в файл все структуры
    c = (char *)st;
```

```
        for (int i = 0; i < size; i++)
        {
            putc(*c, fp);
            c++;
        }
        fclose(fp);
        return 0;
    }

    // загрузка из файла массива структур
    int load(char * filename)
    {
        char *c;    // для считывания отдельных символов
        int m = sizeof(int);     // сколько надо считать для получения размера массива
        int n;  // количество структур в массиве

        FILE * fp = fopen(filename, "r");
        if (!fp)
        {
            printf("Error occured while opening file\n");
            return -1;
        }
        // выделяем память для хранения количества данных
        int *ptr_count = malloc(m);
        // считываем количество структур
        c = (char *)ptr_count;
        // пока не считаем m байт, сохраняем байт в выделенный блок для размера массива
        while (m > 0 && (*c = getc(fp)) != EOF)
        {
            c++;
            m--;
        }
        //получаем число элементов
        n = *ptr_count;
        free(ptr_count);    // освобождаем память
        // выделяем память для считанного массива структур
        struct person * ptr = malloc(n * sizeof(struct person));
```

```
    // устанавливаем указатель на блок памяти, выделенный для массива структур
    c = (char *)ptr;
    // считываем посимвольно из файла
    while ((*c= getc(fp))!=EOF)
    {
      c++;
    }
    // перебор загруженных элементов и вывод на консоль
    printf("\nFound %d people\n\n", n);

    for (int i = 0; i < n; i++)
    {
      printf("%-5d %-10s %5d \n", i + 1, (ptr + i)->name, (ptr + i)->age);
      // или так
      // printf("%-5d %-10s %5d \n", i + 1, ptr[i].name, ptr[i].age);
    }

    free(ptr);
    fclose(fp);
    return 0;
}
```

Данная задача усложнена тем, что нам надо хранить массив структур, количество которых точно может быть неизвестно. Один из вариантов рещения этой проблемы состоит в сохранении некоторой метаинформации о файле в начале файла. В частности, в данном случае в начале файла сохраняется число записанных структур.

Запись во многом аналогична записи одной структуры. Сначала устанавливаем указатель на число n, которое представляет количество структур, и все байты этого числа записываем в файл:

```
c = (char *)&n;
for (int i = 0; i<sizeof(n); i++)
{
  putc(*c++, fp);
}
```

Затем подобным образом записываем все байты из массива структур - устанавливаем указатель на первый байт массива структур и записываем size байт в файл:

```
// посимвольно записываем в файл все структуры
c = (char *)st;
for (int i = 0; i < size; i++)
{
   putc(*c, fp);
   c++;
}
```

При чтении нам придется файктически считывать из файла два значения: количество структур и их массив. Поэтому при чтении два раза выделяется память. Вначале для количества элементов:

```
int *ptr_count = malloc(m);
```

Затем мы считываем первые 4 байта из файла для получения числа:

```
c = (char *)ptr_count;
while (m > 0 && (*c = getc(fp)) != EOF)
{
   c++;
   m--;
}
//получаем число элементов
n = *ptr_count;
```

Затем аналогичные действия проделываем для массива структур.

```
struct person * ptr = malloc(n * sizeof(struct person));
// устанавливаем указатель на блок памяти, выделенный для массива структур
c = (char *)ptr;
// считываем посимвольно из файла
while ((*c= getc(fp))!=EOF)
{
   c++;
}
```

Индивидуальные задания к лабораторной работе №7

1. Для получения места в общежитии формируется список студентов, который включает Ф.И.О. студента, группу, средний балл, доход на члена семьи. Общежитие в первую очередь предоставляется тем, у кого доход на члена семьи меньше двух минимальных зарплат, затем остальным в порядке уменьшения среднего балла. Вывести список очередности предоставления мест в общежитии. Предусмотреть запись в файл.

2. Описать структуру с именем STUDENT, содержащую следующие поля: - фамилия и инициалы; - номер группы; - успеваемость (массив из пяти элементов). Написать программу, выполняющую следующие действия: - ввод с клавиатуры данных; записи должны быть упорядочены по алфавиту. Предусмотреть запись в файл

3. Создать структуру должностей Vacancies {Manager, Boss, Clerk, Salesman, etc.}. Создать структуру «Employee», состоящую из:

- поля name строкового типа;
- поля vacancy типа Vacancies;
- поля зарплата целого типа;
- поля дата приема на работу типа int[3].

Создать массив сотрудников. Длина массива задается пользователем, заполнение массива производится им же. Вывести полную информацию обо всех сотрудниках. Предусмотреть запись в файл

4. Описать структуру с именем STUDENT, содержащую следующие поля: - фамилия и инициалы; - номер группы; - успеваемость (массив из пяти элементов). Написать программу, выполняющую следующие действия: - ввод с клавиатуры данных; - вывод на экран фамилий и номеров групп для всех студентов, имеющих хотя бы одну оценку 2; если таких студентов нет, вывести соответствующее сообщение. Предусмотреть запись в файл

5. Описать структуру для групп студентов. Поля структуры – произвольные. Необходимо вывести информацию о студентах, чьи фамилии начинаются на I, F. Данные вводятся с клавиатуры. Предусмотреть запись в файл

6. Используя структуру определить количество дней в указанном месяце. Предусмотреть запись в файл

7. В программе заданы месяц и год двух дат. Пользователь вводит еще одну дату (только месяц и год). Определить, принадлежит ли третья дата диапазону от первой даты до второй включительно. Задачу решить с использованием структуры данных. Предусмотреть запись в файл

8. Пользователь вводит данные о количестве студентов, их фамилии, имена и балл для каждого. Программа должна определить средний балл и вывести фамилии и имена студентов, чей балл выше среднего. Предусмотреть запись в файл

9. Написать программу, в которой хранятся данные о товарах, их количестве и цене. При запуске программы эта информация выводится на экран. Далее пользователю должно предлагаться вводить номера товаров и их новое количество. После этого все данные о товарах должны снова выводиться на экран. Задачу решить с использованием структуры данных. Предусмотреть запись в файл

10. Используя структуры, написать программу сложения и умножения двух комплексных чисел. Предусмотреть запись в файл

11. Напишите программу, которая сначала по первой букве должности, введенной пользователем, определяет соответствующее значение переменной, помещает это значение в переменную, а затем выводит полностью название должности, первую букву которой ввел пользователь. Предусмотреть запись в файл

12. Пользователь вводит название учебной дисциплины. Вывести всех преподавателей, которые ведут данную дисциплину. Предусмотреть запись в файл

13. Создать структуру и предусмотреть запись в файл. Вывести сведения о машинах, прошедших техосмотра менее года назад.

14. Организовать поиск в структуре по типу самолёта, типы самолёта уже введены в программе. Предусмотреть запись в файл

15. Пользователь вводит название поры года. Необходимо вывести все названия месяцев, которые принадлежат данной поре года. Предусмотреть запись в файл

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретическую часть лабораторной работы.

2. Реализовать индивидуальное задание по вариантам, представленные в теоретических сведениях, сделать скриншоты работающих программ. Написать комментарии.

3. Написать отчет, содержащий:

1. Титульный лист, на котором указывается:

а) полное наименование министерства образование и название учебного заведения;

б) название дисциплины;

в) номер практического занятия;

г) фамилия преподавателя, ведущего занятие;

д) фамилия, имя и номер группы студента;

е) год выполнения лабораторной работы.

2. Индивидуальное задание из раздела «Теоретические сведения» с кодом, комментариями и скриншотами работающих программ.

3. Построение блок-схем.