ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Лабораторная работа №13

Стандартные обобщенные алгоритмы библиотеки STL

Вариант №15

Выполнил студент группы РИС-23-3Б

Асташин Д.А.

Проверила доцент кафедры ИТАС

О. А. Полякова

2024 г.

**Постановка задачи**

**Задача 1**

1. Создать последовательный контейнер.

2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.

3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).

4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).

6. Найти в контейнере заданный элемент (использовать алгоритмы find(), find\_if(), count(), count\_if()).

7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()).

8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

**Задача 2**

1. Создать адаптер контейнера.

2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.

3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).

4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).

6. Найти в контейнере элемент с заданным ключевым полем (использовать алгоритмы find(), find\_if(), count(), count\_if()).

7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()).

8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

**Задача 3**

1. Создать ассоциативный контейнер.

2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.

3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).

4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).

6. Найти в контейнере элемент с заданным ключевым полем (использовать алгоритмы find(), find\_if(), count(), count\_if()).

7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()).

8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

**Задание из варианта №15**

**Задача 1**

1. Контейнер – список

2. Тип элементов Pair (см. лабораторную работу №3).

**Задача 2**

Адаптер контейнера – очередь с приоритетами.

**Задача 3**

Ассоциативный контейнер – словарь

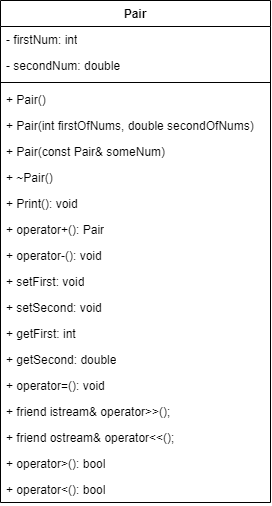
**Задание 3**: Найти среднее арифметическое и добавить его в конец контейнера

**Задание 4**: Найти элементы ключами из заданного диапазона и удалить их из контейнера

**Задание 5**: К каждому элементу добавить сумму минимального и максимального элементов контейнера

**Задача 1**

**UML диаграмма**

****

**Код на языке C++**

#include <iostream>

#include <list>

#include <functional>

#include <algorithm>

#include <ctime>

using namespace std;

class Pair {

int firstNum;

double secondNum;

public:

Pair() {

firstNum = 0;

secondNum = 0;

}

Pair(int firstOfNums, double secondOfNums) {

firstNum = firstOfNums;

secondNum = secondOfNums;

}

Pair(const Pair& someNum) {

firstNum = someNum.firstNum;

secondNum = someNum.secondNum;

}

~Pair() {}

void Print() {

cout << firstNum << " : " << secondNum;

}

Pair operator+(const Pair& other) const {

Pair result;

result.firstNum = firstNum + other.firstNum;

result.secondNum = secondNum + other.secondNum;

return result;

}

void operator-(const Pair& number) {

firstNum -= number.firstNum;

secondNum -= number.secondNum;

}

void setFirst(int num) { firstNum = num; }

void setSecond(double num) { secondNum = num; }

int getFirst() const { return firstNum; }

double getSecond() const { return secondNum; }

void operator=(const Pair& number) {

firstNum = number.firstNum;

secondNum = number.secondNum;

}

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

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

bool operator>(const Pair& p) const { return firstNum > p.firstNum; }

bool operator<(const Pair& p) const { return firstNum < p.firstNum; }

};

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

stream << p.firstNum << " : " << p.secondNum;

return stream;

}

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

stream >> p.firstNum >> p.secondNum;

return stream;

}

int main() {

system("chcp 1251 > Null");

srand(time(0));

int listSize;

do {

cout << "Введите размер списка: "; cin >> listSize;

} while (listSize < 1);

Pair summa;

list<Pair> workList(listSize);

generate(workList.begin(), workList.end(), []() { Pair p; p.setFirst(rand() % 1001); p.setSecond((rand() % 100001) \* 0.01); return p; });

cout << "\nСписок\n\n";

for (auto& p : workList) cout << p << endl;

for\_each(workList.begin(), workList.end(), [&summa](const Pair& p) { summa = summa + p; });

summa.setFirst(summa.getFirst() / workList.size());

summa.setSecond(((summa.getSecond() \* 10000 / workList.size()) \* 0.0001));

cout << "\nСреднее арифметическое: " << summa << endl << endl;

workList.push\_back(summa);

cout << "Список после добавления среднего арифметического\n\n";

for (auto& p : workList) cout << p << endl;

Pair oneEl, twoEl;

cout << "\nВведите две пары чисел, из диапазона которых будут удалены пары из списка\n";

cout << "\nПервая пара: "; cin >> oneEl;

cout << "\nВторая пара: "; cin >> twoEl;

if (oneEl > twoEl) swap(oneEl, twoEl);

list<Pair>::iterator s = remove\_if(workList.begin(), workList.end(), [oneEl, twoEl](Pair& p) { return (p > oneEl) && (p < twoEl); });

workList.erase(s, workList.end());

cout << "\nСписок после удаления из диапазона\n\n";

for (auto& p : workList) cout << p << endl;

auto stl\_pair = minmax\_element(workList.begin(), workList.end());

cout << "\nМинимальный элемент: " << \*stl\_pair.first << endl;

cout << "\nМаксимальный элемент: " << \*stl\_pair.second << endl;

Pair sum\_pair(stl\_pair.first->getFirst() + stl\_pair.second->getFirst(), stl\_pair.first->getSecond() + stl\_pair.second->getSecond());

cout << "\nСумма максимального и минимального элементов: " << sum\_pair << endl;

for\_each(workList.begin(), workList.end(), [&sum\_pair](Pair& p) { p = p + sum\_pair; });

cout << "\nСписок после добавления суммы максимального и минимального к каждой паре\n\n";

for (auto& p : workList) cout << p << endl;

cout << "\nСортировка по возрастанию\n\n";

workList.sort([](const Pair& a, const Pair& b) { return a < b; });

for (auto& p : workList) cout << p << endl;

cout << "\nСортировка по убыванию\n";

workList.sort([](const Pair& a, const Pair& b) { return a > b; });

for (auto& p : workList) cout << p << endl;

int ptr;

cout << "\nВведите значение first, по которому необходимо найти пару: "; cin >> ptr;

auto i = find\_if(workList.begin(), workList.end(), [&ptr](const Pair& p) { return ptr == p.getFirst(); });

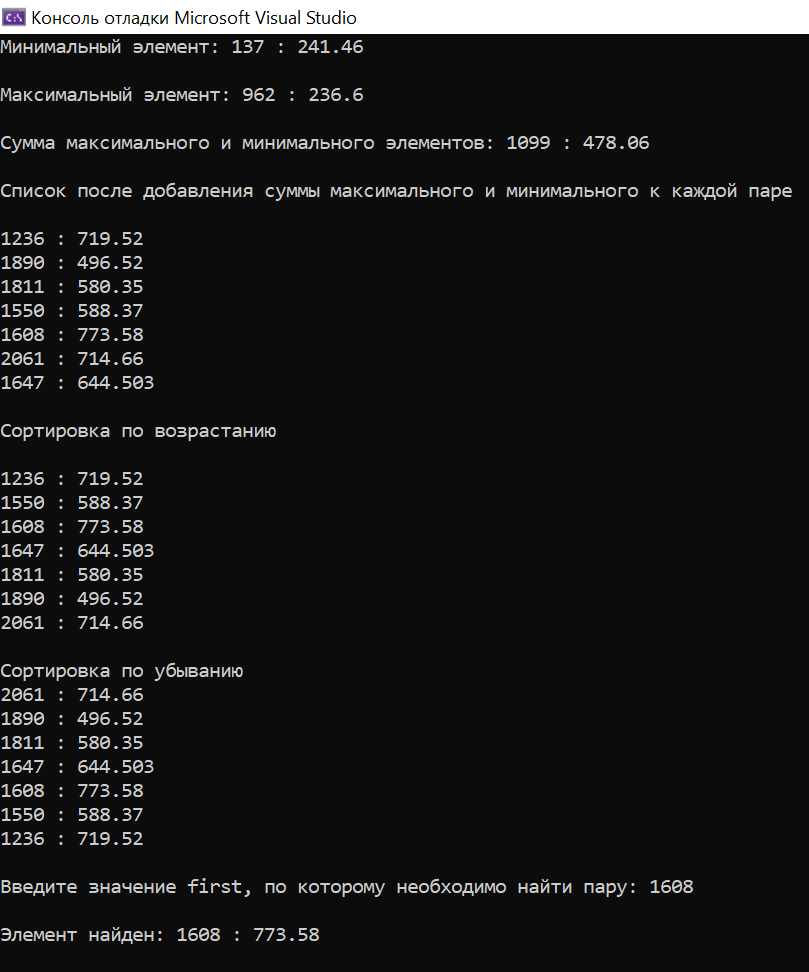
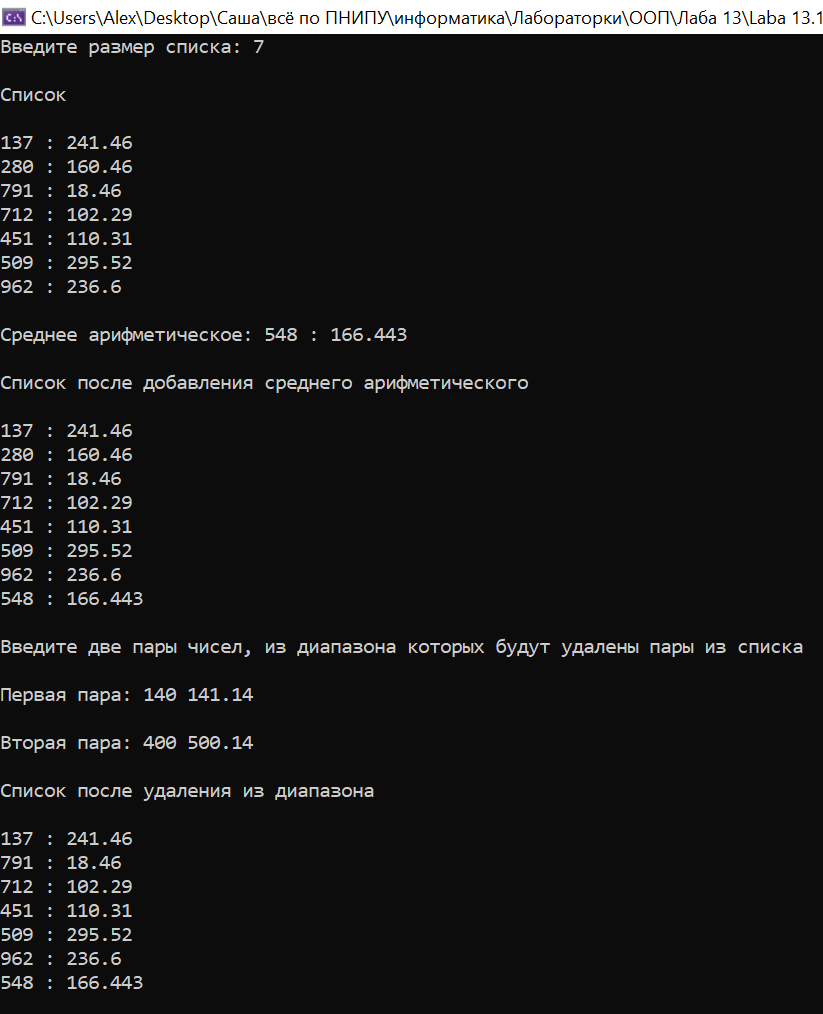
if (i != end(workList)) { cout << "\nЭлемент найден: " << \*i << endl; }

else { cout << "\nЭлемент не найден.\n"; };

return 0;

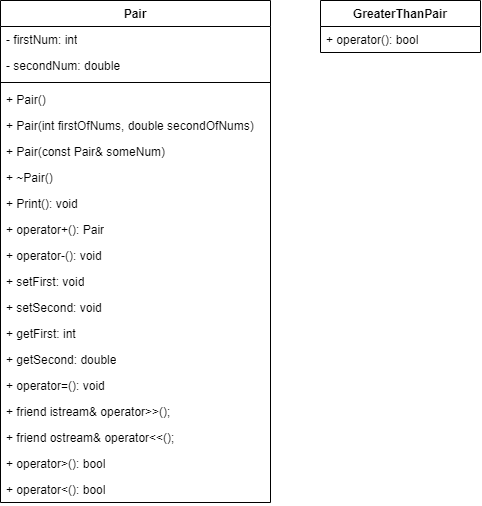
}

**Результаты работы программы**

****

**Задача 2**

**UML диаграмма**

****

**Код на языке C++**

#include <iostream>

#include <queue>

#include <functional>

#include <algorithm>

#include <random>

using namespace std;

class Pair {

int firstNum;

double secondNum;

public:

Pair() {

firstNum = 0;

secondNum = 0;

}

Pair(int firstOfNums, double secondOfNums) {

firstNum = firstOfNums;

secondNum = secondOfNums;

}

Pair(const Pair& someNum) {

firstNum = someNum.firstNum;

secondNum = someNum.secondNum;

}

~Pair() {}

void Print() { cout << firstNum << " : " << secondNum; }

Pair operator+(const Pair& other) const {

Pair result;

result.firstNum = firstNum + other.firstNum;

result.secondNum = secondNum + other.secondNum;

return result;

}

void operator-(const Pair& number) {

firstNum -= number.firstNum;

secondNum -= number.secondNum;

}

void setFirst(int num) { firstNum = num; }

void setSecond(double num) { secondNum = num; }

int getFirst() const { return firstNum; }

double getSecond() const { return secondNum; }

void operator=(const Pair& number) {

firstNum = number.firstNum;

secondNum = number.secondNum;

}

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

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

bool operator>(const Pair& p) const { return firstNum > p.firstNum; }

bool operator<(const Pair& p) const { return firstNum < p.firstNum; }

};

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

stream << p.firstNum << " : " << p.secondNum;

return stream;

}

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

stream >> p.firstNum >> p.secondNum;

return stream;

}

void randomize(Pair& p) {

int x = rand() % 1001;

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

p = Pair(x, y);

}

class GreaterThanPair {

public:

bool operator()(Pair& p1, Pair& p2) { return p1 < p2; }

};

void generateQ(priority\_queue<Pair, vector<Pair>, GreaterThanPair>& myQ, int qSize) {

Pair qwe(0, 0);

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

randomize(qwe);

myQ.push(qwe);

}

}

ostream& operator<<(ostream& stream, priority\_queue<Pair, vector<Pair>, GreaterThanPair> myQ) {

while (!myQ.empty()) {

Pair user = myQ.top();

cout << user << endl;

myQ.pop();

}

return stream;

}

Pair allSum(priority\_queue<Pair, vector<Pair>, GreaterThanPair> myQ) {

Pair p(0, 0);

while (!myQ.empty()) {

Pair upp = myQ.top();

p = p + upp;

myQ.pop();

}

return p;

}

void removeFromRange(priority\_queue<Pair, vector<Pair>, GreaterThanPair>& myQ, Pair& range1, Pair& range2) {

priority\_queue<Pair, vector<Pair>, GreaterThanPair> queueOne;

while (!myQ.empty()) {

Pair upp = myQ.top();

if (!(upp > range1 && upp < range2)) {

queueOne.push(upp);

}

myQ.pop();

}

myQ = queueOne;

}

Pair maxPair(priority\_queue<Pair, vector<Pair>, GreaterThanPair> myQ) {

Pair maxP = myQ.top();

return maxP;

}

Pair minPair(priority\_queue<Pair, vector<Pair>, GreaterThanPair> myQ) {

Pair minP;

while (!myQ.empty()) {

minP = myQ.top();

myQ.pop();

}

return minP;

}

priority\_queue<Pair, vector<Pair>, GreaterThanPair> addToAll(priority\_queue<Pair, vector<Pair>, GreaterThanPair>& myQ, Pair& para) {

priority\_queue<Pair, vector<Pair>, GreaterThanPair> queueOne;

Pair upp = myQ.top();

while (!myQ.empty()) {

upp = myQ.top();

queueOne.push(upp + para);

myQ.pop();

}

return queueOne;

}

bool findPair(priority\_queue<Pair, vector<Pair>, GreaterThanPair> myQ, int& x, Pair& p) {

bool flag = false;

while (!myQ.empty() && !flag) {

p = myQ.top();

if (x == p.getFirst()) { flag = true; }

myQ.pop();

}

return flag;

}

int main() {

system("chcp 1251 > Null");

srand(time(0));

int queueSize;

do {

cout << "Введите размер очереди: "; cin >> queueSize;

} while (queueSize < 1);

priority\_queue<Pair, vector<Pair>, GreaterThanPair> qst;

generateQ(qst, queueSize);

cout << "\nОчередь\n\n" << qst << endl;;

Pair summa = allSum(qst);

summa.setFirst(summa.getFirst() / qst.size());

summa.setSecond(((summa.getSecond() \* 1000) / qst.size()) \* 0.001);

cout << endl << "Средне арифметическое: " << summa << endl << endl;

qst.push(summa);

cout << "После добавления средне арифметического\n\n" << qst << endl;

Pair oneEl, twoEl;

cout << "\nВведите две пары чисел, из диапазона которых будут удалены пары из очереди\n";

cout << "\nПервая пара: "; cin >> oneEl;

cout << "\nВторая пара: "; cin >> twoEl;

if (oneEl > twoEl) swap(oneEl, twoEl);

removeFromRange(qst, oneEl, twoEl);

cout << "\nОчередь после удаления из диапазона\n\n" << qst << endl;

Pair maxEl = maxPair(qst);

Pair minEl = minPair(qst);

cout << "\nМинимальная пара: " << minEl << endl;

cout << "\nМаксимальный пара: " << maxEl << endl;

summa = minEl + maxEl;

cout << "\nСумма max и min элементов: " << summa << endl;

qst = addToAll(qst, summa);

cout << "\nПосле добавления суммы max и min к каждой паре\n\n" << qst << endl;

int ptr;

cout << "Введите значение первого числа из пары, по которому нужно найти пару: "; cin >> ptr;

bool flag = findPair(qst, ptr, summa);

if (flag) {

cout << "\nЭлемент найден: " << summa << endl;

}

else {

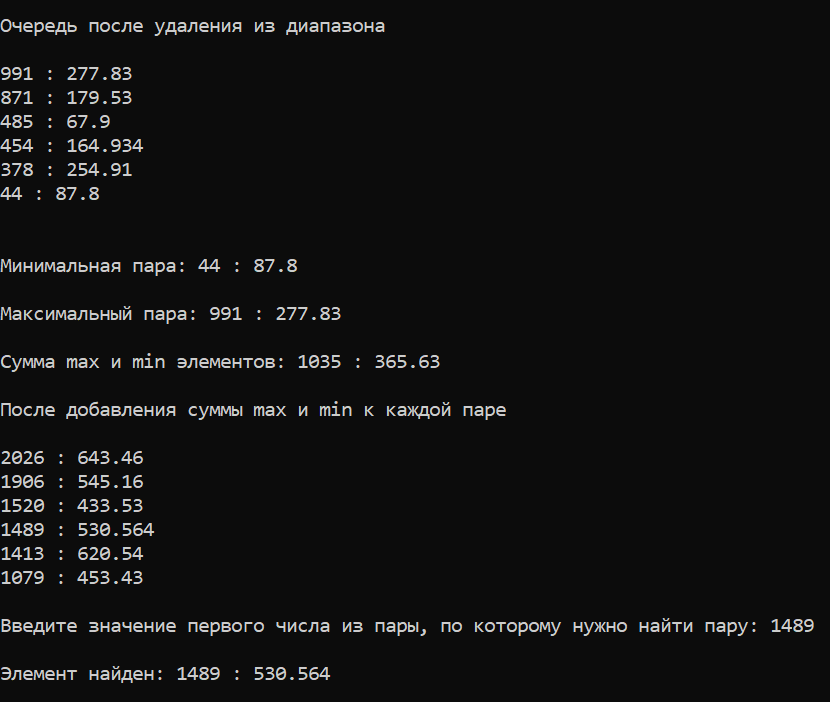
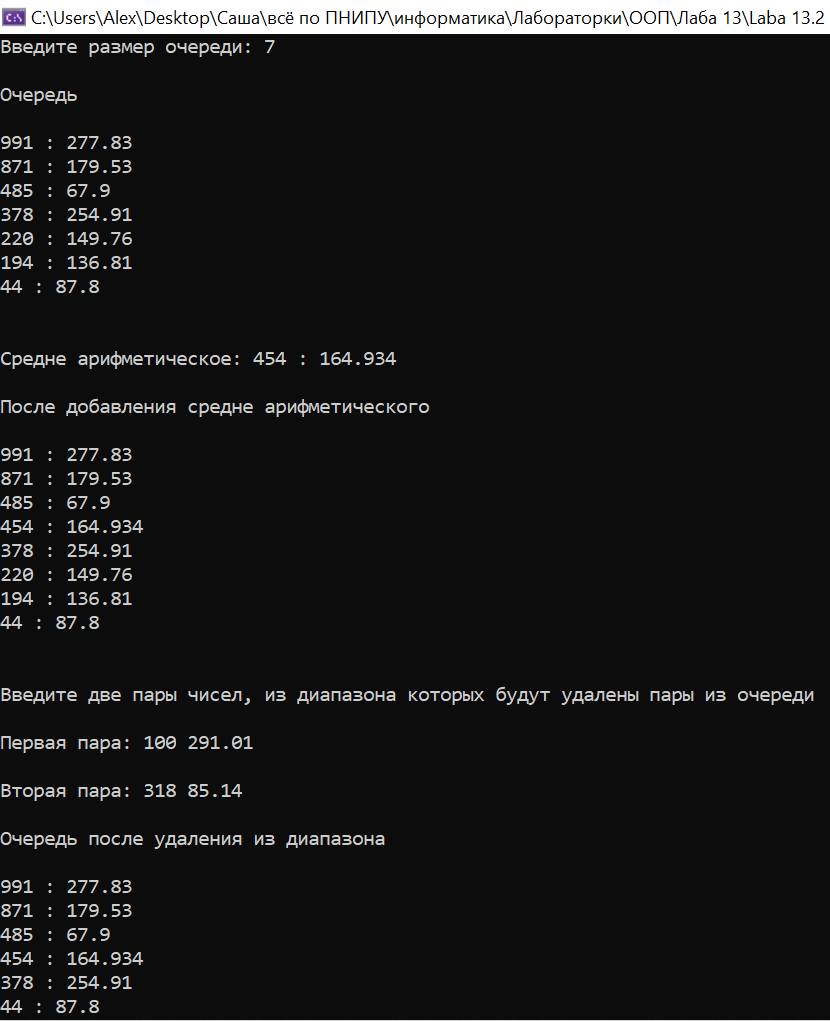
cout << "\nЭлемент не найден.\n";

}

return 0;

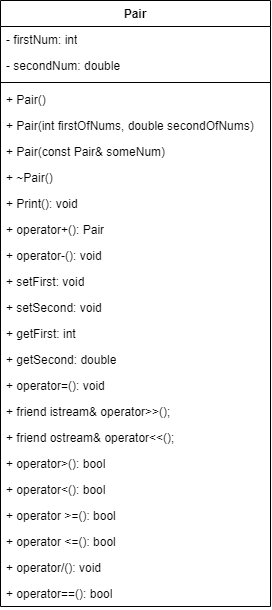
}

**Результаты работы программы**

****

**Задача 3**

**UML диаграмма**

****

**Код на языке C++**

#include <iostream>

#include <map>

#include <algorithm>

#include <random>

#include <ctime>

using namespace std;

class Pair {

int firstNum;

double secondNum;

public:

Pair() {

firstNum = 0;

secondNum = 0;

}

Pair(int firstOfNums, double secondOfNums) {

firstNum = firstOfNums;

secondNum = secondOfNums;

}

Pair(const Pair& someNum) {

firstNum = someNum.firstNum;

secondNum = someNum.secondNum;

}

~Pair() {}

void Print() { cout << firstNum << " : " << secondNum; }

Pair operator+(const Pair& other) const {

Pair result;

result.firstNum = firstNum + other.firstNum;

result.secondNum = secondNum + other.secondNum;

return result;

}

void operator-(const Pair& number) {

firstNum -= number.firstNum;

secondNum -= number.secondNum;

}

void setFirst(int num) { firstNum = num; }

void setSecond(double num) { secondNum = num; }

int getFirst() const { return firstNum; }

double getSecond() const { return secondNum; }

void operator=(const Pair& number) {

firstNum = number.firstNum;

secondNum = number.secondNum;

}

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

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

bool operator>(const Pair& p) const { return firstNum > p.firstNum; }

bool operator<(const Pair& p) const { return firstNum < p.firstNum; }

bool operator>=(const Pair& other) const {

if (firstNum > other.firstNum)

return true;

else if (firstNum < other.firstNum)

return false;

return secondNum >= other.secondNum;

}

bool operator<=(const Pair& other) const {

if (firstNum < other.firstNum) return true;

else if (firstNum > other.firstNum) return false;

return secondNum <= other.secondNum;

}

void operator/(const int& x) {

firstNum /= x;

secondNum /= x;

}

bool operator==(const Pair& p) const { return (firstNum == p.firstNum) && (secondNum == p.secondNum); }

};

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

stream << "(" << p.firstNum << " : " << p.secondNum << ")";

return stream;

}

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

stream >> p.firstNum >> p.secondNum;

return stream;

}

void randomize(Pair& p) {

int x = rand() % 1001;

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

p = Pair(x, y);

}

void generateM(map<Pair, Pair>& myM, int mSize) {

Pair qwe1(0, 0), qwe2(0, 0);

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

randomize(qwe1);

randomize(qwe2);

myM[qwe1] = qwe2;

}

}

ostream& operator<<(ostream& stream, const pair<Pair, Pair> p) {

stream << "Ключ: " << p.first << " \tЗначение: " << p.second << endl;

return stream;

}

ostream& operator<<(ostream& stream, const map<Pair, Pair>& myM) {

for (auto Iter = myM.begin(); Iter != myM.end(); ++Iter) {

stream << \*Iter;

}

return stream;

}

bool isInRange(const Pair& key, const Pair& lower, const Pair& upper) { return key >= lower && key <= upper; }

void removeFromRange(map<Pair, Pair>& myM, const Pair& range1, const Pair& range2) {

auto Iter = myM.begin();

while (Iter != myM.end()) {

const Pair& ptr = Iter->first;

if (!isInRange(ptr, range1, range2)) {

++Iter;

}

else {

Iter = myM.erase(Iter);

}

}

}

Pair minPair(map<Pair, Pair>& myM) {

auto iter = myM.begin();

Pair ptr;

ptr = iter->first;

return ptr;

}

Pair maxPair(map<Pair, Pair>& myM) {

auto iter = myM.begin();

Pair ptr;

while (iter != myM.end()) {

ptr = iter->first;

iter++;

}

return ptr;

}

void allSum(map<Pair, Pair>& myM, Pair& key, Pair& value) {

for (const auto& pair : myM) {

Pair user = pair.first;

key = key + user;

user = pair.second;

value = value + user;

}

}

map<Pair, Pair> addToAll(map<Pair, Pair>& myM, Pair& key, Pair& value) {

map<Pair, Pair> ptr;

for (const auto& pair : myM) {

Pair pair1 = pair.first;

Pair pair2 = pair.second;

ptr[pair1 + key] = (pair2 + value);

}

return ptr;

}

bool findPair(map<Pair, Pair>& myM, Pair& key, Pair& value) {

auto iter = myM.find(key);

if (iter != myM.end()) {

value = Pair(iter->second);

return true;

}

else {

return false;

}

}

int main() {

system("chcp 1251 > Null");

srand(time(0));

int mapSize;

do {

cout << "Введите размер словаря: "; cin >> mapSize;

} while (mapSize < 1);

Pair sum1(0, 0), sum2(0, 0), sumOfKey(0, 0), sumOfValue(0, 0);

map<Pair, Pair> workMap;

generateM(workMap, mapSize);

cout << "\nСловарь\n\n" << workMap << endl;

allSum(workMap, sum1, sum2);

sum1 / workMap.size();

sum2 / workMap.size();

cout << "\nСредне арифметическое ключей: " << sum1 << endl << endl << "Средне арифметическое значений: " << sum2 << endl;

workMap[sum1] = sum2;

cout << "\nПосле добавления средне арифметического\n\n" << workMap << endl;

Pair oneEl, twoEl;

cout << "\nВведите две пары чисел, из диапазона которых будут удалены пары словаря\n";

cout << "\nПервая пара: "; cin >> oneEl;

cout << "\nВторая пара: "; cin >> twoEl;

if (oneEl > twoEl) { swap(oneEl, twoEl); }

removeFromRange(workMap, oneEl, twoEl);

cout << "\nПосле удаления из диапазона\n\n" << workMap << endl;

Pair maxEl = maxPair(workMap);

Pair minEl = minPair(workMap);

findPair(workMap, minEl, sum1);

findPair(workMap, maxEl, sum2);

cout << "\nМинимальный элемент по ключу: " << minEl << endl;

cout << "Ключ: " << minEl << "\tЗначение: " << sum1 << endl;

cout << "\nМаксимальный элемент по ключу: " << maxEl << endl;

cout << "Ключ: " << maxEl << "\tЗначение: " << sum2 << endl;

sumOfKey = minEl + maxEl;

sumOfValue = sum1 + sum2;

cout << "\nСумма max и min элементов\n\n" << "Ключ: " << sumOfKey << "\tЗначение: " << sumOfValue << endl;

workMap = addToAll(workMap, sumOfKey, sumOfValue);

cout << "\nПосле добавления суммы max и min к каждой паре\n\n" << workMap << endl;

cout << "\nВведите ключ, по которому надо найти пару: "; cin >> oneEl;

bool findTo = findPair(workMap, oneEl, twoEl);

if (findTo) {

cout << "\nЭлемент найден\n\n" << "Ключ: " << oneEl << "\tЗначение: " << twoEl << endl;

}

else {

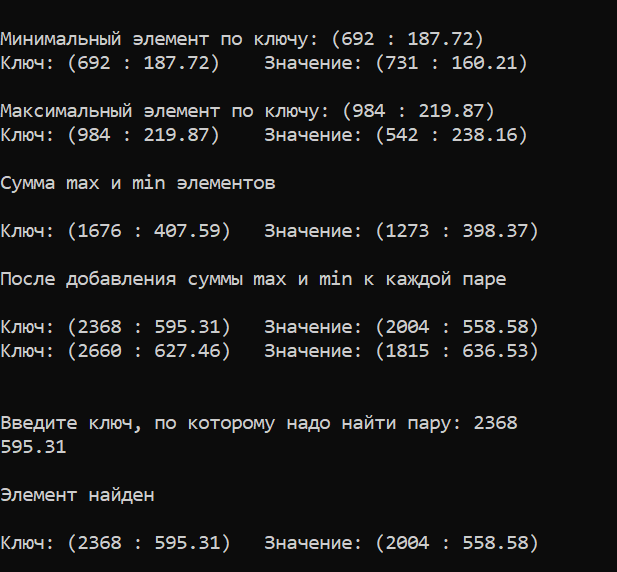
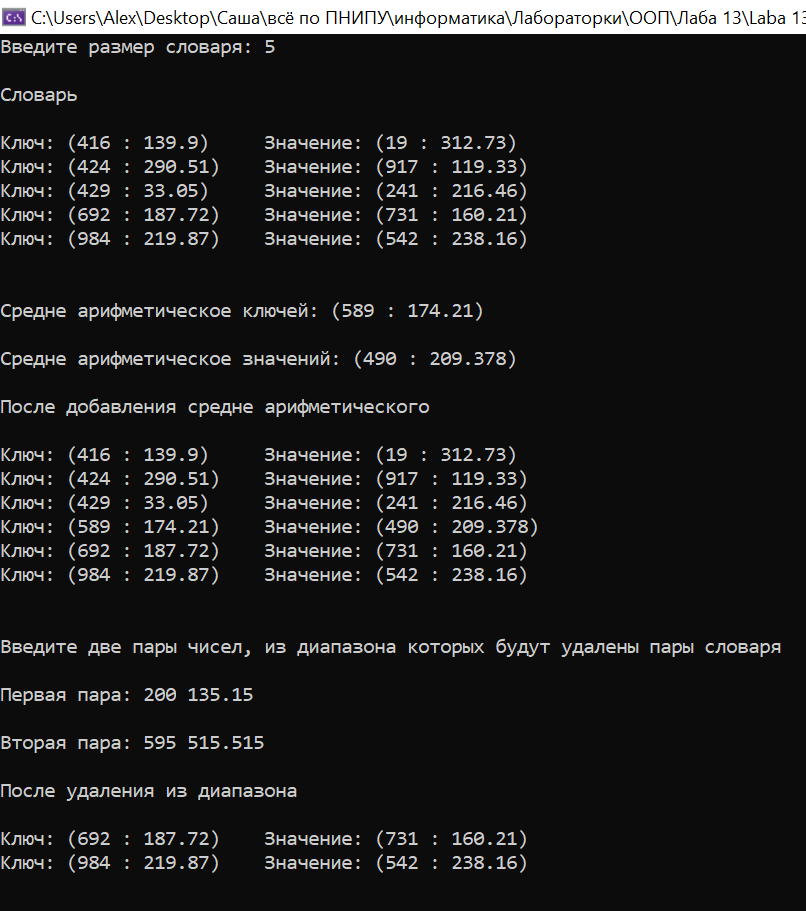
cout << "\nЭлемент не найден.";

}

return 0;

}

**Результаты работы программы**

****

**Выводы**

Все три программы работают успешно, без ошибок

GitHub: https://github.com/DimetriusAsti/OOP13