РОСЖЕЛДОР

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Ростовский государственный университет путей сообщения»

(ФГБОУ ВО РГУПС)

О.Г. Ведерникова

ПРОГРАММИРОВАНИЕ

Учебно-методическое пособие

Часть 4

Файловые потоки ввода-вывода

Ростов-на-Дону
2017

УДК 681.3(07) + 06

Ведерникова, О.Г.

Программирование: [Электронный ресурс] учебно-методическое пособие. В 4 ч. Ч. 4. Файловые потоки ввода-вывода / О.Г. Ведерникова; ФГБОУ ВО РГУПС. – Ростов н/Д, 2017. – 22 с. –

Рассмотрены базовые приемы организации файловых потоков для чтения и записи в среде DevC++. Содержится лекционный материал, а также большое количество исходных текстов программ-примеров.

Приведены задания и методика выполнения лабораторных работ по дисциплине «Программирование», а также примеры выполнения лабораторных работ.

Предназначено для студентов направления подготовки 09.03.02 Информационные системы и технологии и 09.03.01 Информатика и вычислительная техника, а так же для студентов всех специальностей, изучающих дисциплину «Программирование».

Одобрено к внесению в «Электронный университет» кафедрой «Вычислительная техника и автоматизированные системы управления».

ОГЛАВЛЕНИЕ

ФАЙЛОВЫЕ ПОТОКИ ВВОДА-ВЫВОДА

Для хранения информации вводимой или полученной в результате работы программы можно использовать файлы на внешних носителях, таких как жесткий магнитный диск или другие носители. Использование внешних запоминающих устройств для хранения информации является наиболее надёжным и удобным способом хранения информации. Файл в таком понимании называют физическим файлом, а его представитель в программе – логическим файлом.

Существует понятие полного имени файла и относительного имени файла. Полное имя файла – это полный путь к каталогу файла с указанием имени файла с расширением, например: D:\docs\file.txt. Под относительным именем файла понимается имя относительно текущего каталога, то есть если программа располагаемся в той же директории, что и файл, то достаточно указать только имя файла с расширением, например: file.txt

По способу доступа файлы можно разделить на *последовательные* и файлы с *произвольным* доступом. Различают также форматный и бесформатный файловый ввод/вывод.

Файл – можно представить как последовательность компонент одного и того же типа как в одномерном массиве, но в отличие от массива длина файла не определена и является условно бесконечной. Второе отличие от массива – у файла есть текущий файловый указатель или файловое окно, из которого можно считать один текущий элемент. Считав текущий элемент, файловое окно сдвигается на одну позицию вперед. Тогда текущим оказывается следующий элемент. Таким образом, из файла нельзя считать любой элемент по его номеру, нельзя вернуться и считать уже считанный элемент еще раз, за исключением файлов с произвольным доступом.

Основные действия для работы с файловыми потоками

В языке C++ существует несколько способов организации работы с файлами. Рассмотрим один из них в стиле C++ с использованием файловых потоков ввода/вывода.

Стандартная библиотека классов C++ содержит три класса для работы с файловыми потоками:

ifstream – класс входных файловых потоков;

ofstream – класс выходных файловых потоков;

fstream – класс двунаправленных файловых потоков;

Чтобы их использовать необходимо подключить к программе заголовочный файлы **<fstream>, <iostream>** и **<cstdlib>.**

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

#include<cstdlib>
using namespase std;
Использование файлов в программе предполагает следующие операции:
1 создание потока;
2 открытие потока и связывание его с файлом;
3 обмен (ввод/вывод);
4 закрытие файла;

Опишем каждый из них подробнее.
1) Создание потока происходит с помощью следующих операторов
ifstream <имя_входного_потока> для входного потока
или
ofstream<имя_выходного_потока> для выходного потока
Пример:
ifstream f1;
ifstream infile ;
ofstream outfile ;
2) Открытие потока и связывание его с файлом происходит с помощью метода **open** :
имя_потока . open(“адрес_файла”),
где “**адрес_файла**” – это строка текста, переменная типа string, описывающая либо полный адрес файла, например: «D:\docs\file.txt», либо относительный адрес, например: «file.txt»
Пример:
f1.open (“C:\dir\data.txt”); //полный адрес
infile.open (“input.txt”); //относительный адрес
outfile .open (“output.txt”); //относительный адрес
Полный адрес файла – это полный путь от диска к каталогу и к подкаталогам, где расположен файл, с указанием имени файла с расширением.

Под относительным адресом файла понимается имя относительно текущего каталога, например, если программа располагаемся в той же папке, что и файл, то достаточно указать только имя файла с расширением.
Например:
C:\dir1\1.cpp
C:\dir1\dir2\a.txt и **2.cpp**
Полный адрес **: C:\dir1\dir3\a.txt**
В программе **1.cpp** относительный адрес будет **dir2\a.txt**
В программе **2.cpp** относительный адрес будет **a.txt**
Не желательны в записи адреса символы кириллицы и пробелы, поэтому предпочтительнее использовать относительный адрес, а файлы сохранять в одной папке с программой.

3)Обмен (ввод/вывод) происходит таким же образом, как и ввод/вывод из стандартных потоков **cin** и **cout,** то есть с помощью перегруженных операций >> и <<.

Пример:

f1 >> x; вместо **cin>>x; //** ввод, считывание из файлового потока **f1**

Если необходимо считать строку символов, содержащую несколько слов, разделенных пробелами, то стоит воспользоваться функцией **getline(f3,s1).**

Пример:

infile >> dat;

Аналогично происходит запись в файл:

outfile << ”sum=”<<sum; . вместо **cout<<”sum=”<<sum;**

При считывании одного элемента из файла, файловое окно смещается вперед к следующему элементу, и тогда можно считать только второй элемент. Нельзя повторно считать первый элемент или нулевой элемент. Только закрыв поток и открыв его заново можно повторно считывать первый элемент. Также нельзя заглянуть вперед и считать 5-ый элемент, считав только первый.

4 Закрытие файла происходит с помощью метода **close:**

Пример:

f1 . close();
infile.close();
outfile.close();

Проиллюстрируем эти операции в следующем примере.

```
#include<conio.h>
#include<string.h>
#include<iostream>
#include <fstream>
using namespace std;
int main ()
{setlocale(LC_ALL, "rus");// для корректного отображения Кириллицы
  ifstream f_in;//  -создали поток для ввода, чтения
  f_in.open("a1.txt"); // открыли файл для ввода, чтения
 float a;
  f_in >> a; // считали первое значение из файла
  cout << "  1 значение ="<<a<< endl; // напечатали это значение
  string buff;
  f_in >> buff; // считали 2-ое значение из файла в строку
  cout << "buff 2="<<buff << endl; // напечатали эту строку
  getline(f_in, buff); // считали 3-ю строку из файла с пробелами
   cout << "buff 3="<<buff << endl; // напечатали эту 3-ю строку
  f_in.close(); // закрываем файл
 ifstream f_in2; //  -создали поток для ввода, чтения еще раз
 f_in2.open("a1.txt"); // открыли файл для ввода. чтения еще раз
 f_in2 >> a; // считали еще раз первое значение из файла еще раз
```

```
f_in2.close(); // закрываем файл
```

Определения конца файла

Файл считается условно бесконечным набором элементов, количество элементов в нем заранее не известно. Данные из файла должны считываться в цикле, до тех пор, пока не достигнут конец файла. Для определения конца файла существует функция **eof().**

В этом заключается преимущество файлов и этим они и интересны файлы, что их можно обрабатывать, ничем заранее не ограничиваясь, как в массивах – должно быть заранее известно максимальное кол-во эл. И выделять для них память. Поэтому информацию выгоднее хранить в файле, а не в массиве (за исключением сортировки). К тому же не надо вводить снова и снова при каждом новом запуске большие объемы информации, например структуры.

Входной файл, как правило, содержит последовательность однотипных данных, количество которых заранее не определено и может меняться. Обычно данные из файла считывают в цикле, после каждой операции считывания проверяется, не достигнут ли конец файла. Для определения конца файла существует функция **eof()**. Она возвращает **true** если конец файла достигнут, и **false** если конец файла не достигнут. Пример цикла для чтения из файлового потока *f_in*:

while (! *F_in*.eof()) {… тело цикла…}– «*пока не достигнут конец файла продолжать тело цикла*»

Пример №1

В физическом файле **a1.txt**, созданном в «Блокноте», записаны исходные данные: вещественные числа, каждое число с новой строки. Считать данные и вычислить сумму положительных чисел из них.

```
#include<conio.h>
#include<iostream>
#include <fstream>
using namespace std;
int main ()
{   ifstream fin; // создаём объект класса ifstream для чтения
   Fin.open("a1.txt"); //и связываем его с файлом a1.txt , открыли файл для чтения
    float a; float s;
   while (! fin.eof())
   {  fin >> a;   // считали число из файла
     cout << "a="<<a << endl; //вывели на экран считанное число
```

```
      if(a>0) s+=a;
      }
      fin.close(); // закрываем файл
    cout<<"s="<<s;
getch();

}
```

Пример №2

Переписать из одного файла в другой только отрицательные элементы заданного файла **number.txt**. В физическом файле, созданном в «Блокноте», записаны исходные данные: вещественные числа, каждое число с новой строки.

```
#include<conio.h>
#include<iostream>
#include <fstream>
using namespace std;
int main ()
{   ifstream dano; // создаём объект класса ifstream для чтения
   Dano.open ("number.txt", ios::out); //и связываем его с файлом
  ofstream nov; //для записи
nov .open(“nov1.txt”, ios::in | ios::app); //и связываем его с файлом
float buf;
 while (! dano.eof())
   {  dano >> buf;   // считали число из файла
      cout << "buf="<<buf << endl; //вывели на экран считанное число
      if (buf<0)  nov<<buf;// записали в файл nov
      }
      dano.close(); // закрываем файл
      nov.close();
    getch();
}
```

Режимы открытия файлов

Режимы открытия файлов устанавливают характер использования файлов. Для установки режима в классе ios_base предусмотрены константы, которые определяют режим открытия файлов (см. Таблица 1).

Таблица 1 — Режимы открытия файлов

Константа	**Описание**
ios_base::in	открыть файл для чтения
ios_base::out	открыть файл для записи
ios_base::ate	при открытии переместить указатель в конец файла
ios_base::app	открыть файл для записи в конец файла
ios_base::trunc	удалить содержимое файла, если он существует
ios_base::binary	открытие файла в двоичном режиме

Режимы открытия файлов можно устанавливать непосредственно при создании объекта или при вызове функции open()**.**

Пример:

```
ofstream fout("cppstudio.txt", ios_base::app); // открываем файл для добавления информации к концу файла
fout.open("cppstudio.txt", ios_base::app); // второй способ для открытия файла и для добавления информации к концу файла
```

Режимы открытия файлов можно комбинировать с помощью поразрядной логической операции **или** |,

Например: ios_base::out | ios_base::trunc – открытие файла для записи, предварительно очистив его.

Файловый поток типа **ofstream**, при связке с файлами по умолчанию содержат режимы открытия файлов ios_base::out | ios_base::trunc**.** То есть файл будет создан, если не существует. Если же файл существует, то его содержимое будет удалено, а сам файл будет готов к записи.

Файловый поток типа **ifstream** связываясь с файлом, имеют по умолчанию режим открытия файла ios_base::in – файл открыт только для чтения. Если он не существует, то будет ошибка.

Пример:

```
ofstream f;
f.open("a3_1.txt",ios::out | ios::app); //файл для записи и для добавления
int n=10, i, a[n];
```

```
 for(i=0;i<n; i++)
  {a[i]=i;
    if (a[i]%2==0)  f<<a[i]<<endl;   }
```

Пример:

Определить комбинированный тип для представления анкеты ребенка, состоящей из его имени, пола и роста. Дан файл **kids.dat** с информацией о детях. Вывести средний рост мальчиков

```
#include <fstream.h>
#include <iostream.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>

  struct deti {
       string name;
       char pol;
       int rost;
        };
int main()
{
int N, k=0;
cout<<"Vvedite max kolichestvo detei"<<endl;
cin>>N;
deti children[N];
ifstream f_children("F:/kids.txt");//otkrivaetsj vxodnoi potok
while(! F_children.eof())
{
f_children>>children[i].name;
f_children>>children[i].pol;
f_children>>children[i].rost;
 k++;}
int col=0;
float sum=0;
for (i=0;i<k;i++)
 {
    if (children[i].pol==’m’)
    {
     col=col+1;
     sum=sum+children[i].rost;
    }
 }
float avg_rost=sum/col;
```

```
cout<<"srednii rost malchicov sostavljet"<<avg_rost;
getch();
f_children.close();
}
```

Пример:

В исходном файле записаны сначала два целых числа n и m – размерность записанной матрицы, причем n и m <100. Далее с новой строки вещественные числа через пробел, которые являются элементами первой строки матрицы (m-штук). Далее с новой строки вещественные числа через пробел, которые являются элементами следующей строки матрицы, и т.д. вся матрица по строкам (n-штук строк). Считать значения из файла в двумерный массив по строкам (n и m <100). Каждый элемент побочной диагонали матрицы увеличить в два раза;

```
#include<conio.h>
#include<string.h>
#include<iostream>
#include <fstream>
using namespace std;
int main ()
{   setlocale(LC_ALL, "rus"); // корректное отображение Кириллицы

    int i,j,n,m,a[10][10]; float s=0;
   ifstream fin("a3.txt"); // открыли файл для чтения
    fin >> n;fin >> m;
    for(i=0;i<n;i++)
    {
                for(j=0;j<m;j++)
                {
                     fin >> a[i][j];   // считали строку из файла
                    cout << "  "<<a[i][j];
                    if(a[i][j]>0) s+=a[i][j];
                 }cout <<endl;
    }
      fin.close(); // закрываем файл
    cout<<"s="<<s;
   system("pause");
   //return 0;
}
```

ПРОИЗВОЛЬНЫЙ ДОСТУП К ФАЙЛАМ

Произвольный доступ к файлу означает доступ, при котором программа может считывать или записывать данные в любом требуемом месте файла, не обрабатывая его с самого начала, без необходимости закрывать файл и открывать его повторно, чтобы прейти от чтения к файлу.

Для этого необходимо объявить файловый поток двунаправленного типа

fstream имя_потока; – двунаправленный поток для чтения и записи

Пример:

fstream fl;

fl.open(file1.txt, ios::in | ios::out); //– для чтения и записи

Данный класс **fstream** содержит метод **seekp(n)** перемещающий файловый указатель в позицию **n** для записи данных, и метод **seekg(n)** - перемещающий указатель для чтения данных.

fl.seekp(n); //– позиционирование указателя для записи

fl.seekg(n); //- позиционирование указателя для чтения

Величина **n** – измеряется в байтах. Для определения размера объектов в файле можно воспользоваться функцией **sizeof();**

Sizeof(int); sizeof(double); sizeof(mystruct); и т.д.…

Пример:

fl.seekg(nom*sizeof(int)) – переместить указатель для чтения в позицию **nom** в файле, где записаны целые числа.;

fl>>buf; чтения из позиции.

Пример: В цикле считать все числа .

```
For (i=0;!fl.eof();i++)
{fl.seekg(i*sizeof(int));
fl>>buf; cout<<buf}
```

Пример:

```
#include <conio.h>
#include<string.h>
#include<iostream>
#include <fstream>
using namespace std;
int main ()
 {
    float buf;
   fstream fin("a2.txt",ios::in|ios::out); // открыли файл для чтения и записи
    for (int i=0;i<10;i++)
    {
```

```
            a=rand()%11-5;
            fin.seekp(i*sizeof(float));
            fin<<a<<endl;
         }

       for(int i=0;!fin.eof();i++)
         {fin.seekg(i*sizeof(float));
          fin>>buf;
          cout<<"buf="<<buf<<endl;
          }
           fin.close(); // закрываем файл
       system("pause");
}
```

РАБОТА С ФАЙЛАМИ В СТИЛЕ С

Пример. Вычислить количество положительных элементов в файле

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<iostream>
using namespace std;
int main ()
{
    int x, kol=0,c;
    FILE *f;-объявление файловой переменной
   f=fopen("data.txt","w"); открытие файла для записи и связывание его с физическим файлом
   for (int i=1;i<10;i++)
   {fprintf(f,"%d\n",i);запись в файл f целых чисел i
    cout<<endl<<i;}
    fclose(f);
    f=fopen("data2.txt","r"); открытие для чтения
    while (!feof(f)) пока не конец файла
    {fscanf(f,"%d",&x); считать из файла
    if (x<0)kol++;
    cout<<endl<<x;}
     fclose(f);
     cout<<"kol="<<kol;
   getch();
   }
```

ЗАДАНИЯ ДЛЯ ЛАБОРАТОРНОЙ РАБОТЫ №11.

Задание 1. Чтение данных из файла

Для всех вариантов исходные данные набрать в блокноте.

1. Дан файл вещественных чисел a.txt. Найти количество отрицательных и количество положительных элементов.
2. Дан файл вещественных чисел a.txt Найти *количеств*о нулевых элементов и произведение элементов меньших 1 и больших 0.
3. Дан файл вещественных чисел a.txt Найти количество нулевых элементов и сумму отрицательных элементов.
4. Дан файл вещественных чисел a.txt Найти количество элементов равных 5 и сумму положительных элементов.
5. Дан файл вещественных чисел a.txt Найти количество нулевых элементов и сумму положительных элементов.
6. Дан файл вещественных чисел a.txt Найти количество положительных элементов и сумму положительных элементов.
7. Дан файл вещественных чисел a.txt Найти количество отрицательных и количество положительных элементов.
8. Дан файл вещественных чисел a.txt Найти *количеств*о нулевых элементов и произведение элементов меньших 1 и больших 0.
9. Дан файл вещественных чисел a.txt Найти количество нулевых элементов и сумму отрицательных элементов.
10. Дан файл вещественных чисел a.txt Найти количество элементов равных 5 и сумму положительных элементов.
11. Дан файл вещественных чисел a.txt Найти количество нулевых элементов и сумму положительных элементов.
12. Дан файл вещественных чисел a.txt Найти количество положительных элементов и сумму положительных элементов.
13. Дан файл вещественных чисел a.txt Найти *количеств*о положительных элементов и произведение элементов меньших 1 и больших 0.
14. Дан файл вещественных чисел a.txt Переписать положительные элементы в файл b.txt

Задание 2. Чтение и запись данных в файл

1. Дано 2 файла вещественных чисел a1.txt и a2.txt. Найти сумму положительных элементов в двух файлов.
2. Дано 2 файла вещественных чисел a1.txt и a2.txt. Найти произведение отрицательных элементов двух файлов

3. Дано 2 файла вещественных чисел a1.txt и a2.txt. Найти количество нулевых элементов в двух файлов

4. Дано 2 файла вещественных чисел a1.txt и a2.txt. Найти сумму положительных элементов в двух файлов.

5. Дано 2 файла вещественных чисел a1.txt и a2.txt. Найти произведение отрицательных элементов двух файлов

6. Дано 2 файла вещественных чисел a1.txt и a2.txt. Найти количество нулевых элементов в двух файлов

7. Дан файл вещественных чисел a.txt . Переписать в файл a2.txt положительные элементы файла b(n) умноженные на 5.

8. Дан файл вещественных чисел a.txt . Переписать в файл a2.txt все ненулевые элементы файла a.txt

9. Дан файл вещественных чисел a.txt или a.txt. Переписать в файл a2.txt ненулевые элементы файла a.txt разделенные на 5.

10. Дан файл вещественных чисел a.txt Переписать в файл a2.txt отрицательные элементы файла a.txt умноженные на 2.

11. Дано 2 файла вещественных чисел a1.txt и a2.txt. В каком из двух данных файлов больше отрицательных элементов?

12. В данном файле a2.txt каких элементов больше, равных 0 или равных 1?

13. Дано 2 файла вещественных чисел a1.txt и a2.txt. Найти произведение отрицательных элементов двух файлов

14. Дано 2 файла вещественных чисел a1.txt и a2.txt. Найти произведение отрицательных элементов двух файлов

Задание 3. Обработка тестовой информации из файлов

1. Организовать текстовый файл. Заменить в файле все маленькие латинские буквы на большие.(создавая новый дополнительный файл)

2. Из заданного входного файла считать символы и записать в один новый файл только буквы, в другой новый файл только цифры .

3. Организовать текстовый файл. Организовать замену символов в файле. "старый" символ и "новый" символ запрашиваются и вводятся с клавиатуры.(создавая новый дополнительный файл)

4. Организовать текстовый файл. Преобразовать файл, удалив в нем лишние пробелы.(создавая новый дополнительный файл)

5. Организовать текстовый файл, состоящий из строк. Заменить в файле все большие латинские буквы на маленькие . создавая новый дополнительный файл)

6. Организовать текстовый файл. Заменить в файле все цифры на '*'.(создавая новый дополнительный файл)

7. Организовать текстовый файл. Заменить в файле все буквы (нецифры) на '*'.(создавая новый дополнительный файл)

8. Организовать текстовый файл. Удалить в файле все цифры. (создавая новый дополнительный файл)

9. Из заданного входного файла считать символы и записать в один новый файл только большие латинские буквы, в другой новый файл только малые латинские буквы и посчитать количество цифр.

10. Организовать текстовый файл. Удалить в файле все буквы. (создавая новый дополнительный файл)

11. Из заданного входного файла считать символы и записать в новый файл все символы за исключением символов разделителей : пробелы , точки , запятые, двоеточия и т.д.

12. Организовать текстовый файл . Оставив в файле только буквы. (создавая новый дополнительный файл)

13. Из заданного входного файла считать символы и записать в новый файл только большие буквы латинского алфавита .

14. Организовать файл вещественных чисел Заменить все положительные компоненты файла их квадратными корнями, а все отрицательные компоненты их квадратами, создавая новый дополнительный файл.

15. Организовать файл целых чисел . Удалить из файла все отрицательные компоненты. . создавая новый дополнительный файл)

16. организовать файл целых чисел, заменить все элементы файла от -10 до 10 на противоположные. создавая новый дополнительный файл)

17. Организовать файл целых чисел Все числа, кратные 3 заменить их удвоенным произведением. создавая новый дополнительный файл)

18. организовать файл целых чисел, заменить все элементы файла от -2 до 5 на противоположные. создавая новый дополнительный файл)

Задание 4. Передвижение файлового окна

1. Организовать файл целых чисел. В новый файл записать элементы файла, занимающие нечётные позиции, в другой новый файл элементы файла, занимающие чётные позиции.

2. Организовать файл целых чисел. Определить наибольший отрицательный компонент файла среди компонент файла расположенных на чётных позициях.

3. Организовать файл целых чисел. Определить наибольший элемент файла среди элементов файла номера, которых кратны трем.

4. Организовать файл целых чисел. Вычислить количество отрицательных компонентов файла расположенных на нечётных позициях.

5. Организовать файл целых чисел. Определить наименьший положительный компонент файла среди компонент файла расположенных на чётных позициях.

6. Организовать файл целых чисел. Вычислить среднее значение среди положительных значений файла, номера которых кратны трем.

7. Организовать файл целых чисел . Определить наименьший положительный компонент файла среди компонент файла расположенных на нечётных позициях.

8. Организовать файл целых чисел. Вычислить количество отрицательных компонентов файла расположенных на чётных позициях.

9.Организовать файл целых чисел. Вычислить среднее значение среди положительных значений файла расположенных на чётных позициях.

10. Организовать файл целых чисел. Вычислить количество отрицательных компонентов файла , номера которых кратны трем.

11. Организовать файл целых чисел. Найти сумму элементов файла расположенных на чётных позициях.

12. Организовать файл целых чисел. Вычислить среднее значение среди положительных значений файла расположенных на нечётных позициях.

13. Организовать файл целых чисел. Найти сумму элементов файла , номера которых кратны трем.

14. Организовать файл целых чисел. Вычислить количество нулевых элементов файла расположенных на чётных позициях.

15. Организовать файл целых чисел . Определить наибольший отрицательный компонент файла среди компонент файла расположенных на нечётных позициях.

16. Организовать файл целых чисел. Найти сумму элементов файла расположенных на нечётных позициях.

17. Организовать файл целых чисел. Вычислить количество нулевых элементов файла расположенных на нечётных позициях.

Дополнительное задание 1*.

1. Создать файл. Из файла создать массив, элементы которого являются четными числами и расположены после максимального элемента.
2. Создать файл. Массив создать из исходного файла. Внести в него положительные числа, расположенные в файле между минимальным и максимальным элементами
3. Создать файл. Из файла целых чисел сформировать массив, записав в него только четные компоненты, находящиеся до минимального элемента.
4. Создать файл. Из исходного файла сформировать массив, записав в него числа, расположенные в файле до максимального элемента и после минимального.
5. Создать файл. Из файла создать массив, элементы которого не являются отрицательными числами и расположены до максимального значения файла.
6. Создать файл. Из файла целых чисел сформировать массив, записав в него только кратные 5 и 7 значения, находящиеся после максимального элемента файла.
7. Создать й файл. Из файла создать массив, элементы которого являются положительными числами и расположены после максимального элемента.

8. Создать файл. Массив создать из исходного файла. Внести в него отрицательные числа, расположенные в файле между минимальным и максимальным элементами

9. Создать файл. Из файла целых чисел сформировать массив, записав в него только нечетные отрицательные компоненты, находящиеся до минимального элемента.

10. Создать файл. Из исходного файла сформировать массив, записав в него числа, расположенные в файле до максимального элемента и после минимального.

11. Создать файл. Из файла создать массив, элементы которого кратны трём и расположены до максимального значения файла.

12. Создать файл. Из файла целых чисел сформировать массив, записав в него только кратные 5 и 7 значения, находящиеся после максимального элемента файла.

13. Создать файл. Массив создать из исходного файла. Внести в него положительные четные числа, расположенные в файле между минимальным и максимальным элементами

14. Создать файл. Из файла целых чисел сформировать массив, записав в него элементы меньшие 1 и находящиеся до минимального элемента.

15. Создать файл. Из исходного файла сформировать массив, записав в него числа большие 5 и расположенные в файле до максимального элемента и после минимального.

Дополнительное задание 2*.

1) Создать типизированный файл, куда записать n целых чисел. В файле поменять местами минимальный среди положительных элементов и третий по счету четный элемент.

2) Создать типизированный файл, куда записать n целых чисел. Поменять местами последнее четное и третье отрицательное числа в файле.

3) Создать типизированный файл, куда записать n вещественных чисел. Поменять местами последнее отрицательное число в файле с четвертым по счету числом.

4) Создать типизированный файл, куда записать n целых чисел. В файле поменять местами максимальный среди положительных элементов и второй по счету четный элемент.

5) Создать типизированный файл, куда записать n целых чисел. В файле поменять местами минимальный среди положительных элементов и третий по счету отрицательный элемент.

6) Создать типизированный файл, куда записать n целых чисел. Поменять местами последнее положительное и третье отрицательное числа в файле.

7) Создать типизированный файл, куда записать n целых чисел. Поменять местами последнее четное и второе положительное числа в файле.

8) Создать типизированный файл, куда записать n вещественных чисел. Поменять местами последнее положительное число в файле с четвертым по счету числом.

9) Создать типизированный файл, куда записать n целых чисел. Поменять местами первое четное и последнее положительное числа в файле.

10) Создать типизированный файл, куда записать n целых чисел. В файле поменять местами максимальный среди четных элементов и третий элемент.

11) Создать типизированный файл, куда записать n целых чисел. В файле поменять местами максимальный среди четных элементов и третий последний отрицательный элемент.

12) Создать типизированный файл, куда записать n целых чисел. Поменять местами последнее положительное и третье отрицательное числа в файле.

13) Создать типизированный файл, куда записать n целых чисел. Поменять местами последнее четное и второе положительное числа в файле.

14) Создать типизированный файл, куда записать n вещественных чисел. Поменять местами последнее отрицательное число в файле с четвертым по счету числом.

Дополнительное задание 3*

1. Определить комбинированный тип для представления информации по горным вершинам, состоящей из названия вершины и ее высоты. Дан файл Vep.dat с информацией по вершинам. Вывести название самой высокой вершины из всех.
2. Определить комбинированный тип для представления информации по горным вершинам, состоящей из названия вершины и ее высоты. Дан файл Ver1.dat с информацией по вершинам. Вывести название самой низкой вершины из всех.
3. Определить комбинированный тип для представления информации по горным вершинам, состоящей из названия вершины и ее высоты. Дан файл Ver1.dat с информацией по вершинам. Вывести среднее значение высот всех вершин.
4. .Определить комбинированный тип для представления анкеты ребенка, состоящей из его имени, пола и роста. Дан файл kids.dat с информацией о детях. Вывести средний рост девочек
5. .Определить комбинированный тип для представления анкеты ребенка, состоящей из его имени, пола и роста. Дан файл kids.dat с информацией о детях. Вывести имя самого высокого мальчика.
6. .Определить комбинированный тип для представления анкеты ребенка, состоящей из его имени, пола и роста. Дан файл kids.dat с информацией о детях. Вывести средний рост мальчиков

7. .Определить комбинированный тип для представления анкеты ребенка, состоящей из его имени, пола и роста. Дан файл kids.dat с информацией о детях. Вывести имя самой высокой девочки.

8. Определить комбинированный тип для представления анкеты ребенка, состоящей из его имени, пола и роста. Дан файл kids.dat с информацией о детях. Вывести средний рост девочек.

9. Определить комбинированный тип для представления анкеты ребенка, состоящей из его имени, пола и роста. Дан файл kids.dat с информацией о детях. Вывести средний рост всех детей.

10. Определить комбинированный тип для представления анкеты студента, состоящей из его фамилии, дня рождения, месяца рождения и пола. Дан файл students.dat с информацией о студентах группы. Вывести фамилии мальчиков, родившихся в январе.

11. .Определить комбинированный тип для представления анкеты студента, состоящей из его фамилии, дня рождения, месяца рождения и пола. Дан файл students.dat с информацией о студентах группы. Вывести фамилии девочек, родившихся в мае.

12. Определить комбинированный тип для представления анкеты студента, состоящей из его фамилии, дня рождения, месяца рождения и пола. Дан файл students.dat с информацией о студентах группы. Вывести фамилии студентов, родившихся летом.

13. .Определить комбинированный тип для представления анкеты ребенка, состоящей из его имени, пола и роста. Дан файл kids.dat с информацией о детях. Вывести средний рост мальчиков

14. Определить комбинированный тип для представления анкеты студента, состоящей из его фамилии, дня рождения, месяца рождения и пола. Дан файл students.dat с информацией о студентах группы. Вывести фамилии студентов, родившихся весной.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Начальный курс С и С++. Березин Б. И., Березин С. Б. Москва: Диалог-МИФИ, 2008. - 272 с.

2. Как программировать на С++. Х.М. Дейтел, П.Дж. Дейтел. Москва: Бином-Пресс, 2008 г., 1456 с.

3. Скользкие места С++. Как избежать проблем при проектировании и компиляции ваших программ. Дьюхэрст С. Москва: ДМК Пресс, 2006 г. - 264 с.

4. Эффективное использование С++. 55 верных способов улучшить структуру и код ваших программ. Дьюхэрст С. Москва: ДМК Пресс, 2006 г. - 264 с.

5. Современное программирование с нуля. Потопахин В. В. Москва: ДМК Пресс,2010. - 240 с.

6. Сборник задач по программированию. Учебное пособие. Мишенин А. И. Москва: Финансы и статистика, 2009. - 224 с.

7. С++ для чайников, 6-е издание. Стефан Рэнди Дэвис. И.: Диалектика. 2011г. 336 стр.

8. В.И. Медведев. Особенности объектно-ориентированного программирования на С++/CLI, C#. РИЦ «Школа», 2010. - 62с

9. Информатика и программирование : Компьютерный практикум : учеб.пособие для вузов/ А.Н. Гуда, М.А. Бутакова, Н.М. Нечитайло, А.В. Чернов; ред. В.И. Колесников. -М.: Дашков и К°, 2009. -238 с.

10. Бутакова М.А. Сборник задач по программированию : учеб.-метод. пособие/ М. А. Бутакова, В. В. Ильичева; РГУПС. -Ростов н/Д, 2012. -34 с.

11. Павловская Т.А. С/С++. Программирование на языке высокого уровня : учеб. для вузов/ Т.А. Павловская. -М.; СПб.: Питер, 2006. -460 с.: а-ил.

12. Ильичева В.В. Алгоритмизация и программирование : практикум/ В. В. Ильичева; РГУПС. -Ростов н/Д, 2010. -142 с.

13. Ведерникова О.Г. Программирование на языке С/С++ : учеб. пособие / О.Г. Ведерникова; РГУПС. -Ростов н/Д, 2008. -50 с.

14. Дергачева И.В. Алгоритмизация и программирование : учеб.-метод. пособие / И. В. Дергачева; ФГБОУ ВПО РГУПС. -Ростов н/Д, 2013. -96 с.:а-ил.

Учебное издание

Ведерникова Ольга Геннадьевна

ПРОГРАММИРОВАНИЕ
Часть 4

Файловые потоки ввода-вывода

«Электронный университет» ФГБОУ ВО РГУПС

_____0_____

Адрес университета:
344038, Ростов н/Д, пл. Ростовского Стрелкового Полка
Народного Ополчения, 2.