Пермский национальный исследовательский политехнический университет.

Лабораторная работа № 13 по ООП.

«Стандартные обобщенные библиотеки STL».

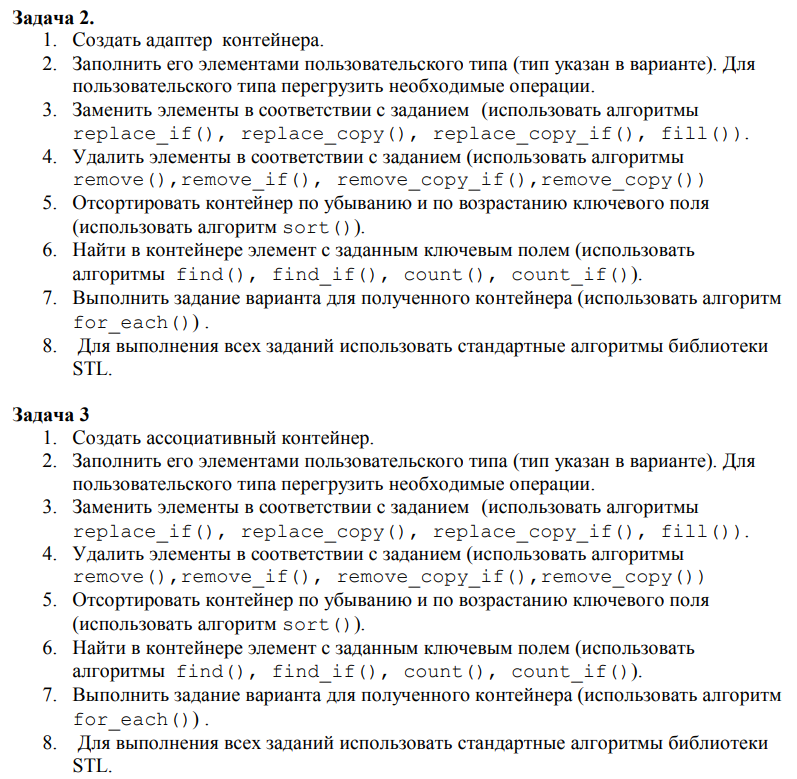
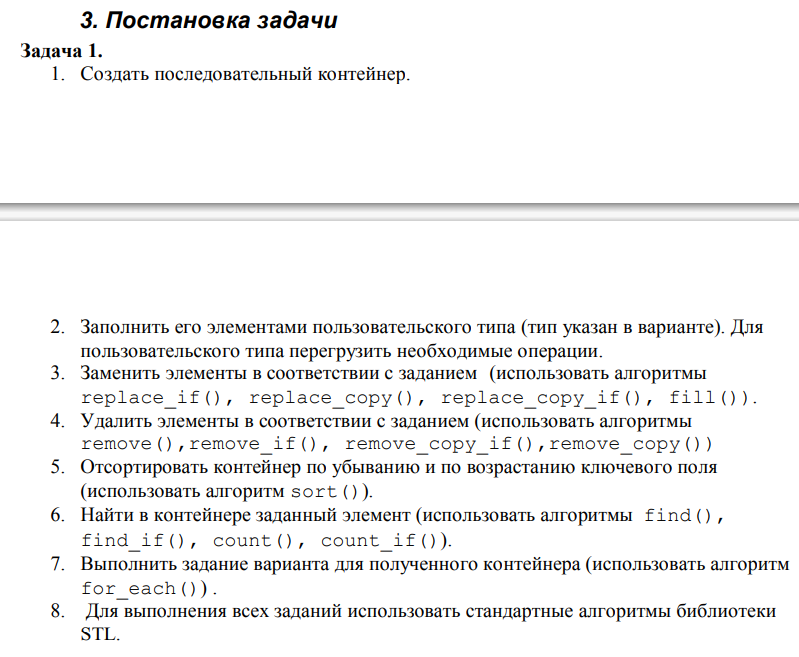
Выполнил: студент группы РИС-23-2б

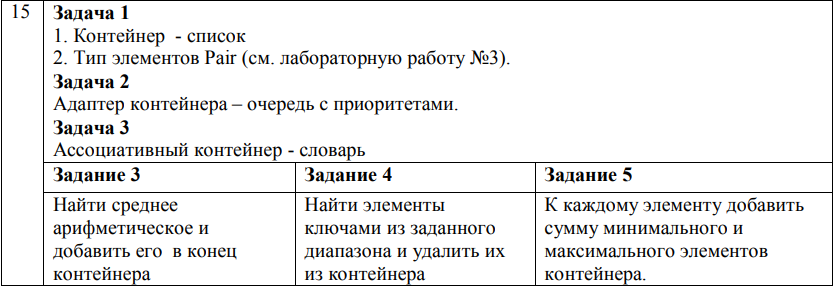
Вековшинин Иван Николаевич

Проверила: доцент кафедры ИТАС

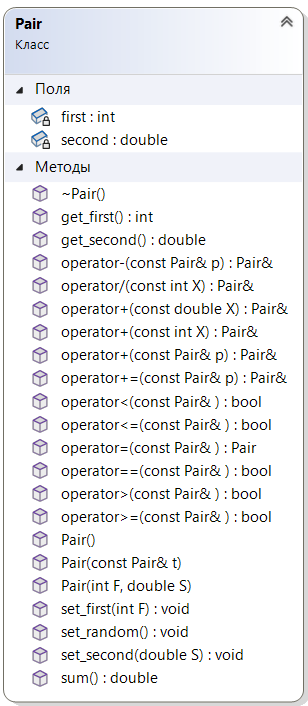
О.А. Полякова.

2024 г.





UML Diagram класса Pair:



**Код:**

**Задача 1.**

Pair.h

#pragma once

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

class Pair {

private:

int first;

double second;

public:

Pair() { first = 0; second = 0; };

Pair(int F, double S) { first = F; second = S; };

Pair(const Pair& t) { first = t.first;second = t.second; };

~Pair() {};

int get\_first() { return first; }

double get\_second() { return second; }

void set\_first(int F) { first = F;}

void set\_second(double S) { second = S;}

void set\_random();

double sum() { double s = second + first; return s; }

Pair operator =(const Pair&);

bool operator <(const Pair&) const;

bool operator >(const Pair&) const;

bool operator <=(const Pair&) const;

bool operator >=(const Pair&) const;

bool operator ==(const Pair&) const;

Pair& operator -(const Pair& p);

Pair& operator +(const Pair& p);

Pair& operator +=(const Pair& p);

Pair& operator +(const int X);

Pair& operator +(const double X);

Pair& operator /(const int X);

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

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

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

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

};

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

first += X;

return \*this;

}

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

second += X;

return \*this;

}

Pair& Pair::operator /(const int X) {

first = first / X;

second = second / X;

return \*this;

}

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

this->first -= p.first;

this->second -= p.second;

return \*this;

}

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

this->first += p.first;

this->second += p.second;

return \*this;

}

Pair& Pair::operator +=(const Pair& p) {

this->first += p.first;

this->second += p.second;

return \*this;

}

istream& operator >>(istream& in, Pair& t) {

cout << "Целое первое = "; in >> t.first;

cout << "Вещественное второе = "; in >> t.second;

return in;

}

ostream& operator <<(ostream& out, const Pair& t) {

return (out << t.first << " : " << t.second << endl);

}

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

if (&p == this) return \*this;

first = p.first;

second = p.second;

return\*this;

}

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

fin >> p.first;

fin >> p.second;

return fin;

}

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

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

return fout;

}

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

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis < sother) return true;

return false;

}

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

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis > sother) return true;

return false;

}

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

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis <= sother) return true;

return false;

}

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

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis >= sother) return true;

return false;

}

bool Pair::operator ==(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis == sother) return true;

return false;

}

void Pair::set\_random() {

first = rand() % 10 + 1;

second = round(((double)(rand()) / RAND\_MAX \* 10 + 1) \* 100) / 100;

}

Main.cpp

#include <iostream>

#include <list>

#include <algorithm>

#include <functional>

#include "Pair.h"

using namespace std;

ostream& operator<<(ostream& out, list<Pair>& l) {

for\_each(l.begin(), l.end(), [](Pair& p) {cout << p; });

return out;

}

bool EQUAL(Pair a, Pair t) {

if (a == t) return true;

return false;

}

void menu(const int c, list<Pair>& l) {

switch (c) {

//sort

case 1: {

cout << l << endl;

cout << "Введите < для сортировки по возрастанию\n Введите > для сортировки по убыванию\n";

char op;

cin >> op;

if (op == '<') l.sort(mem\_fun\_ref(&Pair::operator<));

else if (op == '>') l.sort(mem\_fun\_ref(&Pair::operator>));

else cout << "неправильный характер\n";

cout << l << endl;

break;

}

//find element

case 2: {

cout << l << "\n\n";

cout << "Введите значение:\n";

Pair value;

cin >> value;

auto it = find\_if(l.begin(), l.end(), bind2nd(ptr\_fun(EQUAL), value));

if (it == l.end()) {

cout << "такого элемента нет!" << endl;

break;

}

for (list<Pair>::iterator i = l.begin(); i != l.end(); ++i) {

if (i == it) {

cout << "элемент: " << \*i << endl;

}

}

break;

}

//arithmetical mean

case 3: {

cout << l << endl << endl;

Pair sum;

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

sum.set\_first(sum.get\_first() / l.size());

sum.set\_second(sum.get\_second() / l.size());

l.push\_back(sum);

cout << l << endl;

break;

}

//delete from range

case 4: {

cout << l << endl;

Pair one, two;

cout << "Краткое описание каждого элемента: " << endl;

for\_each(l.begin(), l.end(), [](Pair& p) {cout << p.sum() << ' '; });

cout << "\nВведите первую границу:\n"; cin >> one;

cout << "\nВведите вторую границу:\n"; cin >> two;

if (one > two) swap(one, two);

auto s = remove\_if(l.begin(), l.end(), [one, two](Pair& p) {return p >= one && p <= two; });

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

cout << "Элементы коллекции: " << endl;

for\_each(l.begin(), l.end(), [](Pair& p) {cout << p; });

break;

}

//minmax

case 5: {

cout << l << endl << endl;

auto mini = min\_element(l.begin(), l.end());

auto maxi = max\_element(l.begin(), l.end());

cout << "min: " << \*mini;

cout << "max: " << \*maxi;

Pair sum;

sum = \*mini + \*maxi;

cout << "сумма = " << sum << endl;

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

cout << endl << l << endl;

break;

}

//print

case 6: {

cout << l << endl;

break;

}

//exit

default: {

cout << "\nВыход из меню" << endl;

exit(777);

}

}

}

int main() {

system("color F0");

srand(time(NULL));

setlocale(LC\_ALL, "Ru");

int n;

cout << "Длина контейнера: ";

cin >> n;

list<Pair> l(n);

generate(l.begin(), l.end(), []() {Pair p; p.set\_random(); return p; });

cout << l << endl;

while (true) {

system("pause");

system("cls");

cout << "1. Отсортировать контейнер\n";

cout << "2. Найти пару\n";

cout << "3. Поставьте в конце среднее арифметическое\n";

cout << "4. Удаление элементов из диапазона\n";

cout << "5. Добавьте сводку минимальных и максимальных значений для каждой пары\n";

cout << "6. Вывод контейнера\n";

int c;

cin >> c;

system("cls");

menu(c, l);

}

return 0;

}

Задача 2.

Pair.h

#pragma once

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

class Pair {

private:

int first;

double second;

public:

Pair() { first = 0; second = 0; };

Pair(int F, double S) { first = F; second = S; };

Pair(const Pair& t) { first = t.first;second = t.second; };

~Pair() {};

int get\_first() { return first; }

double get\_second() { return second; }

void set\_first(int F) { first = F; }

void set\_second(double S) { second = S; }

void set\_random();

double sum() { double s = second + first; return s; }

Pair operator =(const Pair&);

bool operator <(const Pair&) const;

bool operator >(const Pair&) const;

bool operator <=(const Pair&) const;

bool operator >=(const Pair&) const;

bool operator ==(const Pair&) const;

Pair& operator -(const Pair& p);

Pair& operator +(const Pair& p);

Pair& operator +=(const Pair& p);

Pair& operator +(const int X);

Pair& operator +(const double X);

Pair& operator /(const int X);

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

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

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

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

};

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

first += X;

return \*this;

}

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

second += X;

return \*this;

}

Pair& Pair::operator /(const int X) {

first = first / X;

second = second / X;

return \*this;

}

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

this->first -= p.first;

this->second -= p.second;

return \*this;

}

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

this->first += p.first;

this->second += p.second;

return \*this;

}

Pair& Pair::operator +=(const Pair& p) {

this->first += p.first;

this->second += p.second;

return \*this;

}

istream& operator >>(istream& in, Pair& t) {

cout << "целое первое = "; in >> t.first;

cout << "вещественное второе = "; in >> t.second;

return in;

}

ostream& operator <<(ostream& out, const Pair& t) {

return (out << t.first << " : " << t.second << endl);

}

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

if (&p == this) return \*this;

first = p.first;

second = p.second;

return\*this;

}

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

fin >> p.first;

fin >> p.second;

return fin;

}

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

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

return fout;

}

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

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis < sother) return true;

return false;

}

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

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis > sother) return true;

return false;

}

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

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis <= sother) return true;

return false;

}

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

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis >= sother) return true;

return false;

}

bool Pair::operator ==(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis == sother) return true;

return false;

}

void Pair::set\_random() {

first = rand() % 10 + 1;

second = round(((double)(rand()) / RAND\_MAX \* 10 + 1) \* 100) / 100;

}

Main.cpp

#include <iostream>

#include <queue>

#include <vector>

#include <algorithm>

#include <functional>

#include "Pair.h"

using namespace std;

ostream& operator<<(ostream& out, priority\_queue<Pair>& q) {

vector<Pair> COPY;

Pair elem;

int num = q.size();

for (size\_t i = 0; i < num; ++i) {

elem = q.top();

cout << elem;

COPY.push\_back(elem);

q.pop();

}

int k = 0;

for (int i = 0; i < COPY.size(); ++i) {

q.push(COPY[k++]);

}

COPY.clear();

return out;

}

bool EQUAL(Pair a, Pair t) {

if (a == t) return true;

return false;

}

void menu(const int c, priority\_queue<Pair>& q) {

switch (c) {

//sort

case 1: {

char op;

cout << "Введите < для сортировки по возрастанию\nВведите > для сортировки по убыванию\n";

cin >> op;

if (op == '>') cout << q << endl;

else if (op == '<') {

vector<Pair> COPY;

Pair elem;

int num = q.size();

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

elem = q.top();

COPY.push\_back(elem);

q.pop();

}

sort(COPY.begin(), COPY.end(), mem\_fun\_ref(&Pair::operator<));

for\_each(COPY.begin(), COPY.end(), [](Pair& p) {cout << p; });

while (!COPY.empty()) {

q.push(COPY[COPY.size() - 1]);

COPY.pop\_back();

}

}

else cout << "неправильный характер!\n";

break;

}

//find element

case 2: {

cout << q << "\n\n";

cout << "Введите значение:\n";

Pair value;

cin >> value;

auto f = [value](priority\_queue<Pair>& q) {

vector<Pair> COPY;

Pair elem;

bool flag = false;

int num = q.size();

for (int i = 0; i < num && !flag; ++i) {

elem = q.top();

if (EQUAL(elem, value)) {

flag = true;

cout << "элемент: " << elem << endl;

}

COPY.emplace\_back(elem);

q.pop();

}

if (!flag) cout << "такого элемента нет!\n";

while (!COPY.empty()) {

q.push(COPY[COPY.size() - 1]);

COPY.pop\_back();

}

};

f(q);

break;

}

//arithmetical mean

case 3: {

cout << q << endl;

vector<Pair> COPY;

Pair elem;

while (!q.empty()) {

elem = q.top();

COPY.push\_back(elem);

q.pop();

}

Pair sum;

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

sum.set\_first(round(sum.get\_first() / COPY.size() \* 100) / 100);

sum.set\_second(round(sum.get\_second() / COPY.size() \* 100) / 100);

cout << "среднее арифметическое равно " << sum << "\n\n";

q.push(sum);

while (!COPY.empty()) {

q.push(COPY[COPY.size() - 1]);

COPY.pop\_back();

}

cout << q << endl;

break;

}

//delete from range

case 4: {

cout << q << endl;

Pair elem;

vector<Pair> COPY;

while (!q.empty()) {

elem = q.top();

COPY.push\_back(elem);

q.pop();

}

Pair one, two;

cout << "Краткое изложение каждого элемента: " << endl;

for\_each(COPY.begin(), COPY.end(), [](Pair& p) {cout << p.sum() << " "; });

cout << "\nВведите первую границу:\n"; cin >> one;

cout << "\nВведите вторую границу:\n"; cin >> two;

if (one > two) swap(one, two);

auto s = remove\_if(COPY.begin(), COPY.end(), [one, two](Pair& p) {return p >= one && p <= two; });

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

while (!COPY.empty()) {

q.push(COPY[COPY.size() - 1]);

COPY.pop\_back();

}

cout << endl << q << endl;

break;

}

//minmax

case 5: {

cout << q << endl << endl;

Pair elem;

vector<Pair> COPY;

while (!q.empty()) {

elem = q.top();

COPY.push\_back(elem);

q.pop();

}

auto mini = COPY[COPY.size() - 1];

auto maxi = COPY[0];

cout << "min: " << mini;

cout << "max: " << maxi;

Pair sum;

sum = mini + maxi;

cout << "сумма = " << sum << endl;

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

while (!COPY.empty()) {

q.push(COPY[COPY.size() - 1]);

COPY.pop\_back();

}

cout << endl << q << endl;

break;

}

//print

case 6: {

cout << q << endl;

break;

}

//exit

default: {

cout << "\nВыход из меню" << endl;

exit(777);

}

}

}

int main() {

srand(time(NULL));

system("color F0");

setlocale(LC\_ALL, "Ru");

int n;

cout << "Введите размер очереди: ";

cin >> n;

priority\_queue<Pair> q;

Pair p;

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

p.set\_random();

q.push(p);

}

cout << q << endl;

while (true) {

system("pause");

system("cls");

cout << "1. Сортировка контейнера" << endl;

cout << "2. Найди себе пару" << endl;

cout << "3. Поставьте в конце среднее арифметическое" << endl;

cout << "4. Удаление элементов из диапазона" << endl;

cout << "5. Добавьте сводку минимальных и максимальных значений для каждой пары" << endl;

cout << "6. Контейнер для печати" << endl;

cout << "Выберите свой вариант: ";

int c;

cin >> c;

system("cls");

menu(c, q);

}

return 0;

}

Задача 3.

Pair.h

#pragma once

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

class Pair {

private:

int first;

double second;

public:

Pair() { first = 0; second = 0; };

Pair(int F, double S) { first = F; second = S; };

Pair(const Pair& t) { first = t.first;second = t.second; };

~Pair() {};

int get\_first() { return first; }

double get\_second() { return second; }

void set\_first(int F) { first = F; }

void set\_second(double S) { second = S; }

void set\_random();

double sum() { double s = second + first; return s; }

Pair operator =(const Pair&);

bool operator <(const Pair&) const;

bool operator >(const Pair&) const;

bool operator <=(const Pair&) const;

bool operator >=(const Pair&) const;

bool operator ==(const Pair&) const;

Pair& operator -(const Pair& p);

Pair& operator +(const Pair& p);

Pair& operator +=(const Pair& p);

Pair& operator +(const int X);

Pair& operator +(const double X);

Pair& operator /(const int X);

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

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

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

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

};

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

first += X;

return \*this;

}

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

second += X;

return \*this;

}

Pair& Pair::operator /(const int X) {

first = first / X;

second = second / X;

return \*this;

}

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

this->first -= p.first;

this->second -= p.second;

return \*this;

}

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

this->first += p.first;

this->second += p.second;

return \*this;

}

Pair& Pair::operator +=(const Pair& p) {

this->first += p.first;

this->second += p.second;

return \*this;

}

istream& operator >>(istream& in, Pair& t) {

cout << "целое первое = "; in >> t.first;

cout << "вещественное второе = "; in >> t.second;

return in;

}

ostream& operator <<(ostream& out, const Pair& t) {

return (out << t.first << " : " << t.second << endl);

}

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

if (&p == this) return \*this;

first = p.first;

second = p.second;

return\*this;

}

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

fin >> p.first;

fin >> p.second;

return fin;

}

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

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

return fout;

}

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

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis < sother) return true;

return false;

}

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

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis > sother) return true;

return false;

}

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

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis <= sother) return true;

return false;

}

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

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis >= sother) return true;

return false;

}

bool Pair::operator ==(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis == sother) return true;

return false;

}

void Pair::set\_random() {

first = rand() % 10 + 1;

second = round(((double)(rand()) / RAND\_MAX \* 10 + 1) \* 100) / 100;

}

Main.cpp

#include <iostream>

#include <list>

#include <map>

#include <vector>

#include <algorithm>

#include <functional>

#include "Pair.h"

using namespace std;

ostream& operator<<(ostream& out, map<double, Pair>& m) {

for (map <double, Pair>::iterator it = m.begin(); it != m.end(); it++) {

cout << "ключ " << it->first << ", value " << it->second;

}

return out;

}

void print\_vect(Pair elem) {

cout << "ключ " << elem.sum() << ", value " << elem << endl;

}

void menu(const int c, map<double, Pair>& m) {

switch (c) {

//sort

case 1: {

cout << m << endl;

cout << "Введите < для сортировки по возрастанию\nВведите > для сортировки по убыванию\n";

char op;

cin >> op;

if (op == '<') cout << m << endl;

else if (op == '>') {

vector<Pair> COPY;

Pair elem;

for (map <double, Pair>::iterator it = m.begin(); it != m.end(); ++it) {

elem = it->second;

COPY.push\_back(elem);

}

sort(COPY.begin(), COPY.end(), mem\_fun\_ref(&Pair::operator>));

for\_each(COPY.begin(), COPY.end(), ptr\_fun(print\_vect));

COPY.~vector();

}

else cout << "неправильный характер\n";

break;

}

//find element

case 2: {

cout << m << "\n\n";

cout << "Введите значение:\n";

Pair value;

cin >> value;

const auto it = find\_if(m.begin(), m.end(), [&value](decltype(\*m.begin())& it)->bool {

return it.second == value;

});

if (it == m.end()) {

cout << "такого элемента нет!" << endl;

break;

}

for (map<double, Pair>::iterator i = m.begin(); i != m.end(); ++i) {

if (i == it) {

cout << "элемент: " << i->second << endl;

}

}

break;

}

//arithmetical mean

case 3: {

cout << m << endl << endl;

Pair sum;

for (auto it = m.begin(); it != m.end(); it++) sum += it->second;

sum.set\_first(sum.get\_first() / m.size());

sum.set\_second(round((double)(sum.get\_second() / m.size()) \* 100) / 100);

cout << "среднее арифметическое равно " << sum.sum() << " in pair " << sum << "\n\n";

m.insert(make\_pair(sum.sum(), sum));

cout << m << endl;

break;

}

//delete from range

case 4: {

cout << m << endl << endl;

int one, two;

cout << "\nВведите первую границу:\n"; cin >> one;

cout << "\nВведите вторую границу:\n"; cin >> two;

if (one > two) swap(one, two);

while (find\_if(m.begin(), m.end(), [&one, &two](decltype(\*m.begin())& it) -> bool {

return it.second.sum() > one && it.second.sum() < two; // comparing with the object

}) != m.end()) {

const auto it = find\_if(m.begin(), m.end(), [&one, &two](decltype(\*m.begin())& it) -> bool {

return it.second.sum() > one && it.second.sum() < two; // comparing with the object

});

m.erase(it->first);

}

cout << endl << m << endl;

break;

}

//minmax

case 5: {

cout << m << endl << endl;

const auto mini = m.begin();

const auto maxi = --m.end();

cout << "min: " << mini->second << endl;

cout << "max: " << maxi->second << endl;

Pair sum;

sum = mini->second + maxi->second;

cout << "сумма = " << sum << endl;

for (auto it = m.begin(); it != m.end(); it++) {

it->second.set\_second(it->second.get\_second() + sum.get\_second());

it->second.set\_first(it->second.get\_first() + sum.get\_first());

}

cout << endl << m << endl;

break;

}

//print

case 6: {

cout << m << endl;

break;

}

//exit

default: {

cout << "\nВыход из меню" << endl;

exit(777);

}

}

}

int main() {

srand(time(NULL));

setlocale(LC\_ALL, "Ru");

int n;

cout << "Введите количество элементов контейнера: ";

cin >> n;

map<double, Pair> m;

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

Pair p;

p.set\_random();

m.insert(make\_pair(p.sum(), p));

}

cout << m << endl;

while (true) {

system("pause");

system("cls");

cout << "1. Сортировка контейнера" << endl;

cout << "2. Найди себе пару" << endl;

cout << "3. Поставьте в конце среднее арифметическое" << endl;

cout << "4. Удаление элементов из диапазона" << endl;

cout << "5. Добавьте сводку минимальных и максимальных значений для каждой пары" << endl;

cout << "6. Контейнер для печати" << endl;

cout << "Выберите свой вариант: ";

int c;

cin >> c;

system("cls");

menu(c, m);

}

return 0;

}

Результаты работы программы:

