Лабораторная работа №12.
Файловый ввод-вывод.

Цель работы: получить представление о возможностях файлового ввода-вывода. Обобщение пройденного материала.

Рекомендации: для более осознанного понимания происходящего тут, разберитесь с тем, как организованна в Си работа со строками.

1. Несколько слов вместо введения

В стандарте языка Си отсутствуют средства ввода-вывода. Все операции ввода-вывода реализуются с помощью функций, находящихся в библиотеке языка Си, поставляемой в составе конкретной системы программирования Си.

Все файлы рассматриваются как неструктурированная последовательность байтов. На уровне потокового ввода-вывода обмен данными производится побайтно. Чаще всего минимальной порцией данных, участвующей в обмене с внешней памятью, являются блоки в 512 байт или 1024 байта. При вводе с диска (при чтении из файла) данные помещаются в буфер операционной системы, а затем побайтно или определенными порциями передаются программе пользователя. При выводе данных в файл они накапливаются в буфере, а при заполнении буфера записываются в виде единого блока на диск за одно обращение к последнему. Буферы операционной системы реализуются в виде участков основной памяти. Поэтому пересылки между буферами ввода-вывода и выполняемой программой происходят достаточно быстро в отличие от реальных обменов с физическими устройствами.

При работе с потоком можно производить следующие действия:

- ☑ открывать и закрывать потоки (связывать указатели на потоки с конкретными файлами);
- ☑ вводить и выводить: символ, строку, форматированные данные, порцию данных произвольной длины;
- ☑ анализировать ошибки потокового ввода-вывода и условие достижения конца потока (конца файла);
- ☑ управлять буферизацией потока и размером буфера;
- ☑ получать и устанавливать указатель (индикатор) текущей позиции в потоке.

Для того чтобы можно было использовать функции библиотеки ввода-вывода языка Си, в программу необходимо включить заголовочный файл `stdio.h`, который содержит прототипы функций ввода-вывода, а также определения констант, типов и структур, необходимых для работы функций обмена с потоком.

В предыдущих лабораторных работах

2. Открытие и закрытие потока

Прежде чем начать работать с потоком, его необходимо инициализировать, т.е. открыть. При этом поток связывается в исполняемой программе со структурой предопределенного типа `FILE`. Определение структурного типа `FILE` находится в заголовочном файле `stdio.h`. В структуре `FILE` содержатся компоненты, с помощью которых ведется работа с потоком, в частности: указатель на буфер, указатель (индикатор) текущей позиции в потоке и другая информация.

При открытии потока в программу возвращается *указатель на поток*, являющийся указателем на объект структурного типа `FILE`. Этот указатель идентифицирует поток во всех последующих операциях.

Указатель на поток, например `fp`, должен быть объявлен в программе следующим образом:

```
#include <stdio.h>
FILE *fp;
```

Указатель на поток приобретает значение в результате выполнения функции открытия потока:

```
fp = fopen (имя файла, режим открытия);
```

Параметры имя файла и режим открытия являются указателями на массивы символов, содержащих соответственно имя файла, связанного с потоком, и строку режимов открытия. Однако эти параметры могут задаваться и непосредственно в виде строк при вызове функции открытия файла:

```
fp = fopen("C:\\temp\\test.txt", "r");
```

где `C:\\temp\\test.txt` – путь и имя некоторого файла, связанного с потоком;

r - обозначение одного из режимов работы с файлом (тип доступа к потоку).

Стандартно файл, связанный с потоком, можно открыть в одном из следующих шести режимов:

"w" - новый текстовый файл открывается для записи. Если файл уже существовал, то предыдущее содержимое стирается, файл создается заново;

"r" - существующий текстовый файл открывается только для чтения;

"a" - текстовый файл открывается (или создается, если файла нет) для добавления в него новой порции информации (добавление в конец файла). В отличие от режима w, режим "a" позволяет открывать уже существующий файл, не уничтожая его предыдущей версии, и писать в продолжение файла;

"w+" - новый текстовый файл открывается для записи и последующих многократных исправлений. Если файл уже существует, то предыдущее содержимое стирается. Последующие после открытия файла запись и чтение из него допустимы в любом месте файла, в том числе запись разрешена и в конец файла, т.е. файл может увеличиваться ("расти");

"r+" - существующий текстовый файл открывается как для чтения, так и для записи в любом месте файла; однако в этом режиме невозможна запись в конец файла, т.е. недопустимо увеличение размеров файла;

"a+" - текстовый файл открывается или создается (если файла нет) и становится доступным для изменений, т.е. для записи и для чтения в любом месте; при этом в отличие от режима "w+" можно открыть существующий файл и не уничтожать его содержимого; в отличие от режима "r+" в режиме "a+" можно вести запись в конец файла, т.е. увеличивать его размеры.

Поток можно открыть в текстовом либо двоичном (бинарном) режиме.

В текстовом режиме прочитанная из потока комбинация символов CR и LF, то есть управляющие коды “возврат каретки” и “перевод строки”, преобразуется в один символ новой строки \n. При записи в поток в текстовом режиме осуществляется обратное преобразование, т.е. символ новой строки \n (LF) заменяется последовательностью CR и LF.

Если файл, связанный с потоком, хранит не текстовую, а произвольную двоичную информацию, то указанные преобразования не нужны и могут быть даже вредными. Обмен без такого преобразования выполняется при выборе двоичного или бинарного режима, который обозначается буквой `b`. Например, "`r+b`" или "`wb`". В некоторых компиляторах текстовый режим обмена обозначается буквой `t`, т.е. записывают "`a+t`" или "`rt`".

Возможные ошибки при открытии потока:

- ☑ указанный файл не найден (для режима "чтение");
- ☑ диск заполнен или диск защищен от записи и т.п.
- ☑ при отсутствии динамической памяти устанавливается признак ошибки "Not enough memory" (недостаточно памяти).

В перечисленных случаях указатель на поток приобретает значение `NULL`. Заметим, что указатель на поток в любом режиме, отличном от аварийного, никогда не бывает равным `NULL`.

Приведем типичную последовательность операторов, которая используется при открытии файла, связанного с потоком:

```
if ((fp = fopen ("C:\\temp\\test.txt", "w"))  = = NULL)
{
    printf("%s", strerror(errno));
    exit(0);
}
```

где `NULL` - нулевой указатель, определенный в файле `stdio.h`.

Для вывода на экран дисплея сообщения об ошибке при открытии потока используется стандартная библиотечная функция `printf()`.

Функция `strerror()` генерирует строку с сообщением об ошибке, полученного из кода ошибки, переданного `errno`. Номер ошибки заносится в глобальную переменную `int errno` (определенную в заголовочном файле `errno.h`) рядом функций библиотеки языка Си, в том числе и функциями ввода-вывода.

Открытые на диске файлы после окончания работы с ними рекомендуется закрыть явно. Для этого используется библиотечная функция

```
int fclose (указатель _на поток);
```

Открытый файл можно открыть повторно (например, для изменения режима работы с ним) только после того, как файл будет закрыт с помощью функции `fclose()`.

Работа с файлами на диске

Для работы с файлами на диске в библиотеку языка Си включены следующие функции:

`fgetc()` - ввод (чтение) одного символа из файла;

`fputc()` - запись одного символа в файл;

`fprintf()` - форматированный вывод в файл;

`fscanf()` - форматированный ввод (чтение) из файла;

`fgets()` - ввод (чтение) строки из файла;

`fputs()` - запись строки в файл.

Двоичный режим обмена.

Двоичный режим обмена организуется с помощью функций `getc()` и `putc()`, обращение к которым имеет следующий формат:

```
c= getc(fp);
putc(c, fp);
```

где fp - указатель на поток, c - переменная типа int для приема очередного символа из файла или для записи ее значения в файл. Прототипы функции:

```
int getc ( FILE *stream );
int putc ( int c, FILE *stream );
```

В качестве примера использования функций `getc()` и `putc()` рассмотрим программы ввода данных в файл с клавиатуры и программу вывода их на экран дисплея из файла.

Программа ввода читает символы с клавиатуры и записывает их в файл. Пусть признаком завершения ввода служит поступивший от клавиатуры символ `'#'`. Имя файла запрашивается у пользователя. Если при вводе последовательности символов была нажата клавиша <Enter>, служащая разделителем строк при вводе с клавиатуры, то в файл записываются управляющие коды "Возврат каретки" (`CR`) и "Перевод строки" (`LF`). Код `CR` в дальнейшем при выводе вызывает перевод маркера (курсора) в начало строки экрана дисплея. Код `LF` служит для перевода маркера на новую строку дисплея. Значения этих кодов в тексте программы обозначены соответственно

идентификаторами CR и LF, т.е. CR и LF - именованные константы. Запись управляющих кодов CR и LF в файл позволяет при последующем выводе файла на экран отделить строки друг от друга.

```
/* Программа ввода символов */
#include   <stdio.h>
int main ()
{
    FILE *fp;    /* Указатель на поток */
    char c;
    /* Восьмеричный код "Возврат каретки"*/
    const char CR= '\015';
    /* Восьмеричный код "Перевод строки"*/
    const char LF = '\012';
    char fname[20]; /* Массив для имени файла*/
    puts("Введите имя файла: \n");/* Запрос имени файла*/
    gets(fname);
    /* Открыть файл для записи*/
    if ((fp = fopen(fname,"w")) == NULL)
    {
        printf("%s", strerror(errno));
        return 1;
    }
    /*Цикл ввода и записи в файл символов*/
    while ((c = getchar()) != '#')
    {
        if (c == '\n')
        {
            putc(CR, fp);
            putc(LF, fp);
        }
        else   putc(c, fp);
    }
    /* Цикл ввода завершен; закрыть файл*/
    fclose(fp); return 0;
}
```

Следующая программа читает поток символов из ранее созданного файла и выводит его на экран дисплея:

```
/* Программа вывода символьного файла на экран дисплея */
#include <stdio.h>
int main ()
{
    FILE *fp; /* Указатель на поток */
    char c;
    char fname[20]; /* Массив для имени файла */
    /* Запрос имени файла*/
    puts("введите имя файла:  \n ");
    gets(fname);
    /* Открыть файл для чтения:   */
```

```
    if ((fp = fopen(fname,"r")) == NULL)
    {
        printf("%s", strerror(errno));
        return 1;
    }
    /* Цикл чтения из файла и вывода символов на экран*/
    while  ((c = getc(fp)) != EOF)
        putchar(c);
    fclose(fp); /* Закрыть файл */
    return 0;
}
```

Программу чтения символов из файла можно усовершенствовать, добавив возможность вывода информации на экран порциями (кадрами):

```
/* Усовершенствованная программа вывода символьного файла на
экран дисплея по кадрам */
#include <stdio.h>
int main ()
{
    FILE *fp;   /* Указатель на поток)  */
    char c;
    /* Восьмеричный код "Перевод строки"*/
    const char LF='\012';
    int MAX=10; /* Размер кадра */
    int nline; /* Счетчик строк */
    char fname[10]; /* Массив для имени файла */
    /* Запрос имени файла:  */
    puts("введите имя файла \n");
    gets(fname);
    /* Открыть файл для чтения:  */
    if ((fp = fopen(fname,"r")) == NULL)
    {
        printf("%s", strerror(errno));
        return 1;
    }
    /* Цикл вывода на экран */
    while (1)
    {
        nline = 1;
        while (nline<MAX) /* Цикл вывода кадра */
        {
            c = getc(fp);
            if (c == EOF)
            {
                fclose(fp);
                return 0;
            }
            if (c = LF) nline++;
            putchar(c);
```

```
            }/* Конец цикла вывода кадра*/
            /* Задержка вывода до нажатия любой клавиши */
            getchar();
        }/* Конец цикла вывода */
}/* Конец программы */
```

В этой программе после вывода очередного кадра из MAX строк для перехода к следующему кадру необходимо нажать любую клавишу.

Используя двоичный обмен с файлами, можно сохранять на диске информацию, которую нельзя непосредственно вывести на экран дисплея (целые и вещественные числа во внутреннем представлении).

Строковый обмен с файлами.

Для работы с текстовыми файлами, кроме перечисленных выше функций, удобно использовать функции fgets() и fputs(). Прототипы этих функций в файле stdio.h имеют следующий вид:

```
int fputs(const char *s, FILE *stream);
char * fgets(char *s, int n, FILE *stream);
```

Функция fputs() записывает ограниченную символом '\0' строку (на которую указывает s) в файл, определенный указателем stream на поток, и возвращает неотрицательное целое. При ошибках функция возвращает EOF. Символ '\0' в файл не переносится, и символ '\n' не записывается в конце строки вместо '\0'.

Функция fgets() читает из определенного указателем stream файла не более (n-1) символов и записывает их в строку, на которую указывает s. Функция прекращает чтение, как только прочтет (n-1) символов или встретит символ новой строки '\n', который переносится в строку s. Дополнительно в конец каждой строки записывается признак окончания строки '\0'. В случае успешного завершения функция возвращает указатель s. При ошибке или при достижении конца файла, при условии, что из файла не прочитан ни один символ, возвращается значение NULL. В этом случае содержимое массива, который адресуется указателем s, остается без изменений. Напомним, что в отличие от fgets() функция gets() отбрасывает символ '\n'.

Проиллюстрируем возможности указанных функций на программе, которая переписывает текст из одного файла в другой. На этом же примере еще раз проиллюстрируем особенность языка Си - возможность передачи информации в выполняемую программу из командной строки.

Предположим, что имя выполняемой программы copyfile.exe и мы хотим с ее помощью переписать содержимое файла f1.dat в файл f2.txt. Вызов программы из командной строки имеет вид:

>copyfile.exe fl.dat f2.txt

Текст программы может быть таким:

```
#include <stdio.h>
main(int argc, char *argv[ ])
{
    char cc[256];/* Массив для обмена с файлами */
    FILE *f1,  *f2; /* Указатели на   потоки */
    if (argc!=3)/* Проверка командной строки */
    {
        printf("\n Формат вызова программы:   ");
        printf("\n copyfile.exe");
        printf("\n файл-источник  файл-приемник);
        return 1;
    }
    /* Открытие входного файла */
    if ((f1 = fopen(argv[1], "r")) == NULL)
    {
        perror(argv[1]);
        return 1;
    }
    /* Открытие выходного файла */
    if ((f2 = fopen(argv[2], "w")) == NULL)
    {
        perror(argv[2]);
        return 1;
    }
    while (fgets(cc, 256, f1) != NULL)
        fputs(cc, f2);
    fclose(f1);
    fclose(f2);
    return 0;
}
```

Обратите внимание, что значение `argc` явно не задается, а определяется автоматически. Так как `argv[0]` определено всегда, то значение `argc` не может быть меньше `1`. При отсутствии в командной строке двух аргументов (имен файлов ввода-вывода) значение `argc`

оказывается не равным трем, и поэтому на экран выводится поясняющее сообщение.

Режим форматного обмена с файлами.

В некоторых случаях информацию удобно записывать в файл в виде, пригодном для непосредственного (без преобразования) отображения на экран дисплея, т.е. в символьном виде. Например, для сохранения результатов работы программы, чтобы затем распечатать их на бумагу (получить "твердую" копию) или когда необходимо провести трассировку программы - вывести значения некоторых переменных во время выполнения программы для их последующего анализа. В этом случае можно воспользоваться функциями форматного ввода (вывода) в файл `fprintf()` и `fscanf()`, которые имеют следующие прототипы:

```
int fprintf (указатель на поток, форматная строка, список-переменных);
int fscanf (указатель на поток, форматная строка, список адресов переменных);
```

Как и в функциях `printf()` и `scanf()`, форматная строка может быть определена вне вызова функции, а в качестве фактического параметра в этом случае будет указатель на нее.

От рассмотренных выше функций `printf()`, `scanf()` для форматного обмена с дисплеем и клавиатурой функции `fprintf()` и `fscanf()` отличаются лишь тем, что в качестве первого параметра в них необходимо задавать указатель на поток, с которым производится обмен. В качестве примера приведем программу, создающую файл int.dat и записывающую в него символьные изображения чисел от 1 до 10 и их квадратов:

```
#include <stdio.h>
int main ()
{
    FILE *fp; /* Указатель на поток */
    int n;
    if ((fp = fopen ("int.dat",'w')) == NULL)
    {
        perror("int.dat");
        return 1;
    }
    for  (n=1; n<11; n++)
    fprintf(fp, "%d %d\n", n, n*n);
    fclose(fp);
```

```
    return 0;
}
```

В результате работы этой программы в файле int.dat формируется точно такая же "картинка", как и в том случае, если бы мы воспользовались вместо функции `fprintf()` функцией `printf()` для вывода результатов непосредственно на экран дисплея. Убедитесь в этом. Несложно написать программу, использующую для форматного чтения данных из файла функцию `fscanf()`. Сделайте это!

Позиционирование в потоке.

Все вышеописанные средства позволяют записать в файл данные и читать из него информацию только последовательно.

Операция чтения (или записи) для потока всегда производится, начиная с текущей позиции в потоке. Начальная позиция чтения/записи в потоке устанавливается при открытии потока и может соответствовать начальному или конечному байту потока в зависимости от режима открытия потока. При открытии потока в режимах `"r"` и `"w"` указатель текущей позиции чтения/записи в потоке устанавливается на начальный байт потока, а при открытии в режиме `"a"` - в конец файла (за конечным байтом). При выполнении каждой операции ввода-вывода указатель текущей позиции в потоке перемещается на новую текущую позицию в соответствии с числом прочитанных (записанных) байтов.

Существуют средства позиционирования в потоке, позволяющие перемещать указатель (индикатор) текущей позиции в потоке на нужный байт. Эти средства дают возможность работать с файлом на диске, как с обычным массивом, осуществляя доступ к содержимому файла в произвольном порядке.

В библиотеку языка Си включена функция `fseek()` для перемещения (установки) указателя текущей позиции в потоке на нужный байт потока (файла). Она имеет следующий прототип:

```
int fseek (указатель на поток, смещение, начало отсчета);
```

Смещение задается переменной или выражением типа `long` и может быть отрицательным, т.е. возможно перемещение по файлу в прямом и обратном направлениях. Начало отсчета задается одной из

предопределенных констант, размещенных в заголовочном файле stdio.h:

- ☑ SEEK_SET - начало файла;
- ☑ SEEK_CUR - текущая позиция;
- ☑ SEEK_END - конец файла.

Здесь следует напомнить некоторые особенности данных типа long. Этот тип определяется для целочисленных констант и переменных, которым в памяти должно быть выделено больше места, чем данным типа int. Обычно переменной типа long выделяется 4 байта, чем и определен диапазон ее значений.

Константа типа long записывается в виде последовательности десятичных цифр, вслед за которыми добавляется разделитель L или l. Примеры констант типа long:

```
0l 0L 10L 688L 331
```

В текстах программ лучше использовать L, чтобы не путать l с цифрой 1.

Функция fseek() возвращает 0, если перемещение в потоке (файле) выполнено успешно, в противном случае возвращается ненулевое значение.

Приведем примеры использования функции fseek(). Перемещение к началу потока (файла) из произвольной позиции:

```
fseek(fp, 0L, SEEK_SET);
```

Перемещение к концу потока (файла) из произвольной позиции:

```
fseek(fp, 0L, SEEK_END);
```

В обоих примерах смещение задано явно в виде нулевой десятичной константы 0L типа long.

При использовании сложных типов данных (таких, как структура) можно перемещаться в потоке (файле) на то количество байтов, которое занимает этот тип данных. Пусть, например, определена структура:

```
struct str
{
    ….
} st;
```

Тогда при следующем обращении к функции fseek() указатель текущей позиции в потоке будет перемещен на одну структуру назад относительно текущей позиции:

```
fseek(fp, (long)sizeof(st), SEEK_CUR);
```

Кроме рассмотренной функции fseek(), в библиотеке функций языка Си находятся следующие функции для работы с указателями текущей позиции в потоке:

```
long ftell(FILE *) - получить значение указателя текущей позиции в потоке;
void rewind(FILE *) - установить указатель текущей позиции в потоке на начало потока.
```

Необходимо иметь в виду, что недопустимо использовать функции работы с указателем текущей позиции в потоке для потока, связанного не с файлом, а с устройством. Поэтому применение описанных выше функций с любым из стандартных потоков приводит к неопределенным результатам.

Требования к лабораторной работе:

- Многофайловый проект.
- Использование заголовочных файлов.
- Динамическое выделение памяти.
- Использование функций для работы со строками языка.
- Запрещается использовать глобальные переменные.

Задания к лабораторной работе:

Номер варианта задания соответствует заданию, полученному вами ранее. Для всех вариантов необходимо реализовать седующие функциональные возможности:

1) Считывание данных в массив структур из текстового файла structures.txt. Структура этого файла имеет вид:

```
10 // Количество элементов
Петров В.С. ИВТ-091 5 3 4 4 4
Иванов И.П. ИВТ-091 3 3 3 4 5
…
```

Под первое поле отводится не менее 30 позиций, каждое следующее поле занимает не менее 10 позиций. Функция должна преобразовывать данные из символьного типа в строковый, если это необходимо.

2) Запись данных в бинарный файл. Структура файла: первая позиция – количество записей (целое число), далее без пропусков размещаются данные.
3) Реализовать функцию, которая извлекает из этого бинарного файла данные о переменной структурного типа с заданными с клавиатуры свойствами (например, ФИО студента) в структуру требуемого типа.
4) Реализовать функцию, извлекающую данные из бинарного файла по номеру требуемой записи. Функция должна возвращать true, если чтение прошло успешно, и false в противном случае.

Список заданий:

Вариант 1

1. Описать структуру с именем STUDENT, содержащую следующие поля:

NAME – фамилия и инициалы;
GROUP – номер группы;
SES – успеваемость (массив из 5 элементов).

2. Написать программу, выполняющую следующие действия:
ввод с клавиатуры данных в массив STUD1, состоящий из десяти структур типа STUDENT; записи должны быть упорядочены по возрастанию содержимого поля GROUP; вывод на дисплей фамилий и номеров групп для всех студентов, включенных в массив, если средний балл студента больше 4,0; если таких студентов нет, вывести соответствующее сообщение.

Вариант 2

1. Описать структуру с именем STUDENT, содержащую следующие поля:

NAME – фамилия и инициалы;
GROUP – номер группы;
SES – успеваемость (массив из 5 элементов).

2. Написать программу, выполняющую следующие действия:
ввод с клавиатуры данных в массив STUD1, состоящий из десяти структур типа STUDENT; записи должны быть упорядочены по возрастанию среднего бала; вывод на дисплей фамилий и номеров групп для всех студентов, имеющих оценки 4 и 5; если таких студентов нет, вывести соответствующее сообщение.

Вариант 3

1. Описать структуру с именем STUDENT, содержащую следующие поля:

NAME – фамилия и инициалы;
GROUP – номер группы;
SES – успеваемость (массив из 5 элементов).

2. Написать программу, выполняющую следующие действия:

ввод с клавиатуры данных в массив STUD1, состоящий из десяти структур типа STUDENT; записи должны быть упорядочены по алфавиту; вывод на дисплей фамилий и номеров групп для всех студентов, имеющих хотя бы одну оценку 2; если таких студентов нет, вывести соответствующее сообщение.

Вариант 4

1. Описать структуру с именем AEROFLOT, содержащую следующие поля:
   NAZN – название пункта назначения рейса;
   NUMR – номер рейса;
   TIP – тип самолета.
2. Написать программу, выполняющую следующие действия:
ввод с клавиатуры данных в массив AIRPORT, состоящий из 7 элементов типа AEROFLOT; записи должны быть упорядочены по возрастанию номера рейса; вывод на дисплей номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры; если таких рейсов нет, вывести соответствующее сообщение.

Вариант 5

1. Описать структуру с именем AEROFLOT, содержащую следующие поля:
   NAZN – название пункта назначения рейса;
   NUMR – номер рейса;
   TIP – тип самолета.
2. Написать программу, выполняющую следующие действия:
ввод с клавиатуры данных в массив AIRPORT, состоящий из 7 элементов типа AEROFLOT; записи должны быть размещены в алфавитном порядке по названиям пунктов назначения; вывод на дисплей пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры; если таких рейсов нет, вывести соответствующее сообщение.

Вариант 6

1. Описать структуру с именем WORKER, содержащую следующие поля:

NAME – фамилия и инициалы работника;
POS – название занимаемой должности;
YEAR – год поступления на работу.

2. Написать программу, выполняющую следующие действия:
ввод с клавиатуры данных в массив TABL, состоящий из 10 структур типа WORKER; записи должны быть размещены в алфавитном порядке; вывод на дисплей фамилий работников, чей стаж работы в организации превышает значение, введенное с клавиатуры; если таких работников нет, вывести соответствующее сообщение.

Вариант 7

1. Описать структуру с именем TRAIN, содержащую следующие поля:

NAZN – название пункта назначения;
NUMR – номер поезда;
TIME – время отправления.

2. Написать программу, выполняющую следующие действия:
ввод с клавиатуры данных в массив RASP, состоящих из 8 элементов типа TRAIN; записи должны быть размещены в алфавитном порядке по названиям пунктов назначения; вывод на дисплей информации о поездах, отправляющихся после введенного с клавиатуры времени; если таких поездов нет, вывести соответствующее сообщение.

Вариант 8

1. Описать структуру с именем TRAIN, содержащую следующие поля:

NAZN – название пункта назначения;
NUMR – номер поезда;
TIME – время отправления.

2. Написать программу, выполняющую следующие действия:
ввод с клавиатуры данных в массив RASP, состоящих из 6 элементов типа TRAIN; записи должны быть упорядочены по времени отправления поезда; вывод на дисплей информации о поездах, направляющихся в пункт, название которого введено с клавиатуры; если таких поездов нет, вывести соответствующее сообщение.

Вариант 9

1. Описать структуру с именем TRAIN, содержащую следующие поля:

NAZN – название пункта назначения;

NUMR – номер поезда;

TIME – время отправления.

2. Написать программу, выполняющую следующие действия:

ввод с клавиатуры данных в массив RASP, состоящих из 8 элементов типа TRAIN; записи должны быть упорядочены по номерам поездов; вывод на дисплей информации о поезде, номер которого введен с клавиатуры; если таких поездов нет, вывести соответствующее сообщение.

Вариант 10

1. Описать структуру с именем MARSH, содержащую следующие поля:

BEGST – название начального пункта маршрута;

TERM – название конечного пункта маршрута;

NUMER – номер маршрута.

2. Написать программу, выполняющую следующие действия:

ввод с клавиатуры данных в массив TRAFIC, состоящих из 8 элементов типа MARSH; записи должны быть упорядочены по номерам маршрутов; вывод на дисплей информации о маршруте, номер которого введен с клавиатуры; если таких маршрутов нет, вывести соответствующее сообщение.

Вариант 11

1. Описать структуру с именем MARSH, содержащую следующие поля:

BEGST – название начального пункта маршрута;

TERM – название конечного пункта маршрута;

NUMER – номер маршрута.

2. Написать программу, выполняющую следующие действия:

ввод с клавиатуры данных в массив TRAFIC, состоящих из 8 элементов типа MARSH; записи должны быть упорядочены по номерам маршрутов; вывод на дисплей информации о маршрутах, которые начинаются или кончаются в пункте, название которого

введено с клавиатуры; если таких маршрутов нет, вывести соответствующее сообщение.

Вариант 12

1. Описать структуру с именем NOTE, содержащую следующие поля:
   NAME – фамилия, имя;
   TELE – номер телефон;
   BDAY – день рождения (массив из 3 чисел).
2. Написать программу, выполняющую следующие действия:
ввод с клавиатуры данных в массив BLOCKNOTE, состоящих из 8 элементов типа NOTE; записи должны быть упорядочены по датам дней рождения; вывод на дисплей информации о человеке, номер телефона которого введен с клавиатуры; если такого нет, вывести соответствующее сообщение.

Вариант 13

1. Описать структуру с именем NOTE, содержащую следующие поля:
   NAME – фамилия, имя;
   TELE – номер телефон;
   BDAY – день рождения (массив из 3 чисел).
2. Написать программу, выполняющую следующие действия:
ввод с клавиатуры данных в массив BLOCKNOTE, состоящих из 8 элементов типа NOTE; записи должны быть упорядочены по алфавиту; вывод на дисплей информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры; если таких нет, вывести соответствующее сообщение.

Вариант 14

1. Описать структуру с именем NOTE, содержащую следующие поля:
   NAME – фамилия, имя;
   TELE – номер телефон;
   BDAY – день рождения (массив из 3 чисел).
2. Написать программу, выполняющую следующие действия:
ввод с клавиатуры данных в массив BLOCKNOTE, состоящих из 8 элементов типа NOTE; записи должны быть упорядочены по трем первым цифрам номера телефона; вывод на дисплей информации о

человеке, чья фамилия введена с клавиатуры; если такого нет, вывести соответствующее сообщение.

Вариант 15

1. Описать структуру с именем ZNAK, содержащую следующие поля:
   NAME – фамилия, имя;
   ZODIAC – номер телефон;
   BDAY – день рождения (массив из 3 чисел).
2. Написать программу, выполняющую следующие действия:
ввод с клавиатуры данных в массив BOOK, состоящих из 8 элементов типа ZNAK; записи должны быть упорядочены по датам дней рождения; вывод на дисплей информации о человеке, чья фамилия введена с клавиатуры; если такого нет, вывести соответствующее сообщение.

Вариант 16

1. Описать структуру с именем ZNAK, содержащую следующие поля:
   NAME – фамилия, имя;
   ZODIAC – номер телефон;
   BDAY – день рождения (массив из 3 чисел).
2. Написать программу, выполняющую следующие действия:
ввод с клавиатуры данных в массив BOOK, состоящих из 8 элементов типа ZNAK; записи должны быть упорядочены по датам дней рождения; вывод на дисплей информации о людях, родившихся под знаком, наименование которого введено с клавиатуры; если таких нет, вывести соответствующее сообщение.

Вариант 17

1. Описать структуру с именем ZNAK, содержащую следующие поля:
   NAME – фамилия, имя;
   ZODIAC – номер телефон;
   BDAY – день рождения (массив из 3 чисел).
2. Написать программу, выполняющую следующие действия:
ввод с клавиатуры данных в массив BOOK, состоящих из 8 элементов типа ZNAK; записи должны быть упорядочены по знакам Зодиака; вывод на дисплей информации о людях, родившихся под знаком,

название которого введено с клавиатуры; если таких нет, вывести соответствующее сообщение.

Вариант 18

1. Описать структуру с именем PRICE, содержащую следующие поля:
   TOVAR – название товара;
   MAG – название магазина, в котором продается товар;
   STOIM – стоимость товара в руб.
2. Написать программу, выполняющую следующие действия:
ввод с клавиатуры данных в массив SPISOK, состоящих из 8 элементов типа PRICE; записи должны быть размещены в алфавитном порядке по названиям товаров; вывод на дисплей информации о товаре, название которого введено с клавиатуры; если таких товаров нет, вывести соответствующее сообщение.

Вариант 19

1. Описать структуру с именем PRICE, содержащую следующие поля:
   TOVAR – название товара;
   MAG – название магазина, в котором продается товар;
   STOIM – стоимость товара в руб.
2. Написать программу, выполняющую следующие действия:
ввод с клавиатуры данных в массив SPISOK, состоящих из 8 элементов типа PRICE; записи должны быть размещены в алфавитном порядке по названиям магазинов; вывод на дисплей информации о товарах, продающихся в магазине, название которого введено с клавиатуры; если такого магазина нет, вывести соответствующее сообщение.

Вариант 20

1. Описать структуру с именем ORDER, содержащую следующие поля:
   PLAT – расчетный счет плательщика;
   POL – расчетный счет получателя;
   SUMMA – перечисляемая сумма в руб.
2. Написать программу, выполняющую следующие действия:
ввод с клавиатуры данных в массив SPISOK, состоящих из 8 элементов типа ORDER; записи должны быть размещены в

алфавитном порядке по расчетным счетам плательщиков; вывод на дисплей информации о сумме, снятой с расчетного счета плательщика, введенного с клавиатуры; если такого расчетного счета нет, вывести соответствующее сообщение.

Рекомендуемая литература

1) Подбельский В. В., Фомин С. С. Программирование на языке С М.: Финансы и статистика, 2004. - 600 с.
2) Павловская Т. А. С/С++. - Программирование на языке высокого уровня. СПб.: Питер, 2002. - 464 с.
3) Павловская Т. А.9 Щупак Ю.А. С/С++. Структурное программирование: Практикум. СПб.: Питер, 2002. - 240 с.
4) Голуб А. И. С и С++. Правила программирования. - М.: Бином, 1996. - 272 с.