Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Информатика»

Тема: Лабораторная работа №10

"Сохранение данных в файле с использованием потоков"

Семестр 2

Выполнил работу

Студенты группы РИС-22-2Б

Третьяков Н. А.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2023

Постановка задачи

1. Создать пользовательский класс с минимальной функциональностью
2. Написать функцию для создания объектов пользовательского класс (ввод исходной информации с клавиатуры) и сохранить их в потоке (файле)
3. Написать функцию для чтения и просмотра объектов в потоке(файле)
4. Написать функцию для удоления 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 функции в основной программе предусмотреть меню.

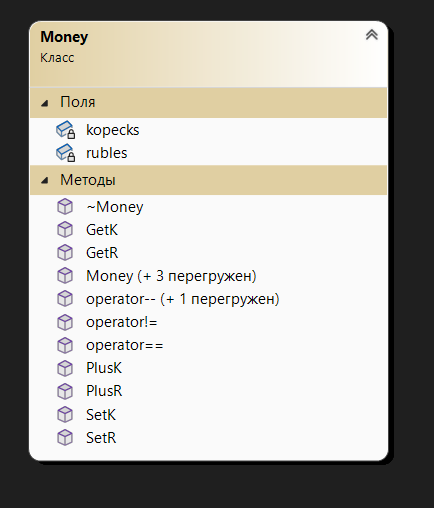
Задание

Удалить все записи больше заданного значения

Увеличить все записи на 1 руб 50 коп

Добавить К записей после записи с номером

UML



Код программы

Money.h

#pragma once

#include <iostream>

#include <string>

#include <cstdio>

using namespace std;

class Money

{

private:

int rubles;

int kopecks;

public:

Money();

Money(string str);

Money(int r, int k);

Money(const Money&);

int GetR();

int GetK();

void SetR(int r);

void SetK(int k);

void PlusR(int r);

void PlusK(int k);

bool operator == (const Money&);

bool operator != (const Money&);

Money& operator --();

Money& operator --(int);

friend std::ostream& operator << (std::ostream& out, const Money&);

friend std::istream& operator >> (std::istream& in, Money&);

~Money() {};

};

Money.cpp

#include <iostream>

#include <string>

#include <cstdio>

#include <cstdlib>

#include "Money.h"

using namespace std;

Money::Money()

{

rubles = 0;

kopecks = 0;

}

Money::Money(int r, int k)

{

if (k < 100)

{

kopecks = k;

rubles = r;

}

else

{

rubles = r + k / 100;

kopecks = k % 100;

}

}

Money::Money(const Money& m)

{

rubles = m.rubles;

kopecks = m.kopecks;

}

Money::Money(string str)

{

string a;

int i = 0;

while (str[i] != ',')

{

i++;

a += str[i];

}

i++;

rubles = stoi(a) ;

a = "";

for (i = i; i < str.size(); i++)

{

a += str[i];

}

kopecks = stoi(a);

}

int Money::GetR()

{

return rubles;

}

int Money::GetK()

{

return kopecks;

}

void Money::SetR(int r)

{

rubles = r;

}

void Money::SetK(int k)

{

kopecks = k;

}

void Money::PlusR(int r)

{

rubles += r;

}

void Money::PlusK(int k)

{

kopecks += k;

}

bool Money::operator!=(const Money& m)

{

if (this->rubles != m.rubles)

{

return 1;

}

else

{

if (this->kopecks != m.kopecks)

{

return 1;

}

else

{

return 0;

}

}

}

bool Money::operator==(const Money& m)

{

if (this->rubles == m.rubles)

{

if (this->kopecks == m.kopecks)

{

return 1;

}

else

{

return 0;

}

}

else

{

return 0;

}

}

Money& Money::operator -- ()

{

kopecks = kopecks - 1;

return \*this;

}

Money& Money::operator--(int)

{

kopecks = kopecks - 1;

return \*this;

}

ostream& operator << (ostream& out, const Money& p)

{

return(out << p.rubles << "," << p.kopecks<< endl);

}

istream& operator >> (istream& in, Money& p)

{

in >> p.rubles;

in >> p.kopecks;

return in;

}

Main.cpp

#include <iostream>

#include <fstream>

#include <string>

#include <cstdio>

#include "Money.h"

using namespace std;

const char\* F = "Main File.txt";

int CoutFile()

{

fstream stream(F, ios::out | ios::trunc);

if (!stream) return -1;

int n;

cout << "How many items to add to the stream" << endl;

cin >> n;

Money m;

for (int i = 0; i < n; i++)

{

cout << "Enter the data" << endl;

cin >> m;

stream << m;

}

stream.close();

}

void Chek(string str, Money& m)

{

string a;

int k = 0;

while (str[k] != ',')

{

a += str[k];

k++;

}

k ++;

int i = stoi(a);

m.SetR(i);

a = "";

for (int n = k; n < str.size(); n++)

{

a += str[n];

}

i = stoi(a);

m.SetK(i);

a = "";

}

int CinFile()

{

string str;

fstream stream(F, ios::in);

if (!stream) return -1;

Money m;

while (stream >> str)

{

Chek(str ,m);

cout << m << '\n';

}

stream.close();

}

int DelFile(int r, int k)

{

string str;

fstream temp("temp.txt", ios::out);

fstream stream(F , ios::in);

if (!stream) return -1;

int i = 0;

Money m;

while (getline(stream, str))

{

Chek(str, m);

if (m.GetR() == r + k / 100)

if (m.GetK() < k % 100)

temp << m;

else

cout << "Del" << endl;

else

if (m.GetR() < r + k / 100)

temp << m;

else

cout << "Del" << endl;

}

stream.close();

temp.close();

remove(F );

rename("temp.txt", F );

return 1;

}

int AddF(int k, Money m1)

{

int i = 0;

fstream temp("temp.txt", ios::out);

fstream stream(F, ios::in);

if (!stream) return -1;

Money m;

string str;

while (getline(stream, str))

{

if (stream.eof())break;

i++;

if (k == i)

{

temp << m1;

}

temp << str << endl;

}

stream.close();

temp.close();

remove(F );

rename("temp.txt", F );

return 1;

}

int ChangeF()

{

string str;

fstream temp("temp.txt", ios::out);

fstream stream(F, ios::in);

if (!stream) return -1;

Money m;

while (getline(stream, str))

{

Chek(str, m);

if (m.GetK() + 50 > 100)

{

m.SetK((m.GetK() + 50) % 100);

m.PlusR(2);

}

else

{

m.PlusK(50);

m.PlusR(1);

}

temp << m;

}

stream.close();

temp.close();

remove(F );

rename("temp.txt", F );

return 1;

}

int main()

{

CoutFile();

CinFile();

DelFile( 100, 30);

CinFile();

Money z(100, 20);

AddF(1, z);

CinFile();

ChangeF();

CinFile();

}

# Ответы на контрольные вопросы

1. Что такое поток?

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.

1. Какие типы потоков существуют?

Потоки бывают

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

1. Какую библиотеку надо подключить при использовании стандартных 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.

1. Какую библиотеку надо подключить при использовании файловых потоков?

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;

1. Какую библиотеку надо подключить при использовании строковых потоков?

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;

1. Какая операция используется при выводе в форматированный поток?

Для форматируемых потоков вывод, как правило, осуществляется перегруженной операцией сдвига влево operator<<.

stream<<3.4;stream<<’\n’;//вывод константы

1. Какая операция используется при вводе из форматированных потоков?

Для форматируемых потоков ввод перегруженной операцией сдвига вправо operator>>.

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’.

1. Какие методы используются при выводе в форматированный поток?

Для вывода символов можно использовать методы:



1. Какие методы используется при вводе из форматированного потока?



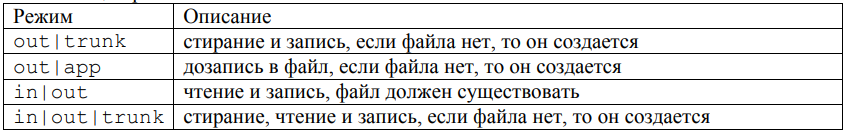


1. Какие режимы для открытия файловых потоков существуют?

Режимы открытия потока:

****

Комбинации режимов:



1. Какой режим используется для добавления записей в файл?

Для добавления записи в конец файла используется режим app.

Также, используется режим ate для чтения и/или записи в конец файла.

1. Какой режим (комбинация режимов) используется в конструкторе ifstream file(“f.txt”)?

Используется режим in - открыть поток для чтения (по умолчанию для ifstream).

1. Какой режим (комбинация режимов) используется в конструкторе fstream file(“f.txt”)?

Используется комбинация режимов in | out - чтение и запись, файл должен существовать.

1. Какой режим (комбинация режимов) используется в конструкторе ofstream file(“f.txt”)?

Используется режим out - открыть поток для записи в файл (по умолчанию для ofstream).

1. Каким образом открывается поток в режиме ios::out|ios::app?

Поток открывается для дозапись в файл, если файла нет, то он создаётся.

1. Каким образом открывается поток в режиме ios::out |ios::trunc?

Поток открывается для стирания и записи, если файла нет, то он создаётся.

1. Каким образом открывается поток в режиме ios::out |ios::in|ios::trunk?

Поток открывается для стирания, чтения и записи, если файла нет, то он создаётся.

1. Каким образом можно открыть файл для чтения?

Файл может быть открыт либо явно методом open(), либо неявно – конструктором при создании потока.

1. Каким образом можно открыть файл для записи?

Файл может быть открыт либо явно методом open(), либо неявно – конструктором при создании потока.

1. Привести примеры открытия файловых потоков в различных режимах.

Ниже приведены примеры открытия файловых потоков в различных режимах:

Открытие файла для записи: ofstream outputFile("example.txt");

Открытие файла для чтения: ifstream inputFile("example.txt");

Открытие файла для добавления записей в конец: ofstream appendFile("example.txt", ios::app);

Открытие файла для чтения и записи: fstream inOutFile("example.txt", ios::in | ios::out);

Открытие файла для чтения и записи, обрезание файла до нулевой длины: fstream truncFile("example.txt", ios::trunc | ios::in | ios::out);

1. Привести примеры чтения объектов из потока.

Чтение объектов из потока в C++ может быть выполнено с помощью операторов извлечения (>>). Ниже приведены примеры чтения различных типов данных из потока:

Чтение целого числа: int number; inputFile >> number;

Чтение строки: string str; getline(inputFile, str);

Чтение символа: char ch; inputFile >> ch;

Чтение числа с плавающей точкой: double value; inputFile >> value;

1. Привести примеры записи объектов в поток.

Запись объектов в поток в C++ может быть выполнена с помощью операторов вставки (<<). Ниже приведены примеры записи различных типов данных в поток:

Запись целого числа: int number = 10; outputFile << number;

Запись строки: string str = "example"; outputFile << str;

Запись символа: char ch = 'a'; outputFile << ch;

Запись числа с плавающей точкой: double value = 3.14; outputFile << value;

1. Сформулировать алгоритм удаления записей из файла.

Алгоритм удаления записей из файла в С++ может быть следующим:

1. Открыть файл для чтения и записи.

2. Создать временный файл.

3. Прочитать записи из исходного файла и проверять каждую запись на соответствие критерию удаления.

4. Записать нужные записи во временный файл.

5. Закрыть исходный файл и временный файл.

6. Удалить исходный файл с помощью функции.

7. Переименовать временный файл в имя исходного файла.

1. Сформулировать алгоритм добавления записей в файл.

1. Открыть файл для записи с помощью функции fstream::open().

2. Проверить, что файл успешно открыт с помощью функции fstream::is\_open().

3. Создать объект записи, который будет содержать данные для записи в файл.

4. Заполнить объект записи данными.

5. Записать объект записи в файл с помощью оператора << или функции fstream::write().

6. Проверить, что запись прошла успешно с помощью функции.

7. Повторять шаги 3-6 для каждой новой записи.

8. Закрыть файл с помощью функции fstream::close().

1. Сформулировать алгоритм изменения записей в файле.

1. Открыть файл для чтения и записи с помощью функции std::fstream.

2. Проверить, что файл успешно открыт.

3. Считать данные из файла в структуру или массив данных.

4. Изменить необходимые данные в структуре или массиве.

5. Перезаписать измененные данные в файл.

6. Закрыть файл.