Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

ПНИПУ

**Лабораторная работа по ООП**

**«№10»**

Выполнил:

студент группы РИС-23-2б

Ившин Максим Сергеевич

Проверила:

доцент кафедры ИТАС

О.А. Полякова

2024 г.

**Разработка алгоритма**

**Постановка задачи**

1. Создать пользовательский класс с минимальной функциональностью.

1. Написать функцию для создания объектов пользовательского класса (ввод исходной информации с клавиатуры) и сохранения их в потоке

(файле).

1. Написать функцию для чтения и просмотра объектов из потока.
2. Написать функцию для удаления объектов из потока в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.
3. Написать функцию для добавления объектов в поток в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.
4. Написать функцию для изменения объектов в потоке в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.
5. Для вызова функций в основной программе предусмотреть меню.

**Задание**

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.

**Анализ задачи**

1. Необходимо реализовать класс Pair. В классе надо прописать конструкторы: по умолчанию, с параметрами и копирования; а также деструктор, для предотвращения утечки памяти.
2. В классе Pair необходимо прописать операторы перегрузки для необходимых операций.
3. Для работы с файлами необходимо подключить директиву

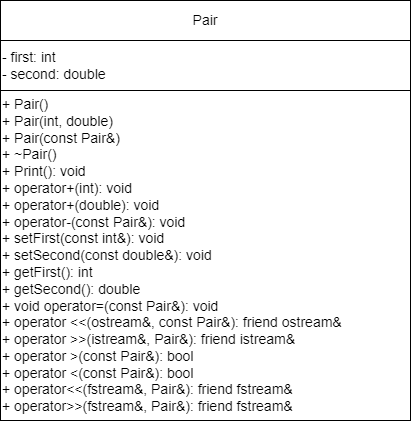
#include <fstream>

1. Написать функцию для создания объектов пользовательского класса (ввод исходной информации с клавиатуры) и сохранения их в потоке

(файле).

* 1. Необходимо написать функцию для чтения и просмотра объектов из потока.
  2. Необходимо написать функцию для удаления объектов из потока
* соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.
  1. Необходимо написать функцию для добавления объектов в поток
* соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.
  1. Необходимо написать функцию для изменения объектов в потоке
* соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.

**UML диаграмма**

****

**Код на C++**

**main.cpp:**

#include <iostream>

#include <fstream>

#include "Pair.h"

#include "Pair.cpp"

#include "file\_work.h"

using namespace std;

int main()

{

    // system("chcp 1251 > Null");

    srand(time(0));

    int N;

    do

    {

        cout << "Введите кол-во элементов: ";

        cin >> N;

    } while (N < 1);

    cout << "\n";

    write\_to\_a\_file(N);

    bool isRunning = true;

    while (isRunning)

    {

        int current\_action = choosing\_an\_action();

        int tmp\_int;

        double tmp\_double;

        Pair pair;

        switch (current\_action)

        {

        case 1:

            cout << "Текущее содержимое файла:\n";

            print\_file();

            cout << endl;

            break;

        case 2:

            do

            {

                cout << "Введите кол-во элементов: ";

                cin >> tmp\_int;

            } while (tmp\_int < 1);

            write\_to\_a\_file(tmp\_int);

            break;

        case 3:

            cout << "Введите минимальное значение пары (int double):\n";

            cin >> pair;

            removing\_el\_smaller\_than(pair);

            break;

        case 4:

            do

            {

                cout << "Какое число вы хотите ввести?\n";

                cout << "1 - целое \n2 - вещественное\n";

                cin >> N;

            } while (N < 1 || N>2);

            switch (N)

            {

            case 1:

                cout << "Введите ЦЕЛОЕ число: ";

                cin >> tmp\_int;

                add\_L(tmp\_int);

                break;

            default:

                cout << "Введите ВЕЩЕСТВЕННОЕ число: ";

                cin >> tmp\_double;

                add\_L(tmp\_double);

                break;

            }

            break;

        case 5:

            do

            {

                cout << "Введите НОМЕР, после которого необходимо ввести k элементов\n";

                cin >> N;

            } while (N < 1);

            do

            {

                cout << "Введите количество элементов, которые необходимо добавить\n";

                cin >> tmp\_int;

            } while (tmp\_int < 1);

            add\_k\_el\_after(N, tmp\_int);

            break;

        default:

            isRunning = false;

            break;

        }

    }

    return 0;

}

**File\_work.h:**

#pragma once

using namespace std;

void randomize(Pair& p)

{

    int x = rand() % 1001;

    double y = (rand() % 100001) \* 0.01;

    p = Pair(x, y);

}

bool write\_to\_a\_file(int N) {

    fstream F1("labWork10.txt", ios::out | ios::trunc);

    if (!F1)

    {

        cout << "Ошибка открытия файла\n";

        return 0;

    }

    for (int i = 0; i < N; i++)

    {

        Pair pair;

        randomize(pair);

        F1 << pair << "\n";

    }

    F1.close();

    return 1;

}

bool print\_file()

{

    fstream F1("labWork10.txt", ios::in);

    //in чтение

    if (!F1)

    {

        cout << "Ошибка открытия файла\n";

        return 0;

    }

    Pair pair;

    while (F1 >> pair)

    {

        cout << pair << "\n";

    }

    return 1;

}

void error\_mes(bool N)

{

    if (!N)

    {

        cout << "Файл не найден\n";

    }

}

bool removing\_el\_smaller\_than(Pair& para)

{

    fstream F1("labWork10.txt", ios::in);

    //in чтение

    if (!F1)

    {

        cout << "Ошибка открытия файла\n";

        return 0;

    }

    fstream tmp("temporary\_file.txt", ios::out | ios::trunc);

    //out - перезаписывание

    //trunk - удалить старое содержимое файла (по умолчанию для оfstream)

    Pair pair;

    while (F1 >> pair)

    {

        if (pair > para)

        {

            tmp << pair << "\n";

        }

    }

    F1.close();

    tmp.close();

    remove("labWork10.txt");

    rename("temporary\_file.txt", "labWork10.txt");

    return 1;

}

template<typename T>

bool add\_L(T L)

{

    fstream F1("labWork10.txt", ios::in);

    //in чтение

    if (!F1)

    {

        cout << "Ошибка открытия файла\n";

        return 0;

    }

    fstream tmp("temporary\_file.txt", ios::out | ios::trunc);

    //out - перезаписывание

    //trunk - удалить старое содержимое файла (по умолчанию для оfstream)

    Pair pair;

    while (F1 >> pair)

    {

        pair = pair + L;

        tmp << pair << endl;

    }

    F1.close();

    tmp.close();

    remove("labWork10.txt");

    rename("temporary\_file.txt", "labWork10.txt");

    return 1;

}

bool add\_k\_el\_after(int number, int quantity)

{

    fstream F1("labWork10.txt", ios::in);

    //in чтение

    if (!F1)

    {

        cout << "Ошибка открытия файла\n";

        return 0;

    }

    fstream tmp("temporary\_file.txt", ios::out | ios::trunc);

    //out - перезаписывание

    //trunk - удалить старое содержимое файла (по умолчанию для оfstream)

    Pair pair;

    int counter = 0;

    while (F1 >> pair)

    {

        counter++;

        tmp << pair << "\n";

        if (counter == number)

        {

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

            {

                Pair new\_pair;

                randomize(new\_pair);

                tmp << new\_pair << "\n";

            }

        }

    }

    if (counter < number)

        cout << "Достигнут конец файла! Элемент с номером " << number << "не найден\n";

    F1.close();

    tmp.close();

    remove("labWork10.txt");

    rename("temporary\_file.txt", "labWork10.txt");

    return 1;

}

int choosing\_an\_action()

{

    cout << "Выберите действие\n";

    cout << "1 - Вывести содержимое файла\n";

    cout << "2 - Перезаписать содержимое файла\n";

    cout << "3 - Удалить все записи меньшие заданного значения\n";

    cout << "4 - Увеличить все записи с заданным значением на число L\n";

    cout << "5 - Добавить K записей после элемента с заданным номером\n";

    cout << "0 - Конец работы\n";

    int choice;

    do

    {

        cout << ">  ";

        cin >> choice;

    } while (choice < 0 || choice > 6);

    cout << "\n";

    return choice;

}

**Pair.cpp:**

#pragma once

#include "Pair.h"

using namespace std;

Pair::Pair()

{

    this->first = 0;

    this->second = 0;

}

Pair::Pair(int first, double second)

{

    this->first = first;

    this->second = second;

}

Pair::Pair(const Pair& other)

{

    this->first = other.first;

    this->second = other.second;

}

void Pair::Print()

{

    cout << "(" << this->first << " : " << this->second << ")\n";

}

Pair Pair::operator+(Pair& p)

{

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

}

Pair Pair::operator-(const Pair& para)

{

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

}

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

{

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

}

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

{

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

}

void Pair::setFirst(const int& x)

{

    this->first = x;

}

void Pair::setSecond(const double& y)

{

    this->second = y;

}

int Pair::getFirst()

{

    return this->first;

}

double Pair::getSecond()

{

    return this->second;

}

void Pair::operator=(const Pair& para)

{

    this->first = para.first;

    this->second = para.second;

}

ostream& operator <<(ostream& stream, const Pair& p)

{

    stream << p.first << " : " << p.second;

    return stream;

}

istream& operator >>(istream& stream, Pair& p)

{

    // cout << "first : second = ";

    stream >> p.first >> p.second;

    return stream;

}

bool Pair::operator >(const Pair& p)

{

    if (this->first < p.first)

    {

        return false;

    }

    if (this->first == p.first && this->second < p.second)

    {

        return false;

    }

    return true;

}

bool Pair::operator <(const Pair& p)

{

    if (this->first > p.first)

    {

        return false;

    }

    if (this->first == p.first && this->second > p.second)

    {

        return false;

    }

    return true;

}

fstream& operator<<(fstream& stream, Pair& para)

{

    stream << para.first << "\n" << para.second << "\n";

    return stream;

}

fstream& operator>>(fstream& stream, Pair& para)

{

    stream >> para.first;

    stream >> para.second;

    return stream;

}

**Pair.h:**

#pragma once

#include <iostream>

#include <fstream>

using namespace std;

class Pair

{

    int first;

    double second;

public:

    Pair();

    Pair(int, double);

    Pair(const Pair&);

    ~Pair() {}

    void Print();

    Pair operator+(Pair& p);

    Pair operator-(const Pair& para);

    Pair operator+(const int& x);

    Pair operator+(const double& y);

    void setFirst(const int& x);

    void setSecond(const double& y);

    int getFirst();

    double getSecond();

    void operator=(const Pair& para);

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

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

    bool operator >(const Pair& p);

    bool operator <(const Pair& p);

    friend fstream& operator<<(fstream& stream, Pair& para);

    friend fstream& operator>>(fstream& stream, Pair& para);

};

**Результаты работы**

**GitHub**

<https://github.com/Korovay4ik/Laboratory-works>

**Контрольные вопросы**

1. **Что такое поток?**

Поток в программировании — это последовательность данных, которая может быть прочитана или записана. 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?**

iostream

1. **Какую библиотеку надо подключить при использовании файловых потоков?**

fstream

1. **Какую библиотеку надо подключить при использовании строковых потоков?**

stream

**6.** **Какая** **операция** **используется** **при** **выводе** **в**

**форматированный поток?**

Операция << используется при выводе в форматированный поток.

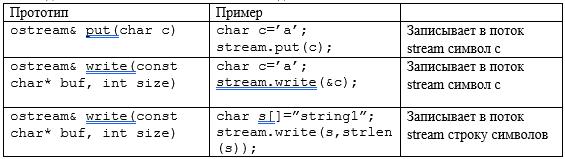
1. **Какая операция используется при вводе из форматированных**

**потоков?**

Операция >> используется при вводе из форматированных потоков.

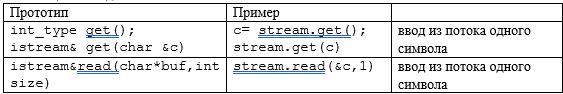
1. **Какие методы используются при выводе в форматированный**

**поток?**

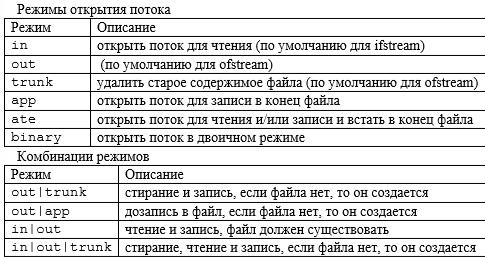


1. **Какие методы используется при вводе из форматированного**

**потока?**



1. **Какие режимы для открытия файловых потоков существуют?**



1. **Какой режим используется для добавления записей в файл?**

Режим ios::out используется для добавления записей в файл.

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

ios::in

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

ios::in

ios::out

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

ios::out

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

ios::out указывает на то, что файл будет открыт для записи, а ios::app гаранти-рует, что при каждой записи данные будут добавляться в конец файла. Если файл не существует, он будет создан. Если файл уже существует, то данные бу-дут добавляться в конец файла, не перезаписывая его содержимое.

1. **Каким образом открывается поток в режиме ios::out**

**|ios::trunc?**

ios::out - файл будет открыт в режиме вывода и при этом, если файл уже суще-ствует, его содержимое будет удалено (ios::trunc). Если же файл не существует, он будет создан.

1. **Каким образом открывается поток в режиме ios::out**

**|ios::in|ios::trunk?**

ios::out указывает на то, что файл будет использоваться для записи, ios::in ука-зывает на то, что файл будет использоваться для чтения, а ios::trunc указывает на то, что содержимое файла будет очищено перед записью в него.

1. **Каким образом можно открыть файл для чтения?**

ifstream file;

file.open("file.txt", ios::in);

1. **Каким образом можно открыть файл для записи?**

ifstream file;

file.open("file.txt", ios::out);

1. **Привести примеры открытия файловых потоков в различных**

**режимах.**

ifstream file;

file.open("file.txt", ios::out | ios::app);

Добавляет новое содержимое в конец файла, не удаляя предыдущее

содержимое

1. **Привести примеры чтения объектов из потока.**



1. **Привести примеры записи объектов в поток**
2. **Сформулировать алгоритм удаления записей из файла.**
3. открыть файл в режиме чтения
4. открыть вспомогательный файл в режиме записи
5. пока не конец файла, копировать записи из файла в вспомогательный,

пропуская те, которые нужно удалить

1. удалить изначальный файл
2. переименовать вспомогательный файл в изначальный
3. **Сформулировать алгоритм добавления записей в файл.**
4. открыть файл в режиме чтения
5. открыть вспомогательный файл в режиме записи
6. пока не конец файла, копировать записи из файла в вспомогательный,

если встретится запись, после которой нужно добавить новую, остановиться, начать добавлять новые, затем продолжить копирование старых записей из изначального файла

1. удалить изначальный файл
2. переименовать вспомогательный файл в изначальный
3. **Сформулировать алгоритм изменения записей в файле.**
4. открыть файл в режиме чтения
5. открыть вспомогательный файл в режиме записи
6. пока не конец файла, копировать записи из файла в вспомогательный,

пропуская те, которые нужно удалить, если встретится запись, после которой нужно добавить новую, остановиться, начать добавлять новые, затем продолжить копирование старых записей из изначального файла

1. удалить изначальный файл

переименовать вспомогательный файл в изначальный

**GitHub:** *https://github.com/geroineee/PNRPU*