Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №18.11**

Дисциплина: Основы теории алгоритмов и структуры данных.

Тема: “ Объектно-ориентированное программирование.

Последовательные контейнеры библиотеки STL”

Вариант.12

Выполнила работу:

студент группы ИВТ-20-2Б

Ананина Арина Юрьевна

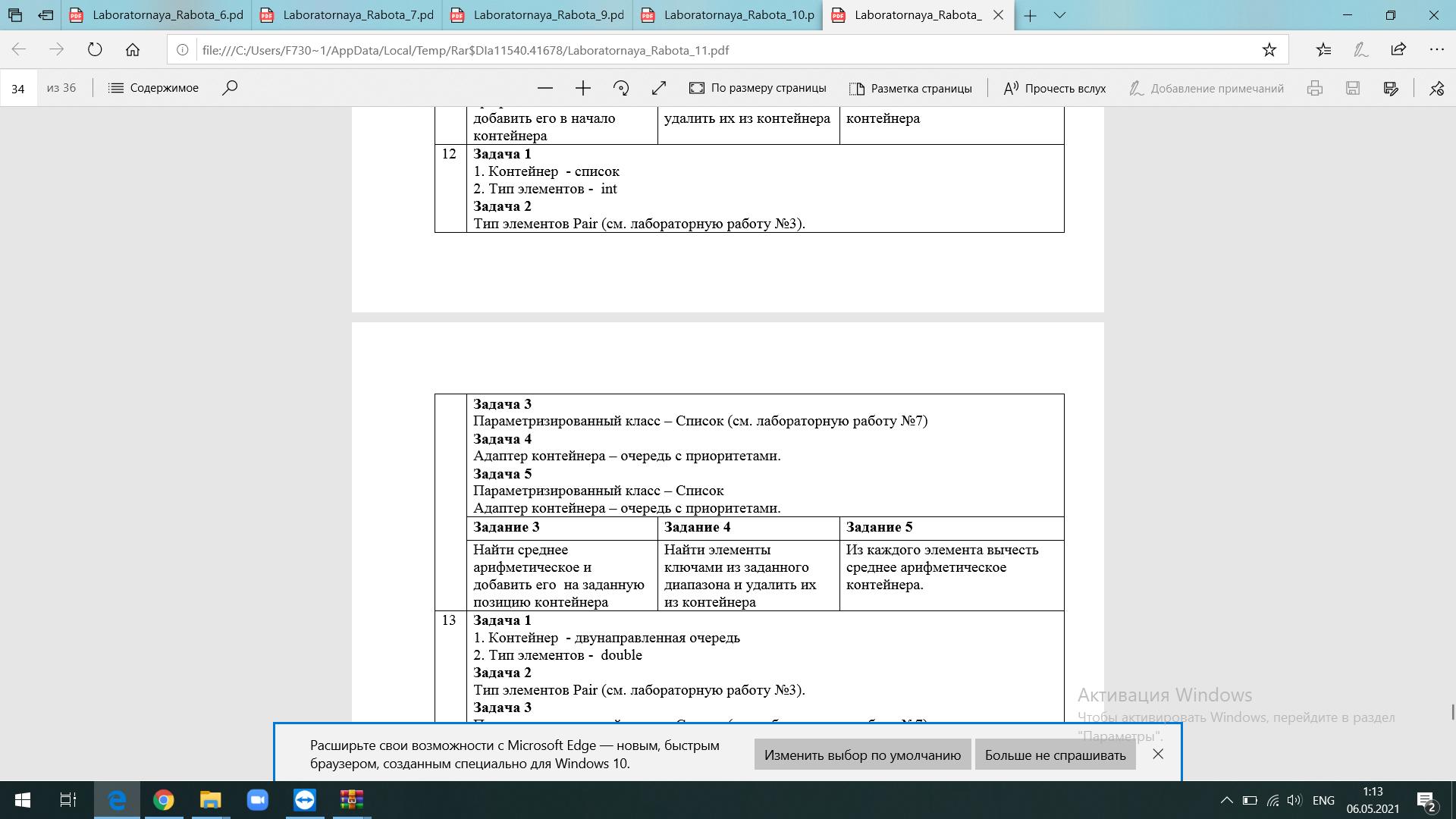
Проверила:

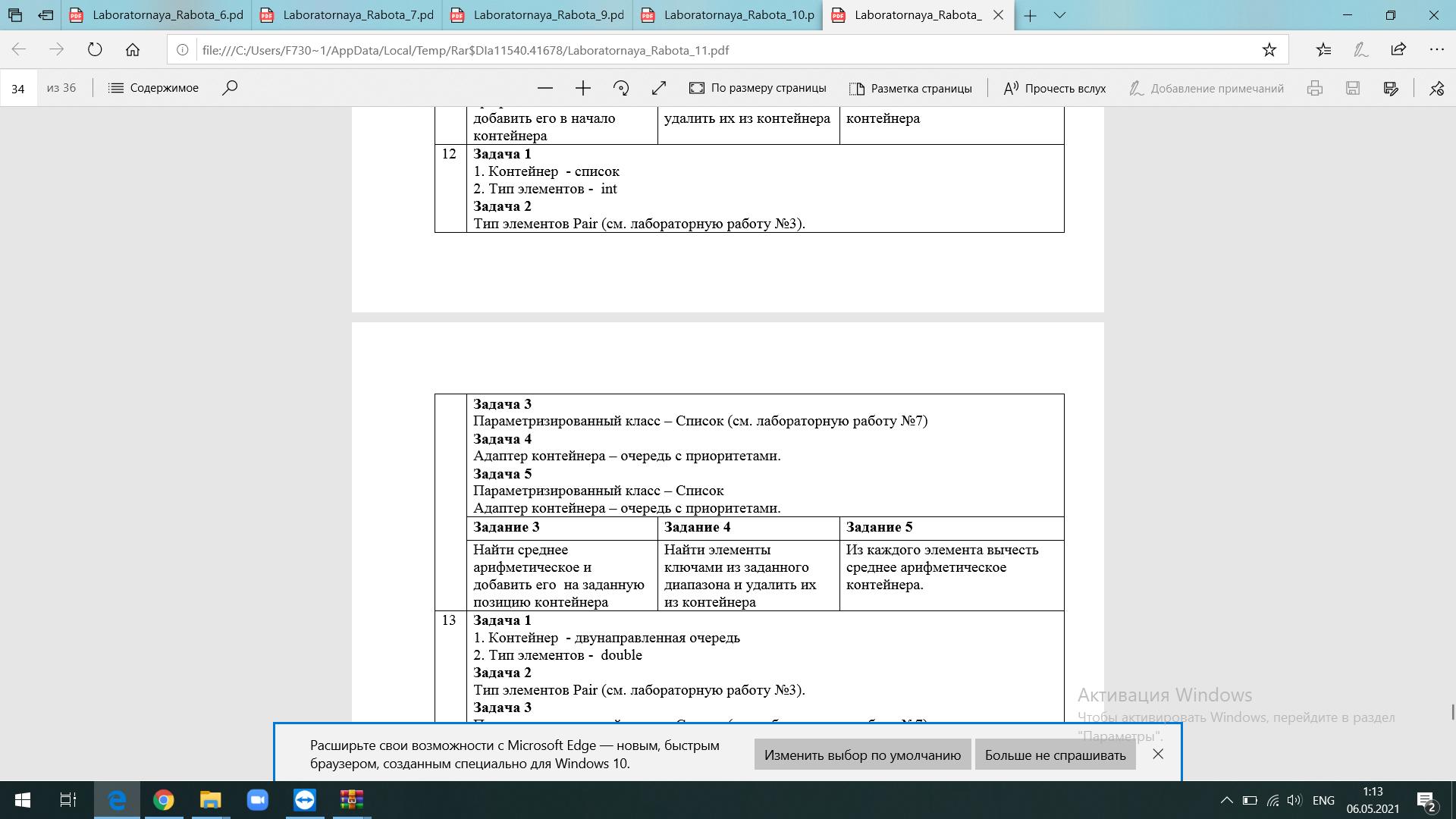
Доцент кафедры ИТАС

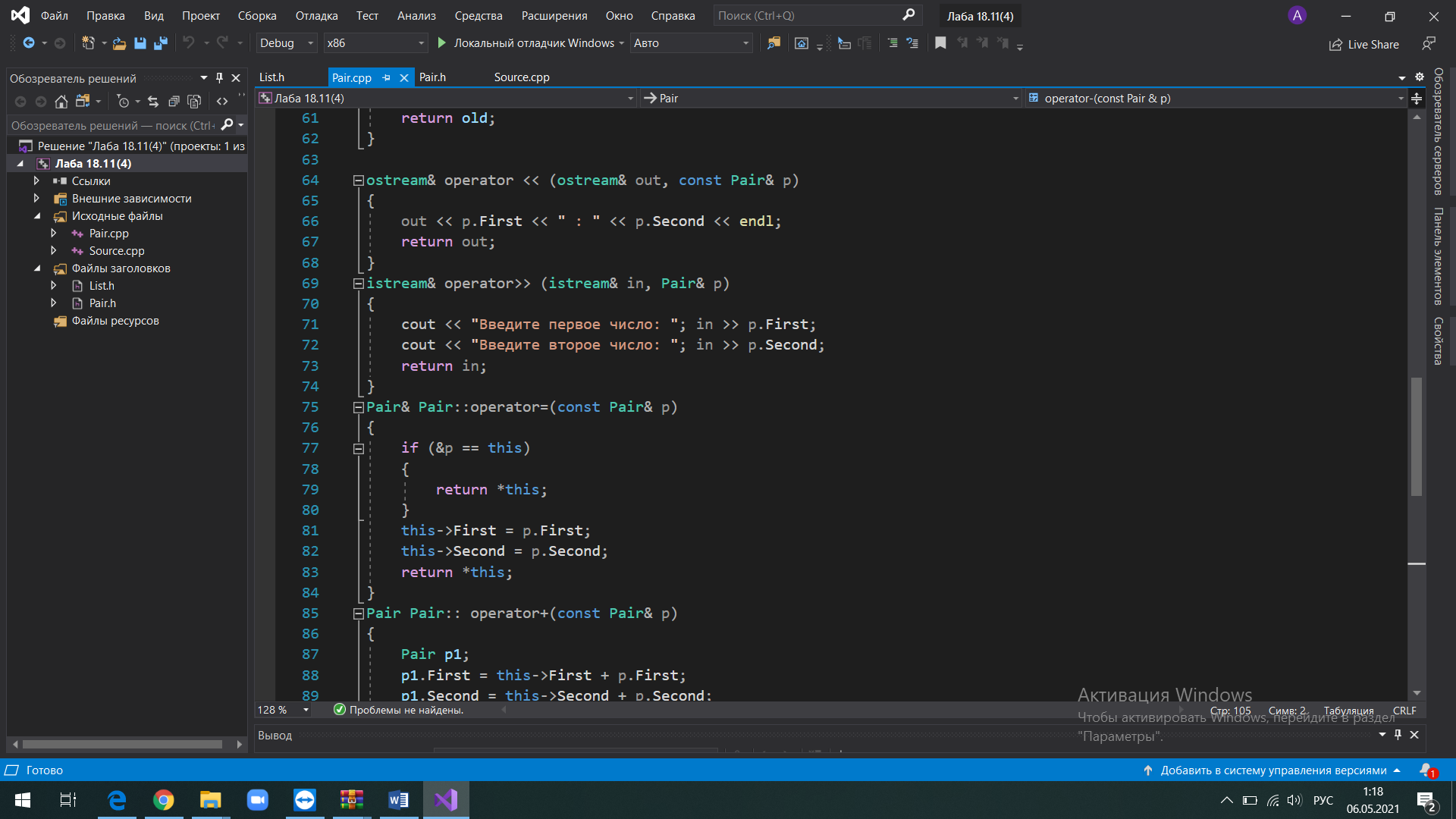
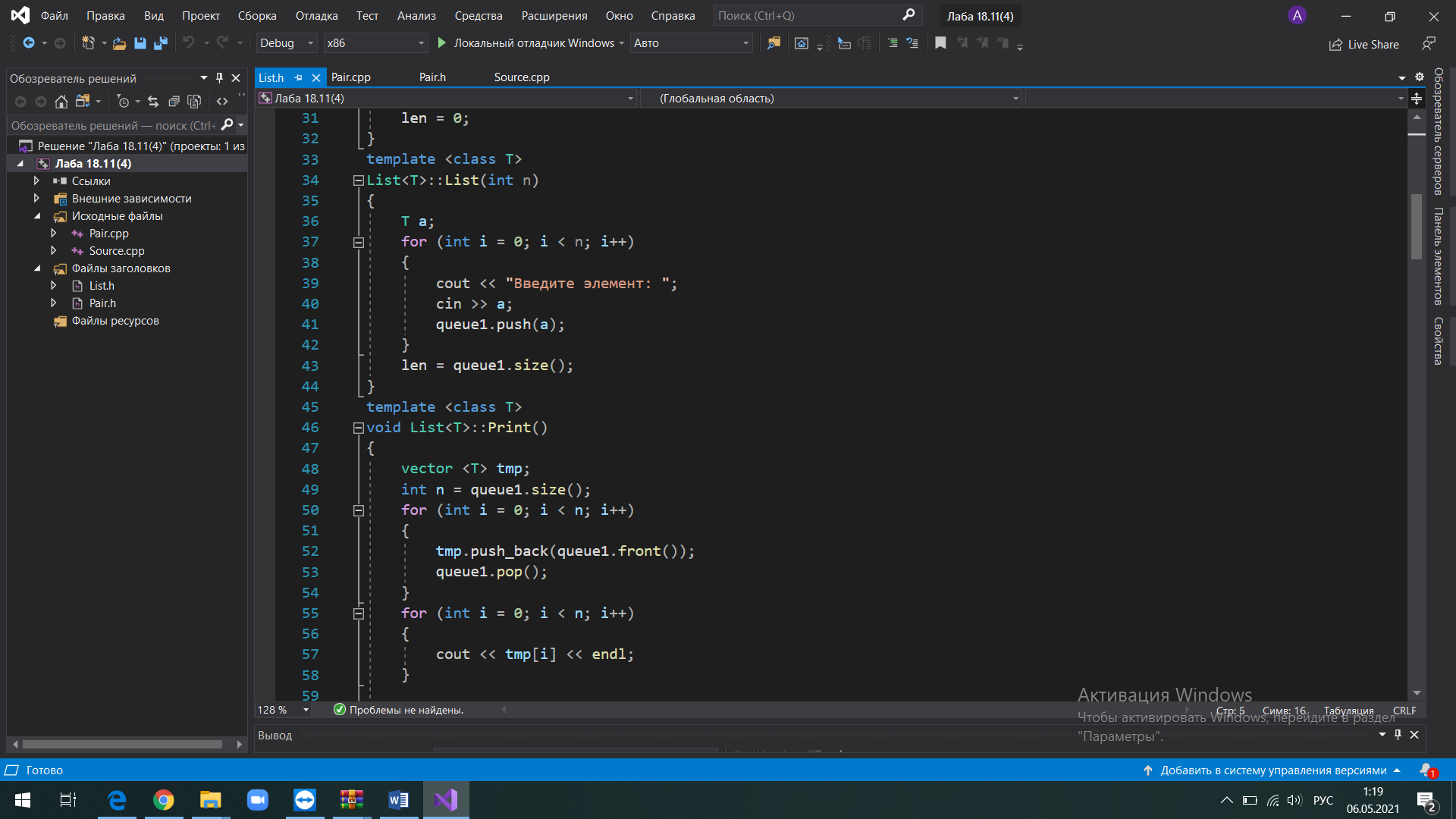
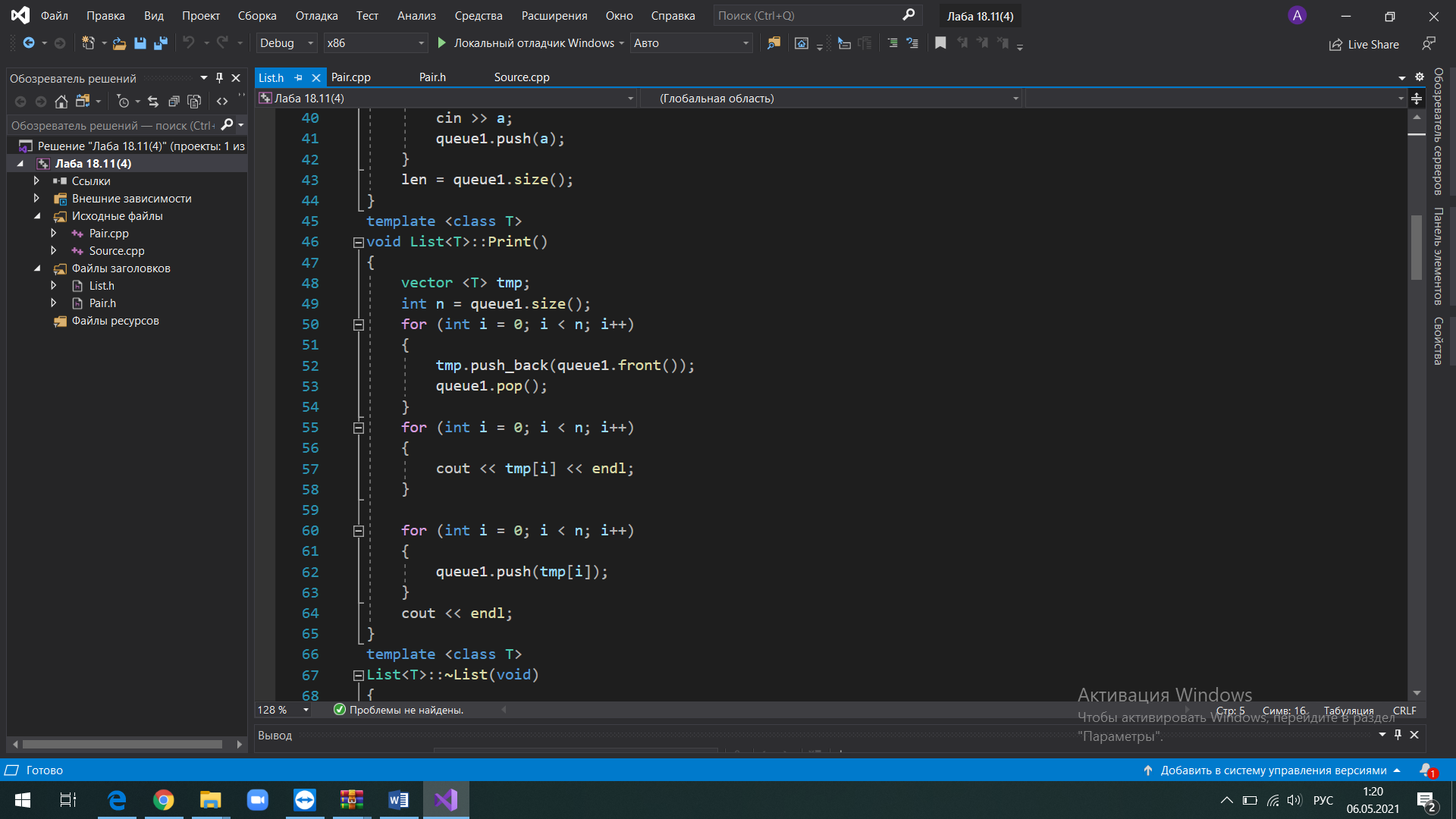
Полякова О.А.

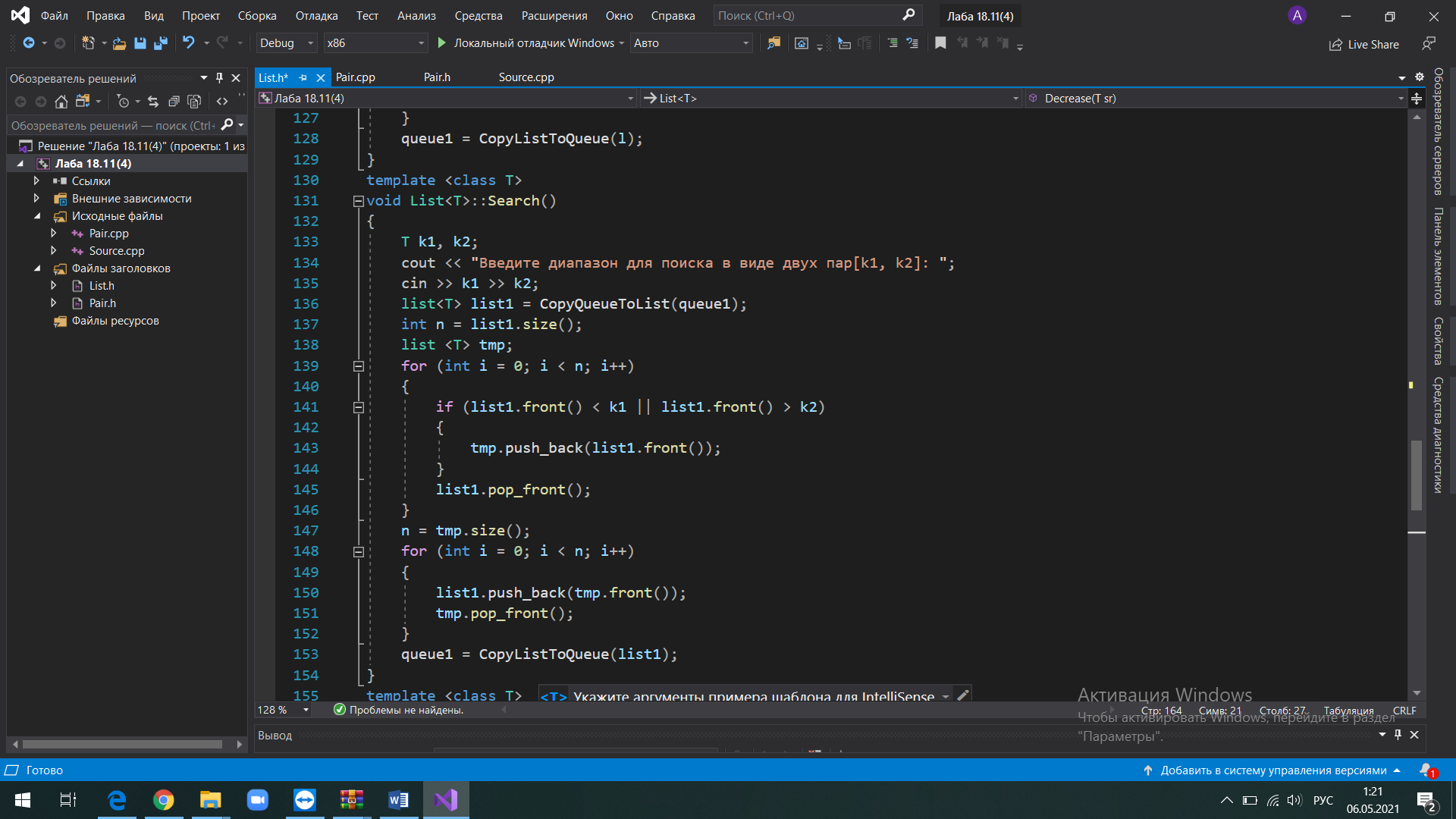
Пермь, 2021

1. **Постановка задачи**

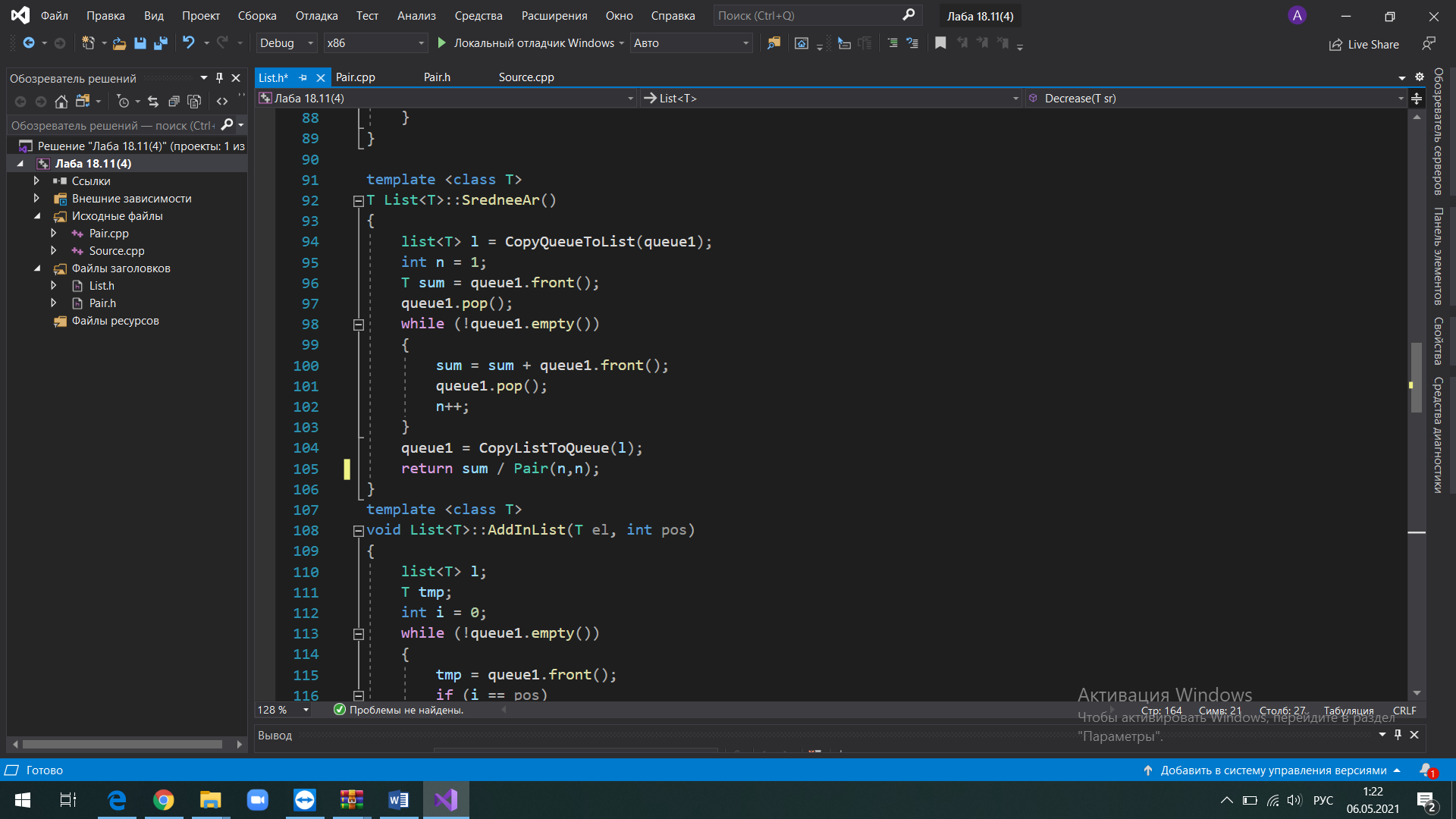




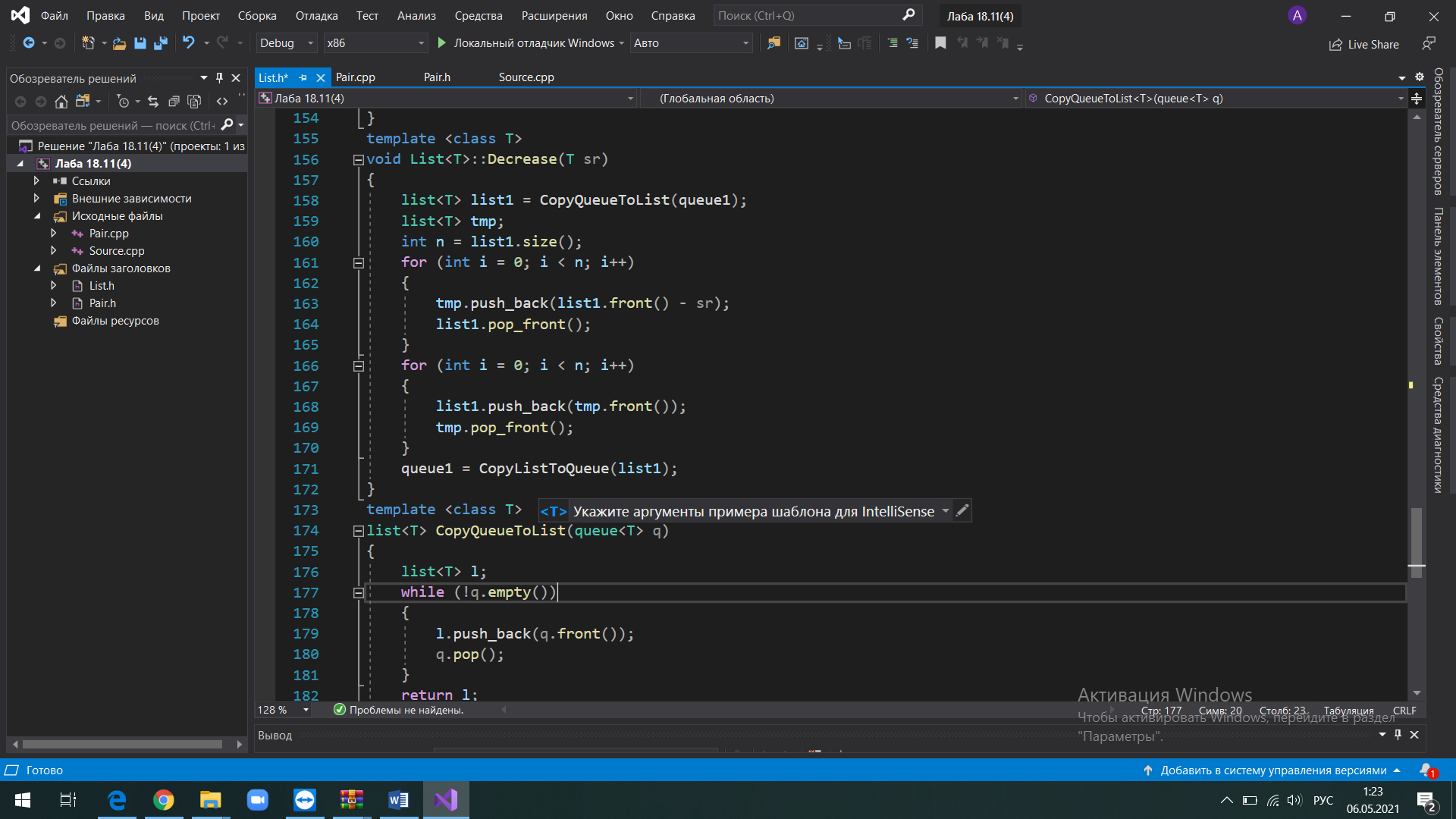
1. **Анализ задачи**
2. Для решения задачи необходимо:
   1. Использовать класс Pair.
   2. Использовать контейнеры библиотеки STL.
3. В ходе работы были использованы следующие типы данных:
   1. Pair, double, int.
4. Для решения задачи данные были представлены в следующем виде:
   1. 
   2. 
   3. 
   4. 
5. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Операторы >> и <<.  
      
6. Поставленные задачи будут решены следующими действиями:
   1. Шаблонная функция для заполнения списка  
      
   2. Шаблонные функции заполнения и вывода очереди с приоритетами:  
      
   3. Функция удаления по ключу в диапозоне:



* 1. Функции нахождения и добавления среднего арифметического на заданную позицию:



* 1. Функция уменьшения всех элементов на заданное число:



1. **Код программы**

List.h:

#pragma once

#include <iostream>

#include <vector>

#include <queue>

#include <list>

using namespace std;

template <class T>

class List

{

queue<T> queue1;

int len;

public:

List(void);

List(int n);

void Print();

~List(void);

List(const List<T>& li);

//3

T SredneeAr();

void AddInList(T el, int pos);

//4

void Search();

//5

void Decrease(T sr);

};

template <class T>

List<T>::List()

{

len = 0;

}

template <class T>

List<T>::List(int n)

{

T a;

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

{

cout << "Введите элемент: ";

cin >> a;

queue1.push(a);

}

len = queue1.size();

}

template <class T>

void List<T>::Print()

{

vector <T> tmp;

int n = queue1.size();

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

{

tmp.push\_back(queue1.front());

queue1.pop();

}

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

{

cout << tmp[i] << endl;

}

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

{

queue1.push(tmp[i]);

}

cout << endl;

}

template <class T>

List<T>::~List(void)

{

}

template <class T>

List<T>::List(const List<T>& li)

{

vector <T> tmp;

int n = li.size();

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

{

tmp.push\_back(li.front());

li.pop();

}

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

{

cout << tmp[i] << endl;

}

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

{

queue1.push(tmp[i]);

}

}

template <class T>

T List<T>::SredneeAr()

{

list<T> l = CopyQueueToList(queue1);

int n = 1;

T sum = queue1.front();

queue1.pop();

while (!queue1.empty())

{

sum = sum + queue1.front();

queue1.pop();

n++;

}

queue1 = CopyListToQueue(l);

return sum / Pair(n,n);

}

template <class T>

void List<T>::AddInList(T el, int pos)

{

list<T> l;

T tmp;

int i = 0;

while (!queue1.empty())

{

tmp = queue1.front();

if (i == pos)

{

l.push\_back(el);

}

l.push\_back(tmp);

queue1.pop();

i++;

}

if (pos >= l.size())

{

l.push\_back(el);

}

queue1 = CopyListToQueue(l);

}

template <class T>

void List<T>::Search()

{

T k1, k2;

cout << "Введите диапазон для поиска в виде двух пар[k1, k2]: ";

cin >> k1 >> k2;

list<T> list1 = CopyQueueToList(queue1);

int n = list1.size();

list <T> tmp;

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

{

if (list1.front() < k1 || list1.front() > k2)

{

tmp.push\_back(list1.front());

}

list1.pop\_front();

}

n = tmp.size();

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

{

list1.push\_back(tmp.front());

tmp.pop\_front();

}

queue1 = CopyListToQueue(list1);

}

template <class T>

void List<T>::Decrease(T sr)

{

list<T> list1 = CopyQueueToList(queue1);

list<T> tmp;

int n = list1.size();

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

{

tmp.push\_back(list1.front() - sr);

list1.pop\_front();

}

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

{

list1.push\_back(tmp.front());

tmp.pop\_front();

}

queue1 = CopyListToQueue(list1);

}

template <class T>

list<T> CopyQueueToList(queue<T> q)

{

list<T> l;

while (!q.empty())

{

l.push\_back(q.front());

q.pop();

}

return l;

}

template <class T>

queue<T> CopyListToQueue(list<T> l)

{

queue<T> q;

int n =l.size();

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

{

q.push(l.front());

l.pop\_front();

}

return q;

}

Main.cpp

#include <iostream>

#include <list>

#include "C:\Users\Арина\source\repos\Лаба 18.11(3)\Pair.h"

#include "List.h"

#include <vector>

#include <queue>

#include <list>

#include <cstdlib>

using namespace std;

typedef list<Pair> ListPair;

int main()

{

setlocale(LC\_ALL, "rus");

List <Pair> list(3);

list.Print();

Pair sr = list.SredneeAr();

cout << sr << endl;

cout << "На какую позицию поставить среднее арифметическое ?" << endl;

int p;

cin >> p;

list.AddInList(sr, p);

list.Print();

list.Search();

list.Print();

List<Pair> list3(4);

sr = list3.SredneeAr();

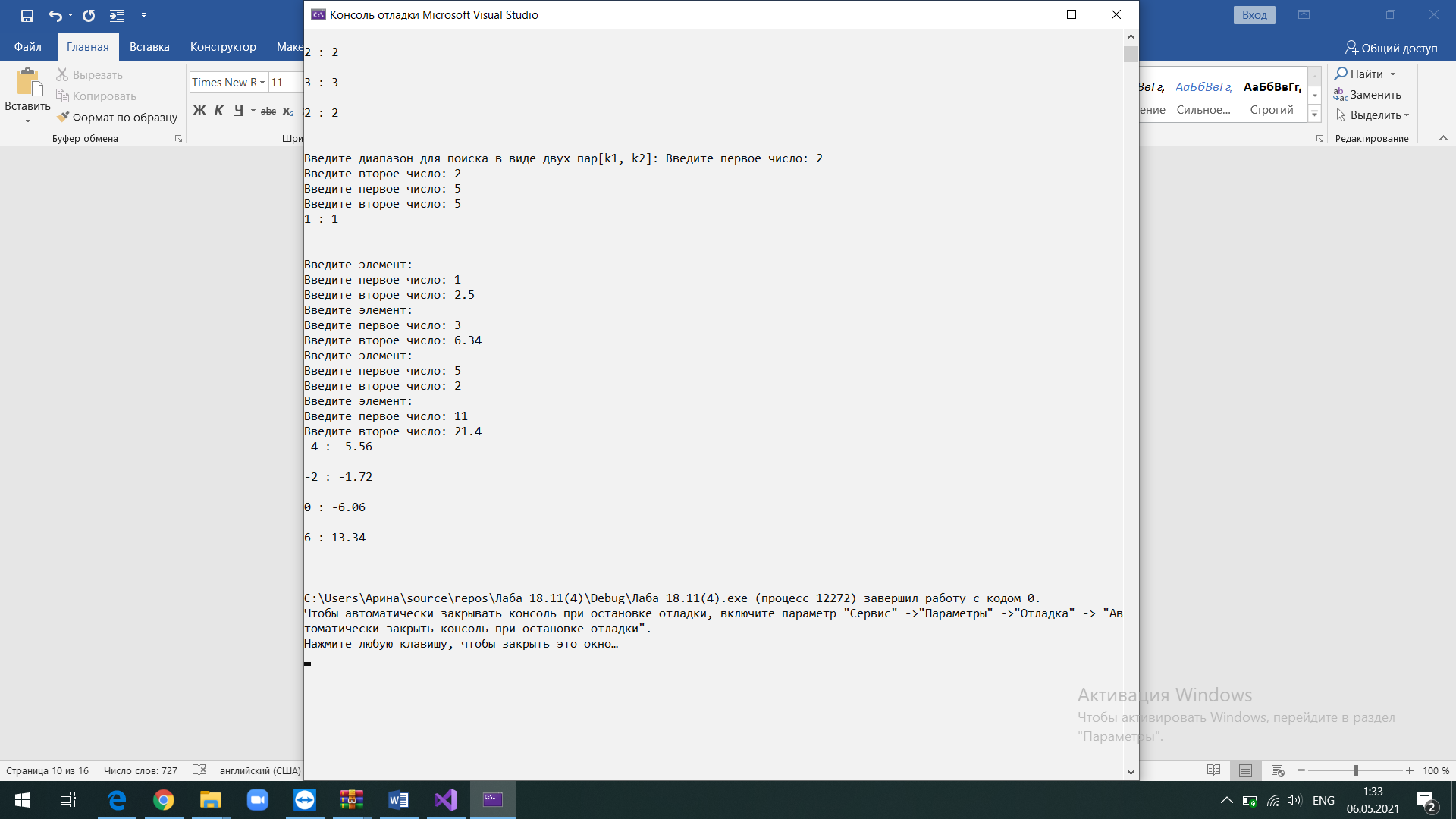
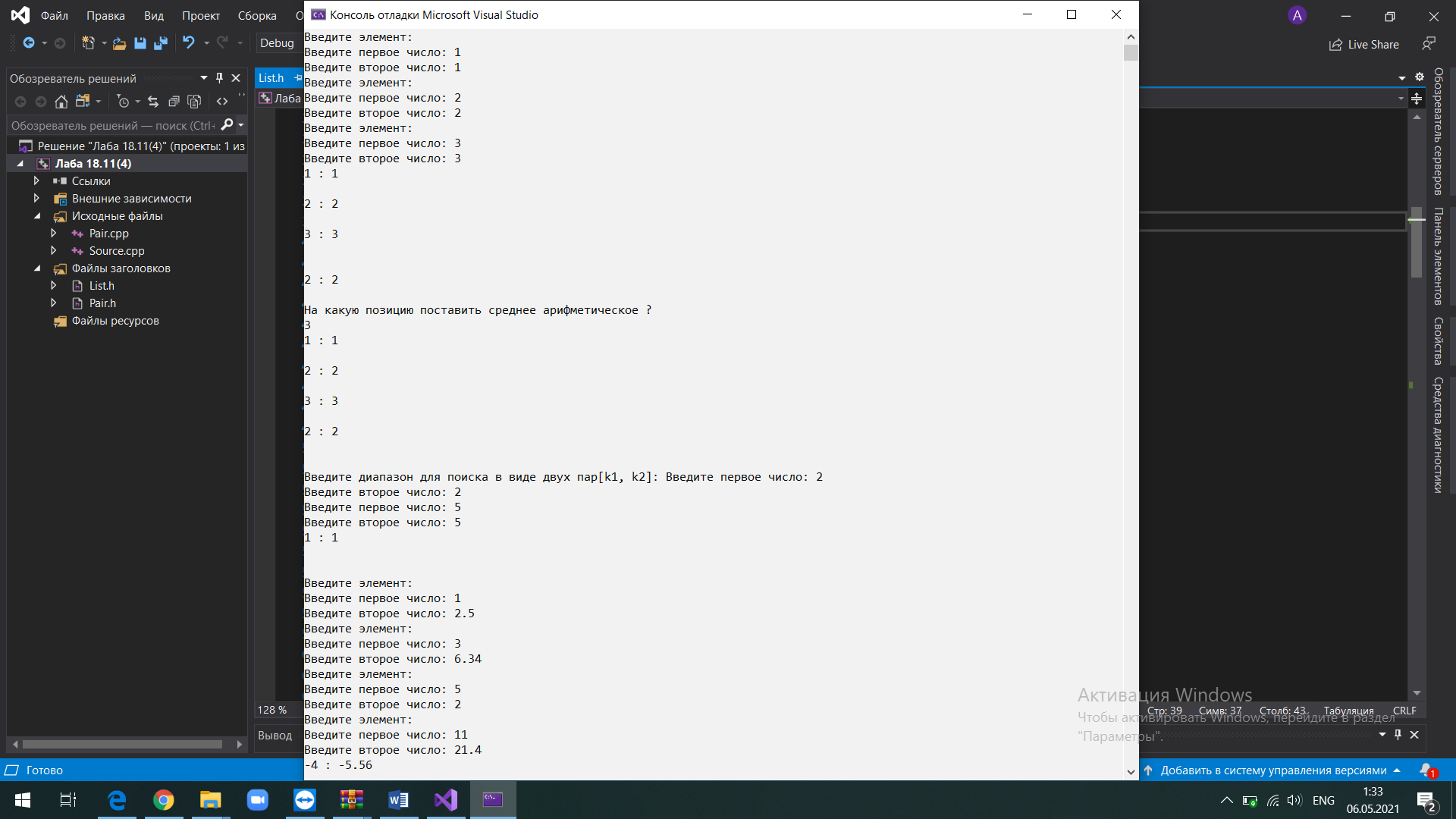
list3.Decrease(sr);

list3.Print();

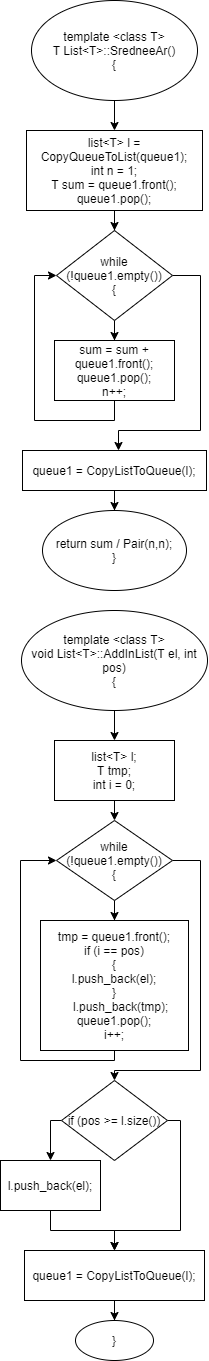
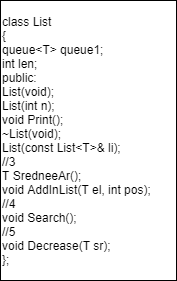
return 0;

}

1. **Выполнение программы**



1. **Блок-схема**



1. **Ответы на вопросы**

