

**Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Севастопольский государственный университет»**

**Методические указания
к лабораторным работам по дисциплине
«Алгоритмизация и программирование»
для студентов дневной и заочной формы обучения направлений
09.03.02 – «Информационные системы и технологии»
и 09.03.03 – «Прикладная информатика»**

2 часть

Севастополь

2022

УДК 004.42 (075.8)

Методические указания к лабораторным работам по дисциплине «Алгоритмизация и программирование» для студентов дневной и заочной формы обучения направлений 09.03.02 — «Информационные системы и технологии» и 09.03.03 — «Прикладная информатика», часть 2 / Сост. В. Н. Бондарев, Т. И. Сметанина, А. Ю. Абрамович. — Севастополь: Изд-во СевГУ, 2022. — 118 с.

Методические указания предназначены для проведения лабораторных работ и обеспечения самостоятельной подготовки студентов по дисциплине «Алгоритмизация и программирование». Целью методических указаний является обучение студентов основным принципам структурного программирования, навыкам разработки программ на языках С/С++.

Методические указания составлены в соответствии с требованиями программы дисциплины «Алгоритмизация и программирование» для студентов направлений 09.03.02 — «Информационные системы и технологии» и 09.03.03 — «Прикладная информатика» и утверждены на заседании кафедры «Информационные системы» протокол № ____ от

Допущено научно-методической комиссией института информационных технологий и систем управления в технических системах в качестве методических указаний.

1 ЛАБОРАТОРНАЯ РАБОТА №1

ПРОГРАММИРОВАНИЕ ОПЕРАЦИЙ НАД СТРОКАМИ И ФАЙЛАМИ

1.1 Цель работы

Изучение основных операций над строками и файлами, программирование операций обработки строк текстовых файлов, исследование свойств файловых указателей.

1.2 Краткие теоретические сведения

1.2.1 Строки

В языке C *строка* представляет собой массив символов, завершающийся символом с кодом ноль ('\\0'). Этот завершающий (терминирующий) ноль в конце строки также называют терминирующим нулем или терминатором – признаком конца строки. При объявлении строкового массива и работе с ним необходимо принимать во внимание наличие терминатора ('\\0') в конце строки, отводя под строку на один байт больше максимального количества символов. Удобно задавать длину строки именованной константой:

```
const int len_str = 80;
char stroka[len_str];
```

В такой переменной **stroka** можно хранить не 80 символов, а только 79.

Строки можно при описании *инициализировать* строковой константой. При этом нуль-символ ('\\0') добавляется в строку автоматически:

```
char a[10] = "язык си";
```

Если строка при определении инициализируется, её длину можно не указывать:

```
char a[] = "язык си";
```

В этом случае компилятор сам выделит память достаточную для размещения всех символов и нуль-литеры.

Для размещения строки в *динамической памяти* необходимо описать указатель на **char**, а затем выделить память с помощью функции `malloc()` или `new`:

```
char *s1 = (char*) malloc(m*sizeof(char));
char *s2 = new char[m];
```

Здесь `m` – переменная, определяющая длину строки.

Для освобождения памяти соответственно:

```
free(s1);
delete []s2; // всегда [] при освобождении памяти массива
```

Для ввода-вывода строк можно использовать как *объекты* `cin` и `cout` языка C++, так и функции языка C.

В языке C для ввода и вывода строк используются функции `scanf` и `printf` со спецификацией `%s`:

```

const int n = 80;
int main() {
    char s[n];
    scanf("%s", s); // считывание до пробела, но не более 79
СИМВОЛОВ
    printf("%s", s);
}

```

В функции `scanf` перед `s` операция взятия адреса (`&`) опущена, потому что имя массива является указателем на его начало. Функция `scanf` выполняет ввод до первого пробельного символа. Чтобы ввести строку, состоящую из нескольких слов, используется спецификация `%c`:

```
scanf("%20c", s);
```

В этом случае количество считываемых символов может быть только константой (и равно 20), и короткие строки придется при вводе дополнять пробелами до требуемой длины строки.

```

const int n = 80;
int main() {
    char s[n];
    scanf("%10c", s); // считывание 10 символов
    printf("%s", s);
}

```

Библиотека языка C содержит функции, специально предназначенные для ввода (`gets`) и вывода (`puts`) строк.

Функция `gets(s)` читает символы с клавиатуры до появления символа `'\n'` и помещает их в строку `s` (сам символ `'\n'` в строку не включается, вместо него вносится `'\0'`). Функция возвращает указатель на `s`, а случае ошибки или конца файла – `NULL`. Однако функция `gets` не контролирует количество введенных символов в `s`, что может приводить к выходу за границы `s`.

Функция `puts(s)` выводит строку `s` на стандартное устройство вывода, заменяя завершающий символ `'\0'` на символ новой строки.

```

const int m = 11;
int main() {
    char t[] = "Язык Си";
    puts(t);
    char *s = new char[m];
    puts("введите не более 10 символов");
    gets(s); //чтение не более 10 символов
    puts(s);
    delete []s;
}

```

Большинство функций работы со строками описаны в заголовочном файле `string.h`. Некоторые из этих функций указаны в таблице 1.1. Здесь аргументы `s` и `t` принадлежат типу `char*`, `cs` и `ct` – типу `const char*`, `n` – типу `size_t`, `c` – типу `int`, приведенному к `char`.

Например, чтобы строке `s` присвоить значение строки `t` можно использовать функции `strcpy` или `strncpy`:

```

const int m = 10;
char t[] = "Язык Си";
char *s = new char[m];
strcpy(s, t);
strncpy(s, t, strlen(t)+1);

```

Функция `strcpy(s, t)` копирует все символы из строки `t`, включая и нуль-литеру, в строку, на которую указывает `s`. Функция `strncpy(s, t, n)` выполняет то же самое, но не более `n` символов. При этом если нуль-символ встретится раньше, копирование прекращается, и оставшиеся символы строки `s` заполняются нуль-символами. Если `n` меньше или равно длине строки `t`, завершающая нуль-литера не добавляется. Обе функции возвращают указатель на результирующую строку.

Функция `strlen(t)` возвращает фактическую длину строки `t`, не включая нуль-литеру. Длина строки это количество ненулевых символов в строке, считая от начала строки, до первого ноля.

Таблица 1.1 — Функции для обработки строк

char* strcpy(s,ct)	копирует строку <i>ct</i> в строку <i>s</i> ; возвращает <i>s</i>
char* strcat(s,ct)	соединяет строку <i>s</i> и <i>ct</i>
char* strcmp(cs,ct)	сравнивает строки <i>cs</i> и <i>ct</i>
char* strchr(cs,c)	возвращает указатель на первое вхождение символа <i>c</i> в строку <i>cs</i>
char* strrchr(cs,c)	возвращает указатель на последнее вхождение символа <i>c</i> в строку <i>cs</i>
size_t strspn(cs,ct)	возвращает число символов в начале строки <i>cs</i> , ни один из которых не входит в строку <i>ct</i>
size_t strcspn(cs,ct)	возвращает число символов в начале строки <i>cs</i> , которые принадлежат множеству символов из строки <i>ct</i>
char* strpbrk(cs,ct)	возвращает указатель в <i>cs</i> на первую литеру, которая совпала с одной из литер, входящих в <i>ct</i>
char* strstr(cs,ct)	возвращает указатель на первое вхождение строки <i>ct</i> в строку <i>cs</i>
size_t strlen(cs)	возвращает длину строки <i>cs</i>
char* strtok(s,ct)	ищет в строке <i>s</i> лексему, ограниченную литерами из строки <i>ct</i>

Рассмотрим ввод и вывод строк с использованием объектов `cin` и `cout` языка C++:

```

#include <iostream>
using namespace std;
const int n = 80;
int main() {
    char s[n];
    cin>>s;
    cout<<s<<endl;
}

```

```
}
```

Таким образом, строка вводится аналогично числовым переменным. Однако, если ввести строку, состоящую из нескольких слов, разделенных пробелами, то будет введено только первое слово. Это связано с тем, что ввод выполняется до первого пробельного символа (' ', '\t', '\n'). Поэтому для ввода строки лучше использовать функции `getline` или `get`:

```
#include <iostream>
using namespace std;
const int n = 80;
int main() {
    char s[n];
    cin.getline(s,n); // 1 способ
    cout<<s<<endl;
    cin.get(s,n); // 2 способ
    cout<<s<<endl;
}
```

Функция `getline` считывает из входного потока $n-1$ символов или менее (если символ '\n' встретится раньше) и записывает их в строковую переменную `s`. Сам символ '\n' тоже считывается, но не записывается в `s`, вместо него записывается нуль-литера ('\0'). Если в потоке более $n-1$ символов, то следующий ввод будет выполняться, начиная с первого нечитанного символа.

Функция `get` работает аналогично, но оставляет в потоке символ '\n'. Поэтому перед повторным вызовом `get` необходимо удалять символ '\n' из входного потока. Для этого нужно вызвать `get` без параметров – `cin.get()`.

1.2.2 Функции файлового ввода-вывода

Язык C++ унаследовал от языка C библиотеку стандартных функций ввода-вывода. Функции ввода-вывода объявлены в заголовочном файле `<stdio.h>`. Операции ввода-вывода осуществляются с файлами. Файл может быть *текстовым* или *бинарным* (двоичным). Различие между ними заключается в том, что текстовый файл разбит на строки. Признаком конца строки является пара символов '\r' (возврат каретки) и '\n' (перевод строки). При вводе или выводе значений из текстового файла они *требуют преобразований* во внутреннюю форму хранения этих значений в соответствии с их типами, объявленными в программе. Бинарный файл – это просто последовательность символов. При вводе-выводе информации в бинарные файлы никаких преобразований не выполняется. Поэтому работа с бинарными файлами выполняется быстрее.

В языке C реализован так называемый *поточный ввод-вывод*. Термин поток происходит из представления процесса ввода-вывода информации в файл в виде последовательности или потока байтов. Все потоковые функции ввода-вывода обеспечивают буферизированный, *форматированный* или *неформатированный ввод-вывод*.

Перед тем, как выполнять операции ввода или вывода в файловый поток его следует открыть с помощью функции `fopen()`. Эта функция имеет следующий прототип:

```
FILE *fopen(const char *filename, const char *mode);
```

Функция `fopen()` открывает файл с именем `filename` и возвращает указатель на файловый поток, используемый в последующих операциях с файлом. Параметр **mode** является строкой, задающей режим, в котором открывается файл. Он может принимать значения, указанные в таблице 1.2.

Таблица 1.2 — Режимы открытия файлов

Режим	Описание режима
r	Файл открывается только для чтения
w	Файл создается только для записи. Если файл с этим именем уже существует, он будет перезаписан. Чтение из файла не разрешено.
a	Режим добавления записей (append). Файл открывается только для записи в конец или создается только для записи, если он еще не существует. Чтение не разрешено.

Чтобы указать, что данный файл открывается или создается как текстовый, добавьте символ `t` в строку режима (например, `"rt"`, `"w+t"` и т.п.).

В случае успеха функция **fopen** возвращает указатель на открытый поток, в случае ошибки – `NULL`. Например:

```
FILE *fptr = fopen("mytxt.txt", "rt");
```

Здесь объявляется переменная `fptr` – указатель на файловый поток и выполняется её инициализация с помощью функции `fopen()`.

Для завершения работы с потоком он должен быть закрыт с помощью функции `fclose()`:

```
fclose(fptr);
```

Рассмотрим функции, осуществляющие ввод-вывод в файловый поток. Функция `fgetc()` имеет следующий прототип:

```
int fgetc(FILE *fptr);
```

Она осуществляет ввод символа из файлового потока `fptr`. В случае успеха функция возвращает код символа. Если делается попытка прочесть конец файла или произошла ошибка, то возвращается `EOF`.

Функция `fputc()` имеет следующий прототип:

```
int fputc(int c, FILE *fptr);
```

Она осуществляет вывод символа в поток.

Функция `fgets()` имеет следующий прототип:

```
char *fgets(char *s, int n, FILE *fptr);
```

Она осуществляет чтение строки символов из файлового потока в строку `s`. Функция прекращает чтение, если прочитано `n-1` символов или встретится символ перехода на новую строку `'\n'`. Если этот символ встретился, то он сохраняется в переменной `s`. В обоих случаях в переменную `s` добавляется символ `'\0'`.

В случае успеха функция возвращает указатель на считанную строку *s*. Если произошла ошибка или считана метка EOF, то она возвращает NULL.

Для записи строки в файл можно использовать функцию **fputs**. При этом символ перехода на новую строку не добавляется, и завершающая нуль-литера в файл не копируется. Функция `fputs()` имеет следующий прототип:

```
int fputs(const *char, FILE *fptr);
```

В случае успеха функция `fputs()` возвращает неотрицательное значение. В противном случае она возвращает EOF.

Форматированный ввод-вывод текстовых файлов организуется в языке C с помощью функций `fscanf()` и `fprintf()`. Эти функции аналогичны функциям `scanf()` и `printf()` соответственно, с той лишь разницей, что их первым аргументом является указатель на файл, открытый в соответствующем режиме.

Функция `feof()` осуществляет проверку достижения метки конца файла. Она имеет прототип:

```
int feof(FILE *fptr);
```

Функция возвращает 0, если конец файла не достигнут.

Рассмотрим пример файлового ввода и вывода с помощью функций `fscanf()` и `fprintf()`:

```
#include <stdio.h>
int main() {
    int i, x;
    FILE *in, *out; // описание указателей на файлы
    if ((in = fopen("c:\\1\\old.txt", "rt")) == NULL) {
        fprintf(stderr, "Не могу открыть входной файл \n");
        return 1;
    }
    if ((out = fopen("c:\\1\\new.txt", "wt")) == NULL) {
        fprintf(stderr, "Не могу открыть выходной файл \n");
        return 1;
    }
    i = 0;
    while (fscanf(in, "%d", &x) != EOF) { // чтение чисел
        i++;
        fprintf(out, "%d\t%d\n", i, x); // печать чисел
    }
    fclose(in); //закрытие файлов
    fclose(out);
    return 0;
}
```

Программа копирует целые числа из входного файла `old.txt` в выходной файл `new.txt`. При этом в выходной файл записываются также порядковые номера чисел. Копирование выполняется до тех пор, пока не будет встречена метка конец файла EOF. Если входной или выходной файл нельзя открыть, то программа выводит соответствующее сообщение в стандартный поток `stderr`, который по умолчанию связан с пользовательским терминалом и предназначен для вывода сообщений об ошибках.

Рассмотрим еще один пример файлового ввода и вывода с помощью функций

`fscanf()` и `fprintf()`:

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

```
int main() {
```

```
    char x;
```

```
    FILE *in,*out; // описание указателей на файлы
```

```
    in = fopen("c:\\1\\old.txt", "rt");
```

```
    if (in == NULL) {
```

```
        fprintf(stderr, " Не могу открыть входной файл \n");
```

```
        return 1;
```

```
    }
```

```
    out = fopen("c:\\1\\new.txt", "wt");
```

```
    if (out== NULL) {
```

```
        fprintf(stderr, "Не могу открыть выходной файл \n");
```

```
        return 1;
```

```
    }
```

```
    while (1) { // вечный цикл
```

```
        fscanf(in, "%c", &x);
```

```
        if (feof(in)) break; // выход из цикла
```

```
        fprintf(out, "%c", x);
```

```
    }
```

```
    fclose(in);
```

```
    fclose(out);
```

```
    return 0;
```

```
}
```

Программа копирует символы из входного файла `old.txt` в выходной файл `new.txt` в цикле пока не встретит метку конца файла.

1.2.3 Файловый ввод-вывод с использованием классов языка C++

Для реализации ввода-вывода с использованием классов языка C++ в программу следует включить заголовочный файл `<fstream>`. Для осуществления операций с файлами библиотека ввода-вывода C++ предусматривает три класса:

`ifstream` – потоковый класс для ввода из файла;

`ofstream` – потоковый класс для вывода в файл;

`fstream` – потоковый класс для ввода-вывода в файл.

Для создания потока ввода, открытия файла и связывания его с потоком можно использовать следующую форму конструктора класса `ifstream`:

```
ifstream fin("text.txt", ios::in);
```

Здесь определяется объект `fin` класса входных потоков `ifstream`. С этим объектом можно работать так же, как со стандартными объектами `cin` и `cout`, то есть использовать операции помещения в поток `<<` и извлечения из потока `>>`, а также рассмотренные выше функции `get`, `getline`. Например:

```
const len = 80;
```

```
char s[len];
```

```
fin.getline(s, len);
```

Предполагается, что файл с именем `text.txt` находится в том же каталоге, что и текст программы, иначе следует указать полный путь, дублируя символ обратной косой черты, так как иначе он будет иметь специальное значение, например:

```
ifstream fin("c:\\work\\text.txt", ios::in);
```

Второй параметр этого конструктора означает режимы открытия файла. В таблице 1.3 приведены возможные режимы открытия и их значение.

Например, для открытия файла вывода в режиме добавления можно использовать инструкцию:

```
ofstream fout("out.txt", ios::out | ios::app);
```

Файлы, которые открываются для вывода, создаются, если они не существуют.

Таблица 1.3 — Режимы открытия и их значения

Режим	Назначение
<code>ios::in</code>	Открыть для чтения
<code>ios::out</code>	Открыть для записи
<code>ios::ate</code>	Начало вывода устанавливается в конец файла (допускает изменение позиции вывода)
<code>ios::app</code>	Открыть для добавления в конец (безотносительно к операциям позиционирования)
<code>ios::trunc</code>	Открыть, удалив содержимое файла
<code>ios::binary</code>	Двоичный режим операций

Если открытие файла завершилось неудачей, объект, соответствующий потоку, будет возвращать значение `false`. Например, если предыдущая инструкция завершилась неудачей, то это можно проверить так:

```
if (!fout)
{ cout<<"Файл не открыт \n"; }
```

Если при открытии файла не указан режим `ios::binary`, то файл открывается как текстовый. В этом случае можно использовать при работе с файлом функции ввода-вывода, принятые в языке C, такие, как `fprintf()` и `fscanf()`.

Для проверки достигнут ли конец файла или нет, можно использовать функцию `eof()` класса `ios`.

Завершив операции ввода-вывода, необходимо закрыть файл, вызвав функцию `close()`:

```
fout.close();
```

Заккрытие файла происходит автоматически при выходе объекта потока из области видимости.

1.2.4 Пример программы обработки текстового файла

Написать программу, которая построчно копирует содержимое файла `text.txt` в файл `out.txt`.

```
#include <fstream>
#include <iostream>
```

```

using namespace std;
const int len = 80;
int main() {
    ifstream fin("text.txt", ios::in);
    if (!fin) {
        cout<<"ошибка открытия";
        return 1;
    }
    ofstream fout("out.txt", ios::out | ios::app);
    if (!fout) {
        cout<<"ошибка открытия";
        return 1;
    }

    char s[len];
    while (1) {
        fin.getline(s, len);
        if (fin.eof()) break;
        fout<<s<<endl;
    }
    fin.close();
    fout.close();
}

```

В заключение отметим, что файл с тестовым примером должен содержать нескольких строк различной длины.

Программа, написанная с использованием функций ввода-вывода языка C, а не классов языка C++ часто является более быстросействующей, но менее защищенной от ошибок ввода данных.

1.3 Варианты заданий

Написать две программы: в первой ввод-вывод осуществлять с помощью средств C, во второй ввод-вывод осуществлять с помощью классов C++.

Вариант 1

Написать программу, которая считывает текст из файла и определяет количество символов X в каждой строке.

Вариант 2

Написать программу, которая считывает текст из файла и заменяет все восклицательные знаки многоточием.

Вариант 3

Написать программу, которая считывает текст из файла и определяет количество арифметических операций (+, -, *, /) в каждой строке.

Вариант 4

Написать программу, которая считывает текст из файла и определяет номер самой длинной строки.

Вариант 5

Написать программу, которая считывает текст из файла и определяет количество строк, содержащих хотя бы одну из арифметических операций (+, -, *, /).

Вариант 6

Написать программу, которая считывает текст из файла и определяет номер самой короткой строки.

Вариант 7

Написать программу, которая считывает текст из файла и определяет количество строк, не содержащих символы препинания (. : ; ! ? ,).

Вариант 8

Написать программу, которая считывает текст из файла и определяет количество строк, начинающихся и заканчивающихся одним и тем же символом.

Вариант 9

Написать программу, которая считывает текст из файла и определяет количество строк, начинающихся с пробела.

Вариант 10

Написать программу, которая считывает текст из файла и заменяет символ А на символ В.

Вариант 11

Написать программу, которая считывает текст из файла и удаляет все символы А.

Вариант 12

Написать программу, которая считывает текст из файла и удаляет все фразы в фигурных скобках {}. Количество открывающихся скобок равно количеству закрывающихся скобок.

Вариант 13

Написать программу, которая считывает текст из файла и заменяет любую последовательность точек одним символом *. Например исходный текст: "9.00...1.....27.3.4..."; результат: "9*00*1*27*3*4*".

Вариант 14

Написать программу, которая считывает текст из файла и заменяет цифры их названиями (например: 1->один, 2->два т.д.).

Вариант 15

Написать программу, которая считывает текст из файла и заменяет символ «+» словом «ПЛЮС».

Вариант 16

Написать программу, которая считывает текст из файла и определяет количество строк, начинающихся с цифры.

Вариант 17

Написать программу, которая считывает текст из файла и определяет количество строк, начинающихся и заканчивающихся цифрой.

Вариант 18

Написать программу, которая считывает текст из файла и определяет: верно ли, что в файле количество открытых скобок равно количеству закрытых скобок.

Вариант 19

Написать программу, которая считывает текст из файла и определяет количество строк, содержащих два пробела подряд.

Вариант 20

Написать программу, которая считывает текст из файла и заменяет строчные гласные буквы на заглавные.

Вариант 21

Написать программу, которая считывает текст из файла и заменяет заглавные гласные буквы на строчные.

Вариант 22

Написать программу, которая считывает текст из файла и удаляет все, кроме фраз, заключенных в фигурные скобки {}. Количество открывающихся скобок равно количеству закрывающихся скобок.

Вариант 23

Написать программу, которая считывает текст из файла и определяет количество строк, начинающихся с гласной буквы.

Вариант 24

Написать программу, которая считывает текст из файла и определяет количество строк, начинающихся и заканчивающихся пробелом.

Вариант 25

Написать программу, которая считывает текст из файла и определяет количество слов в каждой строке.

Вариант 26

Написать программу, которая считывает текст из файла и определяет для каждой строки файла количество символов '*’.

Вариант 27

Написать программу, которая считывает текст из файла и определяет количество запятых в каждой строке.

Вариант 28

Написать программу, которая считывает текст из файла и определяет для каждой строки файла номер первой запятой.

Вариант 29

Написать программу, которая считывает текст из файла и определяет для каждой строки файла номер последней запятой.

Вариант 30

Написать программу, которая считывает текст из файла и определяет количество строк, состоящих из одинаковых символов.

1.4 Порядок выполнения работы

1.4.1 В ходе самостоятельной подготовки изучить основы работы со строками и файлами в языках C/C++.

1.4.2 Выбрать способ представления исходных данных задачи и результатов.

1.4.3 Разработать алгоритм решения задачи, разбив его при необходимости на отдельные функции.

1.4.4 Разработать две программы: одну на языке C, другую на языке C++.

1.4.5 Разработать тестовые примеры, следуя указаниям, изложенным в разделе 3.

1.4.6 Выполнить отладку программы.

1.4.7 Получить результаты работы программы и исследовать её свойства для различных режимов работы, сформулировать выводы.

1.5 Содержание отчета

Цель работы, вариант задания, структурные схемы алгоритма решения задачи с описанием, тексты программ с комментариями, тестовые примеры, выводы.

1.6 Контрольные вопросы

1.6.1 Приведите примеры описания и инициализации строк в языке C.

1.6.2 Как хранится строка языка C в памяти ЭВМ?

1.6.3 Объясните назначение и запишите прототипы функций `gets`, `puts`.

1.6.4 Как выделить динамическую память под строку?

1.6.5 Как выполнить ввод и вывод строки с помощью функций `scanf()` и `printf()`?

1.6.6 Как выполнить ввод и вывод строк с помощью объектов `cin` и `cout`?

1.6.7 Как ввести строку с помощью функций `getline` и `get` объекта `cin`?

1.6.8 Для чего предназначены функции `strcpy()` и `strncpy()`? Как их вызывать?

1.6.9 Для чего предназначена функция `strcmp()`? Как её вызывать? Какие результаты она возвращает?

1.6.10 Объясните, как открыть файл? Какие имеются режимы открытия файлов в языке C?

1.6.11 Опишите функции `fgets` и `fputs`. В чем их отличие от функций `gets` и `puts`?

1.6.12 Приведите примеры чтения и записи значений в файл с помощью функций `fscanf()` и `fprintf()`?

1.6.13 Как открыть файловый поток в режиме чтения в языке C++?

1.6.14 Как открыть файловый поток в режиме записи в языке C++?

2 ЛАБОРАТОРНАЯ РАБОТА №2

ПРОГРАММИРОВАНИЕ ОПЕРАЦИЙ НАД СТРУКТУРАМИ И БИНАРНЫМИ ФАЙЛАМИ

2.1 Цель работы

Исследование особенностей обработки бинарных файлов, хранящих структурные типы данных.

2.2 Краткие теоретические сведения

2.2.1 Бинарные файлы

Содержимое файлов представляет собой набор байтов (чисел от 0 до 255). Для интерпретации этих данных как полезной для человека информации нужно знать правила кодирования – декодирования. Например, для представления текста используемые символы были пронумерованы и сведены в таблицы.

Бинарные (двоичные) файлы содержат данные в представлении, в котором компьютеру удобнее считывать, обрабатывать и сохранять данные. Обычно в бинарном файле данные находятся в том же виде, в котором они лежат в памяти компьютера. И поэтому скорость обработки данных в бинарном виде гораздо выше, так как отсутствуют промежуточные операции преобразования.

Стандартная библиотека C/C++ для работы с бинарными файлами предоставляет специальные функции.

Рассмотрим основные из них: `fread` и `fwrite`. Эти функции позволяют читать и записывать блоки данных любого типа.

```
size_t fread ( void * считываемое_данное,  
               size_t размер_элемента_байты,  
               size_t количество_элементов,  
               FILE * указатель_на_файл );
```

Функция считывает из файла указанное число элементов (блоков) заданного размера. Размер блока задается в байтах. Функция возвращает число прочитанных элементов. Если число прочитанных элементов не равно заданному, то при чтении возникла ошибка или встретился конец файла.

```
size_t fwrite(const void *буфер,  
              size_t размер_элемента_байты,  
              size_t количество_элементов,  
              FILE * указатель_на_файл );
```

Функция `fwrite()` возвращает количество записанных элементов. Если ошибка не произошла, то возвращаемый результат будет равен значению `количество_элементов`.

Одним из самых полезных применений функций `fread()` и `fwrite()` является чтение и запись данных пользовательских типов, особенно структур. Например:

```

#include <stdio.h>
struct some_struct { /* ... */ };
int main () {
// . . . . .
FILE * fp;
char filename[] = "some_file";
struct some_struct buff[64]; // массив структур из 64-х
    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);
    }
    return 0;
}

```

2.2.2 Позиционирование в файле

Каждый открытый файл имеет так называемый указатель на текущую позицию в файле. Все операции над файлом (чтение и запись) выполняются с этой позиции. При каждом выполнении функции чтения или записи, указатель смещается на количество прочитанных или записанных байт, то есть устанавливается сразу за прочитанным или записанным блоком данных в файле. Это так называемый последовательный доступ к данным. Последовательный доступ удобен, когда необходимо последовательно работать с данными в файле. Но иногда необходимо читать или писать данные в произвольном порядке. Это достигается путем установки указателя чтения/записи на некоторую заданную позицию в файле функцией `fseek()`.

```
int fseek(FILE *stream, long offset, int whence);
```

Параметр `offset` задает количество байт, на которое необходимо сместить указатель от точки отсчета, указанной `whence` (таблица 2.1).

Таблица 2.1 – Возможные значения параметра `whence`

Символьное имя	Значение	Точка отсчета
SEEK_SET	0	Смещение от начала файла
SEEK_CUR	1	Смещение от текущей позиции
SEEK_END	2	Смещение от конца файла

Величина смещения может быть как положительной, так и отрицательной, но нельзя сместиться за пределы начала файла (назад от начала файла).

Для перемещения указателя текущей позиции в файле на начало файла, наряду с функцией `fseek()`, может также быть использована функция `rewind()`:

```
void rewind(FILE * указатель_на _файл);
```

Но кроме перемещения указателя позиции чтения/записи на начало, `rewind()` всегда выполняет сброс состояния ошибки в потоке, если до этого при работе с файлом возникла ошибка.

Узнать текущее значение указателя позиции чтения/записи можно с помощью функции `ftell()`:

```
long int ftell(FILE * указатель_на _файл);
```

Она возвращает позицию в байтах. При возникновении ошибки функция `ftell()` возвращает значение -1.

Совместное использование с другими функциями предоставляет дополнительные возможности – например, чтобы узнать размер файла в байтах, нужно переставить указатель на конец файла, а затем узнать его текущее значение.

2.2.3 Пример программы обработки структур

Имеется структура, описывающая рациональную дробь. В структуре два поля: числитель и знаменатель. Требуется написать программу, которая

- вводит дроби с клавиатуры в бинарный файл;
- упорядочивает его по возрастанию;
- отображает содержимое бинарного файла на экран в читаемом человеком виде.

Исходные данные и результаты.

Исходные данные:

Значение одной дроби представим в виде структуры, состоящей из двух полей: числителя и знаменателя. Оба поля будут целочисленными. Поскольку количество дробей не оговорено, то будем хранить все сведения в бинарном файле, а не массиве.

Результаты:

В результате работы программы требуется вывести на экран отсортированные по возрастанию дроби.

Алгоритм решения задачи.

1. Ввести с клавиатуры числитель и знаменатель дроби в бинарный файл. При этом будем прекращать ввод, если пользователь введет символ '0'.

2. Выполнить сортировку файла по возрастанию, используя возможность прямого доступа к элементам бинарного файла. В качестве метода сортировки будем использовать метод прямого обмена (пузырьковую сортировку);

3. Обеспечить вывод отсортированной последовательности дробей:

Каждый из указанных пунктов алгоритма представим в виде функции, которую можно будет вызывать из основной программы. При этом контроль над порядком вызова функций полностью возлагается на пользователя.

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

```

#include <conio.h>
//-----константы и структуры-----
struct drob {
    int ch,zn;
};
const int size_drob=sizeof(drob); //размер структуры
//-----прототипы функций-----
int create_file(FILE *f); //запись в файл
int sort_file(FILE *f); //сортировка файла
int print_file(FILE *f); //вывод файла

//-----основная функция-----
int main() {
    char c;
    //Открытие существующего файла для чтения и записи в конец
    FILE *f=fopen("data.dat", "rb+");
    if (!f) {
        //Создание нового файла для обновления
        f=fopen("data.dat", "wb+");
        if(!f) {
            puts("Не могу открыть (создать) файл\n");
            return 1;
        }
    }
    //вывод меню и запуск соответствующих функций
    while (1) {
        puts("1- Запись в файл");
        puts("2- Сортировка файла");
        puts("3- Вывод файла");
        puts("5- Выход");
        puts("_____");
        puts("Введите номер пункта меню\n");
        c=getch();
        switch (c) {
            case '1':
                create_file(f);
                break;
            case '2':
                sort_file(f);
                break;
            case '3':
                print_file(f);
                break;
            case '5':
                return 0;
        }
    }
}
//-----запись в файл-----

int create_file(FILE *f) {
    drob d;
    fseek(f,0,SEEK_END); //указатель в конец файла

```

```

while (1) {
    puts("Введите числитель (0-exit)");
    scanf("%d",&d.ch);
    if (d.ch==0)
        return 1;
    puts("Введите знаменатель");
    scanf("%i",&d.zn);
    fwrite(&d,size_drob,1,f);
}
}
//-----вывод файла-----
int print_file(FILE *f) {
    drob d;
    int n;
    rewind(f); //указатель в начало файла
    puts("_____");
    puts("| числитель | знаменатель |");
    do {
        n=fread(&d,size_drob,1,f); //чтение структуры из файла
        if (n<1) break; //если n<1, то конец файла
        printf("| %-13d | %-14d|\n",d.ch, d.zn);
    } while (1);
    puts("_____");
    puts("Нажмите любую клавишу");
    getch();
    return 0;
}
int sort_file(FILE *f) {
    long i,j;
    drob d1,d2;

    fseek(f,0,SEEK_END); //указатель в конец
    long len=ftell(f)/size_drob; //определяем количество записей
    rewind(f); //указатель в начало
    //пузырьковая сортировка
    for(i=len-1; i>=1; i--)
        for (j=0; j<=i-1; j++) {
            fseek(f,j*size_drob,SEEK_SET); //указатель на j-ую запись
            fread(&d1,size_drob,1,f); //читаем запись j в d1
            fread(&d2,size_drob,1,f); //читаем след. запись в d2
            if (d1.ch*d2.zn>d2.ch*d1.zn) {
                //указатель на 2 поз. назад
                fseek(f,(-2)*size_drob,SEEK_CUR);
                //обмен значений
                fwrite(&d2,size_drob,1,f); //сначала записываем d2
                fwrite(&d1,size_drob,1,f); // затем записываем d1
            }
        }
    puts("Сортировка успешна завершена");
    getch();
    return 0;
}

```

Подробные комментарии в тексте программы позволят быстро разобраться в её работе. В тексте основной программы осуществляется либо открытие файла (режим «r+b»), либо создание файла (режим «w+b») для обновления, если файл «data.dat» на диске не обнаружен. Это позволяет один раз ввести исходные данные в файл, а затем его многократно использовать. Если потребуется начать работу с новым файлом, то старый файл следует удалить с диска средствами операционной системы.

Все необходимые действия с файлом выполняются с помощью функций, вызываемых из пунктов меню, отображаемых на экране.

2.3 Варианты заданий

Вариант 1

Описать структуру с именем STUDENT, содержащую следующие поля:

- порядковый номер;
- фамилия и инициалы;
- номер группы;
- успеваемость (не менее трех дисциплин).

Написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из структур типа STUDENT;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на дисплей фамилий и номеров групп для всех студентов, чей средний балл больше 4.0, а если таких студентов нет, то вывести соответствующее сообщение;
- отсортировать записи по возрастанию номера группы.

Вариант 2

Для структуры, указанной в варианте 1, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из структур типа STUDENT;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на дисплей фамилий и номеров групп для всех студентов, имеющих оценки 4 и 5, а если таких студентов нет, то вывести соответствующее сообщение;
- отсортировать записи по возрастанию среднего балла.

Вариант 3

Для структуры, указанной в варианте 1, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из структур типа STUDENT;

- чтение данных из этого файла;
- корректировку данных в файле по номеру записи;
- вывод на дисплей фамилий и номеров групп для всех студентов, имеющих хотя бы одну оценку 2, если таких студентов нет, то вывести соответствующее сообщение;
- отсортировать записи по полю фамилия по алфавиту.

Вариант 4

Описать структуру с именем AEROFLOT, содержащую следующие поля:

- порядковый номер;
- название пункта отправления рейса;
- название пункта назначения рейса;
- номер рейса;
- тип самолета.

Написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа AEROFLOT;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры, а если таких рейсов нет, то выдать на дисплей соответствующее сообщение;
- отсортировать записи по возрастанию номера рейса.

Вариант 5

Для структуры, указанной в варианте 4, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа AEROFLOT;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на экран пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры, а если таких рейсов нет, выдать на дисплей соответствующее сообщение;
- отсортировать записи в алфавитном порядке по названиям пунктов назначения.

Вариант 6

Описать структуру с именем WORKER, содержащую следующие поля:

- порядковый номер;
- фамилия и инициалы работника;
- название занимаемой должности;
- год поступления на работу.

Написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из структур типа WORKER;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на дисплей фамилий работников, чей стаж работы в организации превышает значение, введенное с клавиатуры, если таких работников нет, то вывести на дисплей соответствующее сообщение;
- отсортировать записи по названию занимаемой должности.

Вариант 7

Описать структуру с именем TRAIN, содержащую следующие поля:

- порядковый номер;
- название пункта отправления;
- название пункта назначения;
- номер поезда;
- время отправления (часы и минуты).

Написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа TRAIN;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на экран информации о поездах, отправляющихся после введенного с клавиатуры времени, если таких поездов нет, выдать на дисплей соответствующее сообщение;
- отсортировать записи по алфавиту по названиям пунктов назначения.

Вариант 8

Для структуры, указанной в варианте 7, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа TRAIN;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на экран информации о поездах, направляющихся в пункт, название которого введено с клавиатуры, а если таких поездов нет, то выдать на дисплей соответствующее сообщение;
- отсортировать записи по времени отправления поезда.

Вариант 9

Для структуры, указанной в варианте 7, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа TRAIN;
- чтение данных из этого файла;
- корректировку данных в файле по номеру записи;
- вывод на экран информации о поезде, номер которого введен с клавиатуры, а если таких поездов нет, то выдать на дисплей соответствующее сообщение;
- отсортировать записи по номерам поездов.

Вариант 10

Описать структуру с именем MARSH, содержащую следующие поля:

- порядковый номер;
- название начального пункта маршрута;
- название конечного пункта маршрута;
- номер маршрута.

Написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа MARSH;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на экран информации о маршруте, номер которого введен с клавиатуры, а если таких маршрутов нет, то выдать на дисплей соответствующее сообщение;
- отсортировать записи по номерам маршрутов.

Вариант 11

Для структуры, указанной в варианте 10, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа MARSH; записи должны быть упорядочены по номерам маршрутов;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на экран информации о маршрутах, которые начинаются или оканчиваются в пункте, название которого введено с клавиатуры, если таких маршрутов нет, выдать на дисплей соответствующее сообщение.

Вариант 12

Описать структуру с именем NOTE, содержащую следующие поля:

- порядковый номер;
- фамилия, имя;
- номер телефона;
- дата рождения (массив из трех чисел).

Написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа NOTE; записи должны быть упорядочены по датам рождения;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на экран информации о человеке, номер телефона которого введен с клавиатуры, а если такого нет, выдать на дисплей соответствующее сообщение.

Вариант 13

Для структуры, указанной в варианте 12, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа NOTE; записи должны быть размещены по алфавиту;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры, а если таких нет, выдать на дисплей соответствующее сообщение.

Вариант 14

Для структуры, указанной в варианте 12, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа NOTE; записи должны быть упорядочены по трем первым цифрам номера телефона;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на экран информации о человеке, чья фамилия введена с клавиатуры, а если такого нет, выдать на дисплей соответствующее сообщение.

Вариант 15

Описать структуру с именем ZNAK, содержащую следующие поля:

- порядковый номер;
- фамилия, имя;
- знак Зодиака;
- дата рождения (массив из трех чисел).

Написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа ZNAK; записи должны быть упорядочены по датам рождения;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на экран информации о человеке, чья фамилия введена с клавиатуры, а если такого нет, выдать на дисплей соответствующее сообщение.

Вариант 16

Для структуры, указанной в варианте 15, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа ZNAK; записи должны быть упорядочены по датам рождения;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на экран информации о людях, родившихся под знаком, название которого введено с клавиатуры, а если таких нет, выдать на дисплей соответствующее сообщение.

Вариант 17

Для структуры, указанной в варианте 15, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа ZNAK; записи должны быть упорядочены по знакам Зодиака;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на экран информации о людях, родившихся в месяц, значение которого введено с клавиатуры, а если таких нет, выдать на дисплей соответствующее сообщение.

Вариант 18

Описать структуру с именем PRICE, содержащую следующие поля:

- порядковый номер;
- название товара;
- название магазина, в котором продается товар;
- стоимость товара в руб.

Написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа PRICE; записи должны быть размещены в алфавитном порядке по названиям товаров;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на экран информации о товаре, название которого введено с клавиатуры, а если таких товаров нет, выдать на дисплей соответствующее сообщение.

Вариант 19

Для структуры, указанной в варианте 18, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа PRICE; записи должны быть размещены в алфавитном порядке по названиям магазинов;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на экран информации о товарах, продающихся в магазине, название которого введено с клавиатуры, а если такого магазина нет, выдать на дисплей соответствующее сообщение.

Вариант 20

Описать структуру с именем ORDER, содержащую следующие поля:

- порядковый номер;
- расчетный счет плательщика;
- расчетный счет получателя;
- перечисляемая сумма в руб.

Написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа ORDER; записи должны быть размещены в алфавитном порядке по расчетным счетам плательщиков;
- чтение данных из этого файла и вывод их на экран;

- корректировку данных в файле по номеру записи;
- вывод на экран информации о сумме, снятой с расчетного счета плательщика, введенного с клавиатуры, а если такого расчетного счета нет, выдать на дисплей соответствующее сообщение.

Вариант 21

Описать структуру с именем STUDENT, содержащую следующие поля:

- порядковый номер;
- номер;
- фамилия и имя;
- год рождения;
- год поступления в университет;
- структура OCENKI, содержащая четыре поля: физика, математика, программирование, история.

Написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из структур типа STUDENT; записи должны быть упорядочены по алфавиту;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на дисплей анкетных данных студентов отличников, если таких студентов нет, вывести соответствующее сообщение.

Вариант 22

Для структуры, указанной в варианте 21, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из структур типа STUDENT; записи должны быть упорядочены номеру;
- чтение данных из этого файла;
- корректировку данных в файле по номеру записи;
- вывод на дисплей анкетных данных студентов, получивших одну оценку 3, а если таких студентов нет, вывести соответствующее сообщение.

Вариант 23

Для структуры, указанной в варианте 21, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из структур типа STUDENT; записи должны быть упорядочены по фамилиям;
- чтение данных из этого файла и вывод их на экран;
- вывод на дисплей анкетных данных студентов, получивших все двойки, а если таких студентов нет, вывести соответствующее сообщение.

Вариант 24

Для структуры, указанной в варианте 21, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из структур типа STUDENT; записи должны быть упорядочены по номеру;

- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на дисплей анкетных данных студентов, получивших все пятерки, а если таких студентов нет, вывести соответствующее сообщение.

Вариант 25

Для структуры, указанной в варианте 21, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из структур типа STUDENT; записи должны быть упорядочены по алфавиту;
- чтение данных из этого файла;
- корректировку данных в файле по номеру записи;
- вывод на дисплей анкетных данных студентов, получивших одну оценку 4, а все остальные – 5, если таких студентов нет, вывести соответствующее сообщение.

Вариант 26

Для структуры, указанной в варианте 21, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из структур типа STUDENT;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на дисплей фамилий студентов, которые начинаются с литеры 'А', и их оценок, а если таких студентов нет, вывести соответствующее сообщение;
- отсортировать записи по номеру по убыванию.

Вариант 27

Для структуры, указанной в варианте 21, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из структур типа STUDENT;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на дисплей фамилий студентов, которые начинаются с литеры 'Б', и год их рождения, а если таких студентов нет, вывести соответствующее сообщение;
- отсортировать записи по номеру по возрастанию.

Вариант 28

Для структуры, указанной в варианте 21, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из структур типа STUDENT;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;

- вывод на дисплей фамилий студентов, которые начинаются с литеры ‘Б’ или ‘Г’, и год их поступления, а если таких студентов нет, вывести соответствующее сообщение;
- отсортировать записи по фамилиям по убыванию.

Вариант 29

Для структуры, указанной в варианте 21, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из структур типа STUDENT;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на дисплей фамилий студентов, имеющих средний балл выше среднего балла группы, а если таких студентов нет, вывести соответствующее сообщение;
- отсортировать записи по фамилиям по возрастанию.

Вариант 30

Для структуры, указанной в варианте 21, написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из структур типа STUDENT;
- чтение данных из этого файла и вывод их на экран;
- корректировку данных в файле по номеру записи;
- вывод на дисплей фамилий и года рождения студентов, не получивших ни одной тройки, а если таких студентов нет, вывести соответствующее сообщение;
- отсортировать записи по номеру.

2.4 Порядок выполнения работы

2.4.1 В ходе самостоятельной подготовки изучить основы работы со структурами и бинарными файлами в языках C/C++.

2.4.2 Выбрать способ представления исходных данных задачи и результатов. Разработать алгоритм решения задачи, разбив его на отдельные функции.

2.4.3 Разработать программу на языке C/C++.

2.4.4 Разработать тестовые примеры, следуя указаниям раздела 3.

2.4.5 Выполнить отладку программы.

2.4.6 Получить результаты работы программы и исследовать её свойства в различных режимах работы, сформулировать выводы.

2.5 Содержание отчета

Цель работы, вариант задания, структурная схема алгоритма решения задачи с описанием, текст программы с комментариями, тестовые примеры, результаты вычислений, выводы.

2.6 Контрольные вопросы

2.6.1 Что называют структурой в языке C/C++?

2.6.2 Покажите на примерах ввод-вывод структур с помощью функций языка C.

2.6.3 Как выполнить чтение и запись структурных переменных в бинарные файлы?

2.6.4 Как обнаружить конец файла при использовании функции `fread`?

2.6.5. Как определить длину файла с помощью функций `seek` и `ftell`?

Заказ № _____ от « _____ » _____ 2022г. Тираж _____ экз.
Изд-во СевГУ