Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

ПНИПУ

**Лабораторная работа  
“Классы”. №7**

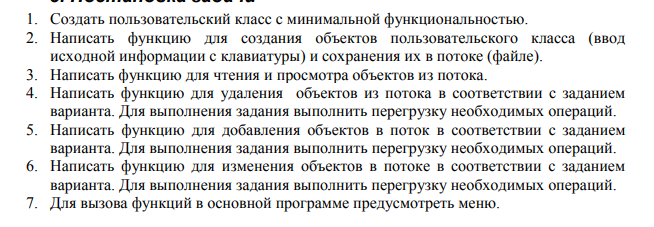
Выполнил:   
студент группы РИС-23-1б   
Фаезов Александр

Проверила:   
доцент кафедры ИТАС   
О.А. Полякова

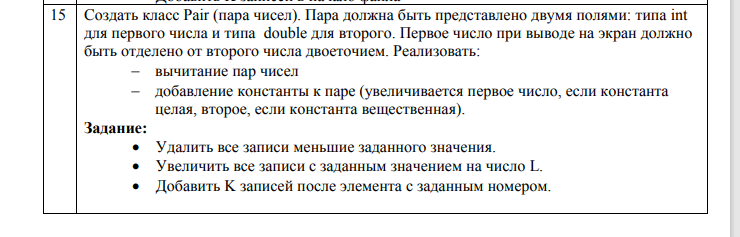
Пермь, 2024 г.

**«Классы и объекты. Файловые потоки»**

**Анализ задачи:**

****

***Вариант 15:***

****

**Код на языке C++:**

#include <iostream>

#include <fstream>

using namespace std;

class Pair {

private:

int x;

double y;

public:

Pair() {

this->x = 0;

this->y = 0;

}

Pair(int i, double j) {

this->x = i;

this->y = j;

cout << "Вызван конструктор с параметрами: " << this << endl;

cout << "X равно: " << this->x << '\t' << "Y равно: " << this->y << endl << endl;

}

Pair(const Pair& d) {

this->x = d.x;

this->y = d.y;

cout << "Вызван конструктор копирования: " << this << endl;

cout << "X равно: " << this->x << '\t' << "Y равно: " << this->y << endl << endl;

}

double diff() {

return this->x - this->y;

}

void setX(int b) {

this->x = b;

}

void setY(double a) {

this->y = a;

}

int returnX() {

return this->x;

}

double returnY() {

return this->y;

}

void dobavlenieConst(int j) {

cout << "К " << this->x << " добавляем const " << j << ": " << endl;

this->x = x + j;

}

void dobavlenieConst(double i) {

cout << "К " << this->y << " добавляем const " << i << ": " << endl;

this->y = y + i;

}

void operator=(const Pair& a) {

this->x = a.x;

this->y = a.y;

}

void operator -(const Pair& a) {

this->x = x - a.x;

this->y = y - a.y;

}

void operator +(int j) {

cout << "К " << this->x << " добавляем const " << j << ": " << endl;

this->x = x + j;

}

void operator +(double k) {

cout << "К " << this->y << " добавляем const " << k << ": " << endl;

this->y = y + k;

}

~Pair() {

cout << "Вызван деструктор: " << this << endl;

}

friend istream& operator >> (istream& is, Pair& p) {

cout << "Вызвана прегрузка оператора >> " << endl << endl;

cout << "Введите int x : ";

is >> p.x;

cout << "Введите double y: ";

is >> p.y;

cout << "Перегрузка закончилась" << endl << endl;

return is;

}

friend ostream& operator << (ostream& os, const Pair& p) {

cout << "Вызвана перегрузка оператор << " << endl << endl;

os << p.x << " : " << p.y << endl << endl;

cout << "Перегрузка закончилась" << endl << endl;

return os;

}

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

fout << p.x << " : " << p.y;

return fout;

}

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

char q;

fin >> p.x;

fin >> q;

fin >> p.y;

return fin;

}

};

Pair& randomPair(Pair& p) {

p.setX(rand() % 10);

p.setY((double)(rand() % 1000) / (double)(rand() % 100) + 1);

return p;

}

int make\_file(string f\_name) {

fstream stream(f\_name + ".txt", ios::out | ios::trunc);

if (!stream) {

cout << "Ошибка открытия файла";

return -1;

}

int n;

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

cin >> n;

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

Pair p;

randomPair(p);

stream << p << endl;

}

stream.close();

return n;

}

int print\_file(string f\_name) {

fstream stream(f\_name + ".txt", ios::in);

if (!stream) {

cout << "Ошибка открытия файла";

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, int k) {

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

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

if (!stream) {

cout << "Ошибка открытия файла";

return -1;

}

int i = 0;

Pair p;

while (stream >> p) {

if (p.returnX() >= k && p.returnY() >= k) {

temp << p << endl;

i++;

}

}

stream.close();

temp.close();

remove(f\_name);

rename("temp.txt", f\_name);

return i;

}

int change\_file(const char\* f\_name, int k, int L) {

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

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

if (!stream) {

cout << "Ошибка открытия файла";

return -1;

}

Pair p;

int i = 0;

while (stream >> p) {

if (p.returnX() == k || p.returnY() == k) {

p.setX(p.returnX() + L);

p.setY(p.returnY() + L);

i++;

}

temp << p << endl;

}

stream.close();

temp.close();

remove(f\_name);

rename("temp.txt", f\_name);

return i;

}

int add\_file(const char\* f\_name, int index, int K) {

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

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

if (!stream) {

cout << "Ошибка открытия файла";

return -1;

}

Pair p;

int i = 0;

while (stream >> p) {

temp << p << endl;

i++;

if (i == index) {

for (int j = 0; j < K; j++) {

cout << "Введите новые значения: " << endl;

cin >> p;

temp << p << endl;

}

}

}

stream.close();

temp.close();

remove(f\_name);

rename("temp.txt", f\_name);

return i;

}

int main() {

system("chcp 1251 > NULL");

srand(time(NULL));

string f\_name;

int с, size = 0;

do {

cout << "\n1. Создать файл";

cout << "\n2. Вывести содержимое файла";

cout << "\n3. Удалить записи меньше определенного значения";

cout << "\n4. Увеличить все записи с заданным значением на число К";

cout << "\n5. Добавить К записей после элемента с заданным номером";

cout << "\n0. Выход из программы\n";

cin >> с;

switch (с) {

case 0:

cout << "Иди нахуй"; break;

case 1:

cout << "Введите имя файла: ";

cin >> f\_name;

size = make\_file(f\_name);

if (size < 0) cout << "Файл не создан( ";

break;

case 2:

cout << "Введите имя файла: ";

cin >> f\_name;

size = print\_file(f\_name);

if (size == 0) cout << "Файл пуст ";

if (size < 0) cout << "Хуйня не работает ";

break;

case 3:

cout << "Введите имя файла: ";

cin >> f\_name;

int k;

cout << "Введите минимальное значение: ";

cin >> k;

f\_name += ".txt";

size = del\_file(f\_name.c\_str(), k);

if (size < 0) cout << "Хуйня не работает ";

break;

case 4:

cout << "Введите имя файла: ";

cin >> f\_name;

int value;

cout << "Введите изменяемое значение: ";

cin >> value;

int K;

cout << "Введите К: ";

cin >> K;

f\_name += ".txt";

size = change\_file(f\_name.c\_str(), value, K);

if (size < 0) cout << "Хуйня не работает ";

break;

case 5:

cout << "Введите имя файла: ";

cin >> f\_name;

int index;

cout << "Введите номер элемента: ";

cin >> index;

cout << "Введите К: ";

do

{

cin >> K;

} while (K < 1);

f\_name += ".txt";

size = add\_file(f\_name.c\_str(), index, K);

if (size < 0) cout << "Хуйня не работает ";

break;

default:

cout << "Ты еблан или как, где ты сука увидел эту цифру/число в выборе???";

break;

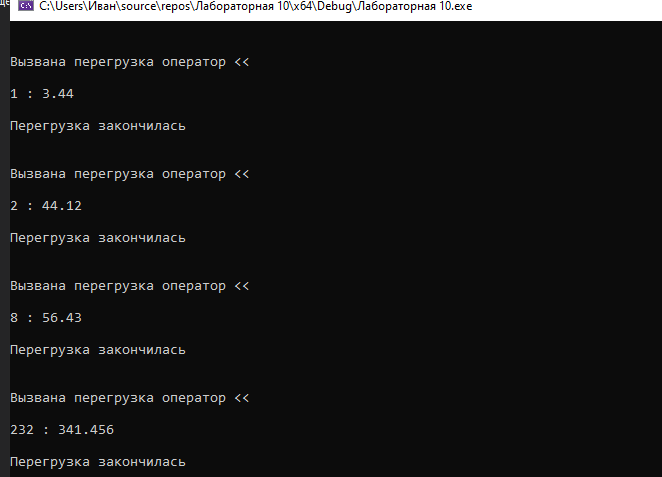
}

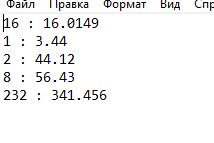
} while (с != 0);

return 0;

}

**Работа программы:**





**Ответ на вопросы:**

1. Поток в программировании - это абстракция для передачи, получения или обработки данных. Поток представляет собой последовательность данных, которая может быть считана из источника (ввод), записана в него (вывод), либо обработана.

2. В C++ существуют основные типы потоков:

- Потоки ввода (`istream`) для считывания данных.

- Потоки вывода (`ostream`) для записи данных.

- Потоки ввода-вывода (`iostream`), комбинация потоков ввода и вывода.

- Файловые потоки (`fstream`), работающие с файлами.

- Строковые потоки (`stringstream`), оперирующие со строками в памяти.

3. Для использования стандартных потоков в C++ (`cin`, `cout`, `cerr`, `clog`) необходимо подключить библиотеку `<iostream>`.

4. Для использования файловых потоков в C++ (`ifstream`, `ofstream`, `fstream`) необходимо подключить библиотеку `<fstream>`.

5. Для использования строковых потоков в C++ (`stringstream`) необходимо подключить библиотеку `<sstream>`.

6. При выводе в форматированный поток используется операция `<<`.

7. При вводе из форматированных потоков используется операция `>>`.

8. Методы, используемые при выводе в форматированный поток, включают `width`, `precision`, `setf`, и другие методы, позволяющие установить формат вывода данных.

9. Методы, используемые при вводе из форматированного потока, включают `width`, `setf`, `get`, и другие для управления форматом вводимых данных.

10. Режимы для открытия файловых потоков в C++:

- `ios::in` - открытие для чтения.

- `ios::out` - открытие для записи.

- `ios::app` - открытие для добавления записей в конец файла.

- `ios::ate` - открытие и установка позиции в конец файла.

- `ios::trunc` - открытие и обрезка файла.

11. Режим `ios::app` используется для добавления записей в файл, при этом новые данные будут дописываться в конец файла.

12. В конструкторе `ifstream file("f.txt")` используется режим `ios::in`, который открывает файл только для чтения.

13. В конструкторе `fstream file("f.txt")` нет явно указанных режимов, что означает открытие файла для чтения и записи без обрезки.

14. В конструкторе `ofstream file("f.txt")` используется режим `ios::out`, который открывает файл только для записи, а существующий файл будет перезаписан.

15. Поток открывается в режиме `ios::out | ios::app`.

16. Поток открывается в режиме `ios::out | ios::trunc`.

17. Поток открывается в режиме `ios::out | ios::in | ios::trunc`.

18. Файл открывается для чтения с использованием режима `ios::in`.

19. Файл открывается для записи с использованием режима `ios::out`.

20. Примеры открытия файловых потоков в различных режимах:

- `ifstream input("data.txt", ios::in);` // открытие файла для чтения.

- `ofstream output("result.txt", ios::out | ios::app);` // открытие файла для записи в конец.

21. Примеры чтения объектов из потока:

- `int num;`

- `cin >> num;`

22. Примеры записи объектов в поток:

- `double value = 5.678;`

- `cout << value;`

23. Алгоритм удаления записей из файла:

1. Открываем файл в режиме записи с обрезкой.

2. Считываем данные из исходного файла, пропуская записи, которые нужно удалить.

3. Записываем считанные данные обратно в файл.

24. Алгоритм добавления записей в файл:

1. Открываем файл в режиме добавления записей.

2. Создаем новую запись или считываем данные для добавления.

3. Записываем новые данные в конец файла.

25. Алгоритм изменения записей в файле:

1. Открываем файл в режиме чтения и записи.

2. Считываем данные и находим запись, которую нужно изменить.

3. Изменяем данные записи.

4. Записываем измененные данные обратно в файл.

**Вывод:** Задача была выполнена. Всё получилось.

