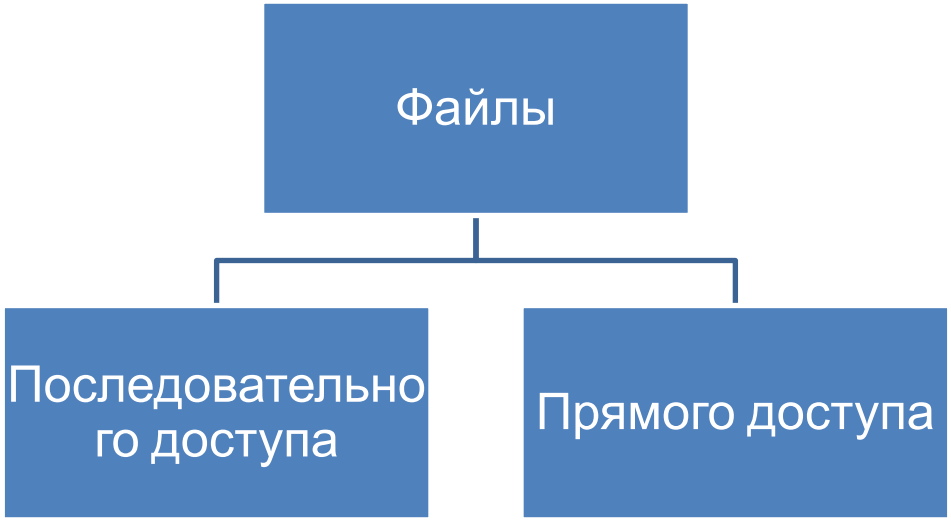
Файли

Классификация файлов по доступу



При последовательном доступе файл рассматривается как последовательность значений, которые передаются в порядке их поступления (от программы или из окружения). Если файл открыт, то передача начинается с начала файла.

При прямом доступе файл рассматривается как набор элементов, занимающих последовательные позиции в линейном порядке; значение может быть передано в элемент файла (или из него), находящийся в любой выбранной позиции. Позиция элемента задается его *индексом*

Классификация файлов в C++

Текстовый файл — файл, в котором каждый символ из используемого набора символов хранится в виде одного байта (кода, соответствующего символу).
Текстовые файлы разбиваются на несколько строк с помощью специального
символа «конец строки». Текстовый файл заканчивается специальным символом

При записи в двоичный файл символы и числа записываются в виде последовательности байт (в своем внутреннем двоичном представлении в памяти компьютера).

Основные концепции при работе с файлами в C++

- Файл – именованный набор байтов, который может быть сохранен на некотором накопителе.
- Используя выходной файловый поток, вы можете писать информацию в файл с помощью оператора вставки (<<).
- Используя входной файловый поток, вы можете читать хранимую в файле информацию с помощью оператора извлечения (>>).
- Для открытия и закрытия файла вы используете методы файловых классов.
- Для работы с файлами необходимо подключить заголовочный файл **<fstream>**
- В <fstream> определены несколько классов и подключены заголовочные файлы **<ifstream>** - файловый ввод и **<ofstream>** - файловый вывод.

Функции работы с файлами

Файл открывается функцией:

имя файловой переменной.open(имя файла)

Файл закрывается функцией:

имя файловой переменной.close()

Проверка ошибок при выполнении файловых операций:

имя файловой переменной.fail()

Если в процессе файловой операции ошибок не было, функция возвратит ложь (0). Однако, если встретилась ошибка, функция fail возвратит истину

Определение конца файла:

имя файловой переменной.eof()

функция возвращает значение 0, если конец файла еще не встретился, и 1, если встретился конец файла.

Используя цикл *while,* ваши программы могут непрерывно читать содержимое файла, пока не найдут конец файла, как показано ниже:

```
while (! input_file.eof())
    {
      // Операторы
    }
```

Открытие файла

Кроме уже описанных процедур файл можно открывать другим способом, а имен сразу после объявления файловой переменной. Однако в этом случае необходимо указывать режимы открытия и тип переменной будет: *fstream.*

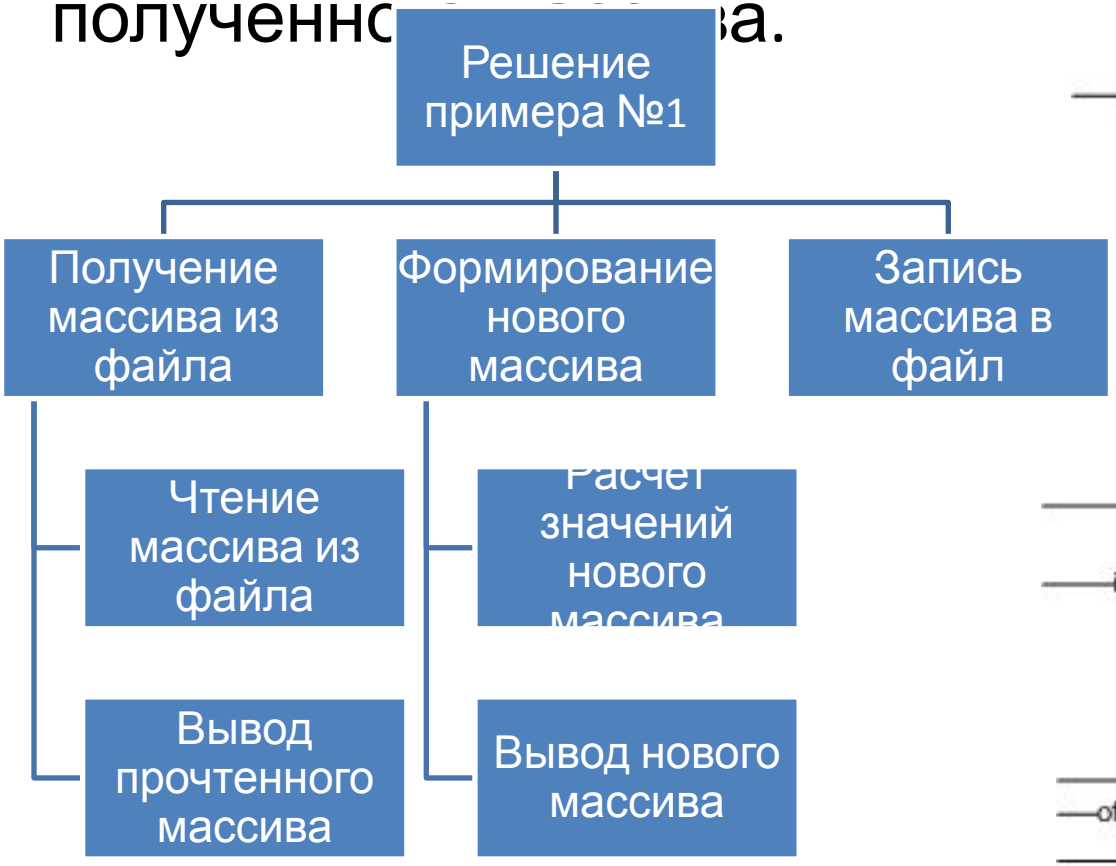
fstream имя переменной («имя файла», режим);

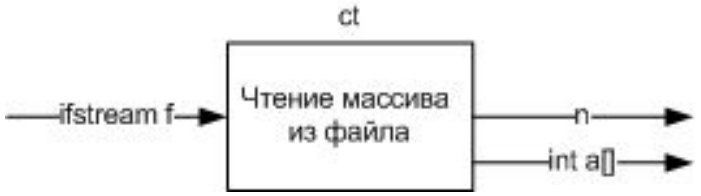
Следующая операция открытия файла открывает файл для вывода, используя режим *ios::noreplace,* чтобы предотвратить перезапись существующего файла:

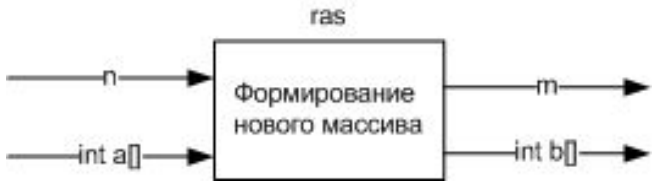
ifstream output_file("FIlename.EXT", ios::out | ios::noreplace);

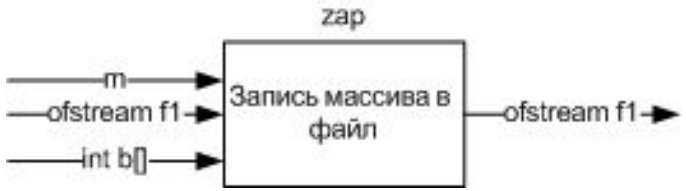
Режим открытия	Назначение
ios::app	Открывает файл в режиме добавления, располагая файловый указатель в конце файла.
ios::ate	Располагает файловый указатель в конце файла.
ios::in	Указывает открыть файл для ввода.
ios::nocreate	Если указанный файл не существует, не создавать файл и возвратить ошибку.
ios::noreplace	Если файл существует, операция открытия должна быть прервана и должна возвратить ошибку.
ios::out	Указывает открыть файл для вывода.
ios::trunc	Сбрасывает (перезаписывает) содержим, з существующего файла.

Пример. Даны целые числа a1,...,a16. Получить новый массив по правилу (a1*a9,a2*a10,..., a8+a16). Найти минимальный элемент полученного массива.





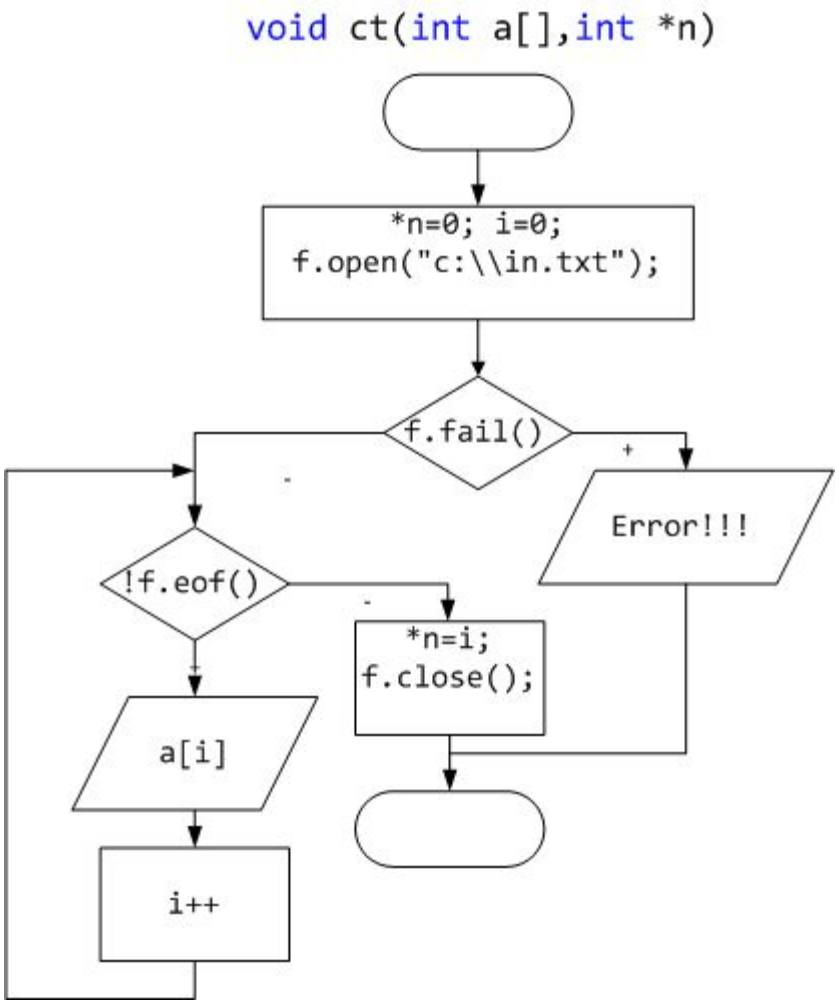


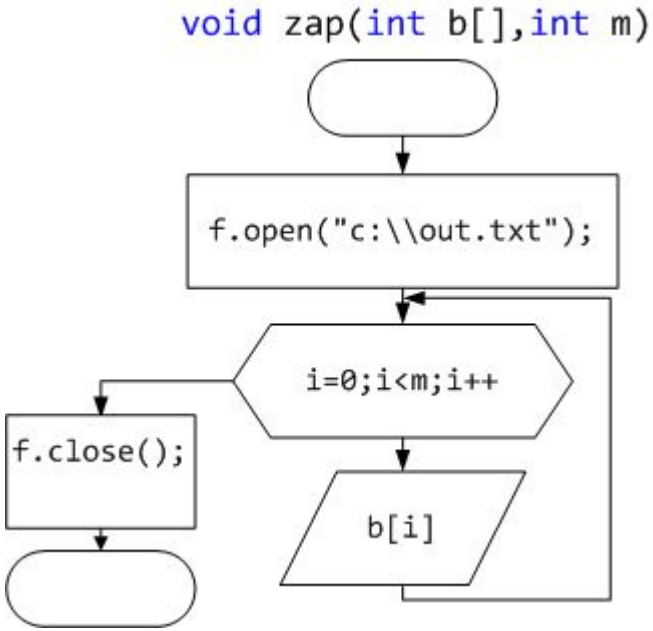


Вход	Действия	Выход
f – текстовый файл	1. Открытие файла для чтения 2. Если есть ошибка открытия Тогда Выход Иначе 2.1 пока не конец файла 2.1.1 Чтение из файла a[i] 2.2.2 i++ 2.2 n=i 2.3 Закрытие файла	a – масив целых n – количество элеметов

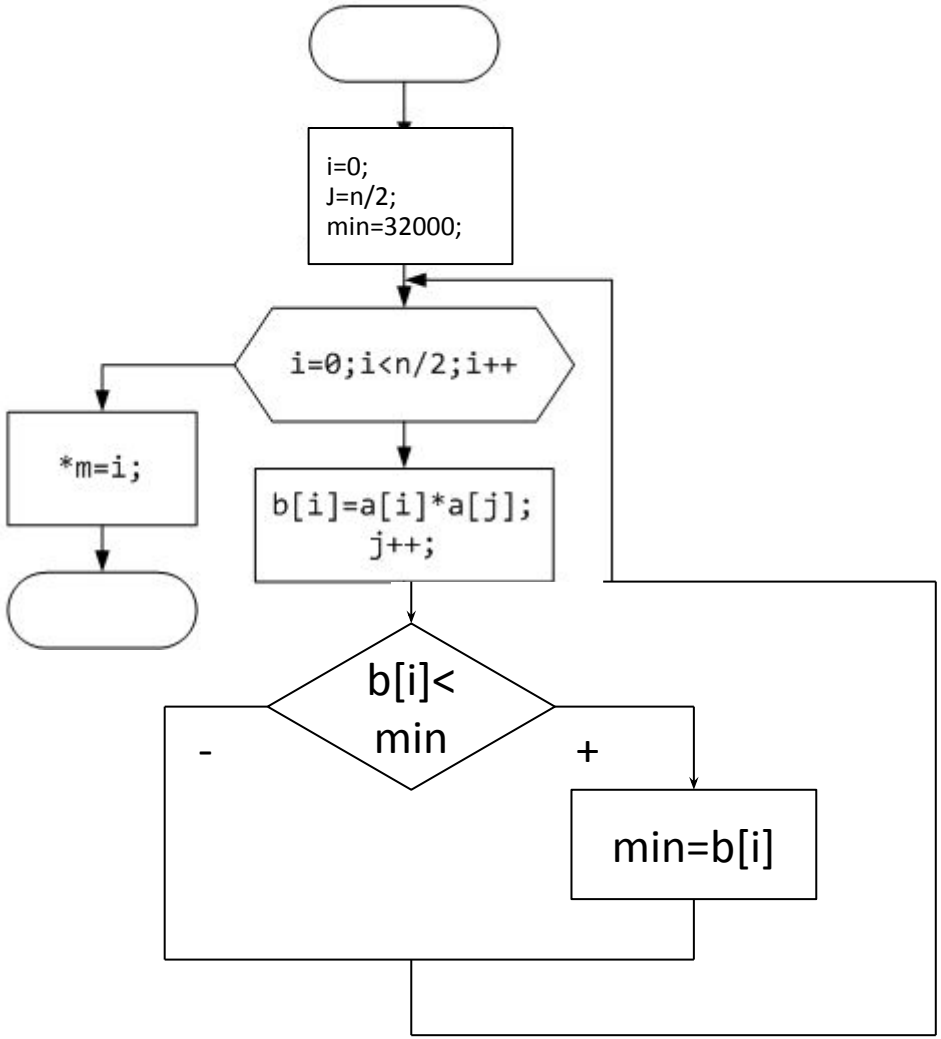
Вход	Действия	Выход
b– масив целых m – количество элеметов	1.i=0 2. j=n/2 3.min=32000 4. i=0..n/2 4.1 b[i]=a[i]*a[j] 4.2 j++ 4.3 Если b[i]<min min=b[i] 5. m=i	b– масив целых m – количество элеметов

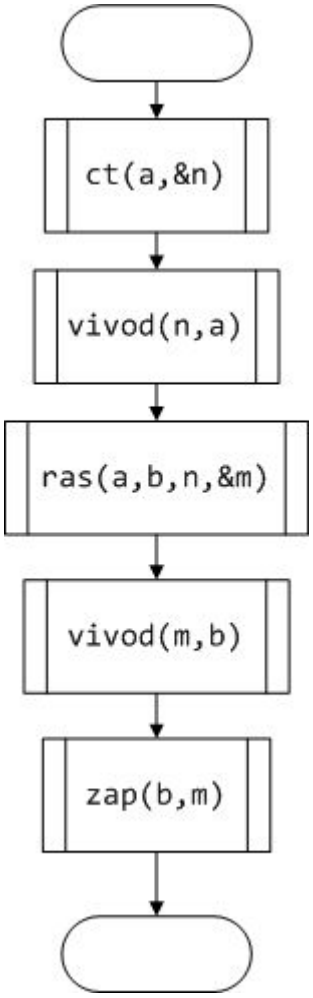
Вход	Действия	Выход
a – масив целых n – количество элеметов	1. Открытие файла для записи 2. i=0..m 2.1 Запись b[i] в файл 3. Закрытие файла	f – текстовый файл


void ct(int a[],int *n)
*n=0; i=0;
f.open("c:\\in.txt");
f.fail()
+
-
Error!!!
!f.eof()
-
+
*n=i;
f.close();
a[i]
i++


void zap(int b[],int m)
f.open("c:\\out.txt");
i=0;i<m;i++
f.close();
b[i]

```
void ras(int a[],int b[],int n,int *m)
```






```
void ct( int a[],int *n)
{
    *n=0;
    int i=0;
    ifstream f;
    f.open("d:\\in.txt");
    if (f.fail())
    {
        cout<<"Error!!! File not open";
        _getch();
        exit(1);
    }
    else
    {
        while(!f.eof())
        {
            f>>a[i];
            i++;
        }
        *n=i;
        f.close();
    }
}
```

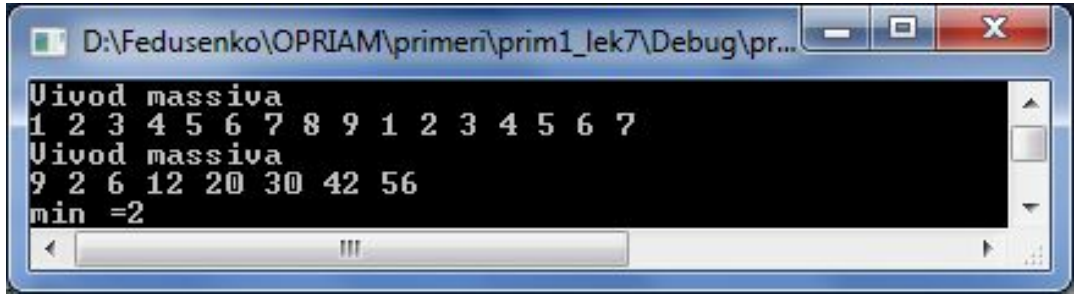
Lister - [c:\in.txt]

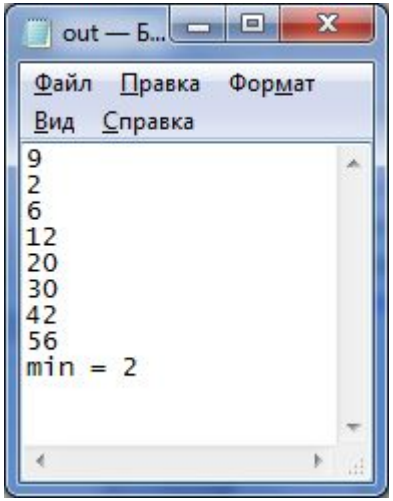
Файл Правка Вид Коди

1
2
3
4
5
6
7
8
9
1
2
3
4
5
6
7

```
void ras (int a[],int b[],int n,int *m,int *min)
{
    int i=0,j=n/2;
    *m=n/2;
    *min=32000;
    for (i=0;i<n/2;i++)
    {
        b[i]=a[i]*a[j];
        j++;
        if (b[i]<*min) *min=b[i];
    }
}
void zap(int b[],int m,int min)
{
    ofstream f;
    f.open ("d:\\out.txt");
    for (int i=0;i<m;i++)
        f<<b[i]<<endl;
    f<<"min = "<<min<<endl;
    f.close();
}
void vivod (int a[],int n)
{
    cout<<"Vivod massiva"<<endl;
    for (int i=0;i<n;i++)
        cout<<a[i]<<" ";
    cout<<endl;
}
```

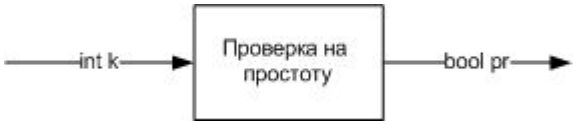
```
int _tmain(int argc, _TCHAR* argv[])
{
    int a[1000],b[1000],n,m,min;
    ct(a,&n);
    vivod(a,n);
    ras(a,b,n,&m,&min);
    vivod(b,m);
    cout<<"min ="<<min<<endl;
    zap(b,m,min);
    _getch();
    return 0;
}
```

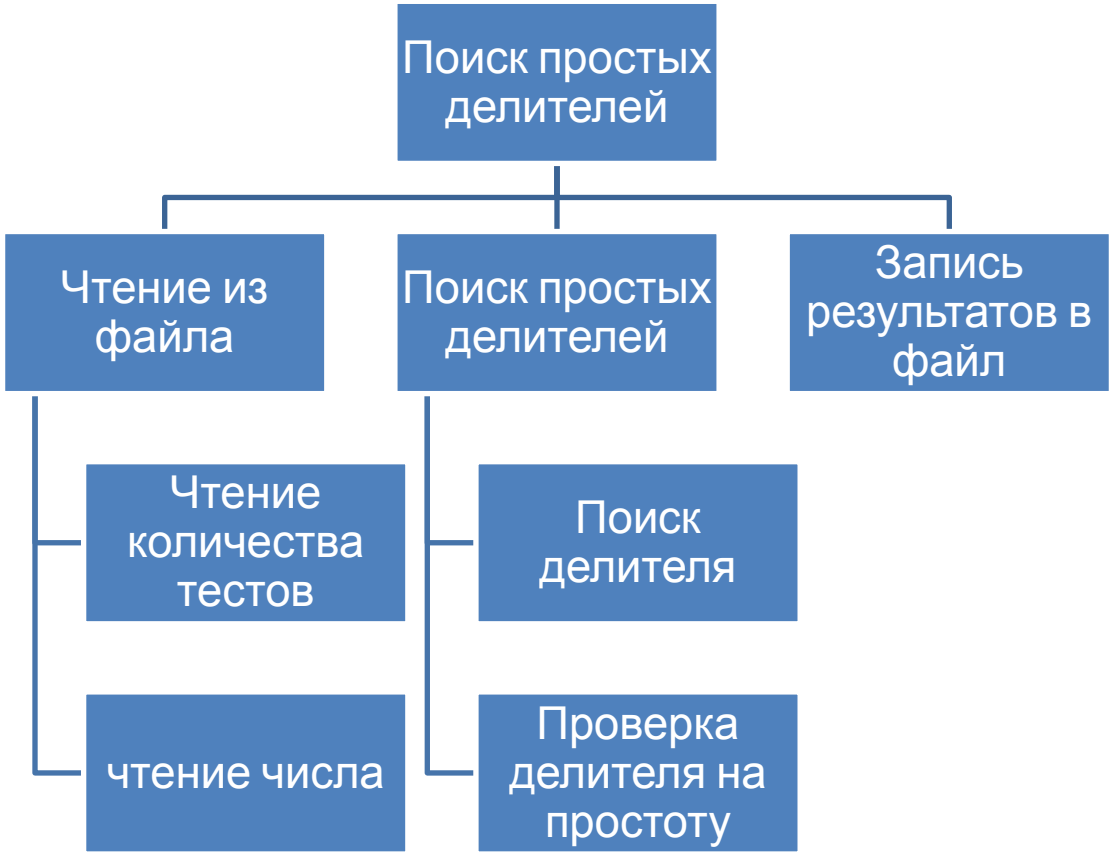




Пример. Найти все простые делители числа M. Ввод и вывод осуществляются в текстовые файлы.

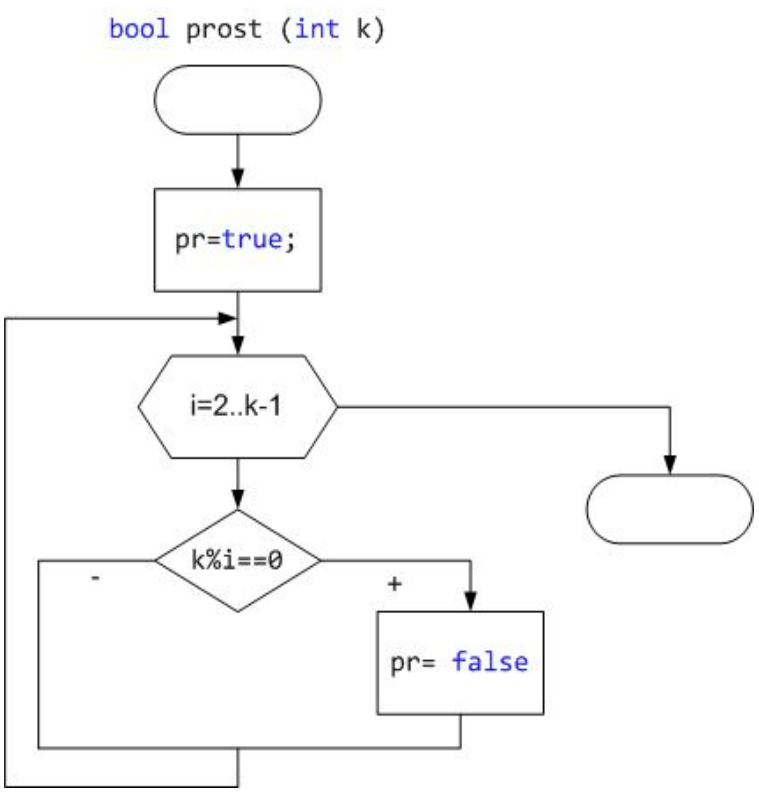
Во входном файле m.txt вначале указано количество тестов. После чего идет:
строка1. число M
Рядок 2 .. N. Число M.
Пример ввода
2
124
3090

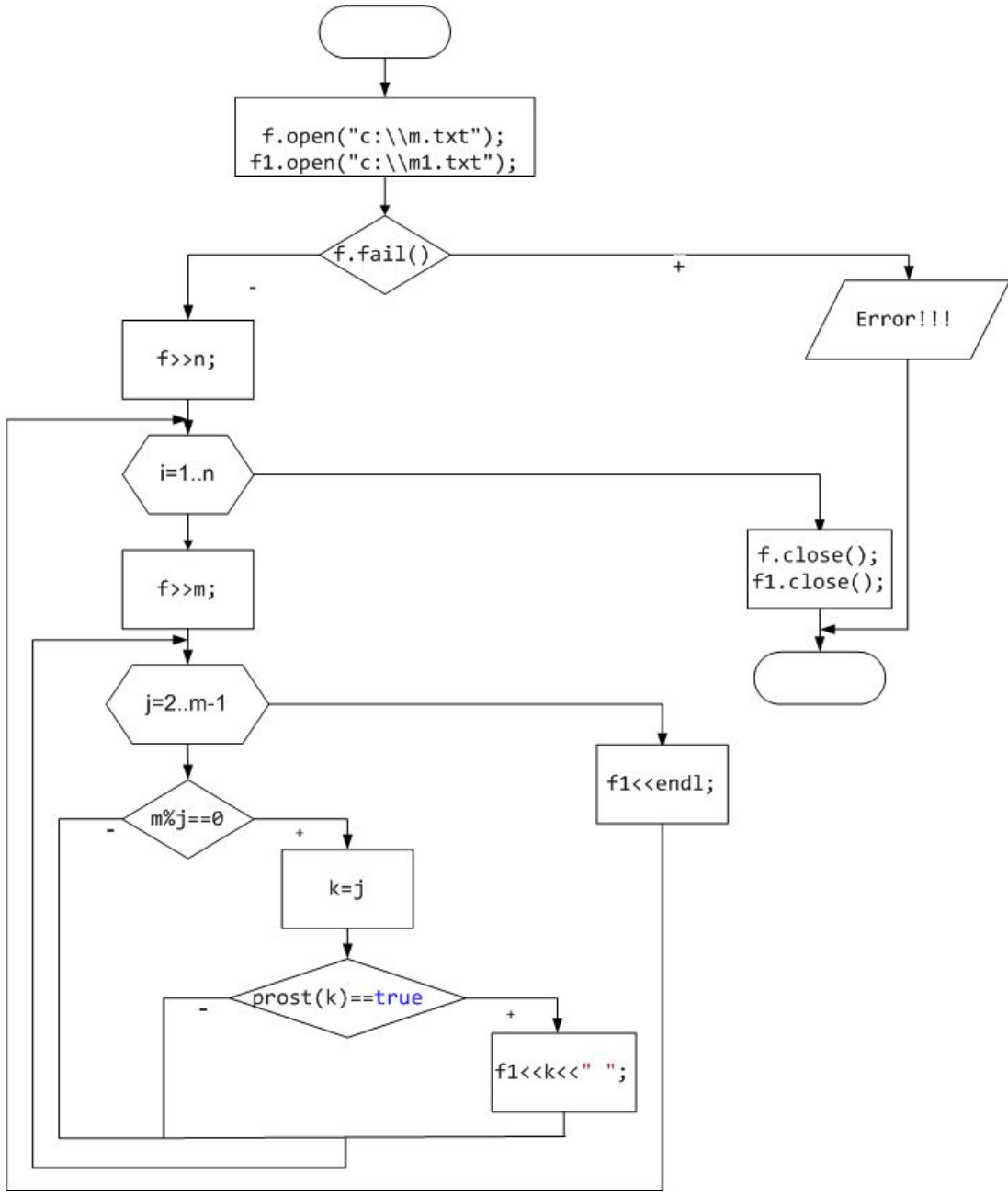




Вход	Действия	Выход
f – текстовый файл	1. Открытие файла для чтения 2. Если есть ошибка открытия Тогда Выход Иначе 2.1 Чтение количества тестов N 2.2. i=1..N 2.2.1 Чтение числа M 2.2.2. j=2..M-1 2.2.2.1 Если M%j=0 то K=j 2.2.2.2 Если K простой делитель, то запись его в f 1	f 1– текстовый файл

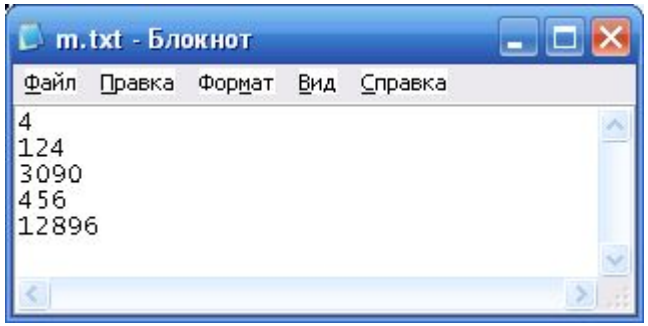
Вход	Действия	Выход
K - целое	1. pr=true 2. i=2..K-1 2.1 Если K%i=0 то pr=false	pr - логическое

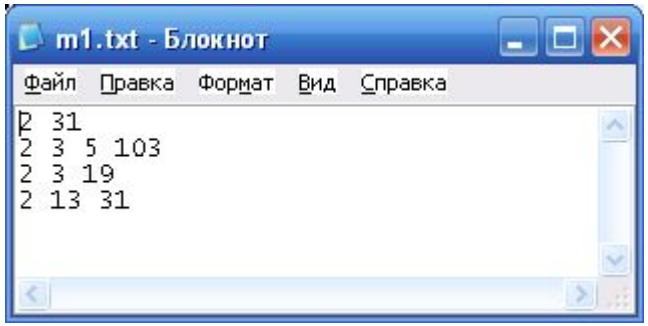

bool prost (int k)
pr=true;
i=2..k-1
k%i==0
-
+
pr= false


f.open("c:\\m.txt");
f1.open("c:\\m1.txt");
f.fail()
-
+
Error!!!
f>>n;
i=1..n
f.close();
f1.close();
f>>m;
j=2..m-1
f1<<endl;
m%j==0
-
+
k=j
prost(k)==true
-
+
f1<<k<<" ";

```
bool prost (int k)
{
	bool pr=true;
	for (int i=2;i<=k-1;i++)
		if (k%i==0) pr= false;
	return pr;
}
int _tmain(int argc, _TCHAR* argv[])
{
	int n,m,k;
	ifstream f;
	f.open("c:\\m.txt");
	ofstream f1;
	f1.open("c:\\m1.txt");
	if (f.fail())
	{
	cout<<"Error!!! Fail not open";
	_getch();
	exit(1);
	}
		return 0;
}
```

```
else
{
f>>n;
for (int i=1;i<=n;i++)
{
	f>>m;
	for (int j=2;j<=m-1;j++)
	{
	if (m%j==0)
	{
		k=j;
		if (prost(k)==true)
		f1<<k<<" ";
	}
}
f1<<endl;
}
f.close();
f1.close();
}
```





ДЗ

В файле записан массив целых чисел a1,..,an необходимо:

- Определить количество целых чисел, входящих в последовательность a1,...,an по одному разу.
- Из модулей членов данной последовательности выбрать наибольший.
- Перенести в хвост одномерного массива максимальный элемент.

Двоичные (бинарные) файлы

1. Открытие файла для чтения

ifstream имя файловой переменной (“путь”,ios::binary|ios::in)

2. Открытие файла для записи

ofstream имя файловой переменной (“путь”,ios::binary|ios::out)

3. Закрытие файла

имя файловой переменной.close();

4. Ввод/вывод. Stream-библиотека C++ имеет перегруженные потоковые функции-элементы write и read для последовательного двоичного файлового ввода/вывода.

Функция-элемент write

Функция **write** посылает ряд байт в выходной поток. Эта функция может записывать любую переменную или экземпляр в поток.

Прототип перегруженной функции-элемента:

ostream& write(const char* buff, int num);

ostream& write(const signed char* buff, int num);

ostream& write(const unsigned char* buff, int num);

Параметр buff - это указатель на буфер, содержащий данные, которые будут посылаться в выходной поток.

Параметр num указывает число байт в буфере, которые передаются в этот поток.

Функция-элемент read

Функция **read** считывает некоторое количество байт из входного потока. Эта функция может считывать любую переменную или экземпляр из потока.

Прототип перегруженной функции-элемента read:

```
ostream& read(char* buff, int num);
ostream& read(signed char* buff, int num);
ostream& read(unsigned char* buff, int num);
```

Параметр buff - это указатель на буфер, который принимает данные из входного потока.

Параметр num указывает число считываемых из потока байт.

Оператор **sizeof**

Для определения числа байт используется оператор **sizeof.**

Результат оператора **sizeof** имеет тип **size_t**, целочисленный тип. Благодаря этому оператору можно избежать жесткого прописывания размеров данных, которые часто зависят от типа компьютера.

Оператор **sizeof** может иметь один из следующих операндов.

- Имя типа. Если оператор **sizeof** используется с именем типа, оно должно быть заключено в скобки.
- Выражения. Если оператор **sizeof** используется с выражением, его можно определять как со скобками, так и без них. Значение выражения не вычисляется.

Если оператор **sizeof** применяется к объекту типа **char**, он дает результат 1. Если оператор **sizeof** применяется к массиву, то результатом является не размер указателя, представленного идентификатором массива, а общее количество байтов в этом массиве.

Если оператор **sizeof** применяется к объекту типа **class**, **struct** или **union**, то результатом будет число байт в объекте этого типа, плюс любое заполнение, которое добавляется для выравнивания членов в границах слова.

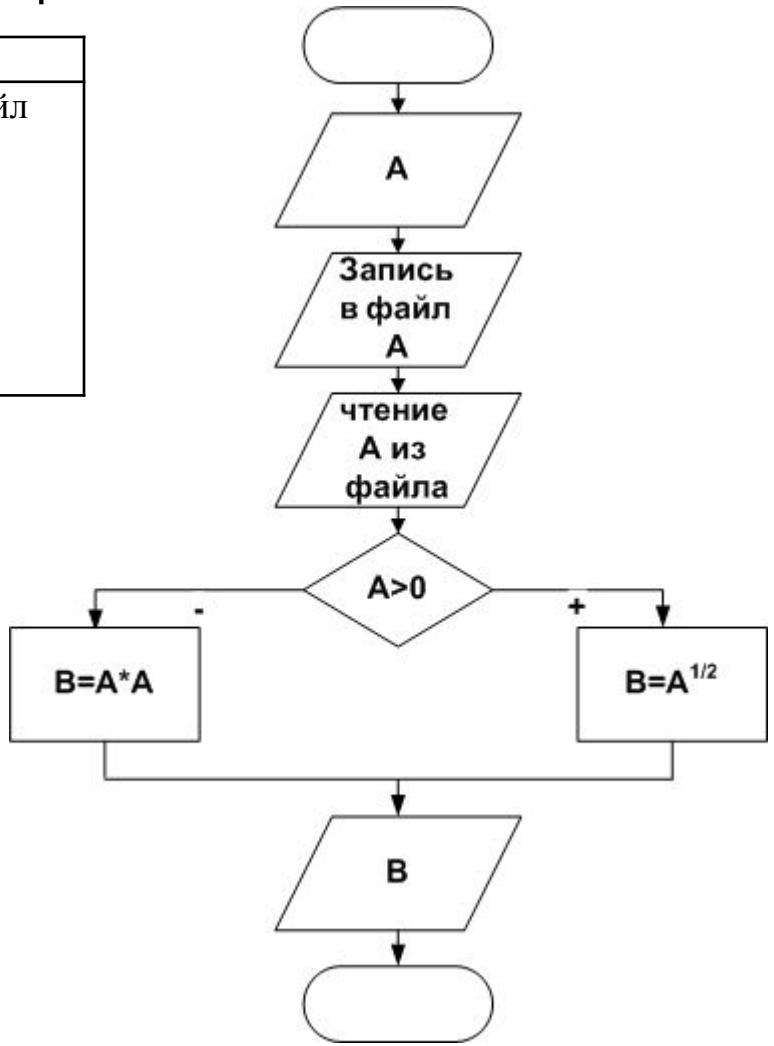
Если оператор **sizeof** применяется к ссылке, он создает такой же результат, как если бы **sizeof** был применен к самому объекту.

Оператор **sizeof** часто используется для вычисления количества элементов в массиве с помощью выражения следующего вида.

sizeof array / sizeof array[0]

Пример. Ввести число А и записать его в бинарный файл. Считать число из файла, если оно положительное то получить его корень, а если отрицательное то возвести в квадрат.

Вход	Действия	Выход
А – целое число	1. Ввод числа А 2. Запись его в бинарный файл f 3. Чтение числа из бинарного файла 4. Если А>0 то В=корень из А иначе В=А*А 5. Вывод В	f – бинарный файл В - целое



```
#include "stdafx.h"
#include<iostream>
#include<fstream>
#include<stdio.h>
#include<conio.h>
#include <iomanip>
#include <math.h>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
        system("CLS"); //Очистка экрана
       int A=0;
       double B=0;
       cout<<"A =" ;
       cin>>A;
       //Открываем файл в двоичном режиме для записи
       ofstream out("C://1.txt",ios::binary|ios::out);
       //Записываем в файл число A
        out.write((char*)&A,sizeof A);
       out.close(); //Закрываем файл
       cout<<"A = "<<A<<endl; //ПоказываемA до его изменений
       //Открываем файл в двоичном режиме только для чтения
       ifstream in("C://1.txt",ios::binary|ios::in);
       //Читаем оттуда информацию и запоминаем её в A
       in.read((char*)&A,sizeof A);
        in.close(); //Закрываем файл
       if (A>0) B=pow(A,1.0/2);
       else B=A*A;
       cout<<"B = "<<B<<endl; //Показываем B
       system("PAUSE");
       return 0;
}
```