Лабораторная работа №10

Сохранение данных в файле с использованием потоков

1. Цель задания:

1) Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2) Разработка программы, в которой данные сохраняются в файле, корректируются и выводятся из файла на печать. Работа с файлом осуществляется с использованием потоковых классов.

2. Теоретические сведения

2.1. Классификация потоков.

Поток - определяется как последовательность байтов и не зависит от конкретного устройства, с которым производится обмен (оперативная память, файл на диске, клавиатура или принтер). Обмен с потоком для увеличения скорости передачи данных производится, как правило, через специальную область оперативной памяти — буфер. Буфер накапливает байты, и фактическая передача данных выполняется после заполнения буфера. При вводе это дает возможность исправить ошибки, если данные из буфера еще не отправлены в программу.

Потоки бывают

- Стандартные: только однонаправленные, либо входные, либо выходные.
- Строковые: могут быть и однонаправленными и двунаправленными
- Файловые: могут быть и однонаправленными и двунаправленными.

Стандартные потоки обозначаются стандартными именами. Эти имена привязаны к стандартным устройствам: клавиатуре и экрану; стандартные потоки можно перенаправить на другие устройства, например, на файл на диске.

Потоки других видов надо объявлять как переменные соответствующего типа. Переменная файлового потока связывается со стандартным файлом на диске.

Стандартные потоки - форматируемые, строковые тоже. Файловые потоки могут быть и форматируемыми, и не форматируемыми.

2.2. Подключение потоков

Для использования стандартных потоков надо задать в программе директиву

```
# include <iostream>
```

В заголовочном файле iostream содержатся описания классов ввода/вывода и четыре стандартных системных объекта:

`cin` – объект класса `istream`, по умолчанию связан с клавиатурой.

`cout` - объект класса `ostream`, по умолчанию связан с экраном.

`clog` - объект класса `ostream`, соответствующий стандартному выводу для ошибок, по умолчанию связан с экраном.

`cerr`- объект класса `ostream`, соответствующий стандартному выводу для ошибок, по умолчанию связан с экраном.

Объект `cout` используется для обычного вывода, а `clog` и `cerr` для вывода сообщений об ошибок. Используются он также, как и `cout`. Специально объявлять в программе стандартные потоки не требуется.

Для использования строковых потоков надо задать в программе директиву

```
# include <sstream>
```

После этого в программе можно объявлять объекты – строковые потоки трех типов:

- входной `istringstream;`
- выходной `ostringstream`;
- двунаправленный `stringstream`;

Для использования файловых потоков надо задать в программе директиву
`# include <fstream>`
После этого в программе можно объявлять объекты – файловые потоки трех типов:
- входной `ifstream`;
- выходной `ofstream`;
- двунаправленный `fstream`;

2.3. Операции ввода-вывода

2.3.1. Вывод

Для форматируемых потоков вывод, как правило, осуществляется перегруженной операцией сдвига влево `operator<<`, а ввод перегруженной операцией сдвига вправо `operator>>`. Операции перегружены для всех стандартных типов. Для несимвольных типов при выводе выполняется преобразование из двоичного формата в символьный тип, при вводе – преобразование из символьного типа во внутренний двоичный.

```
stream<<3.4;stream<<'\n';//вывод константы
stream<<3.45/1.23+0.67; stream<<'\n';//вывод выражения
int a=10;stream<<a; stream<<'\n';//вывод целой переменной
double b=4.123e-2; stream<<b; stream<<'\n';//вывод
вещественной константы
char c='a'; stream<<'a'; stream<<'\n';//вывод символьной
переменной
stream<<a<<b<<'\n';//вывод нескольких значений
char S[]="string1\n";
stream<<S;//вывод строки символов
```

Символы выводятся в поток без преобразования
Для вывода символов можно использовать методы:

Прототип	Пример	
`ostream& put(char c)`	`char c='a';` `stream.put(c);`	Записывает в поток stream символ c
`ostream& write(const char* buf, int size)`	`char c='a';` `stream.write(&c);`	Записывает в поток stream символ c
`ostream& write(const char* buf, int size)`	`char s[]="string1";` `stream.write(s,strlen(s));`	Записывает в поток stream строку символов

Методы вывода возвращают ссылку на поток. Их можно соединять в одно выражение-оператор.
`stream.write(s,strlen(s)).put('\n);`
Вывод из двоичных файлов (режим binary) выполняется методом write:
`ostream& write(const char* buf, streamsize size)`
Метод записывает size символов символьного массива в поток данных. Символы разделители на вывод не влияют.

2.3.2. Ввод

Ввод значений осуществляется из входного потока в переменные программы. Он выполняется одинаково для всех потоков:

```
stream>>a;
stream>>a>>b;
```

Ввод в переменную завершается, если очередной символ в потоке не соответствует типу вводимого значения. Стандартными разделителями в потоке являются: ‘ ‘(пробел), ‘\n’, ‘\t’.

Ввод символов и строк. Символы читаются из потока без преобразования. Поместить в символьную переменную любой символ (в том числе и символ разделитель) можно с помощью методов

Прототип	Пример	
`int_type get();` `istream& get(char &c)`	`c= stream.get();` `stream.get(c)`	ввод из потока одного символа
`istream&read(char*buf,int size)`	`stream.read(&c,1)`	ввод из потока одного символа

Ввод символьных массивов и строк выполняется с помощью операции >> до первого символа-разделителя (обычно пробела). Для ввод строк с пробелами используют методы `get() и getline()`.

Прототип	Пример	
`istream& get(char*str, streamsize count)`	`stream.get(s,50)`	ввод из потока строки с пробелами (пока не встретиться признак конца файла eof или не будет введено 50 символов)
`istream& get(char*str, streamsize count, char lim)`	`stream.get(s,50, ';')`	ввод из потока строки с пробелами (пока не встретиться признак конца файла eof, или не будет введено 50 символов, или не встретится ;)
`istream& getline(char*str, streamsize count)`	`stream.getline(s, 50)`	Работает также как get, но удаляет из входного потока символ ‘\n’
`istream& get(char*str, streamsize count, char lim)`	`stream.getline(s, 50,';')`	

Ввод в двоичные файлы производится методом read:

```
istream&read(char*buf, streamsize size)
```

Метод читает `size` символов в массив `buf`. Символы разделители на ввод не влияют.

2.4. Файловые потоки

Алгоритм работы с файлом:

1. Открыть файл.
2. Выполнить операции обмена между программой и файлом.
3. Закрыть файл.

Файловый поток представлен в программе переменной потокового типа. Файл на диске представлен именем файла. В программе имя файла – это константа-строка или символьный массив. Файловый поток всегда должен быть связан с файлом на диске. Эта связь устанавливается при открытии файла и разрывается при его закрытии.

Файл может быть открыт либо явно методом open(), либо неявно – конструктором при создании потока. Закрыть файл также можно либо явно, с помощью метода close(), либо неявно – деструктором.
Операции обмена между файлом и программой зависят от типа связываемого с файлом потока.
Режимы открытия потока

Режим	Описание
in	открыть поток для чтения (по умолчанию для ifstream)
out	(по умолчанию для ofstream)
trunk	удалить старое содержимое файла (по умолчанию для ofstream)
app	открыть поток для записи в конец файла
ate	открыть поток для чтения и/или записи и встать в конец файла
binary	открыть поток в двоичном режиме

Комбинации режимов

Режим	Описание
out|trunk	стирание и запись, если файла нет, то он создается
out|app	дозапись в файл, если файла нет, то он создается
in|out	чтение и запись, файл должен существовать
in|out|trunk	стирание, чтение и запись, если файла нет, то он создается

2.5. Примеры работы с файловыми потоками

-
```
//создаем выходной текстовый поток, открываем текстовый файл
//и связываем поток с файлом на диске
ofstream stream (”d:/files/number.txt”);
```
-
```
ifstream stream;//создаем входной поток
stream.open(”c:/files/number.txt”);//открываем файл
//обработка – ввод информации из файла
stream.close();//закрываем файл
```
-
```
//создаем выходной поток, открываем файл для дозаписи в конец
//файла и связываем его с потоком
ofstream stream (”d:/files/number.txt”, std::ios::app);
```
-
```
 //проверка открытия потока
if (stream.is_open()) //проверка открытия
{…….}//файл открылся
```
-
```
 //проверка открытия потока
if (stream) //проверка открытия
{…….}//файл открылся
```
-
```
 //проверка открытия потока
if (!stream) //проверка открытия
{…….}//файл не открылся
```
-
```
 if (!(stream>>x)) {….}//ввод завершился неудачей
```
-
```
stream>>x;
if(!stream){…..} //ввод завершился неудачей
```

-
```
 //чтение из потока in
while(!in.eof())
{…….}//операции чтения из потока
```

-
```
int nl=0;
while(stream.get(c))//читать все символы
{
if (c=='\n')nl++;
}
//В этом случае цикл закончится при возникновении ситуации end
of file.
```

-
```
//копирование файлов
#include <fstream>
using namespace std;

//функция копирования из потока in в поток out
void filecopy(ifstream &in, ofstream&out)
{
char c;
while(in.get(c))
out.put(c);
}

int main()
{
ifstream istream("c:/text/number1.txt");
ofstream ostream("c:/text/number2.txt");
if(istream)filecopy(istream,ostream);
return 1;
}
```

3. Постановка задачи

1. Создать пользовательский класс с минимальной функциональностью.
2. Написать функцию для создания объектов пользовательского класса (ввод исходной информации с клавиатуры) и сохранения их в потоке (файле).
3. Написать функцию для чтения и просмотра объектов из потока.
4. Написать функцию для удаления объектов из потока в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.
5. Написать функцию для добавления объектов в поток в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.
6. Написать функцию для изменения объектов в потоке в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.
7. Для вызова функций в основной программе предусмотреть меню.

4. Ход работы

Задача

Пользовательский класс Person с атрибутами:

имя (name) – string
возраст (age) – int

1. Создать пустой проект.
2. Добавить в него класс Person.
3. В файл Person.h добавить описание класса Person:

```
#pragma once
#include <iostream>//библиотека для работы со стандартными потоками
#include <fstream> //библиотека для работы со файловыми потоками
#include <string>
using namespace std;
class Person
{
public:
	Person();
	Person(string, int);
	Person(const Person&);
	Person operator =(const Person&);
	friend ostream& operator <<(ostream &out, const Person&p);
	friend istream& operator>>(istream &in, Person &p);
	public:
	~Person();
private:
	string name;
	int age;
};
```

4. В файл Person.cpp добавить определение методов класса Person:

```
#include "Person.h"
Person::Person()
{
	name=""; age=0;
}
Person::Person(string N,int A)
{
	name=N;age=A;
}
Person::Person(const Person&p)
{
	name=p.name;age=p.age;
}
Person Person::operator =(const Person&p)
{
	if(&p==this) return *this;
	name=p.name;age=p.age;
	return*this;
}
Person::~Person()
{
}

ostream& operator<<(ostream& out, const Person &p)
{
	out<<"name: "<<p.name<<"  age:"<<p.age<<"\n";
	return out;
}

istream& operator>>(istream& in, Person&p)
{
	cout<<"name?";in>>p.name;
```

```
	cout<<"age?";in>>p.age;
	return in;
}
```

5. Добавить в проект файл lab10_main.cpp. В файл записать функцию main(), создающую объекты класса Person для тестирования работы методов класса.

```
#include "Person.h"
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
void main()
{
Person x;//создать объект
cin>>x;//ввести значения атрибутов
cout<<x;//вывести значения атрибутов
Person y;//создать объект
y=x;//операция присваивания
cout<<y; //вывести значения атрибутов
}
```

6. Написать функцию для создания объектов пользовательского класса и сохранения их в файле. Для этого нужно добавить функции в описание класса Person (файл Person.h).

```
friend fstream& operator>>(fstream &fin, Person &p);
friend fstream& operator <<(fstream &fout, const Person&p);
```

7. В файл Person.cpp добавить реализацию этих функций.

```
//дружественные функции для работы с файловыми потоками
fstream& operator>>(fstream& fin, Person&p)
{
	fin>>p.name;
	fin>>p.age;
	return fin;
}
fstream& operator<<(fstream& fout, const Person &p)
{
	fout<<p.name<<"\n"<<p.age<<"\n";
	return fout;
}
```

8. Добавить в проект файл file_work.h
9. В файл file_work.h добавить функцию для сохранения объектов класса Person в потоке.

```
#include "Person.h"
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int make_file(const char* f_name)
{
	fstream stream(f_name,ios::out|ios::trunc);//открыть для записи
	if(!stream)return -1;//ошибка открытия файла
	int n;
	Person p;
	cout<<"N?"; cin>>n;
	for(int i=0;i<n;i++)
	{
		cin>>p;//ввод атрибутов объекта из стандартного потока
		stream<<p<<"\n";//запись объекта в файловый поток
	}
	stream.close();//закрыть поток
	return n;//вернуть количество записанных объектов
}
```

10. Для выполнения тестирования изменить файл lab10_main.cpp следующим образом:

```
#include "Person.h"
#include <iostream>
#include <fstream>
```

```
#include <string>
#include "file_work.h"
using namespace std;
void main()
{	Person p;
	int k,c;
	char file_name[30];
	do
	{		//Меню
		cout<<"\n1. Make file";//создание файла
		cout<<"\n0. Exit\n";
		cin>>c;
		switch(c)
		{
		case 1: cout<<"file name?"; cin>>file_name;//задаем имя файла
			k=make_file(file_name);//вызов функции для записи в файл
			if(k<0)cout<<"Can't make file";//вывод сообщения об ошибке
			break;
		}
	}
		while(c!=0);

}
```

11. Написать функцию для чтения объектов из потока. Для этого добавить в файл file_work.h следующую функцию:

```
int print_file(const char*f_name)
{
	fstream stream(f_name,ios::in);//открыть для чтения
	if(!stream)return -1;//ошибка открытия файла
	Person p; int i=0;
	while(stream>>p)
	{
		cout<<p<<"\n";
		i++;
	}
	stream.close();
	return i;
}
```

12. Для выполнения тестирования изменить файл lab10_main.cpp следующим образом:

```
void main()
{	Person p;
	int k,c;
	char file_name[30];
	do
	{		//Меню
		cout<<"\n1. Make file";
		cout<<"\n2. Print file";//печать файла
		cout<<"\n0. Exit\n";
		cin>>c;
		switch(c)
		{
		case 1: cout<<"file name?"; cin>>file_name;
			k=make_file(file_name);
			if(k<0)cout<<"Can't make file";
			break;
		case 2: cout<<"file name?"; cin>>file_name;//задаем имя файла
			k=print_file(file_name);//вызов функции для печати файла
			if(k==0)cout<<"Empty file\n";//если файл пустой
			if(k<0)cout<<"Can't read file\n";//если файл нельзя открыть
			break;

		}
	}
```

```
            while(c!=0);

}
```

13. Написать функцию для удаления объектов из потока. Для этого добавить в файл file_work.h следующую функцию:

```
int del_file(const char*f_name, int k)
{
      fstream temp("temp", ios::out);//открыть для записи
      fstream stream(f_name,ios::in);//открыть для чтения
      if(!stream)return -1;//ошибка открытия файла
      int i=0;Person p;
      while(stream>>p)//пока нет конца файла выполняем чтение объекта
      {
//если прочитан признак конца файла,то выход из цикла
            if (stream.eof())break;
            i++;
//если номер объекта не равен k, то записываем его во вспомогательный файл
      if(i!=k) temp<<p;
      }
      //закрыть файлы
      stream.close(); temp.close();
      remove(f_name);//удалить старый файл
      rename("temp", f_name);// переименовать temp
      return i;//количество прочитанных
}
```

14. Для выполнения тестирования изменить файл lab10_main.cpp следующим образом:

```
void main()
{     Person p;
      int k,c;
      char file_name[30];
      do
      {           //Меню
            cout<<"\n1. Make file";
            cout<<"\n2. Print file";
            cout<<"\n3. Delete record from file";
            cout<<"\n0. Exit\n";
            cin>>c;
            switch(c)
            {
            case 1: cout<<"file name?"; cin>>file_name;
                  k=make_file(file_name);
                  if(k<0)cout<<"Can't make file";
                  break;
            case 2: cout<<"file name?"; cin>>file_name;
                  k=print_file(file_name);
                  if(k==0)cout<<"Empty file\n";
                  if(k<0)cout<<"Can't read file\n";
                  break;
            case 3: cout<<"file name?"; cin>>file_name;
                  int nom; cout<<"nom?"; cin>>nom;
                  k=del_file(file_name,nom);
                  if(k<0)cout<<"Can't read file";
                  break;
            }
      }
      while(c!=0);
}
```

15. Написать функцию для добавления объектов в поток. Добавление объектов в конец файла отличается от добавления объектов начало и середину, поэтому следует написать две разные функции. Добавим в файл file_work.h функцию для добавления в середину файла, которая будет использовать вспомогательный файл:

```
int add_file(const char*f_name, int k,Person pp)
```

```
{
	fstream temp("temp", ios::out);//открыть для записи
	fstream stream(f_name,ios::in);//открыть для чтения
	if(!stream)return -1;//ошибка открытия файла
	Person p; int i=0, l=0;
	while(stream>>p)
	{
		if (stream.eof())break;
		i++;
		if(i==k)
		{
			temp<<pp;//записать в temp новую запись
			l++;
		}
		temp<<p;
	}
	stream.close(); temp.close();
	remove(f_name);
	rename("temp", f_name);
	return l;//количество добавленных
}
```

16. Добавим в файл file_work.h функцию для добавления в середину файла, которая будет использовать вспомогательный файл:

```
int add_end(const char*f_name, Person pp)
{
	fstream stream(f_name, ios::app);//открыть для добавления
	if(!stream)return -1;//ошибка открытия файла
	stream<<pp; //записать новую запись
	return 1;
}
```

17. Для выполнения тестирования изменить файл lab10_main.cpp следующим образом:

```
void main()
{	Person p,p1;
	int k,c;
	char file_name[30];
	do
	{		//Меню
		cout<<"\n1. Make file";
		cout<<"\n2. Print file";
		cout<<"\n3. Delete record from file";
		cout<<"\n4. Add record to file";
		cout<<"\n0. Exit\n";
		cin>>c;
		switch(c)
		{
		case 1: cout<<"file name?"; cin>>file_name;
			k=make_file(file_name);
			if(k<0)cout<<"Can't make file";
			break;
		case 2: cout<<"file name?"; cin>>file_name;
			k=print_file(file_name);
			if(k==0)cout<<"Empty file\n";
			if(k<0)cout<<"Can't read file\n";
			break;
		case 3: cout<<"file name?"; cin>>file_name;
			int nom; cout<<"nom?"; cin>>nom;
			k=del_file(file_name,nom);
			if(k<0)cout<<"Can't read file";
			break;
		case 4: cout<<"file name?";
			cin>>file_name;
			cout<<"nom?"; cin>>nom;
			cout<<"New person:";
```

```
                cin>>p1;
                k=add_file(file_name,nom,p1);
                if(k<0) cout<<"Can't read file";
                if(k==0) k=add_end(file_name,pp);
                break;
            }
        }
            while(c!=0);
}
```

18. Написать функцию для изменения объектов в потоке. Изменение выполняется с использованием вспомогательного файла. Добавим в файл file_work.h функцию для изменения объектов в потоке:

```
int change_file(const char*f_name, int k,Person pp)
{
    fstream temp("temp", ios::out);//открыть для записи
    fstream stream(f_name,ios::in);//открыть для чтения
    if(!stream)return -1;//ошибка открытия файла
    Person p; int i=0, l=0;
    char x;
    while(stream>>p)
    {
        if (stream.eof())break;
        i++;
        if(i==k)
        {
            cout<<p<<" - is changing... Continue[y/n]?\n";
            cin>>x;
            if(x=='n'||x=='N')break;
            temp<<pp;
            l++;

        }
        else temp<<p;
    }
    stream.close(); temp.close();
    remove(f_name);
    rename("temp", f_name);
    return l;//количество измененных элементов
}
```

19. Для выполнения тестирования изменить файл lab10_main.cpp следующим образом:

```
void main()
{   Person p,p1,p2;
    int k,c;
    char file_name[30];
    do
    {       //Меню
        cout<<"\n1. Make file";
        cout<<"\n2. Print file";
        cout<<"\n3. Delete record from file";
        cout<<"\n4. Add record to file";
        cout<<"\n5. Change record in file";
        cout<<"\n0. Exit\n";
        cin>>c;
        switch(c)
        {
        case 1: cout<<"file name?"; cin>>file_name;
            k=make_file(file_name);
            if(k<0)cout<<"Can't make file";
            break;
        case 2: cout<<"file name?"; cin>>file_name;
            k=print_file(file_name);
            if(k==0)cout<<"Empty file\n";
            if(k<0)cout<<"Can't read file\n";
```

```
                break;
            case 3: cout<<"file name?"; cin>>file_name;
                int nom; cout<<"nom?"; cin>>nom;
                k=del_file(file_name,nom);
                if(k<0)cout<<"Can't read file";
                break;
            case 4: cout<<"file name?";
                cin>>file_name;
                cout<<"nom?"; cin>>nom;
                cout<<"New person:";
                //Person p1;
                cin>>p1;
                k=add_file(file_name,nom,p1);
                if(k<0) cout<<"Can't read file";
                if(k==0) k=add_end(file_name,p1);
                break;
            case 5: cout<<"file name?";
                cin>>file_name;
                cout<<"nom?"; cin>>nom;
                cout<<"New person:";
                //Person p2;
                cin>>p2;
                k=change_file(file_name,nom,p2);
                if(k<0) cout<<"\nCan't read file";
                if(k==0) cout<<"\nNot such record";
                break;
            }
        }
            while(c!=0);
}
```

5. Варианты

№	Задание
1	Создать класс Time для работы с временными интервалами. Интервал должен быть представлен в виде двух полей: минуты типа int и секунды типа int. при выводе минуты отделяются от секунд двоеточием. Реализовать: – сложение временных интервалов (учесть, что в минуте не может быть более 60 секунд) – сравнение временных интервалов (==) **Задание:** • Удалить все записи равные заданному значению. • Увеличить все записи с заданным значением на 1 минуту 30 секунд. • Добавить K записей после элемента с заданным номером.
2	Создать класс Time для работы с временными интервалами. Интервал должен быть представлен в виде двух полей: минуты типа int и секунды типа int. при выводе минуты отделяются от секунд двоеточием. Реализовать: – вычитание временных интервалов (учесть, что в минуте не может быть более 60 секунд) – сравнение временных интервалов (!=) **Задание:** • Удалить все записи не равные заданному значению. • Уменьшить все записи с заданным значением на 1 минуту 30 секунд. Значение интервала не должно быть меньше 0 минут 0 секунд. • Добавить K записей в начало файла.

3	Создать класс Time для работы с временными интервалами. Интервал должен быть представлен в виде двух полей: минуты типа int и секунды типа int. при выводе минуты отделяются от секунд двоеточием. Реализовать: – добавление секунд (учесть, что в минуте не может быть более 60 секунд) – вычитание секунд **Задание:** • Удалить все записи равные заданному значению. • Уменьшить все записи с заданным значением на 1 минуту 30 секунд. Значение интервала не должно быть меньше 0 минут 0 секунд. • Добавить K записей после элемента с заданным значением.
4	Создать класс Time для работы с временными интервалами. Интервал должен быть представлен в виде двух полей: минуты типа int и секунды типа int. при выводе минуты отделяются от секунд двоеточием. Реализовать: – добавление секунд (учесть, что в минуте не может быть более 60 секунд) – сравнение временных интервалов (== и !=) **Задание:** • Удалить все записи из интервала от k1 до k2, где k1 и k2 переменные типа Time. • Увеличить все записи с заданным значением на 1 минуту 30 секунд. • Добавить K записей в начало файла.
5	Создать класс Time для работы с временными интервалами. Интервал должен быть представлен в виде двух полей: минуты типа int и секунды типа int. при выводе минуты отделяются от секунд двоеточием. Реализовать: – вычитание секунд – сравнение временных интервалов (== и !=) **Задание:** • Удалить все записи большие заданного значения. • Увеличить все записи с заданным значением на 1 минуту 30 секунд. • Добавить K записей после записи с номером N.
6	Создать класс Money для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа int для копеек. Дробная часть числа при выводе на экран должна быть отделена от целой части запятой. Реализовать: – сложение денежных сумм, – вычитание денежных сумм, **Задание:** • Удалить все записи равные заданному значению. • Увеличить все записи с заданным значением на 1 рубль 50 копеек. • Добавить K записей после элемента с заданным номером.
7	Создать класс Money для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа int для копеек. Дробная часть числа при выводе на экран должна быть отделена от целой части запятой. Реализовать: – деление сумм, – умножение суммы на дробное число. **Задание:** • Удалить все записи не равные заданному значению. • Уменьшить все записи с заданным значением в два раза. • Добавить K записей в начало файла.

8	Создать класс Money для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа int для копеек. Дробная часть числа при выводе на экран должна быть отделена от целой части запятой. Реализовать: – сложение суммы и дробного числа – операции сравнения (>, <, ==). **Задание:** • Удалить все записи из интервала от k1 до k2, где k1 и k2 переменные типа Money. • Увеличить все записи с заданным значением в два раза. • Добавить K записей в начало файла.
9	Создать класс Money для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа int для копеек. Дробная часть числа при выводе на экран должна быть отделена от целой части запятой. Реализовать: – вычитание дробного числа из суммы – операции сравнения (==, !=). **Задание:** • Удалить все записи равные заданному значению. • Уменьшить все записи с заданным значением на 1 рубль 50 копеек. Значение интервала не должно быть меньше 0 рублей 0 копеек. • Добавить K записей после элемента с заданным значением.
10	Создать класс Money для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа int для копеек. Дробная часть числа при выводе на экран должна быть отделена от целой части запятой. Реализовать: – операции сравнения (==, !=). – вычитание копеек (--) (постфиксная и префиксная формы) **Задание:** • Удалить все записи большие заданного значения. • Увеличить все записи с заданным значением на 1 рубль 50 копеек. • Добавить K записей после записи с номером N.
11	Создать класс Money для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа int для копеек. Дробная часть числа при выводе на экран должна быть отделена от целой части запятой. Реализовать: – операции сравнения (<, >). – добавление копеек (++) (постфиксная и префиксная формы) **Задание:** • Удалить все записи большие заданного значения. • Уменьшить все записи с заданным значением в два раза. • Добавить K записей после элемента с заданным номером.
12	Создать класс Pair (пара чисел). Пара должна быть представлено двумя полями: типа int для первого числа и типа double для второго. Первое число при выводе на экран должно быть отделено от второго числа двоеточием. Реализовать: – операции сравнения (<, >). – операция ++, которая работает следующим образом: если форма операции префиксная, то увеличивается первое число, если форма операции постфиксная, то увеличивается второе число. **Задание:** • Удалить все записи большие заданного значения. • Увеличить все записи с заданным значением на число L. • Добавить K записей после записи с номером N.

13	Создать класс Pair (пара чисел). Пара должна быть представлено двумя полями: типа int для первого числа и типа double для второго. Первое число при выводе на экран должно быть отделено от второго числа двоеточием. Реализовать: – операции сравнения (<, >). – операция --, которая работает следующим образом: если форма операции префиксная, то уменьшается первое число, если форма операции постфиксная, то уменьшается второе число. **Задание:** • Удалить все записи равные заданному значению. • Уменьшить все записи с заданным значением на число L. • Добавить K записей после элемента с заданным значением
14	Создать класс Pair (пара чисел). Пара должна быть представлено двумя полями: типа int для первого числа и типа double для второго. Первое число при выводе на экран должно быть отделено от второго числа двоеточием. Реализовать: – операции сравнения (==,!=). – вычитание константы из пары (уменьшается первое число, если константа целая, второе, если константа вещественная). **Задание:** • Удалить все записи из интервала от k1 до k2, где k1 и k2 переменные типа Pair. • Увеличить все записи с заданным значением в два раза. • Добавить K записей в начало файла
15	Создать класс Pair (пара чисел). Пара должна быть представлено двумя полями: типа int для первого числа и типа double для второго. Первое число при выводе на экран должно быть отделено от второго числа двоеточием. Реализовать: – вычитание пар чисел – добавление константы к паре (увеличивается первое число, если константа целая, второе, если константа вещественная). **Задание:** • Удалить все записи меньшие заданного значения. • Увеличить все записи с заданным значением на число L. • Добавить K записей после элемента с заданным номером.

6. Контрольные вопросы

1. Что такое поток?
2. Какие типы потоков существуют?
3. Какую библиотеку надо подключить при использовании стандартных потоков?
4. Какую библиотеку надо подключить при использовании файловых потоков?
5. Какую библиотеку надо подключить при использовании строковых потоков?
6. Какая операция используется при выводе в форматированный поток?
7. Какая операция используется при вводе из форматированных потоков?
8. Какие методы используются при выводе в форматированный поток?
9. Какие методы используется при вводе из форматированного потока?
10. Какие режимы для открытия файловых потоков существуют?
11. Какой режим используется для добавления записей в файл?
12. Какой режим (комбинация режимов) используется в конструкторе ifstream file(“f.txt”)?
13. Какой режим (комбинация режимов) используется в конструкторе fstream file(“f.txt”)?

14. Какой режим (комбинация режимов) используется в конструкторе ofstream file(“f.txt”)?
15. Каким образом открывается поток в режиме ios::out|ios::app?
16. Каким образом открывается поток в режиме ios::out |ios::trunc?
17. Каким образом открывается поток в режиме ios::out |ios::in|ios::trunk?
18. Каким образом можно открыть файл для чтения?
19. Каким образом можно открыть файл для записи?
20. Привести примеры открытия файловых потоков в различных режимах.
21. Привести примеры чтения объектов из потока.
22. Привести примеры записи объектов в поток.
23. Сформулировать алгоритм удаления записей из файла.
24. Сформулировать алгоритм добавления записей в файл.
25. Сформулировать алгоритм изменения записей в файле.

7. Содержание отчета

1) Постановка задачи (общая и конкретного варианта).
2) Описание пользовательского класса.
3) Перегруженные операции пользовательского класса.
4) Определение функций для работы с файлом (создание, вывод, удаление, добавление, изменение).
5) Функция main().
6) Объяснение результатов работы программы.
7) Ответы на контрольные вопросы.