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 №18.6**

Дисциплина: Основы теории алгоритмов и структуры данных.

Тема: “ Объектно-ориентированное программирование.

АДФ. Контейнеры.”

Вариант.12

Выполнила работу:

студент группы ИВТ-20- 2Б

Ананина Арина Юрьевна

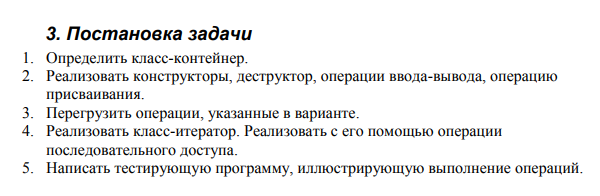
Проверила:

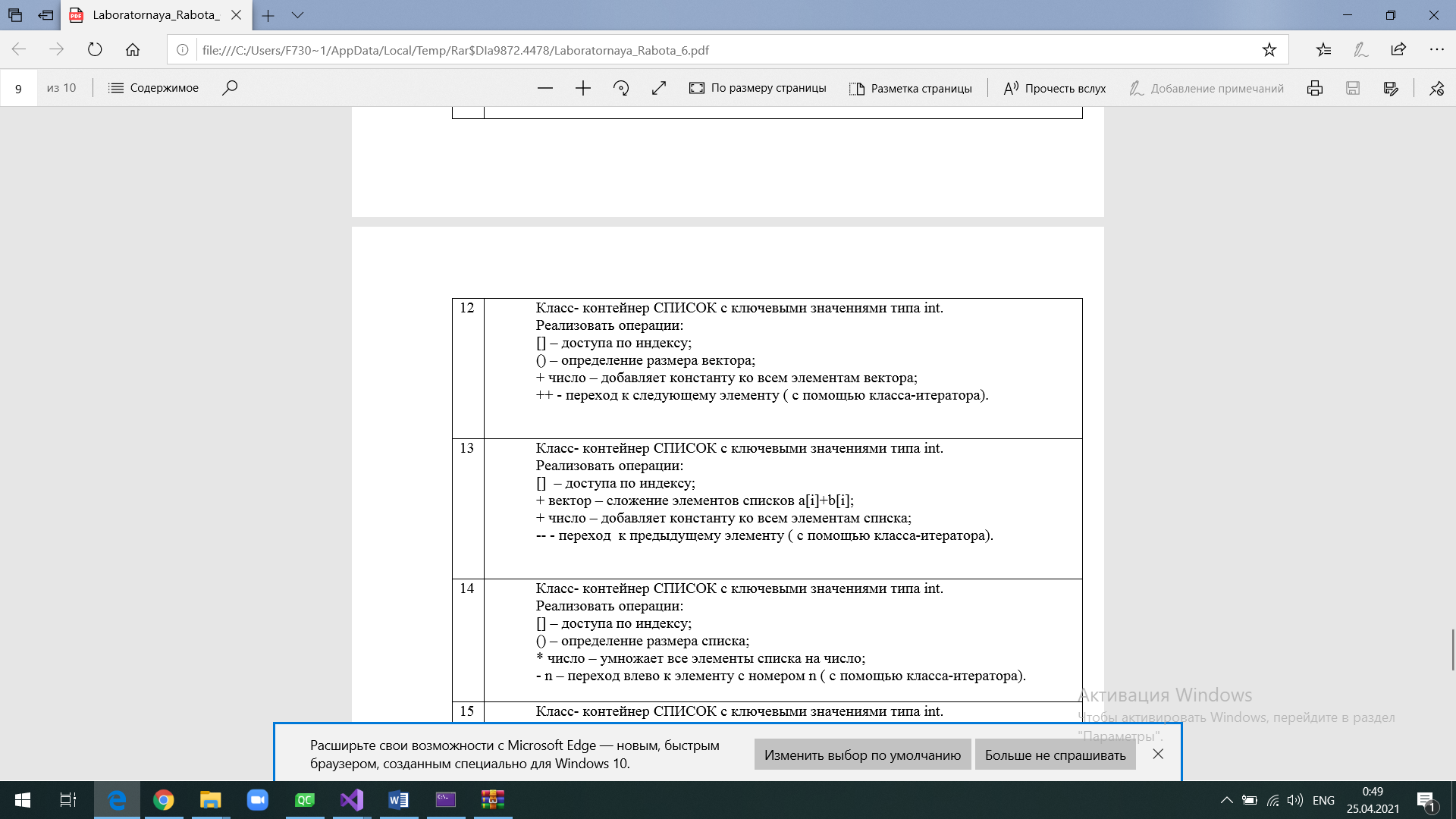
Доцент кафедры ИТАС

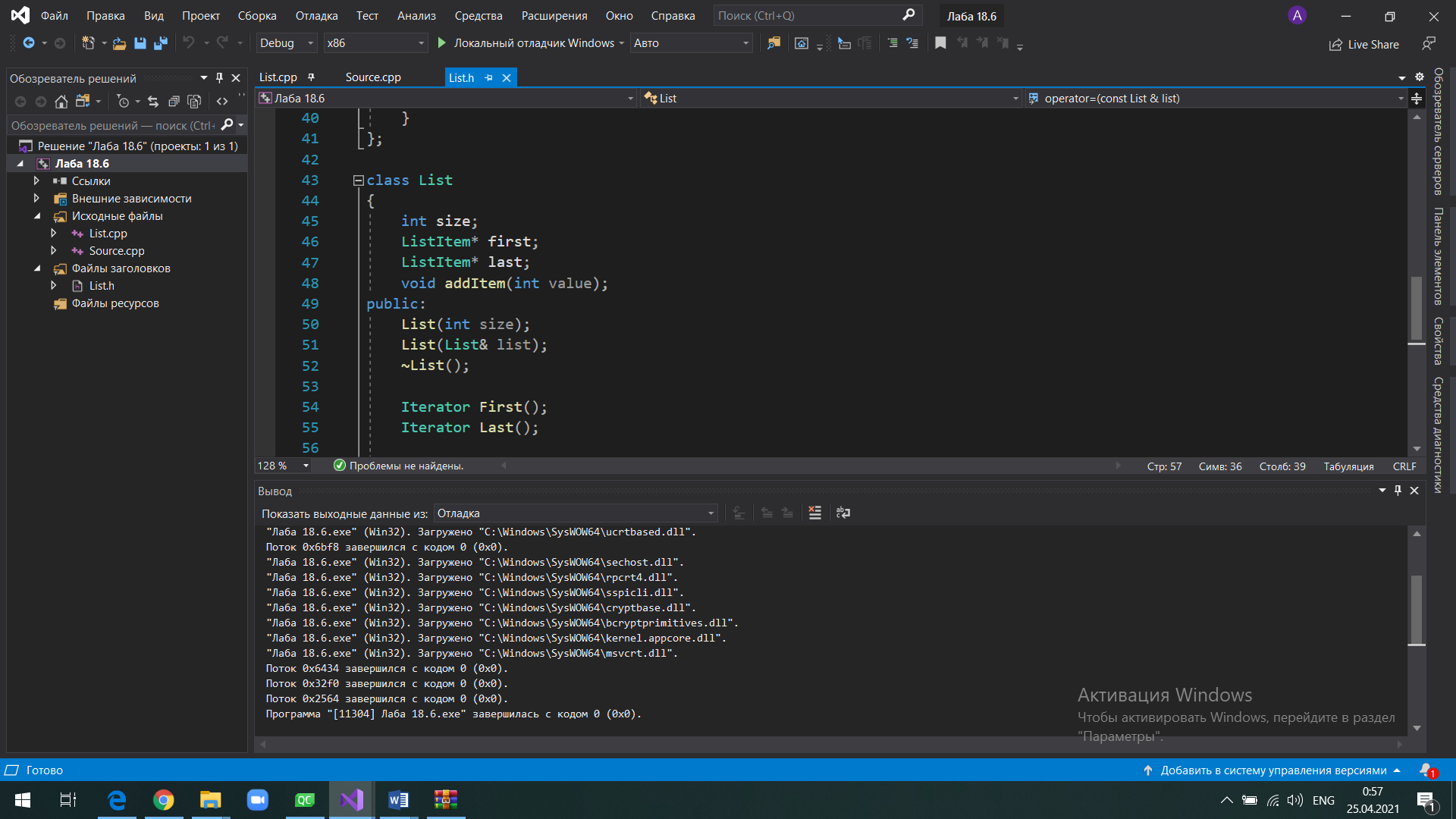
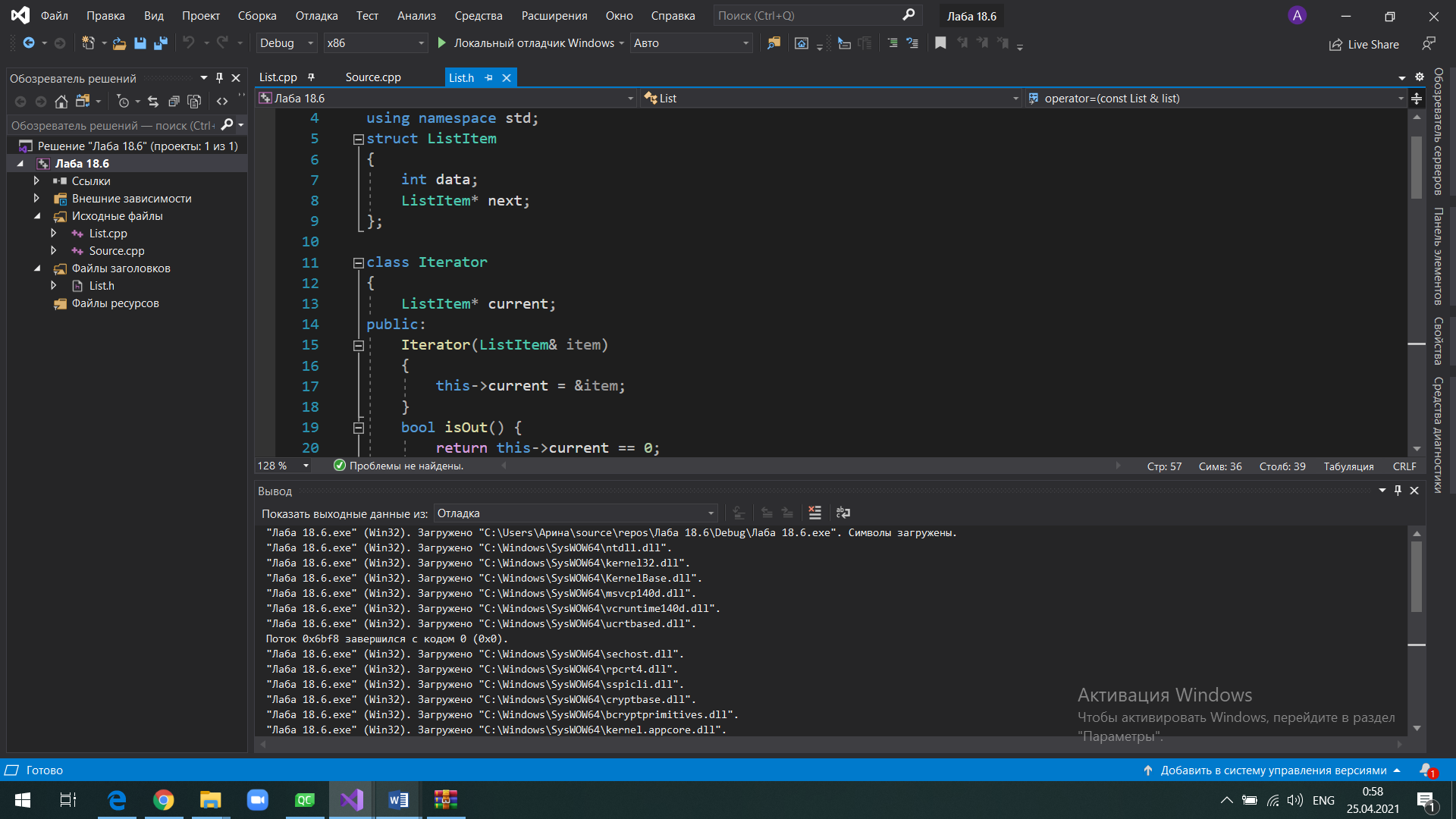
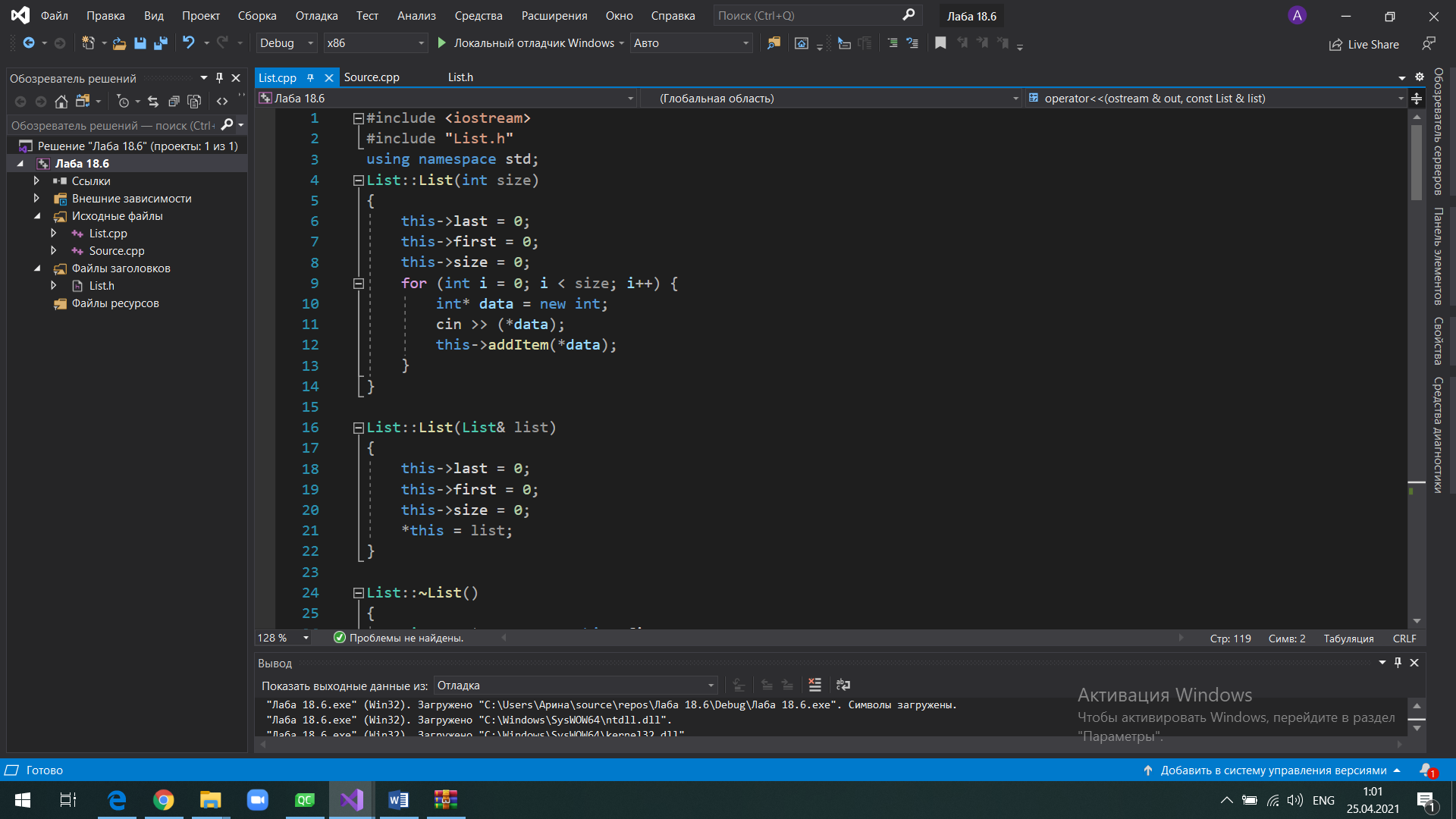
Полякова О.А.

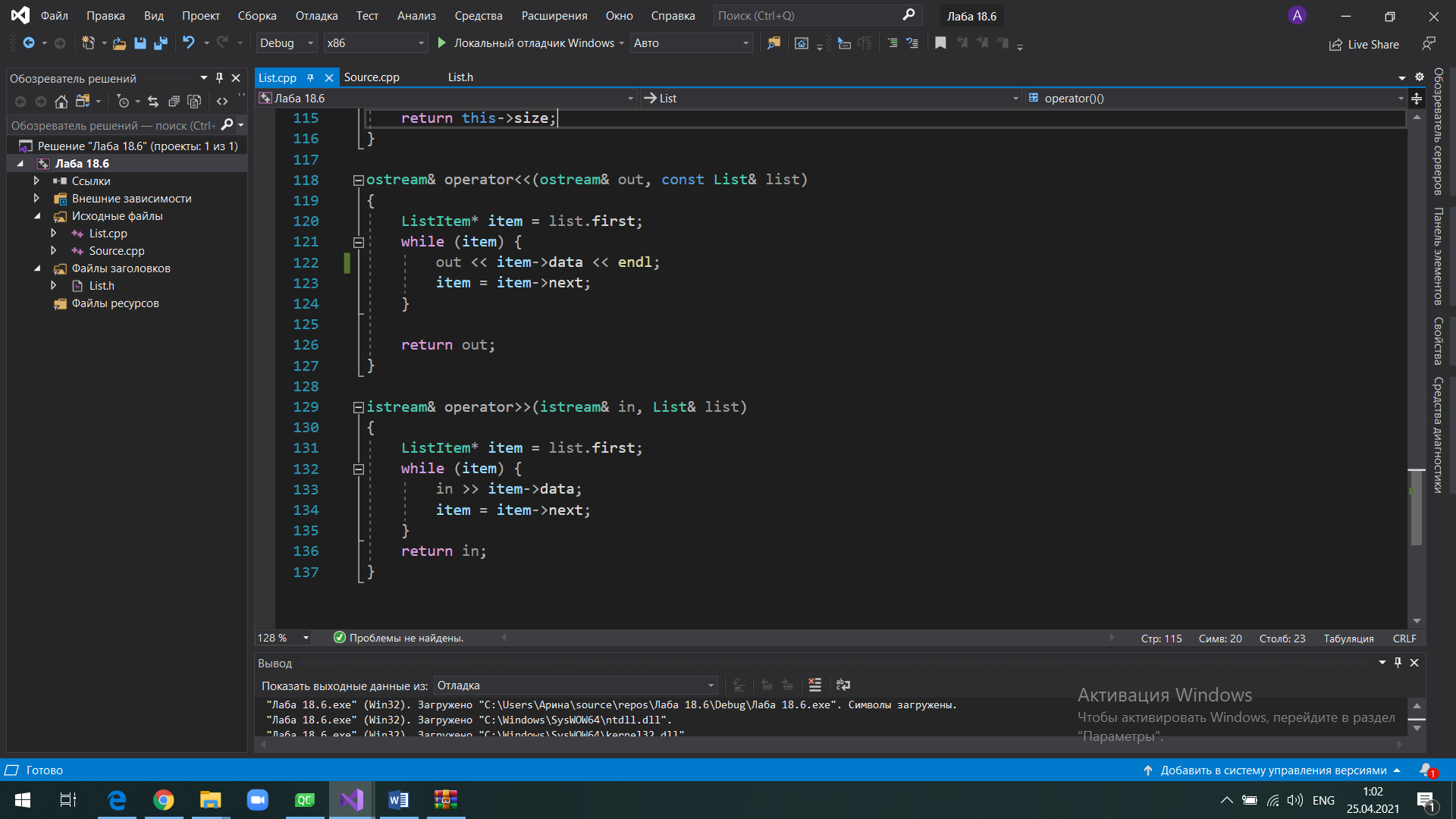
Пермь, 2021

1. **Постановка задачи**



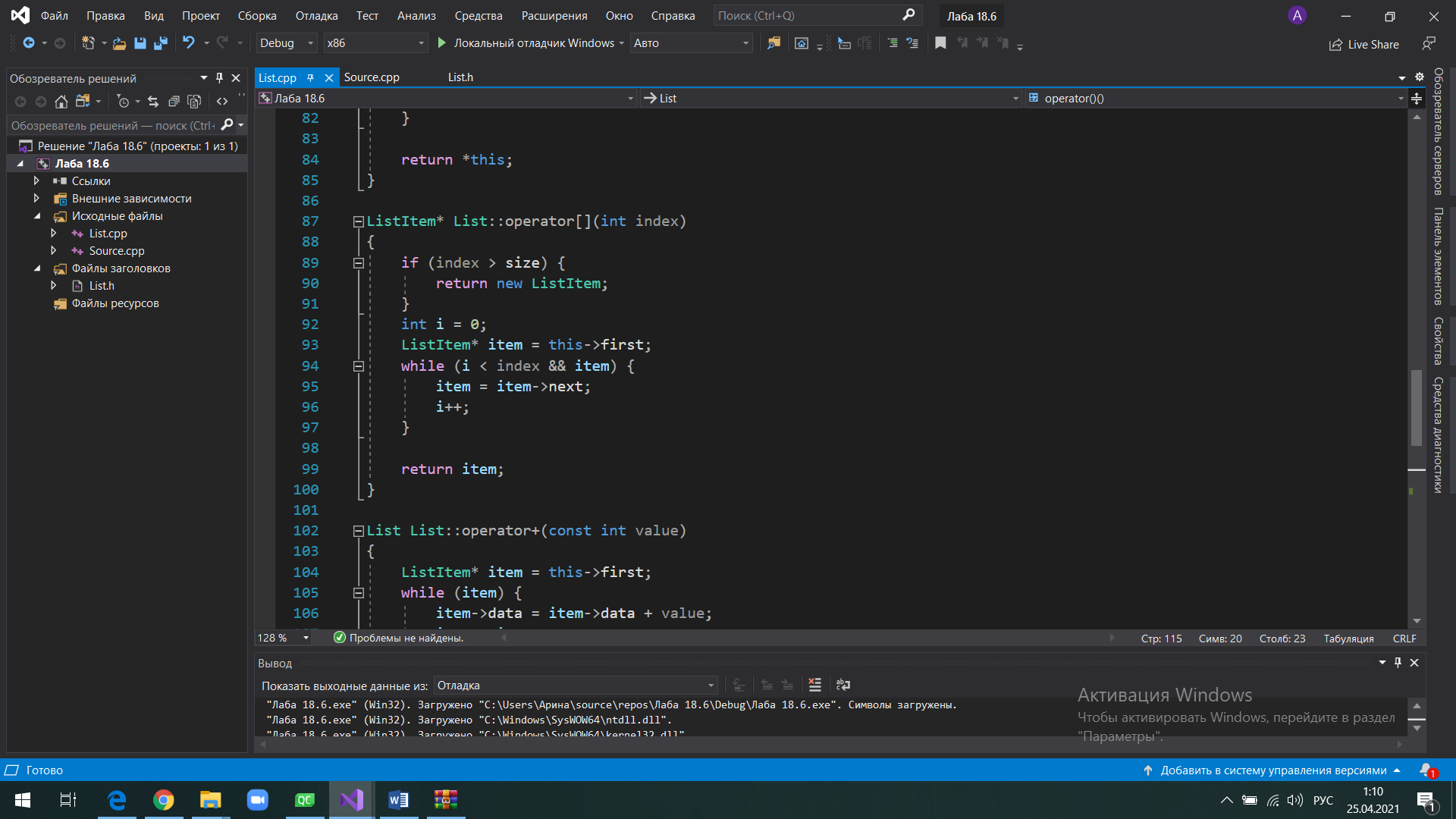


1. **Анализ задачи**
2. Для решения задачи необходимо:
   1. Организовать класс List.
   2. Организовать класс Iterator, дружественный классу List.
   3. Организовать перегрузку оператора [] для класса List.
   4. Организовать перегрузку оператора <<, дружественной классу List.
   5. Организовать перегрузку оператора ++ для класса Iterator.
   6. Организовать перегрузку оператора == для класса Iterator.
   7. Организовать перегрузку оператора != для класса Iterator.
   8. Организовать перегрузку оператора \* для класса Iterator.
3. В ходе работы были использованы следующие типы данных:
   1. Для класса list описаны следующие поля:  
      
   2. Для класса Iterator:  
        
      
4. Для решения задачи данные были представлены в следующем виде:
   1. Для инициализации объектов классов Iterator и List используются конструкторы с параметрами и по умолчанию:  
      
5. Для операций ввода и вывода использовались следующие операторы и функции:

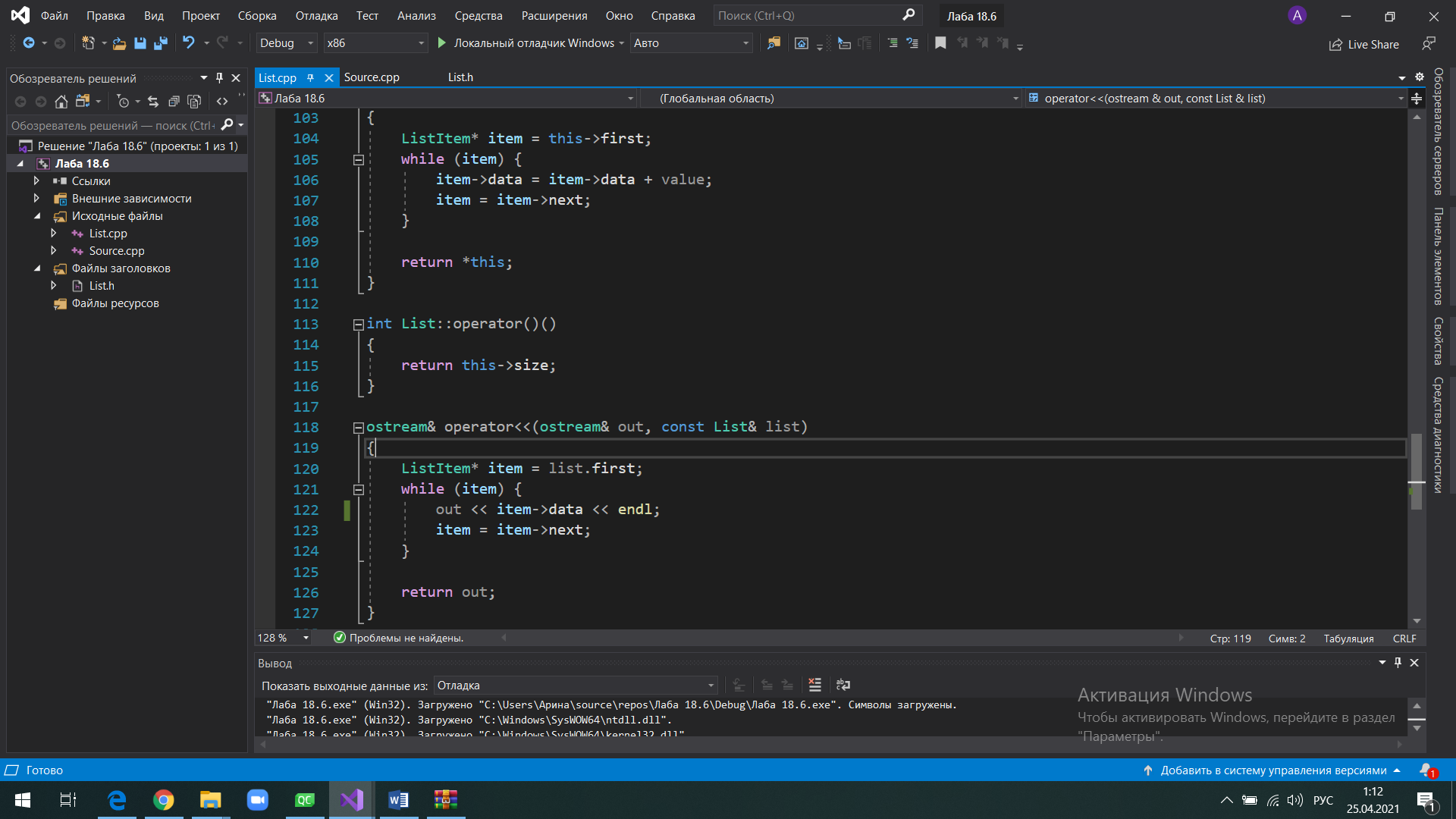


1. Поставленные задачи будут решены следующими действиями:

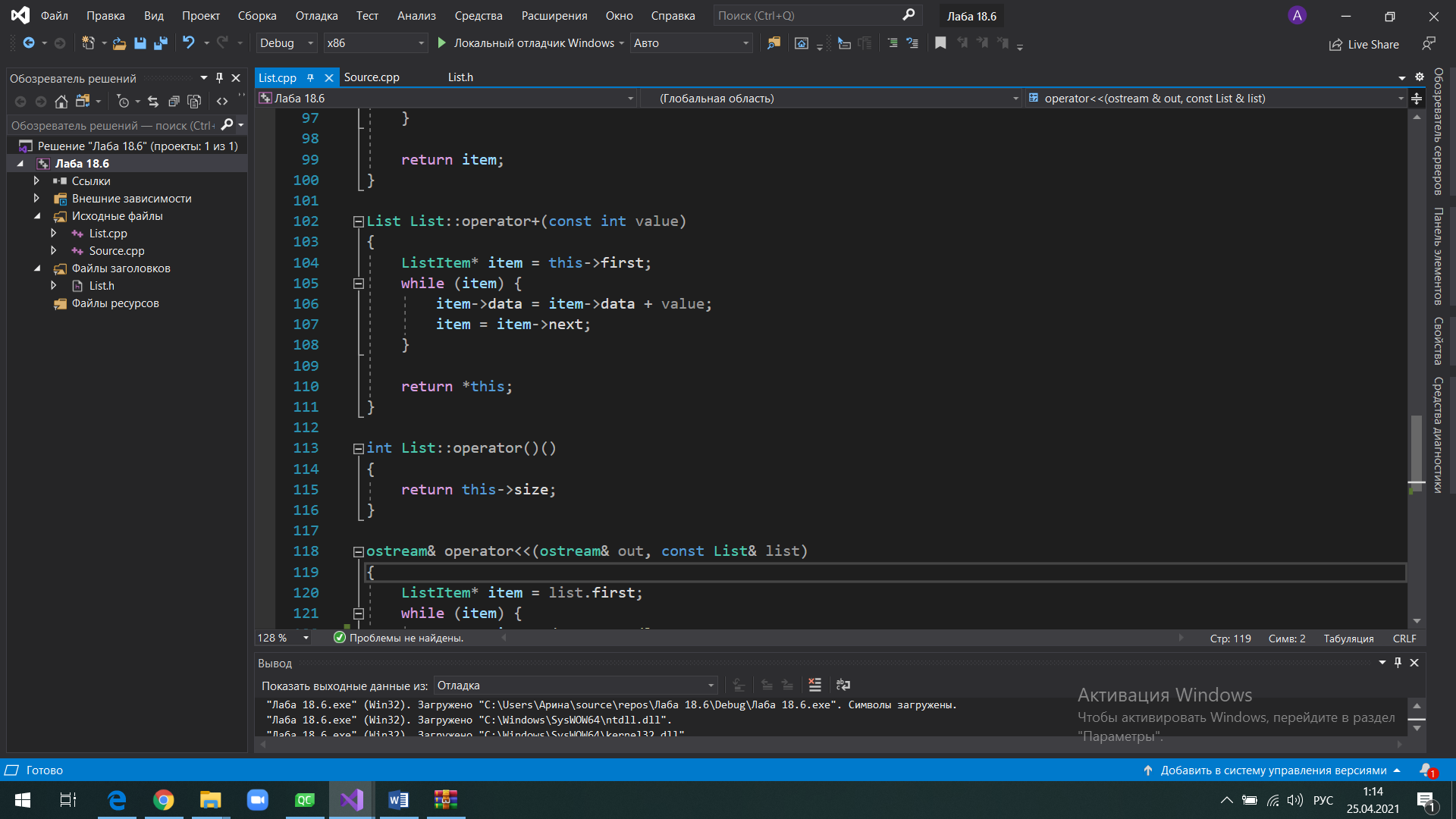
Для обращения по индексу организован оператор:



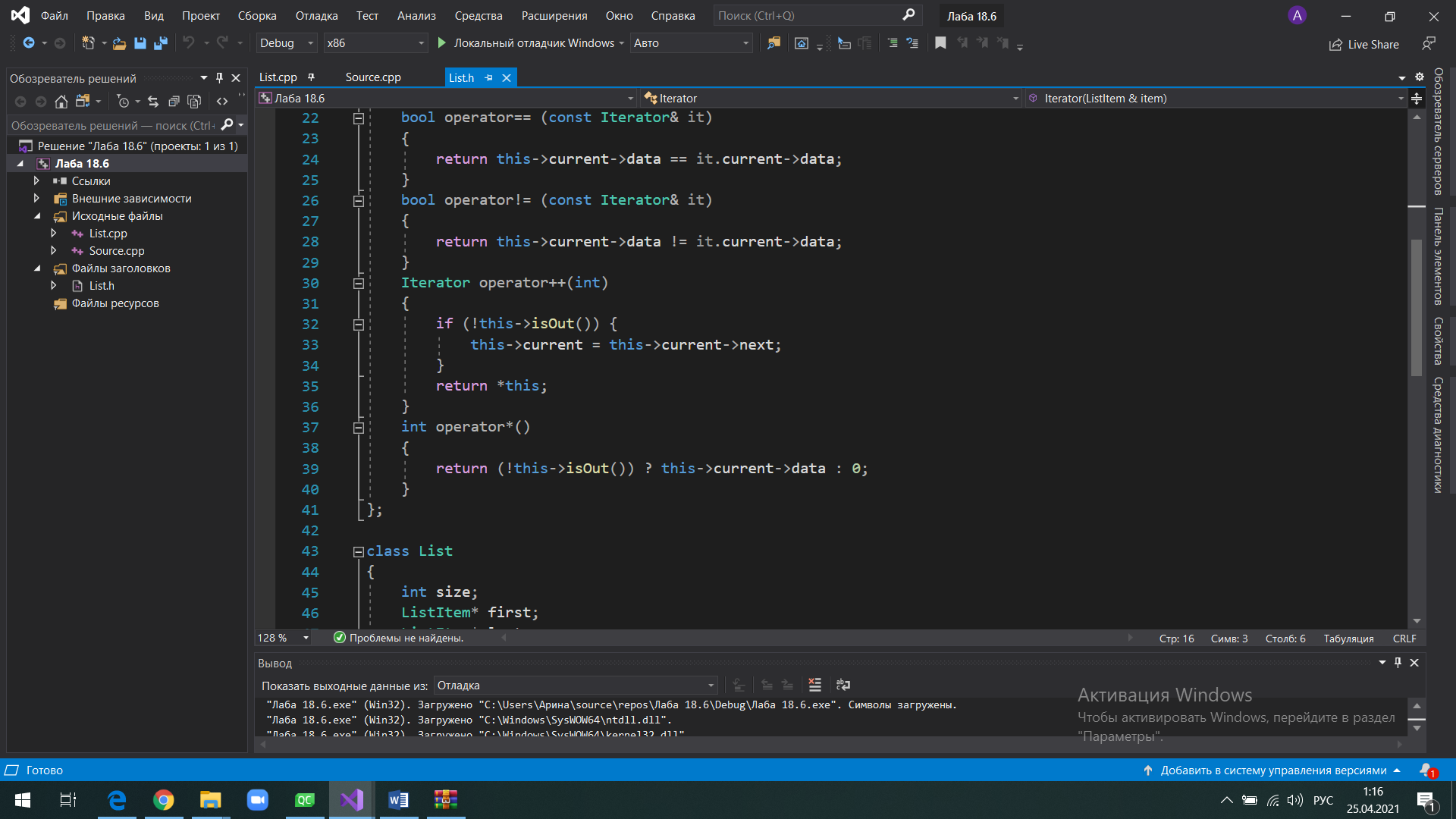
Для определения размера организован гетер размера:



Для добавления константы ко всем элементам списка организован оператор:



Для итератора ++ - переход к следующему элементу, в классе Iterator организован метод:



1. **Код программы**

**Файл Main.cpp:**

#include <iostream>

#include "List.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

List list\_a(5);

List list\_b = list\_a;

List list(list\_b);

cout << list << endl;

list + 5;

cout << "Item[2]: " << endl << list[2]->data << endl;

cout << "Size: " << list() << endl;

Iterator iter = list.First();

iter++;

iter++;

cout << \*iter << endl << endl;

for (Iterator i = list.First(); !i.isOut(); i++) {

cout << \*i << endl;

}

return 0;

}

**Файл List.h:**

#pragma once

#include <iostream>

using namespace std;

struct ListItem

{

int data;

ListItem\* next;

};

class Iterator

{

ListItem\* current;

public:

Iterator(ListItem& item)

{

this->current = &item;

}

bool isOut() {

return this->current == 0;

}

bool operator== (const Iterator& it)

{

return this->current->data == it.current->data;

}

bool operator!= (const Iterator& it)

{

return this->current->data != it.current->data;

}

Iterator operator++(int)

{

if (!this->isOut()) {

this->current = this->current->next;

}

return \*this;

}

int operator\*()

{

return (!this->isOut()) ? this->current->data : 0;

}

};

class List

{

int size;

ListItem\* first;

ListItem\* last;

void addItem(int value);

public:

List(int size);

List(List& list);

~List();

Iterator First();

Iterator Last();

List& operator=(const List& list);

ListItem\* operator[](int index);

List operator+(const int value);

int operator()();

friend ostream& operator<< (ostream& out, const List& list);

friend istream& operator>> (istream& in, List& list);

};

**Файл List.cpp:**

#include <iostream>

#include "List.h"

using namespace std;

List::List(int size)

{

this->last = 0;

this->first = 0;

this->size = 0;

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

int\* data = new int;

cin >> (\*data);

this->addItem(\*data);

}

}

List::List(List& list)

{

this->last = 0;

this->first = 0;

this->size = 0;

\*this = list;

}

List::~List()

{

ListItem\* current = this->first;

while (current) {

ListItem\* rem = current;

current = current->next;

delete rem;

}

}

Iterator List::First() {

return Iterator(\*first);

}

Iterator List::Last() {

return Iterator(\*last);

}

void List::addItem(int value)

{

ListItem\* item = new ListItem;

item->data = value;

item->next = 0;

if (!this->first) {

this->first = item;

}

if (this->last) {

this->last->next = item;

this->last = item;

}

else {

this->last = item;

}

this->size++;

}

List& List::operator=(const List& list)

{

if (this == &list)

{

return \*this;

}

// Очистка

if (this->size) {

this->~List();

this->size = 0;

this->last = 0;

this->first = 0;

}

// Наполнение

ListItem\* item = list.first;

while (item) {

this->addItem(item->data);

item = item->next;

}

return \*this;

}

ListItem\* List::operator[](int index)

{

if (index > size) {

return new ListItem;

}

int i = 0;

ListItem\* item = this->first;

while (i < index && item) {

item = item->next;

i++;

}

return item;

}

List List::operator+(const int value)

{

ListItem\* item = this->first;

while (item) {

item->data = item->data + value;

item = item->next;

}

return \*this;

}

int List::operator()()

{

return this->size;

}

ostream& operator<<(ostream& out, const List& list)

{

ListItem\* item = list.first;

while (item) {

out << item->data;

item = item->next;

}

return out;

}

istream& operator>>(istream& in, List& list)

{

ListItem\* item = list.first;

while (item) {

in >> item->data;

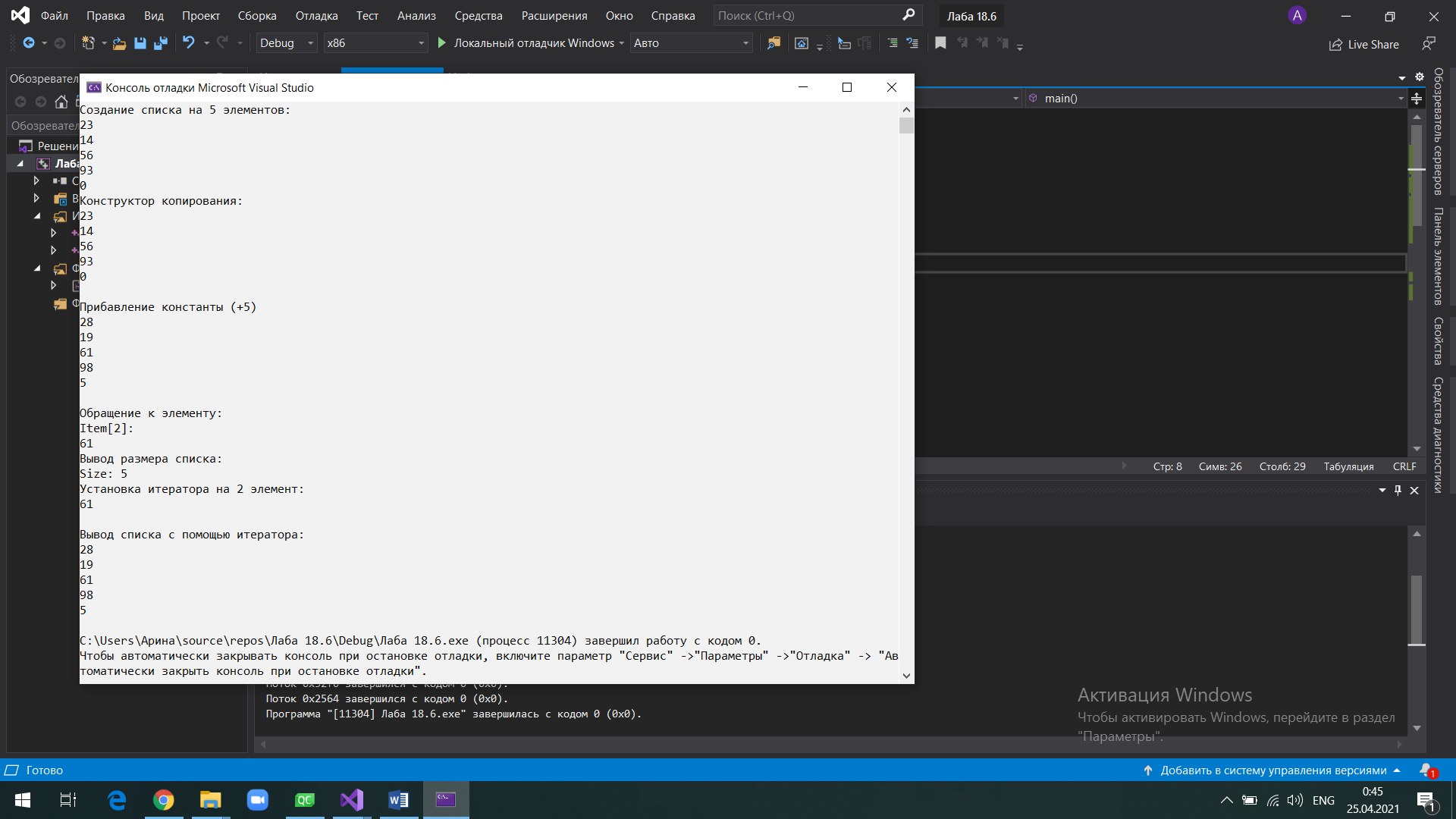
item = item->next;

}

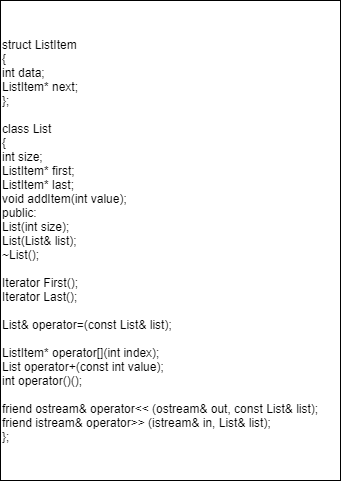
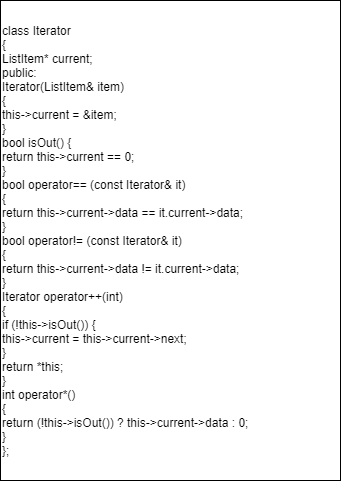
return in;

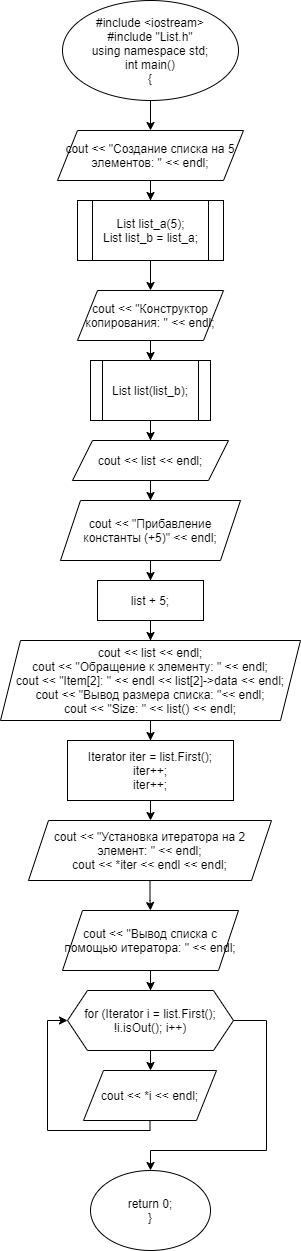
}

1. **Выполнение программы**



1. **Блок-схема**





1. **Ответы на вопросы**

