Федеральное государственное автономное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет», ПНИПУ

ЛАБОРАТОРНАЯ РАБОТА

СОХРАНЕНИЕ ДАННЫХ В ФАЙЛЕ С ИСПОЛЬЗОВАНИЕМ ПОТОКОВ

Выполнил: студент группы РИС-23-3б

Артем Владимирович Швецов

Проверила: доцент кафедры ИТАС

Ольга Андреевна Полякова

Пермь 2024

**Постановка задачи**

1. Создать пользовательский класс с минимальной функциональностью.
2. Написать функцию для создания объектов пользовательского класса и сохранения их в потоке.
3. Написать функцию для чтения и просмотра объектов из потока.
4. Написать функцию для удаления объектов из потока в соответствии с заданием варианта.
5. Написать функцию для добавления объектов в поток в соответствии с заданием варианта.
6. Написать функцию для изменения объектов в потоке в соответствии с заданием варианта.
7. Для вызова функций предусмотреть меню.

Для варианта 15:

Создать класс Pair (пара чисел). Пара должна быть представлена двумя полями: первое – int, второе – double. При выводе на экран первое число должно быть отделено от второго двоеточием. Реализовать:

* Вычитание пар чисел
* Добавление константы к паре (если константа типа int, то прибавление к первому полю, если double – ко второму)

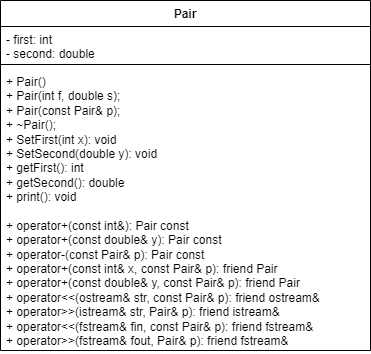
**Задание:**

* Удалить все записи меньше заданного значения.
* Увеличить все значения с заданным значением на число L.
* Добавить K записей после элемента с заданным номером.

**Анализ задачи**

1. Для работы с файловыми потоками подключается библиотеку fstream.
2. Функции для работы с файлами определяются в отдельном заголовочном файле.
3. При удалении и изменении объектов в потоке используется временный файл.

**UML**



**Код**

Pair.h

#pragma once

#include <iostream>

#include <fstream>

using namespace std;

class Pair

{

private:

int first;

double second;

public:

Pair();

Pair(int f, double s);

Pair(const Pair& p);

~Pair();

void SetFirst(int x);

void SetSecond(double y);

int GetFirst();

double GetSecond();

void Print();

//перегруженные операции

Pair operator+(const int& x) const;

Pair operator+(const double& y) const;

Pair operator-(const Pair& p) const;

friend Pair operator+(const int& x, const Pair& p);

friend Pair operator+(const double& y, const Pair& p);

friend ostream& operator<<(ostream& str, const Pair& p);

friend istream& operator>> (istream& str, Pair& p);

friend fstream& operator>> (fstream& fin, Pair& p);

friend fstream& operator<< (fstream& fout, Pair& p);

};

Pair.cpp

#pragma once

#include "Pair.h"

Pair::Pair() {

first = 0;

second = 0;

}

Pair::Pair(int f = 0, double s = 0) {

this->first = f;

this->second = s;

}

Pair::Pair(const Pair& p) {

this->first = p.first;

this->second = p.second;

}

Pair::~Pair() {

}

int Pair::GetFirst() { return this->first; }

void Pair::SetFirst(int x) { this->first = x; }

double Pair::GetSecond() { return this->second; }

void Pair::SetSecond(double y) { this->second = y; }

void Pair::Print() {

cout << this->first << ":" << this->second << endl;

}

Pair Pair::operator+(const int& x) const {

return Pair(this->first + x, this->second);

}

Pair Pair::operator+(const double& y) const {

return Pair(this->first, this->second + y);

}

Pair Pair::operator-(const Pair& p) const {

return Pair(this->first - p.first, this->second - p.second);

}

Pair operator+(const int& x, const Pair& p) {

return p + x;

}

Pair operator+(const double& y, const Pair& p) {

return p + y;

}

ostream& operator<<(ostream& str, const Pair& p) {

str << " " << p.first << " : " << p.second << endl;

return str;

}

istream& operator>> (istream& str, Pair& p) {

cout << endl << "Enter pair values" << endl;

cout << "First value: ";

str >> p.first;

cout << "Second value: ";

str >> p.second;

return str;

}

fstream& operator>> (fstream& fin, Pair& p) {

fin >> p.first;

fin >> p.second;

return fin;

}

fstream& operator<< (fstream& fout, Pair& p) {

fout << p.first << "\n" << p.second << "\n";

return fout;

}

file\_work.h

#pragma once

#include <iostream>

#include <fstream>

#include <string>

#include "Pair.h"

using namespace std;

int make\_file(const char\* f\_name)

{

fstream stream(f\_name, ios::out | ios::trunc);

if (!stream) return -1;

int n;

Pair p;

cout << "Number of pairs: ";

cin >> n;

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

{

cin >> p;

stream << p << '\n';

}

stream.close();

return n;

}

int print\_file(const char\* f\_name)

{

fstream stream(f\_name, ios::in);

if (!stream) return -1;

Pair p;

int i = 0;

while (stream >> p)

{

cout << p << endl;

i++;

}

stream.close();

return i;

}

int del\_file(const char\* f\_name, double k)

{

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

fstream stream(f\_name, ios::in);

if (!stream) return -1;

Pair p;

int i = 0;

while (stream >> p)

{

if (!stream.eof()) {

i++;

if (p.GetFirst() + p.GetSecond() >= k) {

temp << p;

}

}

}

stream.close();

temp.close();

remove(f\_name);

rename("temp", f\_name);

return i;

}

int add\_file(const char\* f\_name, int k, Pair adp)

{

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

fstream stream(f\_name, ios::in);

if (!stream) return -1;

Pair p;

int i = 0, j = 0;

if (k == 0)

{

temp << adp;

j++;

}

while (stream >> p)

{

if (!stream.eof()) {

i++;

temp << p;

if (i == k)

{

temp << adp;

j++;

}

}

}

stream.close();

temp.close();

remove(f\_name);

rename("temp", f\_name);

return j;

}

int add\_end(const char\* f\_name, Pair adp)

{

fstream stream(f\_name, ios::app);

if (!stream) return -1;

stream << adp;

return 1;

}

template <typename TT>

int change\_file(const char\* f\_name, double k, TT l)

{

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

fstream stream(f\_name, ios::in);

if (!stream) return -1;

Pair p;

int i = 0;

while (stream >> p)

{

if (!stream.eof()) {

if (p.GetFirst() + p.GetSecond() == k) {

p = p + l;

i++;

}

temp << p;

}

}

stream.close();

temp.close();

remove(f\_name);

rename("temp", f\_name);

return i;

}

Main.cpp

#include <iostream>

#include <fstream>

#include "Pair.h"

#include "file\_work.h"

int main() {

Pair p, p1;

int k, c, pos;

double l, value;

char file\_name[30];

do

{

cout << endl << "1. Change work file name";

cout << endl << "2. Make file";

cout << endl << "3. Print file";

cout << endl << "4. Delete record from file";

cout << endl << "5. Add record to file";

cout << endl << "6. Change records in file";

cout << endl << "0. Exit" << endl;

cin >> c;

switch (c)

{

case 1:

cout << "Enter file name: ";

cin >> file\_name;

break;

case 2:

k = make\_file(file\_name);

if (k < 0) cout << "Error in file creation" << endl;

break;

case 3:

k = print\_file(file\_name);

if (k == 0) cout << "File is empty" << endl;

if (k < 0) cout << "Error in file reading" << endl;

break;

case 4:

cout << "Enter target value: ";

cin >> value;

k = del\_file(file\_name, value);

if (k < 0) cout << "Error in file reading" << endl;

break;

case 5:

cout << "Enter adding position: ";

cin >> pos;

cout << "Enter adding pair: ";

cin >> p1;

k = add\_file(file\_name, pos, p1);

if (k < 0) cout << "Error in file reading" << endl;

break;

case 6:

cout << "Enter target value: ";

cin >> value;

cout << "Enter adding value: ";

cin >> l;

k = change\_file(file\_name, value, l);

if (k == 0) cout << "No record changed" << endl;

if (k < 0) cout << "Error in file reading" << endl;

break;

default:

break;

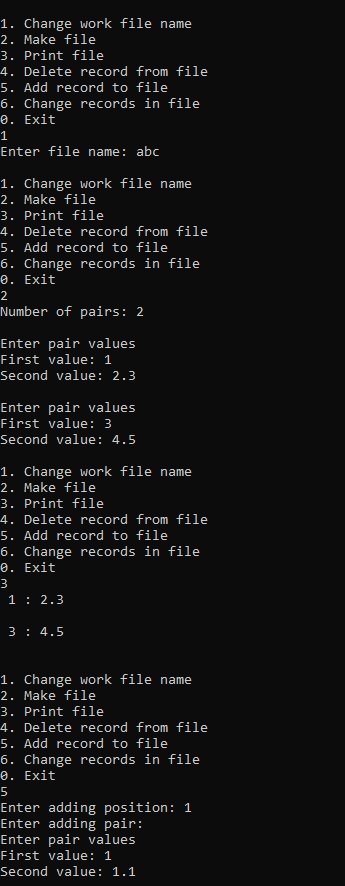
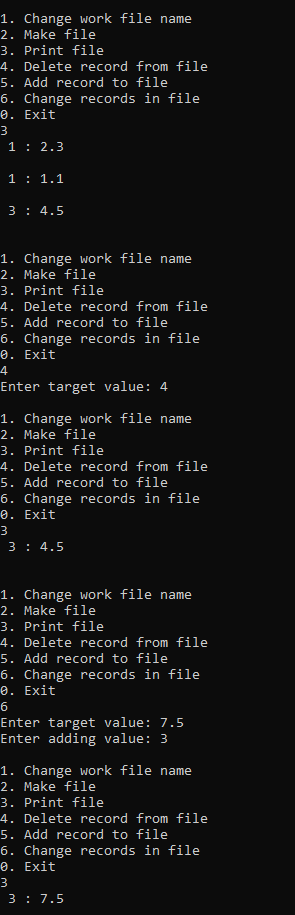
}

} while (c != 0);

return 0;

}

**Решение**

**Выводы**

В ходе выполнения лабораторной работы были изучены файловые потоки и файловый потоковый ввод-вывод данных.

**Ответы на контрольные вопросы:**

1. Поток – последовательность байтов.
2. Стандартные, строковые, файловые.
3. Iostream
4. Fstream
5. Sstream
6. Операция сдвига влево (operator<<)
7. Операция сдвига вправо (operator>>)
8. put(), write()
9. get(), read()
10. in, out, app, trunk, ate, binary
11. out
12. in
13. in|out|trunk
14. out|trunk
15. Открывается для записи, с возможностью добавления в конец файла.
16. Открывается для записи, с удалением предыдущего содержимого.
17. Открывается для чтения и записи, с удалением предыдущего содержимого.
18. Используя конструктор с ifstream.
19. Используя конструктор с ofstream.
20. Fstream finapp(“file.txt”, std::ios::in| std::ios::app)/ fstream fout(“file.txt”, std::ios::out| std::ios::trunk)
21. Пример чтения

fstream finapp(“file.txt”, std::ios::in| std::ios::app)

CustomType buffer;

finapp >> buffer;

1. Пример записи

fstream fout(“file.txt”, std::ios::out| std::ios::trunk)

CustomType buffer(10, 10);

fout << buffer;

1. Алгоритм удаления записей
   1. открыть в потоке файл для чтения
   2. открыть в потоке буферный файл
   3. запись в буферный файл записей с игнорированием соответствующих описанию на удаление
   4. удаление старого файла и переименование буферного
2. Алгоритм добавления записей
   1. открыть в потоке файл для чтения
   2. открыть в потоке буферный файл
   3. запись в буферный файл записей до тех пор пока не достигнута нужная позиция или конец файла
   4. ввод заданной записи в буферный файл
   5. если в изначальном файле еще остались непрочитанные записи, продолжение чтения и записи в буферный
   6. удаление старого файла и переименование буферного
3. Алгоритм добавления записей
   1. открыть в потоке файл для чтения
   2. открыть в потоке буферный файл
   3. запись в буферный файл записей, измененных в соответствии условиям
   4. удаление старого файла и переименование буферного

**Github**

<https://github.com/Hitikov/Lab_OOP_10>