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

ПНИПУ

**Лабораторная работа  
“Классы”. №13**

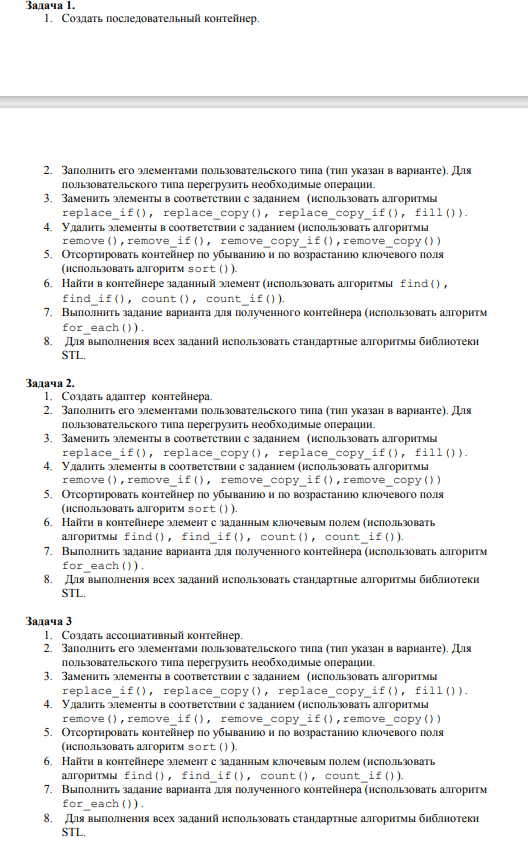
Выполнил:   
студент группы ИВТ-23-2б   
Чудинов Данил Николаевич

Проверила:   
доцент кафедры ИТАС   
О.А. Полякова

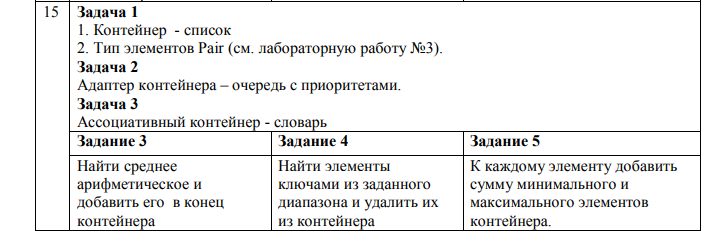
Пермь, 2024 г.

**«Стандартные обобщённые алгоритмы библиотеки STL»**

**Анализ задачи:**

****

***Вариант 15:***

****

**Код на языке C++:**

**13.1:**#include <iostream>

#include <list>

#include <string>

#include <algorithm>

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;

}

Pair(const Pair& d) {

this->x = d.x;

this->y = d.y;

}

double diff() {

return this->x - 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 GetInfo() {

cout << this->x << " : " << this->y << endl;

}

void operator=(const Pair& a) {

this->x = a.x;

this->y = a.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 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 operator + (Pair& p) {

x = x + p.x;

y = y + p.y;

return \*this;

}

bool operator > (const Pair& other){

if (this->x != other.x) {

return this->x > other.x;

}

else {

return this->y > other.y;

}

}

bool operator < (const Pair& other){

if (this->x != other.x) {

return this->x < other.x;

}

else {

return this->y < other.y;

}

}

~Pair() {

}

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

cout << "Введите первое (целое) число: ";

is >> p.x;

cout << "Введите второе (вещественное) число: ";

is >> p.y;

return is;

}

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

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

return os;

}

};

void RandomPair(Pair& p) {

int x = rand() % 1000;

double y = (rand() % 1000) \* 0.01;

p = Pair(x, y);

}

ostream& operator <<(ostream& os, list<Pair>& list) {

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

return os;

}

int main() {

system("chcp 1251>NULL");

cout << "Введите размер списка: ";

int N;

cin >> N;

Pair sum;

list<Pair> list\_1(N);

generate(list\_1.begin(), list\_1.end(), []() {Pair p; RandomPair(p); return p; });

cout << "Созданный список: " << endl << list\_1 << endl << endl;

for\_each(list\_1.begin(), list\_1.end(), [&sum](Pair& p) {sum = sum + p; });

sum.setX(sum.returnX() / list\_1.size());

sum.setY(((sum.returnY() \* 10000) / list\_1.size()) \* 0.0001);

cout << "Среднее арифмитическое: " << endl << sum << endl << endl;

list\_1.push\_back(sum);

cout << "Список, с добавленным в конец средним арифмитическим: " << endl << list\_1 << endl << endl;

cout << "Введите две пары чисел, которые будут служить диапазоном удаления: " << endl;

Pair one;

Pair two;

cout << "Первая пара: " << endl;

cin >> one;

cout << one << endl << endl;

cout << "Вторая пара: " << endl;

cin >> two;

cout << two << endl << endl;

if (one > two) {

swap(one, two);

}

list<Pair>::iterator s = remove\_if(list\_1.begin(), list\_1.end(), [one, two](Pair& p) {return p > one && p < two; });

list\_1.erase(s, list\_1.end());

cout << "Полученный список: " << endl << list\_1 << endl << endl;

pair<list<Pair>::iterator, list<Pair>::iterator> stl\_pair = minmax\_element(list\_1.begin(), list\_1.end());

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

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

Pair sum\_pair(stl\_pair.first->returnX() + stl\_pair.second->returnX(), stl\_pair.first->returnY() + stl\_pair.second->returnY());

cout << "Сумма max и min: " << endl << sum\_pair << endl;

for\_each(list\_1.begin(), list\_1.end(), [&sum\_pair](Pair& p) {p = p + sum\_pair; });

cout << "Полученный список, каждый элемент которого увеличен на сумму max+min: " << endl << list\_1 << endl << endl;

list\_1.sort([](Pair& a, Pair& b) {return a < b; });

cout << "Список, отсортированный по возрастанию: " << endl << list\_1 << endl << endl;

list\_1.sort([](Pair& a, Pair& b) {return a > b; });

cout << "Список, отсортированный по убыванию: " << endl << list\_1 << endl << endl;

cout << "Введите значение (целочисленное), по которому надо найти элемент в списке: " << endl;

int x;

cin >> x;

auto i = find\_if(list\_1.begin(), list\_1.end(), [&x](Pair& p) {return x == p.returnX(); });

if (i != end(list\_1)) {

cout << "Элемент найден: " << \*i << endl;

}

else {

cout << "Элемент не найден";

}

return 0;

}

**13.2:**

#include <iostream>

#include <algorithm>

#include <random>

#include <queue>

#include <functional>

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;

}

Pair(const Pair& d) {

this->x = d.x;

this->y = d.y;

}

double diff() {

return this->x - 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 GetInfo() {

cout << this->x << " : " << this->y << endl;

}

void operator=(const Pair& a) {

this->x = a.x;

this->y = a.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 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 operator + (Pair& p) {

x = x + p.x;

y = y + p.y;

return \*this;

}

bool operator > (const Pair& other) {

if (this->x != other.x) {

return this->x > other.x;

}

else {

return this->y > other.y;

}

}

bool operator < (const Pair& other) {

if (this->x != other.x) {

return this->x < other.x;

}

else {

return this->y < other.y;

}

}

~Pair() {

}

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

cout << "Введите первое (целое) число: ";

is >> p.x;

cout << "Введите второе (вещественное) число: ";

is >> p.y;

return is;

}

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

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

return os;

}

};

void RandomPair(Pair& p) {

int x = rand() % 1000;

double y = (rand() % 1000) \* 0.01;

p = Pair(x, y);

}

class Greater\_ThanPair {

public:

bool operator () (Pair& p1, Pair& p2) {

return p1 < p2;

}

};

void genetare\_Q(priority\_queue<Pair, vector<Pair>, Greater\_ThanPair>& q, int& size) {

Pair qwe(0, 0);

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

RandomPair(qwe);

q.push(qwe);

}

}

ostream& operator<<(ostream& os, const priority\_queue<Pair, vector<Pair>, Greater\_ThanPair>& q) {

auto temp\_q = q;

while (!temp\_q.empty()) {

Pair user = temp\_q.top();

os << user << endl;

temp\_q.pop();

}

return os;

}

Pair all\_sum(priority\_queue<Pair, vector<Pair>, Greater\_ThanPair>& q) {

Pair p(0, 0);

auto temp\_q = q;

while (!temp\_q.empty()) {

Pair user = temp\_q.top();

p = p + user;

temp\_q.pop();

}

return p;

}

void remove\_from\_range(priority\_queue<Pair, vector<Pair>, Greater\_ThanPair>& q, Pair& range1, Pair& range2) {

priority\_queue<Pair, vector<Pair>, Greater\_ThanPair> q1;

auto temp\_q = q;

while (!temp\_q.empty()) {

Pair user = temp\_q.top();

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

q1.push(user);

}

temp\_q.pop();

}

q = q1;

}

Pair max(priority\_queue<Pair, vector<Pair>, Greater\_ThanPair>& q) {

Pair max\_p = q.top();

return max\_p;

}

Pair min(priority\_queue<Pair, vector<Pair>, Greater\_ThanPair>& q) {

auto temp\_q = q;

Pair min\_p;

while (!temp\_q.empty()) {

min\_p = temp\_q.top();

temp\_q.pop();

}

return min\_p;

}

priority\_queue<Pair, vector<Pair>, Greater\_ThanPair> add(priority\_queue<Pair, vector<Pair>, Greater\_ThanPair>& q, Pair& para) {

priority\_queue<Pair, vector<Pair>, Greater\_ThanPair> q1;

auto temp\_q = q;

while (!temp\_q.empty()) {

Pair temp = temp\_q.top();

q1.push(temp + para);

temp\_q.pop();

}

return q1;

}

bool find\_para(priority\_queue<Pair, vector<Pair>, Greater\_ThanPair>& q, int& x, Pair& p) {

auto temp\_q = q;

while (!temp\_q.empty()) {

p = temp\_q.top();

if (x == p.returnX()) {

return true;

}

temp\_q.pop();

}

return false;

}

int main() {

system("chcp 1251>NULL");

cout << "Введите размер очереди: ";

int N;

cin >> N;

priority\_queue<Pair, vector<Pair>, Greater\_ThanPair> q;

int x = 0;

double y = 0;

Pair p(x, y);

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

cin >> p;

q.push(p);

}

cout << q;

p = all\_sum(q);

p.setX(p.returnX() / N);

p.setY(((p.returnY() \* 1000) / N) \* 0.001);

cout << endl;

cout << "Среднее арифмитическое: " << endl << p << endl << endl;

q.push(p);

cout << "После добавления среднего арифметического: " << endl << q << endl << endl;

Pair one, two;

cout << "Введите пары чисел, диапазон которыз будет служить удалением чисел: " << endl;

cout << "Первая пара: " << endl;

cin >> one;

cout << "Вторая пара: " << endl;

cin >> two;

if (one > two) {

swap(one, two);

}

remove\_from\_range(q, one, two);

cout << "После удаления: " << endl << q << endl << endl;

Pair maxel = max(q);

Pair minel = min(q);

cout << "Максимальный элемент: " << endl << maxel << endl;

cout << "Минимальный элемент: " << endl << minel << endl << endl;

p = maxel + minel;

cout << "Сумма max и min: " << endl << p << endl << endl;

q.push(p);

cout << "Очередь после добавления max и min: " << endl << q << endl << endl;

cout << "Введите целочисленное X, которое нужно найти: " << endl;

Pair sum(0, 0);

int X;

cin >> X;

bool flag = find\_para(q, X, p);

if (flag) {

cout << "Элемент найден: " << p;

}

else {

cout << "Элемент не найден";

}

return 0;

}

**13.3:**

#include <iostream>

#include <list>

#include <string>

#include <algorithm>

#include <map>

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;

}

Pair(const Pair& d) {

this->x = d.x;

this->y = d.y;

}

double diff() {

return this->x - 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 GetInfo() {

cout << this->x << " : " << this->y << endl;

}

void operator=(const Pair& a) {

this->x = a.x;

this->y = a.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 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 operator + (Pair& p) {

x = x + p.x;

y = y + p.y;

return \*this;

}

bool operator > (const Pair& other) {

if (this->x != other.x) {

return this->x > other.x;

}

else {

return this->y > other.y;

}

}

bool operator < (const Pair& other) {

if (this->x != other.x) {

return this->x < other.x;

}

else {

return this->y < other.y;

}

}

~Pair() {

}

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

cout << "Введите первое (целое) число: ";

is >> p.x;

cout << "Введите второе (вещественное) число: ";

is >> p.y;

return is;

}

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

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

return os;

}

friend bool operator ==(const Pair& a, const Pair& b) {

if (a.x == b.x) {

return a.x == b.x;

}

}

void operator /(const int& a) {

this->x = x / a;

}

friend bool operator >(const Pair& a, const Pair& b) {

if (a.x != b.x) {

return a.x > b.x;

}

else {

return a.y > b.y;

}

}

friend bool operator <(const Pair& a, const Pair& b) {

if (a.x != b.x) {

return a.x < b.x;

}

else {

return a.y < b.y;

}

}

};

void RandomPair(Pair& p) {

int x = rand() % 1000;

double y = (rand() % 1000) \* 0.01;

p = Pair(x, y);

}

void generate1(map<Pair, Pair>& my1, int size) {

Pair qwe\_1(0, 0), qwe\_2(0, 0);

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

RandomPair(qwe\_1);

RandomPair(qwe\_2);

my1[qwe\_1] = qwe\_2;

}

}

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

os << "Ключ - " << p.first << " Значение - " << p.second << endl;

return os;

}

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

map<Pair, Pair>::const\_iterator It = p.begin();

while (It != p.end()) {

cout << \*It++;

}

return os;

}

bool isInRange(Pair& key, Pair& l, Pair& u) {

return key > l && key < u;

}

void remove\_from\_range(map<Pair, Pair>&p, Pair & l, Pair & u) {

auto it = p.begin();

Pair ptr;

while (it != p.end()) {

ptr = it->first;

if (!isInRange(ptr, l, u)) {

it++;

}

else

{

it = p.erase(it);

}

}

}

Pair minimum\_pair(map<Pair, Pair>& p) {

auto it = p.begin();

Pair ptr;

ptr = it->first;

return ptr;

}

Pair maximum\_pair(map<Pair, Pair>& p) {

auto it = p.begin();

Pair ptr;

while (it != p.end()) {

ptr = it->first;

it++;

}

return ptr;

}

void all\_sum(map<Pair, Pair>& p, Pair& key, Pair& value) {

for (const auto& pair : p) {

Pair asd = pair.first;

key = key + asd;

asd = pair.second;

value = value + asd;

}

}

map<Pair, Pair> add\_to\_all(map<Pair, Pair>& p, Pair& key, Pair& value) {

map<Pair, Pair> ptr;

for (const auto& pair : p) {

Pair asd\_1 = pair.first;

Pair asd\_2 = pair.second;

ptr[asd\_1 + key] = (asd\_2 + value);

}

return ptr;

}

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

auto it = p.find(key);

if (it != p.end()) {

value = Pair(it->second);

return true;

}

else {

return false;

}

}

int main() {

system("chcp 1251>NULL");

cout << "Введите кол-во элементов: ";

int N;

cin >> N;

Pair sum\_1(0, 0), sum\_2(0, 0), sum\_key(0, 0), sum\_value(0, 0);

map<Pair, Pair> adj;

generate1(adj, N);

cout << "Полученный словарь: " << endl << adj << endl << endl;

all\_sum(adj, sum\_1, sum\_2);

sum\_1 / adj.size();

sum\_2 / adj.size();

cout << "Cреднее арифмитическое ключей: " << endl << sum\_1 << endl << endl;

cout<< "Cреднее арифмитическое значений: " << endl << sum\_2 << endl << endl;

adj[sum\_1] = sum\_2;

cout << "Контейнер с средним арифмитическим: " << endl << adj << endl << endl;

cout << "Введите две пары чисел, которые будут служить диапазоном удаления: " << endl;

Pair one;

Pair two;

cout << "Первая пара: " << endl;

cin >> one;

cout << one << endl << endl;

cout << "Вторая пара: " << endl;

cin >> two;

cout << two << endl << endl;

if (one > two) {

swap(one, two);

}

remove\_from\_range(adj, one, two);

cout << "После удаления из диапазона: " << endl << adj << endl << endl;

Pair max = maximum\_pair(adj);

Pair min = minimum\_pair(adj);

findPair(adj, min, sum\_1);

findPair(adj, max, sum\_2);

cout << "Минимальный элемент по ключу: " << endl;

cout << "Ключ - " << min << " Значение - " << sum\_1 << endl << endl;

cout << "Максимальный элемент по ключу: " << endl;

cout << "Ключ - " << max << " Значение - " << sum\_2 << endl << endl;

sum\_key = max + min;

sum\_value = sum\_1 + sum\_2;

cout << "Сумма max и min: " << endl;

cout << "Ключ - " << sum\_key << " Значение - " << sum\_value << endl << endl;

adj = add\_to\_all(adj, sum\_key, sum\_value);

cout << "После добавления max и min: " << endl << adj << endl << endl;

cout << "Поиск элемента по ключу, введите ключ, по которому нужно найти элемент: " << endl;

Pair o;

cin >> o;

bool find = findPair(adj, o, two);

if (find) {

cout << "Элемент найден: " << endl;

cout << "Ключ - " << o << " Значение - " << two << endl << endl;

}

else {

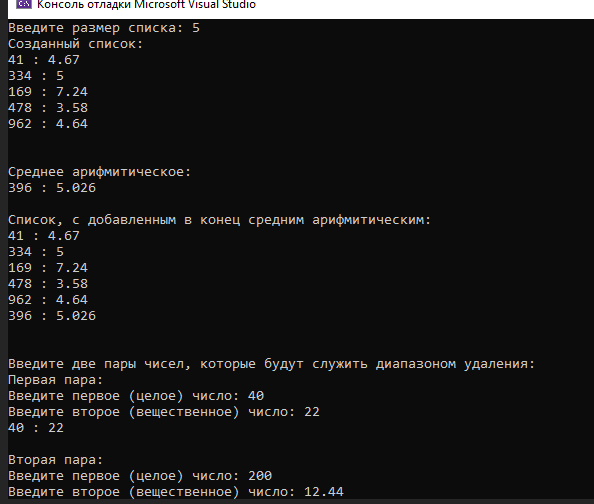
cout << "Элемент не найден";

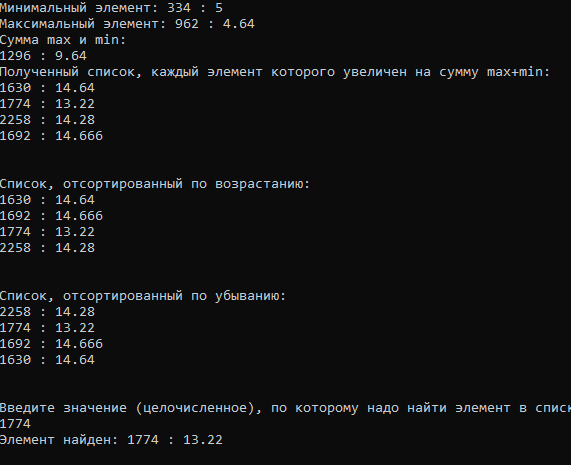
}

}

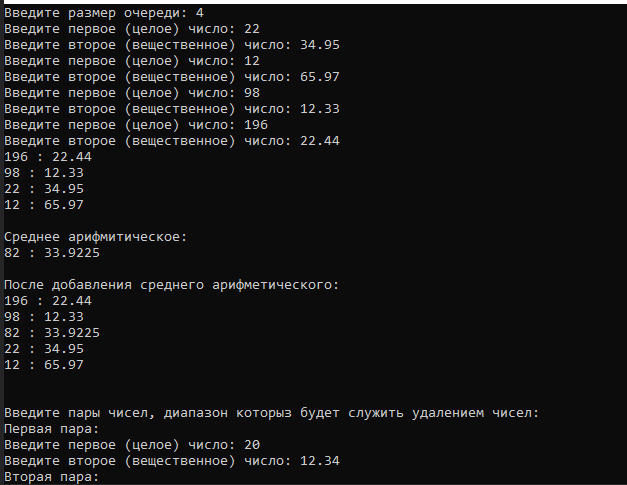
**Работа программы:**

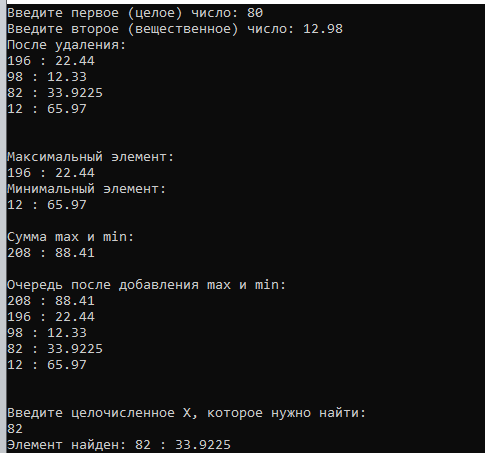
**13.1:**

****

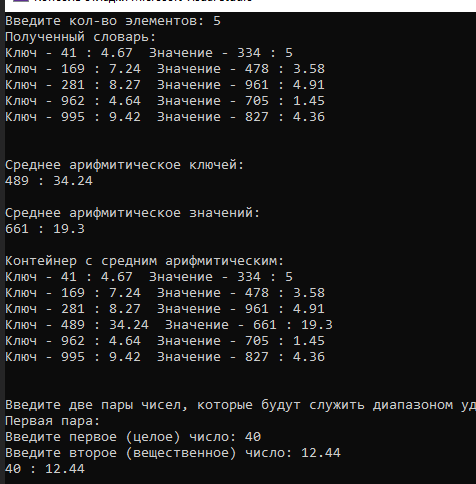


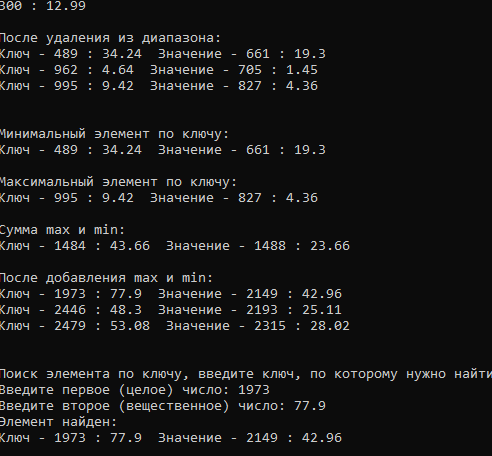
**13.2:**

****

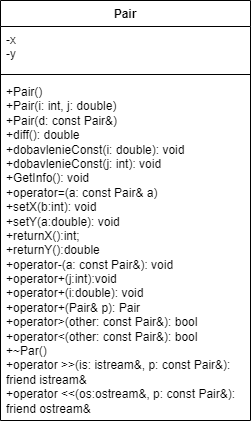
****

**13.3:**

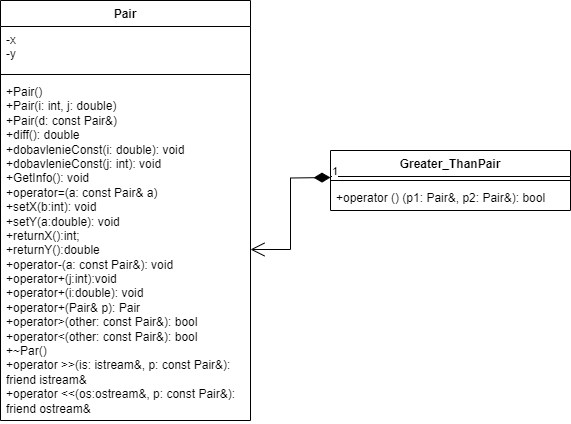
****

****

**13.1:**



**13.2:**

****

**13.3:**

