МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Лабораторная работа № 13:**

"Стандартные обобщенные алгоритмы библиотеки STL"

Выполнила: студентка гр.РИС-23-3Б

Мокроусова Ангелина Андреевна

Проверила: доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь 2024

**Постановка задачи:**

Общая часть:

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()).

1. Удалить элементы в соответствии с заданием (использовать алгоритмы

remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

1. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).
2. Найти в контейнере заданный элемент (использовать алгоритмы find(), find\_if(), count(), count\_if()).
3. Выполнить задание варианта для полученного контейнера (использовать алгоритм

for\_each()) .

1. 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()).

1. Удалить элементы в соответствии с заданием (использовать алгоритмы

remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

1. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).
2. Найти в контейнере элемент с заданным ключевым полем (использовать алгоритмы find(), find\_if(), count(), count\_if()).
3. Выполнить задание варианта для полученного контейнера (использовать алгоритм

for\_each()) .

1. 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()).

1. Удалить элементы в соответствии с заданием (использовать алгоритмы

remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

1. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).
2. Найти в контейнере элемент с заданным ключевым полем (использовать

алгоритмы find(), find\_if(), count(), count\_if()).

1. Выполнить задание варианта для полученного контейнера (использовать алгоритм

for\_each()) .

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

Задание варианта:

|  |  |  |
| --- | --- | --- |
| **Задача 1:**   1. Контейнер - двунаправленная очередь 2. Тип элементов Pair (см. лабораторную работу №3).   **Задача 2:**  Адаптер контейнера – стек.  **Задача 3:**  Ассоциативный контейнер - множество | | |
| **Задание 3** | **Задание 4** | **Задание 5** |
| Найти максимальный элемент и добавить его в конец контейнера | Найти элементы ключами из заданного диапазона и удалить их  из контейнера | К каждому элементу добавить среднее арифметическое контейнера. |

**Анализ задачи:**

Задача 1:

Класс Pair: Определен класс Pair, который представляет собой пару чисел. Класс содержит конструкторы, селекторы, модификаторы и перегруженные операторы сравнения и присваивания, что позволяет использовать объекты этого класса в стандартных алгоритмах STL.

Инициализация и заполнение контейнера: Контейнер deque инициализируется и заполняется случайными элементами типа Pair с помощью функции RandomPair, которая использует генератор случайных чисел.

Поиск и добавление максимального элемента: С помощью алгоритма max\_element находится максимальный элемент в контейнере, который затем добавляется в его конец.

Удаление элементов из диапазона: Функция removeInRange удаляет из deque все элементы, значения которых попадают в заданный диапазон.

Вычисление и добавление среднего значения: Вычисляется среднее арифметическое всех элементов контейнера, после чего каждый элемент увеличивается на это среднее значение.

Поиск элемента: Функция findElement осуществляет поиск элемента в контейнере по заданным значениям.

Сортировка контейнера: Контейнер сортируется по возрастанию первого элемента пары с использованием алгоритма sort.

Задача 2:

Инициализация и заполнение стека: Стек заполняется случайными объектами Pair с помощью функции RandomPair.

Копирование в deque: Элементы стека копируются в deque для дальнейшей работы с итераторами.

Поиск и добавление максимального элемента: Находится максимальный элемент в deque и добавляется в его конец.

Удаление элементов в диапазоне: Из deque удаляются элементы, попадающие в заданный диапазон значений.

Вычисление и добавление среднего значения: Вычисляется среднее значение элементов deque и каждый элемент увеличивается на это значение.

Поиск элемента: Осуществляется поиск заданного элемента в deque.

Копирование обратно в стек: Элементы deque копируются обратно в стек.

Задача 3:

Генерация случайных пар чисел: Функция RandomPair генерирует случайные пары чисел для заполнения контейнера set.

Удаление элементов в заданном диапазоне: Функция removeInRange удаляет элементы из set, которые попадают в заданный диапазон значений.

Поиск элемента: Функция findElement осуществляет поиск заданного элемента в set.

Вычисление среднего значения: Вычисляется среднее значение всех чисел в set.

Модификация элементов: Создается новый set с элементами, увеличенными на среднее значение.

**Код на С++ с комментариями:**

Задача 1:

#include <iostream>

#include <list>

#include <string>

#include <algorithm>

#include <deque>

#include <stack>

#include <set>

#include <cstdlib>

#include <ctime>

using namespace std;

// Класс Pair представляет собой пару чисел

class Pair {

private:

int first; // Первое число

double second; // Второе число

public:

// Конструктор по умолчанию инициализирует поля нулями

Pair() : first(0), second(0.0) {}

// Конструктор с параметрами инициализирует поля заданными значениями

Pair(int first, double second) : first(first), second(second) {}

// Конструктор копирования создает копию существующего объекта Pair

Pair(const Pair& other) : first(other.first), second(other.second) {}

// Деструктор вызывается, когда объект Pair удаляется

~Pair() {}

// Селекторы возвращают значения полей объекта

int getFirst() const { return first; }

double getSecond() const { return second; }

// Модификаторы изменяют значения полей объекта

void setFirst(int first) { this->first = first; }

void setSecond(double second) { this->second = second; }

// Операция присваивания присваивает значения полей одного объекта значениям полей другого

Pair& operator=(const Pair& other) {

first = other.first;

second = other.second;

return \*this;

}

// Операции сравнения сравнивают поля объектов

bool operator<(const Pair& other) const { return first < other.first; } // Сравнение по первому числу

bool operator>(const Pair& other) const { return second > other.second; } // Сравнение по второму числу

// Перегрузка оператора ввода >> для ввода пары чисел из потока ввода

friend istream& operator>>(istream& in, Pair& pair) {

in >> pair.first >> pair.second;

return in;

}

// Перегрузка оператора вывода << для вывода пары чисел в поток вывода

friend ostream& operator<<(ostream& out, const Pair& pair) {

out << "[" << pair.first << ":" << pair.second<< "]" << " ";

return out;

}

};

// Функция RandomPair генерирует случайную пару чисел и присваивает ее параметру p

void RandomPair(Pair& p) {

int x = rand() % 10;

double y = (rand() % 10);

p = Pair(x, y);

}

// Функция удаления элементов из диапазона

void removeInRange(deque<Pair>& p, int a, int b) {

auto it = p.begin();

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

if ((it->getFirst() >= a && it->getFirst() <= b) || (it->getSecond() >= a && it->getSecond() <= b)) {

it = p.erase(it);

}

else {

++it;

}

}

}

// Функция поиска элемента в деке

void findElement(const deque<Pair>& p, int a, double b) {

int i = 0;

for (const Pair& elem : p) {

if (elem.getFirst() == a && elem.getSecond() == b) {

cout << "Нужный элемент находится по индексу: " << i << endl;

return;

}

i++;

}

cout << "Элемент не найден" << endl;

}

int main() {

setlocale(LC\_ALL, "ru");// Установка локали для вывода сообщений на русском языке

srand(time(0));// Инициализация генератора случайных чисел

// Ввод размера контейнера

cout << "Введите размер очереди: ";

int N;

cin >> N;

// Заполнение очереди случайными парами чисел

deque<Pair> queue(N);

for\_each(queue.begin(), queue.end(), RandomPair);

// Вывод содержимого очереди

cout<< endl << "Содержимое очереди: " << endl;

for (const Pair& p : queue)

cout << p;

// Нахождение максимального элемента в очереди и добавление его в конец очереди

auto max\_it = max\_element(queue.begin(), queue.end());

queue.push\_back(\*max\_it);

cout << endl;

// Вывод содержимого очереди

cout << endl << "После добавления максимальной пары в конец (по 1 элементу): " << endl;

for (const Pair& p : queue)

cout << p;

cout << endl;

// Ввод диапазона и удаление элементов

int a, b;

cout << endl << "Введите диапазон для удаления: ";

cin >> a >> b;

removeInRange(queue, a, b);

// Вывод содержимого очереди

cout << endl << "После удаления элементов входящих в ранее введенный диапозон (не включительно): " << endl;

for (const Pair& p : queue)

cout << p;

// Вычисление среднего значения элементов очереди

double sum = 0;

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

int average = sum/ (queue.size()\*2);

cout << endl;

cout << endl << "Cреднее значение элементов: " << average << endl;

// Увеличение каждого элемента очереди на среднее значение

for\_each(queue.begin(), queue.end(), [average](Pair& p) { p.setFirst(p.getFirst() + average); p.setSecond(p.getSecond() + average); });

// Вывод содержимого очереди

cout << endl << "После увеличения каждого элемента на среднее значение: " << endl;

for (const Pair& p : queue)

cout << p;

cout << endl;

// Ввод целого числа для поиска в множестве

cout << endl << "Введите целое число для поиска: ";

int x;

double y;

cin >> x >> y;

findElement(queue, x, y);

sort(queue.begin(), queue.end());

// Вывод содержимого очереди

cout << endl << "После сортировки (по 1 элементу): " << endl;

for (const Pair& p : queue)

cout << p;

cout << endl;

return 0;

}

Задача 2:

#include <iostream>

#include <list>

#include <string>

#include <algorithm>

#include <deque>

#include <stack>

#include <set>

#include <cstdlib>

#include <ctime>

using namespace std;

// Класс Pair представляет собой пару чисел

class Pair {

private:

int first; // Первое число

double second; // Второе число

public:

// Конструктор по умолчанию инициализирует поля нулями

Pair() : first(0), second(0.0) {}

// Конструктор с параметрами инициализирует поля заданными значениями

Pair(int first, double second) : first(first), second(second) {}

// Конструктор копирования создает копию существующего объекта Pair

Pair(const Pair& other) : first(other.first), second(other.second) {}

// Деструктор вызывается, когда объект Pair удаляется

~Pair() {}

// Селекторы возвращают значения полей объекта

int getFirst() const { return first; }

double getSecond() const { return second; }

// Модификаторы изменяют значения полей объекта

void setFirst(int first) { this->first = first; }

void setSecond(double second) { this->second = second; }

// Операция присваивания присваивает значения полей одного объекта значениям полей другого

Pair& operator=(const Pair& other) {

first = other.first;

second = other.second;

return \*this;

}

// Операции сравнения сравнивают поля объектов

bool operator<(const Pair& other) const { return first < other.first; } // Сравнение по первому числу

bool operator>(const Pair& other) const { return second > other.second; } // Сравнение по второму числу

// Перегрузка оператора ввода >> для ввода пары чисел из потока ввода

friend istream& operator>>(istream& in, Pair& pair) {

in >> pair.first >> pair.second;

return in;

}

// Перегрузка оператора вывода << для вывода пары чисел в поток вывода

friend ostream& operator<<(ostream& out, const Pair& pair) {

out << "[" << pair.first << ":" << pair.second << "]" << " ";

return out;

}

};

// Функция RandomPair генерирует случайную пару чисел и присваивает ее параметру p

void RandomPair(Pair& p) {

int x = rand() % 10;

double y = (rand() % 10);

p = Pair(x, y);

}

// Функция для удаления элементов из диапазона

void removeInRange(deque<Pair>& p, int a, int b) {

p.erase(remove\_if(p.begin(), p.end(), [a, b](const Pair& elem) {

return (elem.getFirst() >= a && elem.getFirst() <= b) || (elem.getSecond() >= a && elem.getSecond() <= b);

}), p.end());

}

// Функция поиска элемента в деке

void findElement(const deque<Pair>& p, int a, double b) {

int i = 0;

for (const Pair& elem : p) {

if (elem.getFirst() == a && elem.getSecond() == b) {

cout << "Нужный элемент находится по индексу: " << i << endl;

return;

}

i++;

}

cout << "Элемент не найден" << endl;

}

int main() {

setlocale(LC\_ALL, "ru");// Установка локали для вывода сообщений на русском языке

srand(time(0));// Инициализация генератора случайных чисел

// Ввод размера контейнера

cout << "Введите размер стека: ";

int N;

cin >> N;

// Создание стека контейнеров

stack<Pair> stk;

// Заполнение стека случайными парами чисел

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

Pair p;

RandomPair(p);

stk.push(p);

}

// Вывод содержимого стека

cout << endl << "Содержимое стека: " << endl;

stack<Pair> tmpStack = stk;

while (!tmpStack.empty()) {

cout << tmpStack.top();

tmpStack.pop();

}

// Копирование элементов стека в deque для работы с итераторами

deque<Pair> queue;

while (!stk.empty()) {

queue.push\_front(stk.top());

stk.pop();

}

// Нахождение максимального элемента и добавление его в конец

auto max\_it = max\_element(queue.begin(), queue.end());

queue.push\_back(\*max\_it);

// Вывод содержимого

cout << endl << "Добавления маскимального элемента в конец: " << endl;

for (const Pair& p : queue)

cout << p;

// Ввод диапазона и удаление элементов

int a, b;

cout << endl << "Введите диапазон для удаления: ";

cin >> a >> b;

removeInRange(queue, a, b);

// Вывод содержимого deque после удаления элементов

cout << endl << "Удаления элементов в заданном диапазоне: " << endl;

for (const auto& elem : queue) {

cout << elem;

}

// Вычисление среднего значения элементов

double sum = 0;

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

int average = sum / (queue.size() \* 2);

cout << endl;

cout << endl << "Cреднее значение элементов: " << average << endl;

// Увеличение каждого элемента на среднее значение

for\_each(queue.begin(), queue.end(), [average](Pair& p) { p.setFirst(p.getFirst() + average); p.setSecond(p.getSecond() + average); });

// Вывод содержимого

cout << endl << "Увеличения каждого элемента на среднее значение: " << endl;

for (const Pair& p : queue)

cout << p;

cout << endl;

// Ввод целого числа для поиска в множестве

cout << endl << "Введите целое число для поиска: ";

int x;

double y;

cin >> x >> y;

findElement(queue, x, y);

// Копирование элементов обратно в стек

while (!queue.empty()) {

stk.push(queue.front());

queue.pop\_front();

}

return 0;

}

Задача 3:

#include <iostream>

#include <list>

#include <string>

#include <algorithm>

#include <deque>

#include <stack>

#include <set>

#include <cstdlib>

#include <ctime>

using namespace std;

// Класс Pair представляет собой пару чисел

class Pair {

private:

int first; // Первое число

double second; // Второе число

public:

// Конструктор по умолчанию инициализирует поля нулями

Pair() : first(0), second(0.0) {}

// Конструктор с параметрами инициализирует поля заданными значениями

Pair(int first, double second) : first(first), second(second) {}

// Конструктор копирования создает копию существующего объекта Pair

Pair(const Pair& other) : first(other.first), second(other.second) {}

// Деструктор вызывается, когда объект Pair удаляется

~Pair() {}

// Селекторы возвращают значения полей объекта

int getFirst() const { return first; }

double getSecond() const { return second; }

// Модификаторы изменяют значения полей объекта

void setFirst(int first) { this->first = first; }

void setSecond(double second) { this->second = second; }

// Операция присваивания присваивает значения полей одного объекта значениям полей другого

Pair& operator=(const Pair& other) {

first = other.first;

second = other.second;

return \*this;

}

// Операции сравнения сравнивают поля объектов

bool operator<(const Pair& other) const { return first < other.first; } // Сравнение по первому числу

bool operator>(const Pair& other) const { return second > other.second; } // Сравнение по второму числу

// Перегрузка оператора ввода >> для ввода пары чисел из потока ввода

friend istream& operator>>(istream& in, Pair& pair) {

in >> pair.first >> pair.second;

return in;

}

// Перегрузка оператора вывода << для вывода пары чисел в поток вывода

friend ostream& operator<<(ostream& out, const Pair& pair) {

out << "[" << pair.first << ":" << pair.second << "]" << " ";

return out;

}

};

// Функция RandomPair генерирует случайную пару чисел и присваивает ее параметру p

void RandomPair(Pair& p) {

int x = rand() % 1000;

double y = rand() % 1000;

p = Pair(x, y);

}

// Функция удаления элементов из диапазона

void removeInRange(set<Pair>& s, int a, int b) {

auto it = s.begin();

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

if ((it->getFirst() >= a && it->getFirst() <= b) || (it->getSecond() >= a && it->getSecond() <= b)) {

it = s.erase(it);

}

else {

++it;

}

}

}

// Функция поиска элемента в деке

void findElement(const set<Pair>&p, int a, double b) {

int i = 0;

for (const Pair& elem : p) {

if (elem.getFirst() == a && elem.getSecond() == b) {

cout << "Нужный элемент находится по индексу: " << i << endl;

return;

}

i++;

}

cout << "Элемент не найден" << endl;

}

int main() {

setlocale(LC\_ALL, "ru"); // Установка локали для вывода сообщений на русском языке

srand(time(0));// Инициализация генератора случайных чисел

// Ввод размера контейнера

cout << "Введите размер множества: ";

int N;

cin >> N;

set<Pair> mset;

// Заполнение множества случайными парами чисел

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

Pair p;

RandomPair(p);

mset.insert(p);

}

// Вывод содержимого множества

cout << endl << "Содержимое множества: " << endl;

for (const Pair& p : mset)

cout << p;

// Нахождение максимального элемента в множестве и добавление его в конец множества

auto max\_it = max\_element(mset.begin(), mset.end());

Pair maxElem = \*max\_it;

mset.insert(maxElem);

// Ввод диапазона и удаление элементов

int a, b;

cout << endl << "Введите диапазон для удаления: ";

cin >> a >> b;

removeInRange(mset, a, b);

// Вывод содержимого множества после удаления элементов

cout << endl << "Содержимое множества после удаления элементов в заданном диапазоне: " << endl;

for (const Pair& p : mset)

cout << p;

// Вычисление среднего значения элементов очереди

double sum = 0;

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

int average = sum / (mset.size() \* 2);

cout << endl;

cout << endl << "Cреднее значение элементов: " << average << endl;

// Создание нового множества для хранения измененных элементов

set<Pair> modifiedSet;

// Увеличение каждого элемента множества на среднее значение и добавление в новое множество

for (const Pair& p : mset) {

Pair modifiedPair(p.getFirst() + average, p.getSecond() + average);

modifiedSet.insert(modifiedPair);

}

// Вывод содержимого нового множества

cout << endl << "Содержимое множества после увеличения каждого элемента на среднее значение: " << endl;

for (const Pair& p : modifiedSet)

cout << p;

// Ввод целого числа и дробного числа для поиска в множестве

cout << endl << "Введите целое число и дробное число для поиска: ";

int x;

double y;

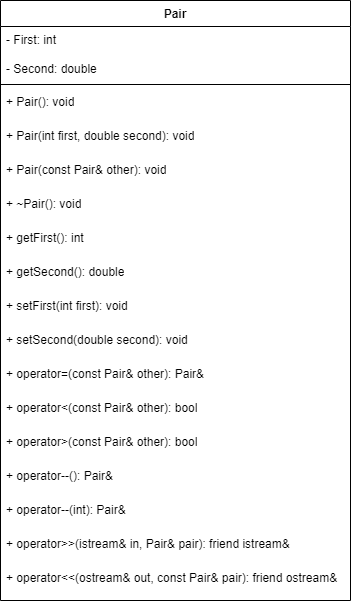
cin >> x >> y;

findElement(modifiedSet, x, y);

return 0;

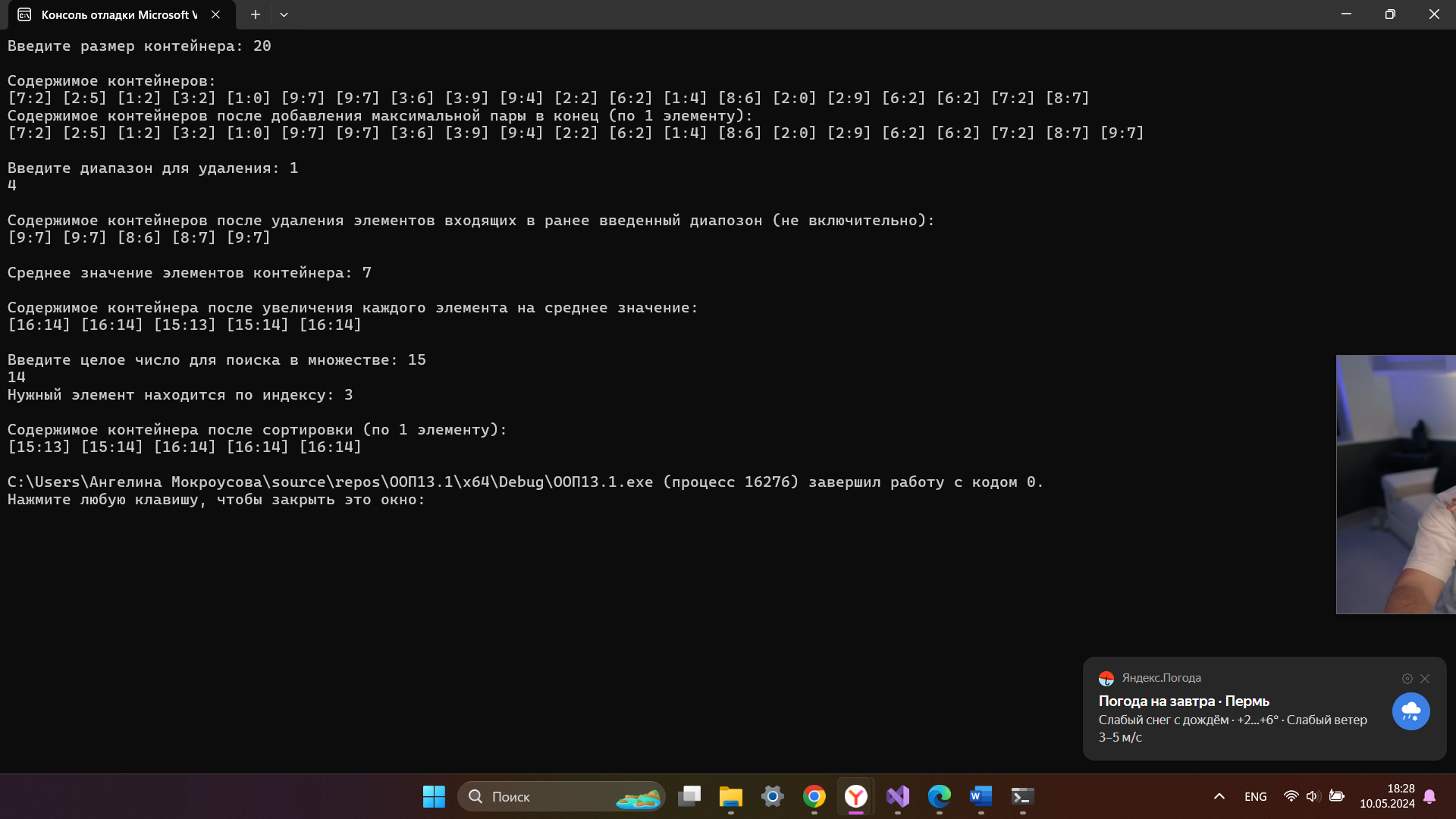
}

**UML диаграмма:**

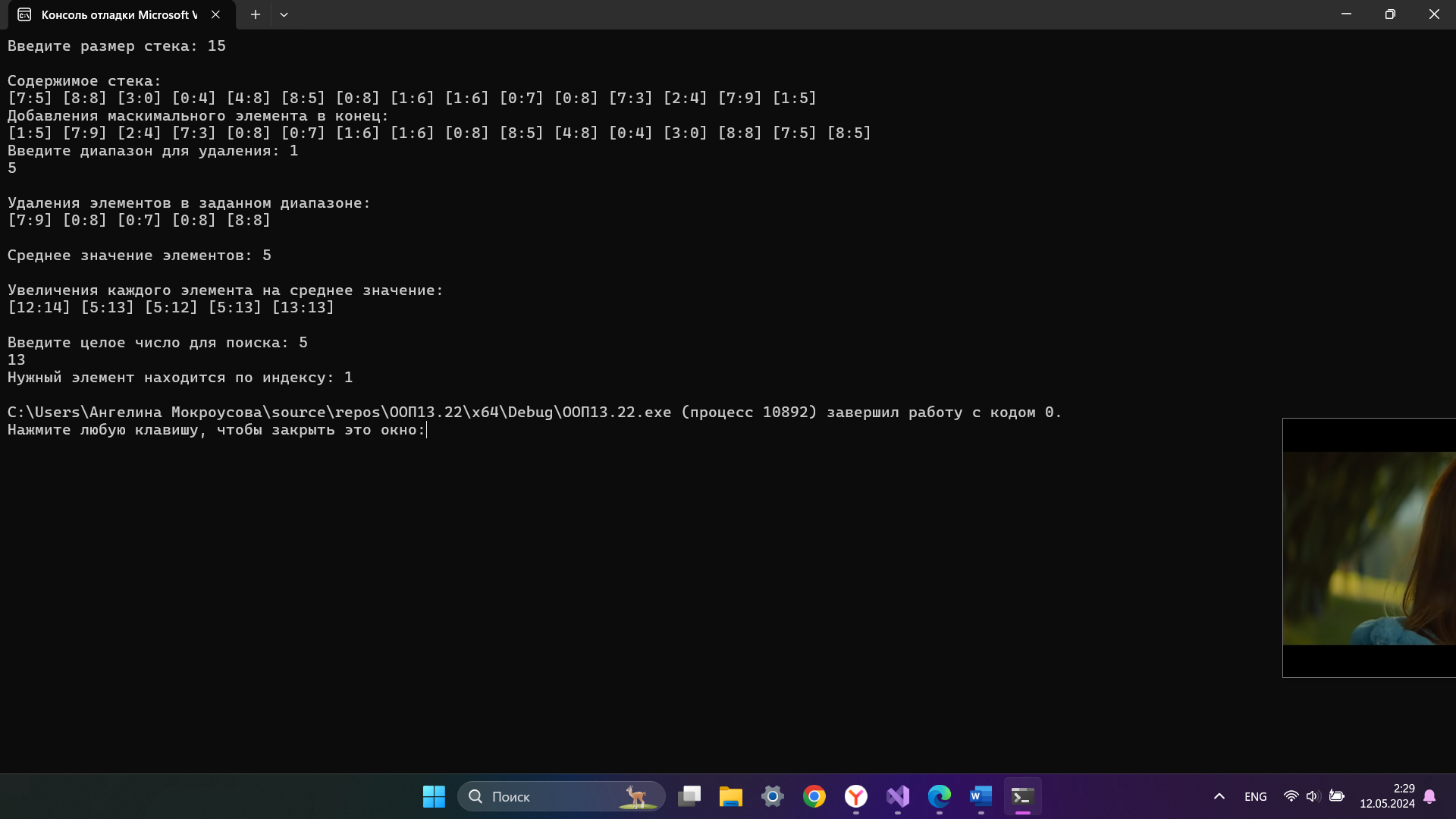
****

**Скриншот работы программы:**

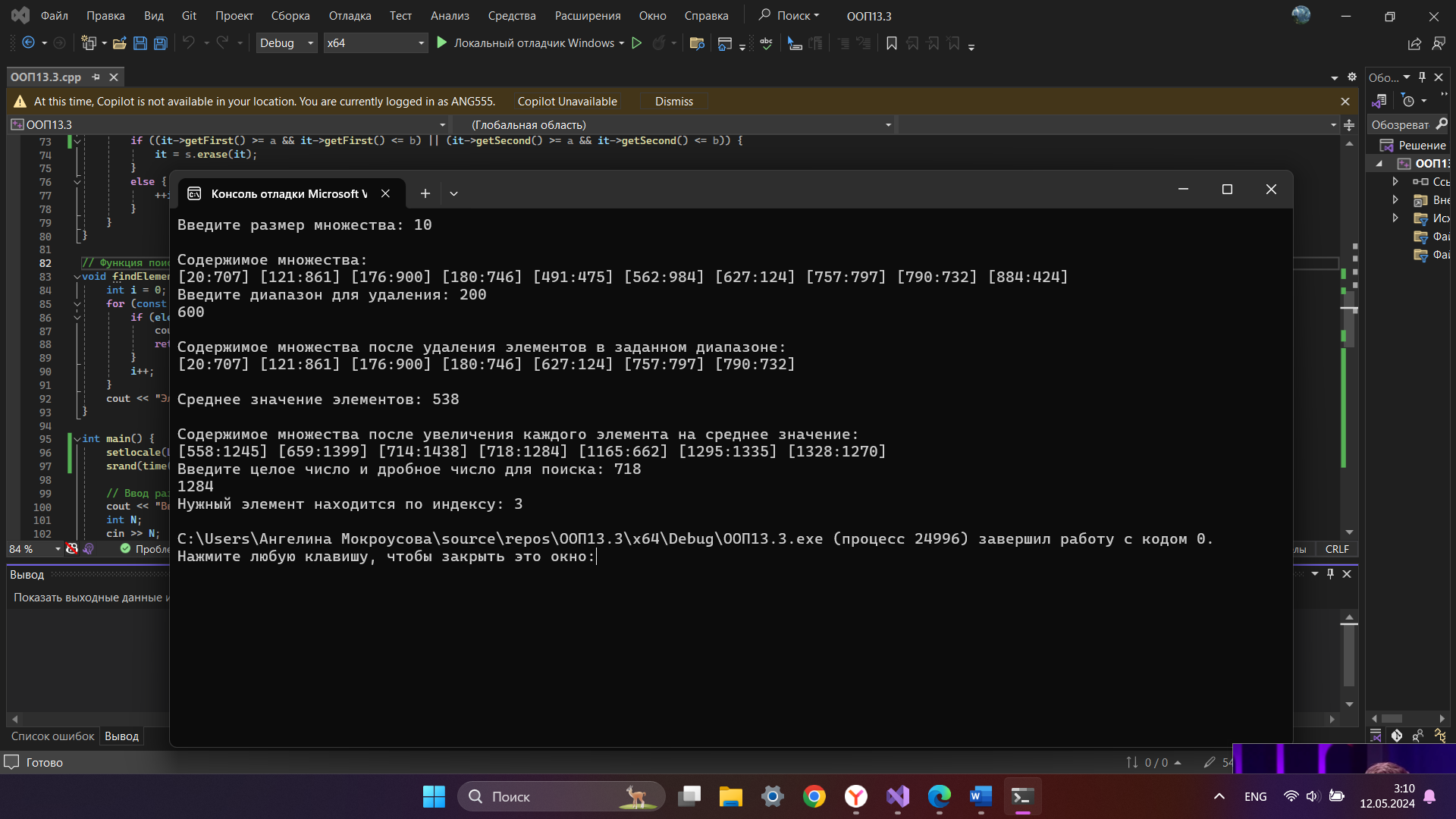
Задача 1:



Задача 2:



Задача 3:



**Вывод:**

Мною была аписана программа, которая успешно выполняет поставленную перед ней задачу. В процессе написания был повторен и закреплен на практике пройденный ранее материал.

**Скриншот с GitHab:**

