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 №18.8**

Дисциплина: Основы теории алгоритмов и структуры данных.

Тема: “ Программа, управляемая событиями”

Вариант.12

Выполнила работу:

студент группы ИВТ-20-2Б

Ананина Арина Юрьевна

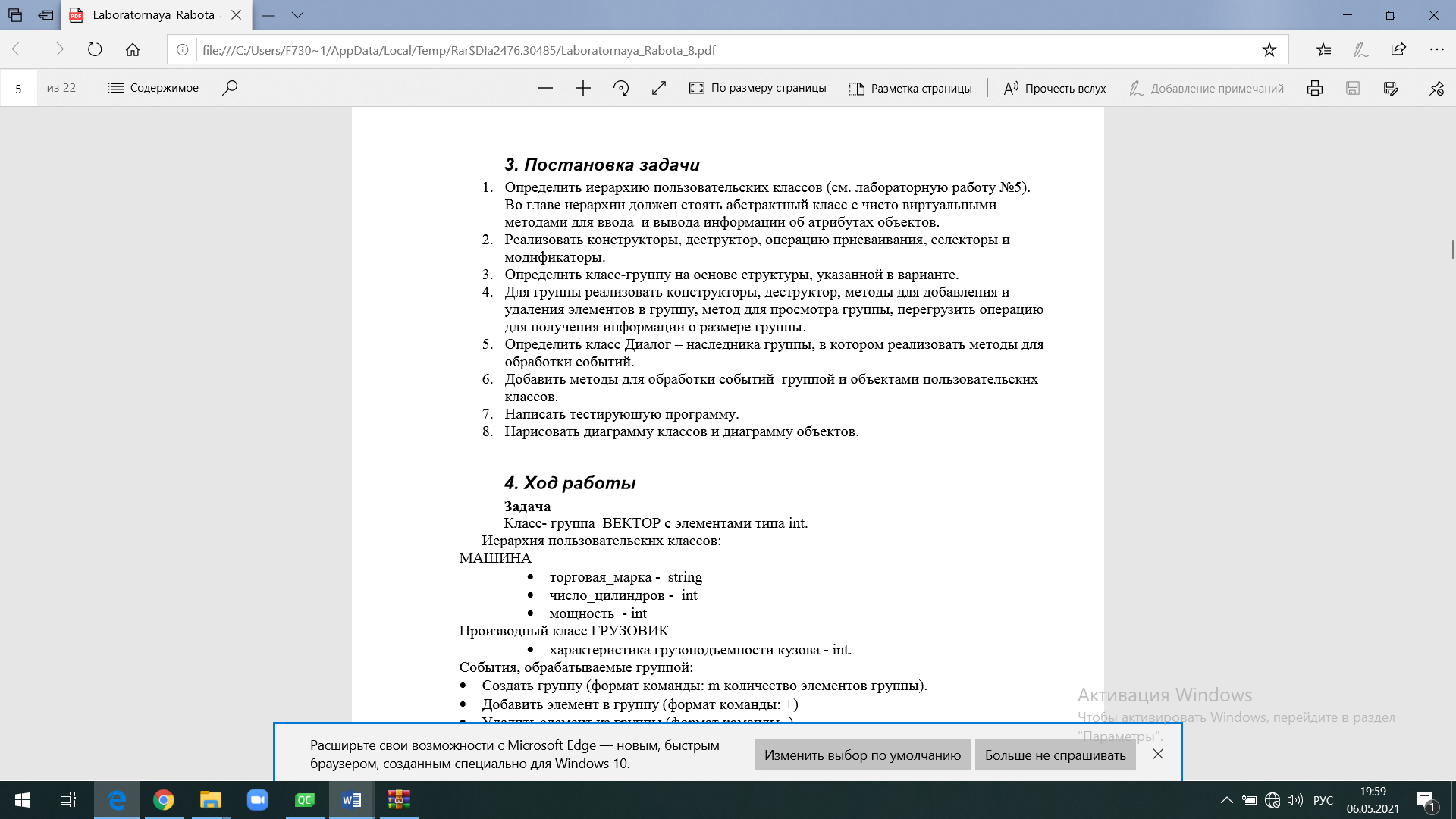
Проверила:

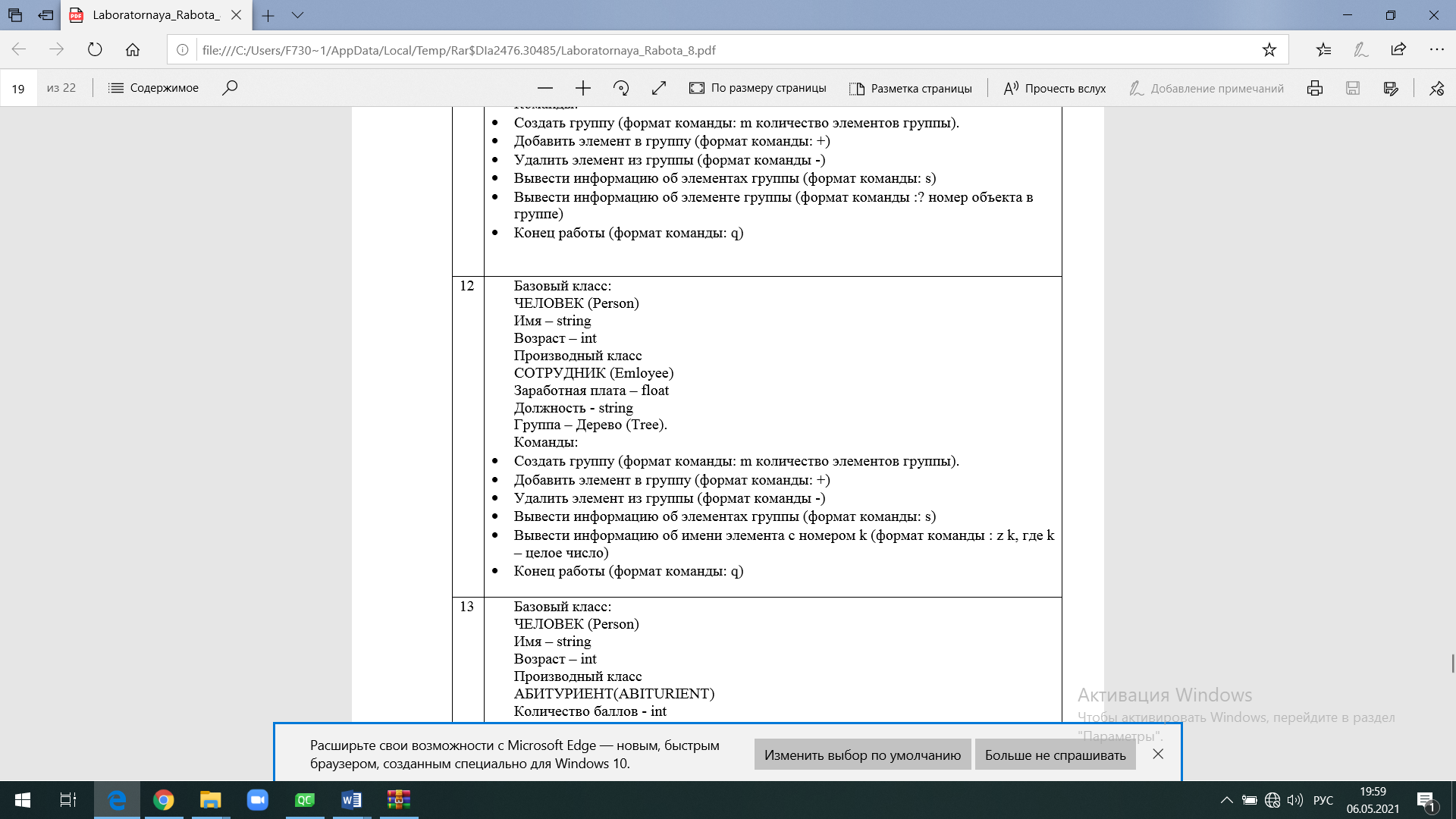
Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

1. **Постановка задачи**





1. **Анализ работы**
2. Типы данных, используемые в ходе работы
   1. Тип int, char, float, string.
3. Действия, выполняемые с этими данными
   1. Данные типа int используются в качестве счетчиков для циклов, хранения размера вектора и номера его текущего элемента, для хранения возраста в классе Person.
   2. Данные типа float и string хранят данные о зарплате и имени в классах Person и Employee.
   3. Данные типа char используются для хранения и обработки команды, которую ввел пользователь.
4. В каком виде будут представлены данные для решения задачи
   1. Данные представлены в виде классов:

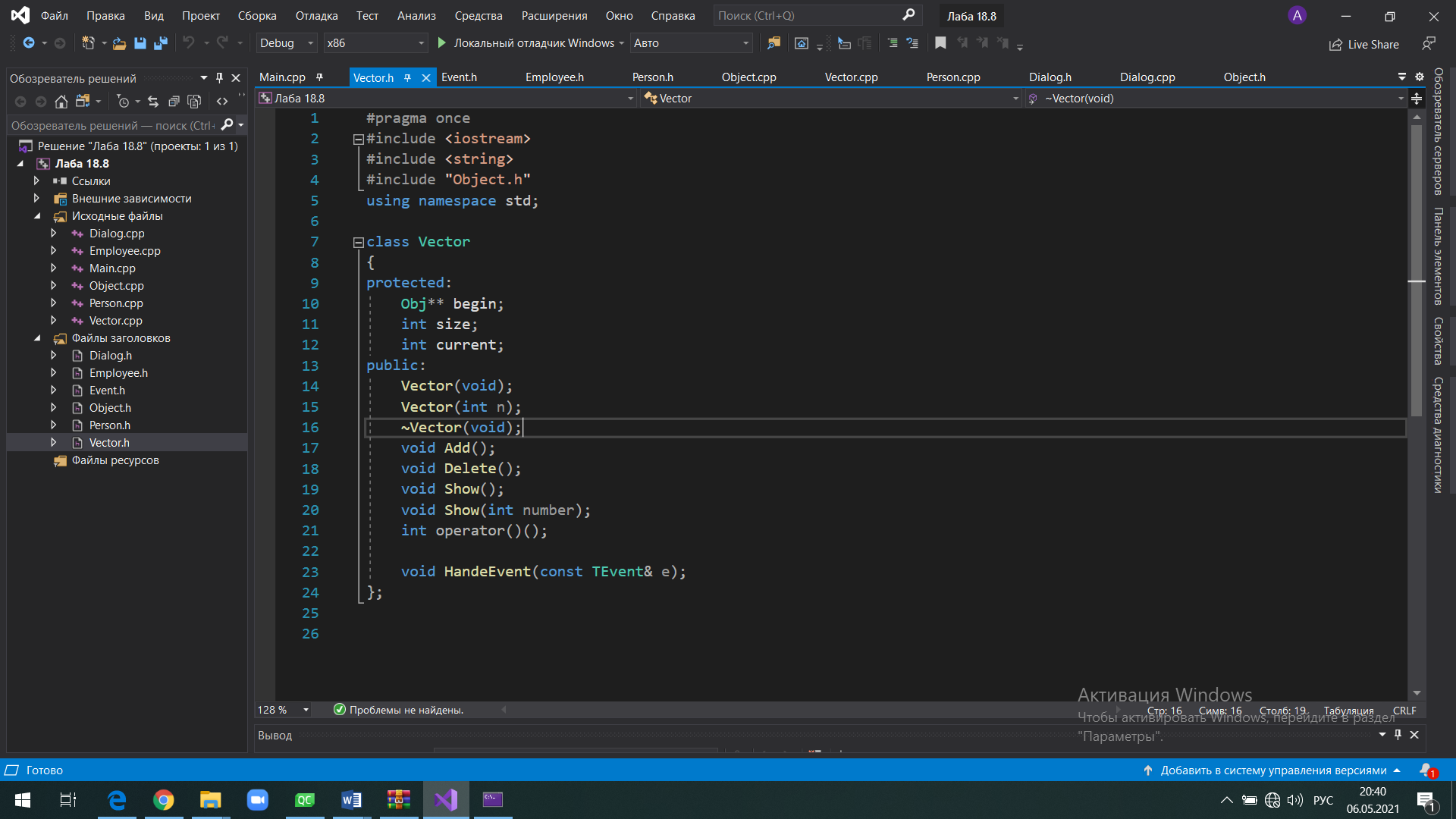
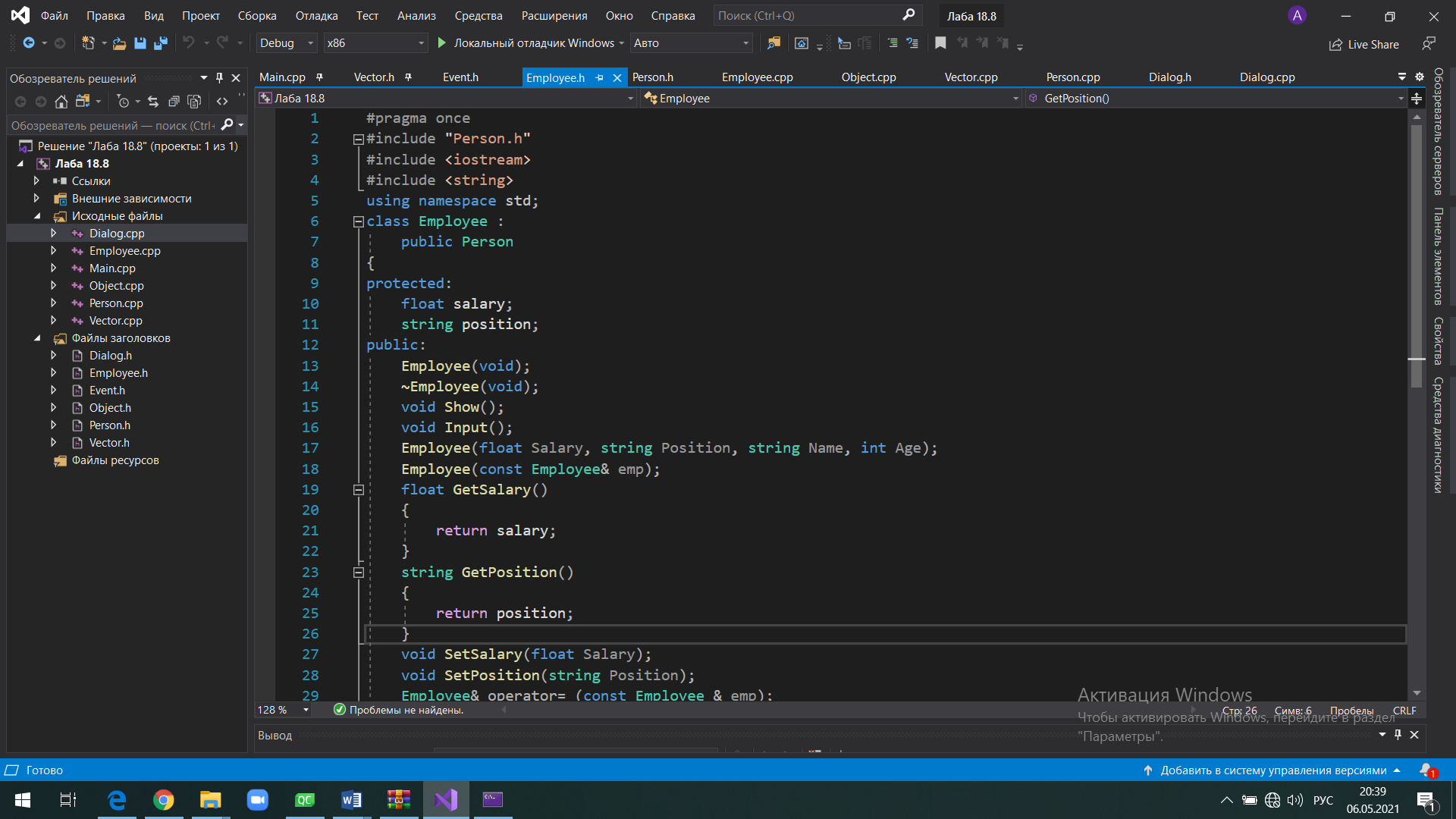
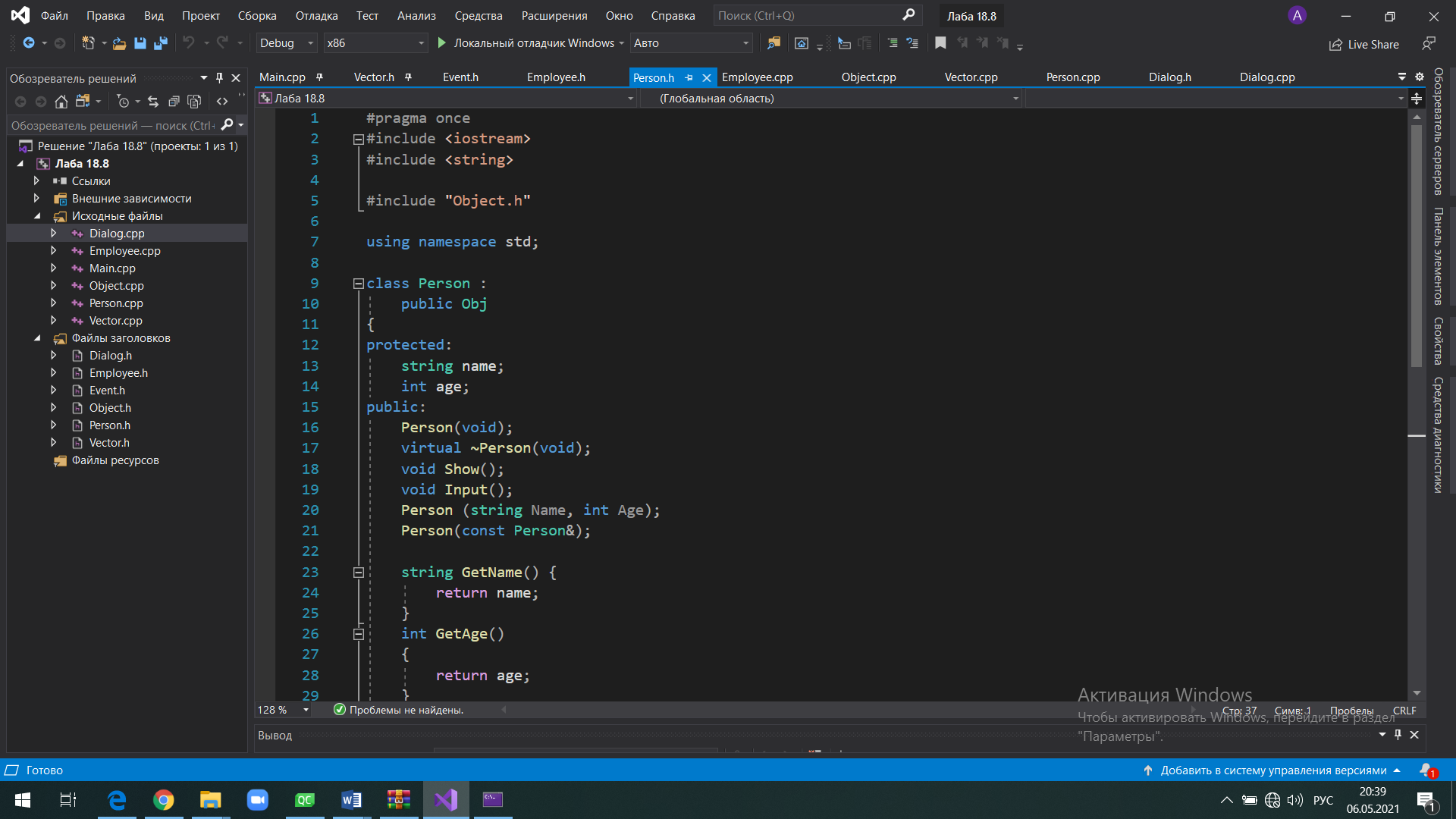
class Vector

class Obj

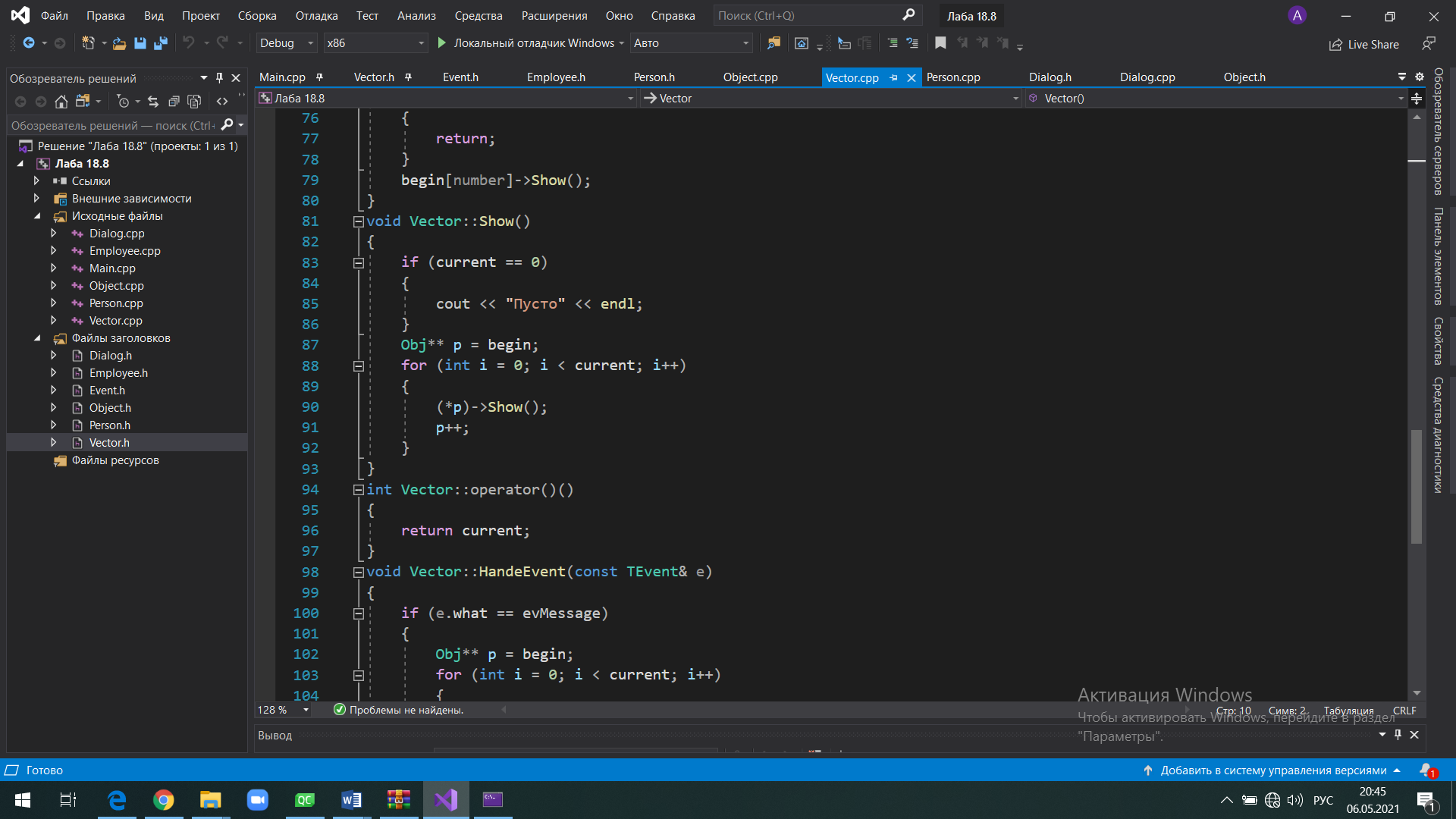
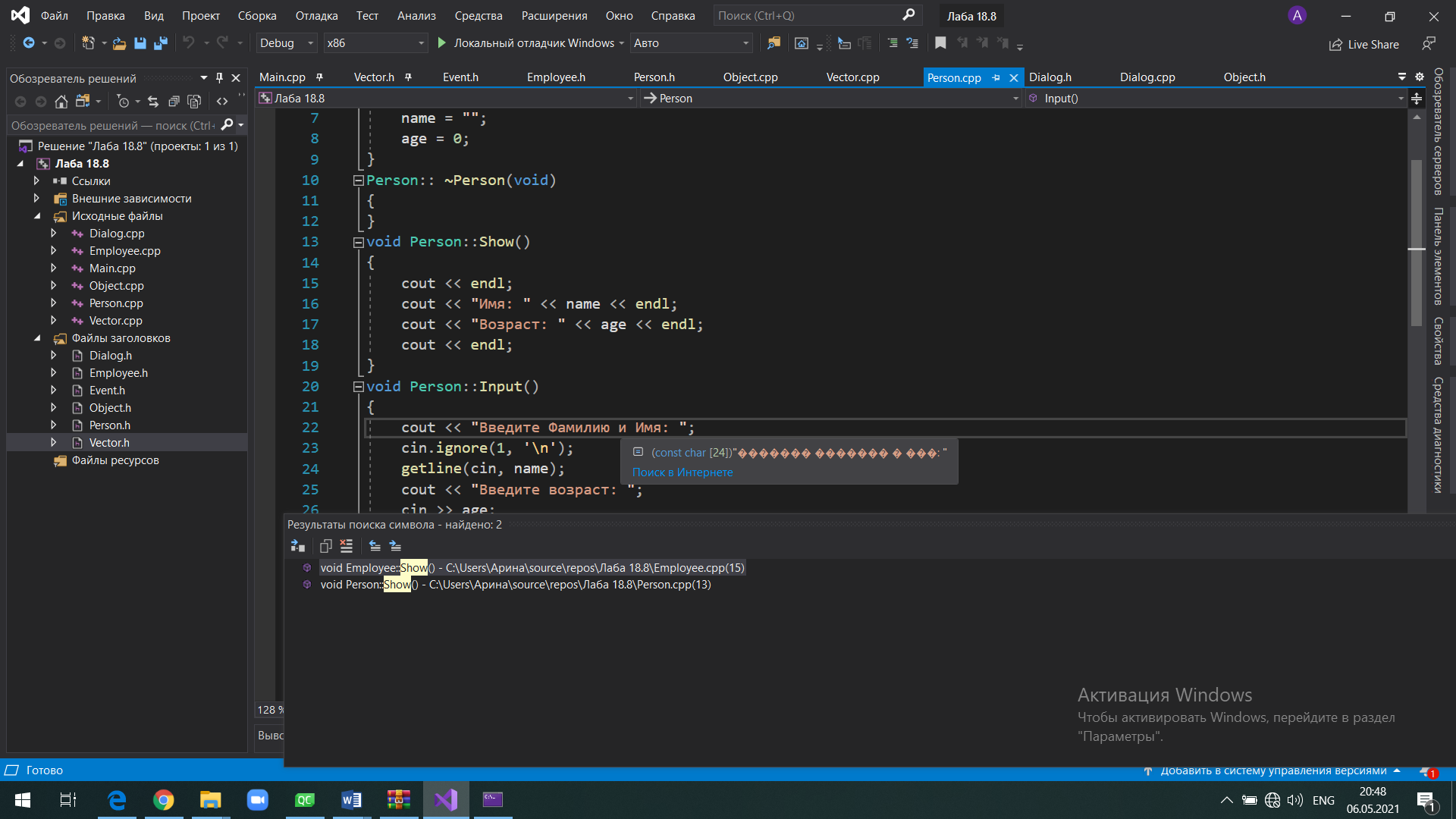
class Person

class Emloyee

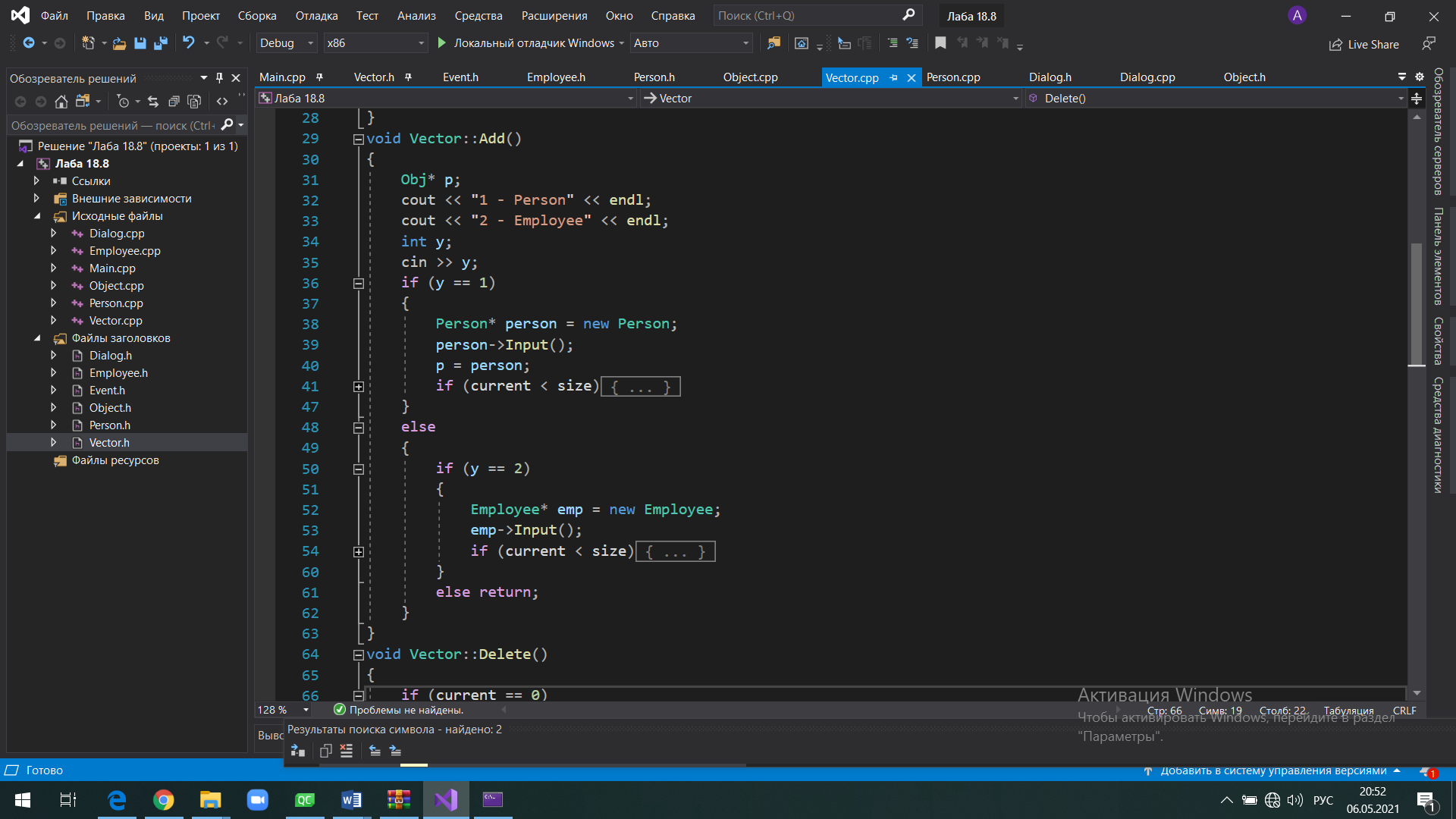
1. Поля, используемые в программе:



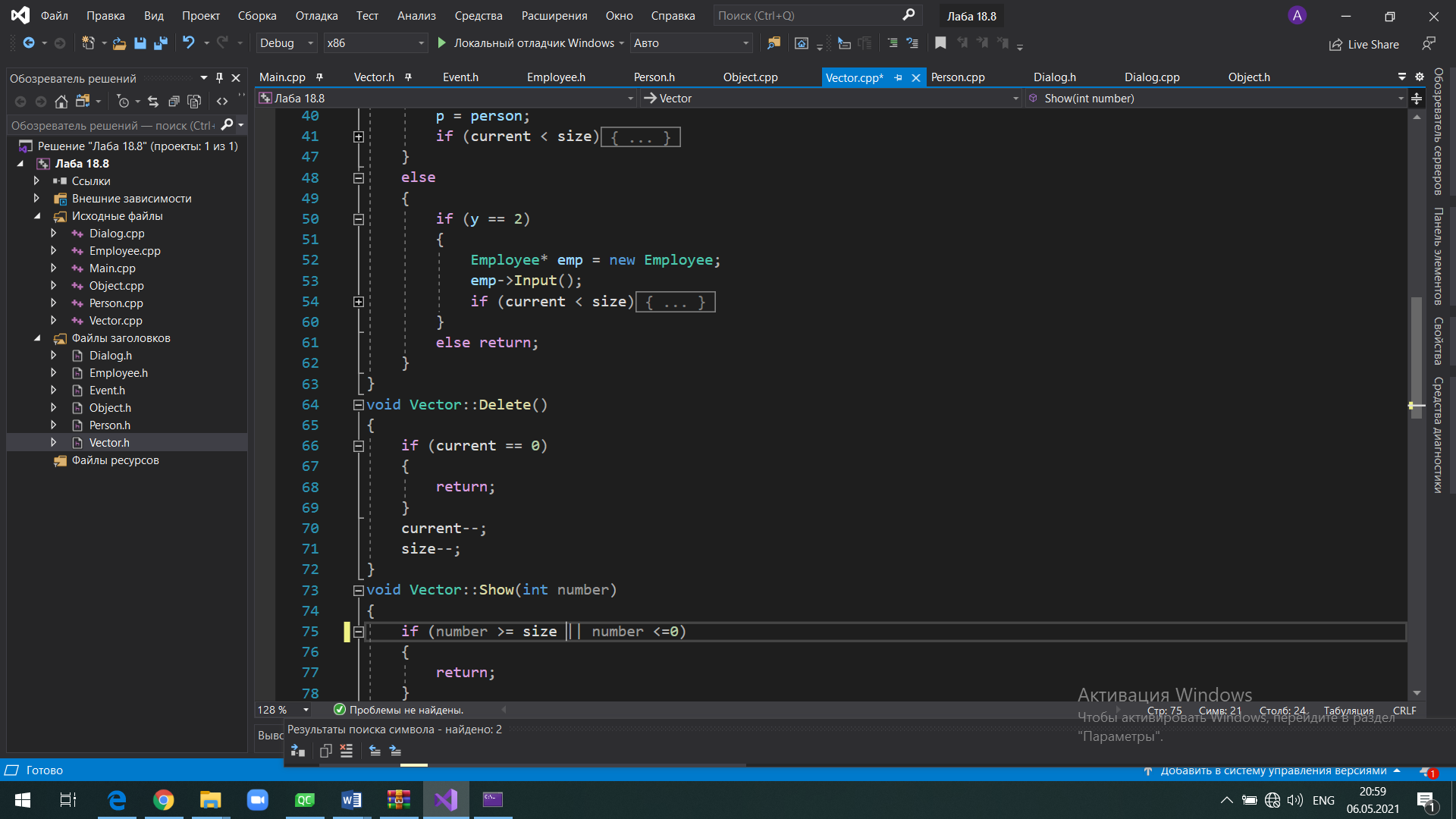
1. Операторы, с помощью которых организован ввод и вывод
   1. Вывод элементов вектора организован с помощью методов Show() и оператора cout

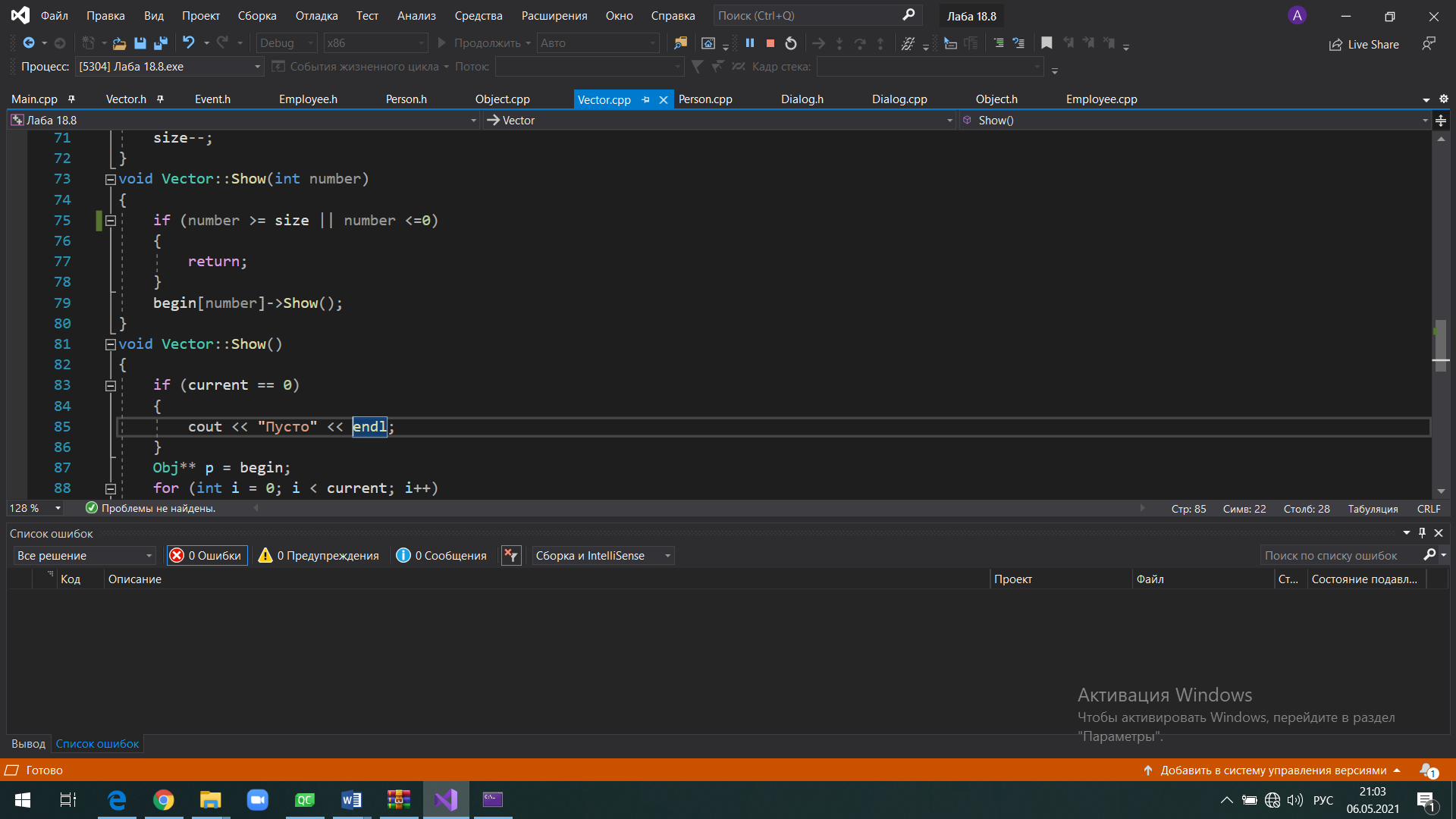
* 1. Ввод элементов организован с помощью метода Add в классе вектора, а внутри него используются метода InPut классов Person и Employee.



1. Каакими действиями будут решены поставленные задачи
   1. Выполнение команды + - добавление элемента в группу, организованно с помощью функции Add, где у пользователя спрашивается какого типа будет добавленный элемент, и в соответсвии с выбором организуется добавление элемента класса Person или класса Employee.
   2. Удаление элемента происходит с помощью метода Delete и удаляется последний элемент группы.



* 1. Выполнение команды z – вывод информации об элементе группы с номером k организованно с помощью метода Show(k), где в качестве входных параметров поступает номер элемента, информацию о котором необходимо вывести на экран.



1. **Код программы**

Object.h

#pragma once

#include "Event.h"

#include <iostream>

#include <string>

using namespace std;

class Obj

{

public:

Obj(void);

virtual void Show() = 0;

virtual void Input() = 0;

virtual ~Obj(void);

virtual void HandleEvent(const TEvent& e) = 0;

};

Object.cpp

#include <iostream>

#include <string>

#include "Object.h"

using namespace std;

Obj::Obj(void)

{

}

Obj::~Obj(void)

{

}

Person.h

#pragma once

#include <iostream>

#include <string>

#include "Object.h"

using namespace std;

class Person :

public Obj

{

protected:

string name;

int age;

public:

Person(void);

virtual ~Person(void);

void Show();

void Input();

Person (string Name, int Age);

Person(const Person&);

string GetName() {

return name;

}

int GetAge()

{

return age;

}

void SetName(string Name);

void SetAge(int Age);

Person& operator= (const Person& per);

void HandleEvent(const TEvent& e);

};

Person.cpp

#include "Person.h"

#include <iostream>

#include <string>

using namespace std;

Person::Person(void)

{

name = "";

age = 0;

}

Person:: ~Person(void)

{

}

void Person::Show()

{

cout << endl;

cout << "Имя: " << name << endl;

cout << "Возраст: " << age << endl;

cout << endl;

}

void Person::Input()

{

cout << "Введите Фамилию и Имя: ";

cin.ignore(1, '\n');

getline(cin, name);

cout << "Введите возраст: ";

cin >> age;

}

Person::Person(string Name, int Age)

{

this->name = Name;

this->age = Age;

}

Person::Person(const Person& per)

{

name = per.name;

age = per.age;

}

void Person::SetName(string Name)

{

name = Name;

}

void Person::SetAge(int Age)

{

age = Age;

}

Person& Person::operator= (const Person& per)

{

if (&per == this)

{

return \*this;

}

name = per.name;

age = per.age;

return \*this;

}

void Person::HandleEvent(const TEvent& e)

{

if (e.what == evMessage)

{

switch (e.command)

{

case cmGet: cout << endl;

break;

}

}

}

Employee.h

#pragma once

#include "Person.h"

#include <iostream>

#include <string>

using namespace std;

class Employee :

public Person

{

protected:

float salary;

string position;

public:

Employee(void);

~Employee(void);

void Show();

void Input();

Employee(float Salary, string Position, string Name, int Age);

Employee(const Employee& emp);

float GetSalary()

{

return salary;

}

string GetPosition()

{

return position;

}

void SetSalary(float Salary);

void SetPosition(string Position);

Employee& operator= (const Employee & emp);

};

Employee.cpp

#include <iostream>

#include <string>

#include "Employee.h"

using namespace std;

Employee::Employee(void) :Person()

{

salary = 0;

position = "no";

}

Employee::~Employee(void)

{

}

void Employee::Show()

{

cout << endl;

cout << "Имя: " << name << endl;

cout << "Возраст: " << age << endl;

cout << "Зарплата: " << salary << endl;

cout << "Должность: " << position << endl;

cout << endl;

}

void Employee::Input()

{

cout << "Введите Фамилию и Имя: ";

cin.ignore(1, '\n');

getline(cin, name);

cout << "Возраст: ";

cin >> age;

cout << "Зарплата: ";

cin >> salary;

cout << "Должность: ";

cin >> position;

}

Employee::Employee(float Salary, string Position, string Name, int Age) : Person (Name, Age)

{

salary = Salary;

position = Position;

}

Employee::Employee(const Employee& emp)

{

name = emp.name;

age = emp.age;

salary = emp.salary;

position = emp.position;

}

void Employee::SetSalary(float Salary)

{

salary = Salary;

}

void Employee::SetPosition(string Position)

{

position = Position;

}

Employee& Employee:: operator= (const Employee & emp)

{

if (&emp == this)

{

return \*this;

}

name = emp.name;

age = emp.age;

salary = emp.salary;

position = emp.position;

return \*this;

}

Vector.h

#pragma once

#include <iostream>

#include <string>

#include "Object.h"

using namespace std;

class Vector

{

protected:

Obj\*\* begin;

int size;

int current;

public:

Vector(void);

Vector(int n);

~Vector(void);

void Add();

void Delete();

void Show();

void Show(int number);

int operator()();

void HandeEvent(const TEvent& e);

};

Vector.cpp

#include "Vector.h"

#include "Object.h"

#include "Person.h"

#include "Employee.h"

#include <iostream>

#include <string>

using namespace std;

Vector::Vector()

{

begin = new Obj\* [1];

current = 0;

size = 1;

}

Vector::Vector(int n)

{

begin = new Obj\* [n];

current = 0;

size = n;

}

Vector::~Vector(void)

{

if (begin != 0)

{

delete[] begin;

}

begin = 0;

}

void Vector::Add()

{

Obj\* p;

cout << "1 - Person" << endl;

cout << "2 - Employee" << endl;

int y;

cin >> y;

if (y == 1)

{

Person\* person = new Person;

person->Input();

p = person;

if (current < size)

{

begin[current] = p;

current++;

size++;

}

}

else

{

if (y == 2)

{

Employee\* emp = new Employee;

emp->Input();

if (current < size)

{

begin[current] = emp;

current++;

size++;

}

}

else return;

}

}

void Vector::Delete()

{

if (current == 0)

{

return;

}

current--;

size--;

}

void Vector::Show(int number)

{

if (number >= size || number <=0)

{

return;

}

begin[number]->Show();

}

void Vector::Show()

{

if (current == 0)

{

cout << "Пусто" << endl;

}

Obj\*\* p = begin;

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

{

(\*p)->Show();

p++;

}

}

int Vector::operator()()

{

return current;

}

void Vector::HandeEvent(const TEvent& e)

{

if (e.what == evMessage)

{

Obj\*\* p = begin;

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

{

(\*p)->HandleEvent(e);

p++;

}

}

}

Event.h

#pragma once

const int evNothing = 0;

const int evMessage = 100;

const int cmAdd = 1;

const int cmDel = 2;

const int cmGet = 3;

const int cmShow = 4;

const int cmMake = 5;

const int cmQuit = 101;

struct TEvent

{

int what;

union

{

int command;

struct

{

int message;

int a;

};

};

};

Dialog.h

#pragma once

#include "Event.h"

#include "Vector.h"

using namespace std;

class Dialog : public Vector

{

protected:

int EndState;

public:

Dialog(void);

virtual ~Dialog();

virtual void GetEvent(TEvent& event);

virtual int Execute();

virtual void HandleEvent(TEvent& event);

virtual void ClearEvent(TEvent& event);

int Valid();

void EndExec();

};

Dialog.cpp

#include "Dialog.h"

using namespace std;

Dialog::Dialog(void):Vector()

{

EndState = 0;

}

Dialog:: ~Dialog()

{

}

void Dialog::GetEvent(TEvent& event)

{

string OpInt = "+-szqm";

string s;

string param;

char code;

cout << '>';

cin >> s;

code = s[0];

if (OpInt.find(code) >= 0)

{

event.what = evMessage;

switch (code)

{

case 'm': event.command = cmMake; break;

case '+': event.command = cmAdd; break;

case '-': event.command = cmDel; break;

case 's': event.command = cmShow; break;

case 'z': event.command = cmGet; break;

case 'q': event.command = cmQuit; break;

}

if (s.length() > 1)

{

param = s.substr(1, s.length() - 1);

int A = atoi(param.c\_str());

event.a = A;

}

}

else

{

event.what = evNothing;

}

}

int Dialog::Execute()

{

TEvent event;

do

{

EndState = 0;

GetEvent(event);

HandleEvent(event);

} while (!Valid());

return EndState;

}

void Dialog::HandleEvent(TEvent& event)

{

if (event.what == evMessage)

{

switch (event.command)

{

case cmMake:

size = event.a;

begin = new Obj \* [size];

ClearEvent(event);

break;

case cmAdd:

Add();

ClearEvent(event);

break;

case cmDel:

Delete();

ClearEvent(event);

break;

case cmShow:

Show();

ClearEvent(event);

break;

case cmGet:

Show(event.a);

ClearEvent(event);

break;

case cmQuit:

EndExec();

ClearEvent(event);

break;

default: Vector::HandeEvent(event);

}

}

}

void Dialog::ClearEvent(TEvent& event)

{

event.what = evNothing;

}

int Dialog::Valid()

{

if (EndState == 0)

{

return 0;

}

else return 1;

}

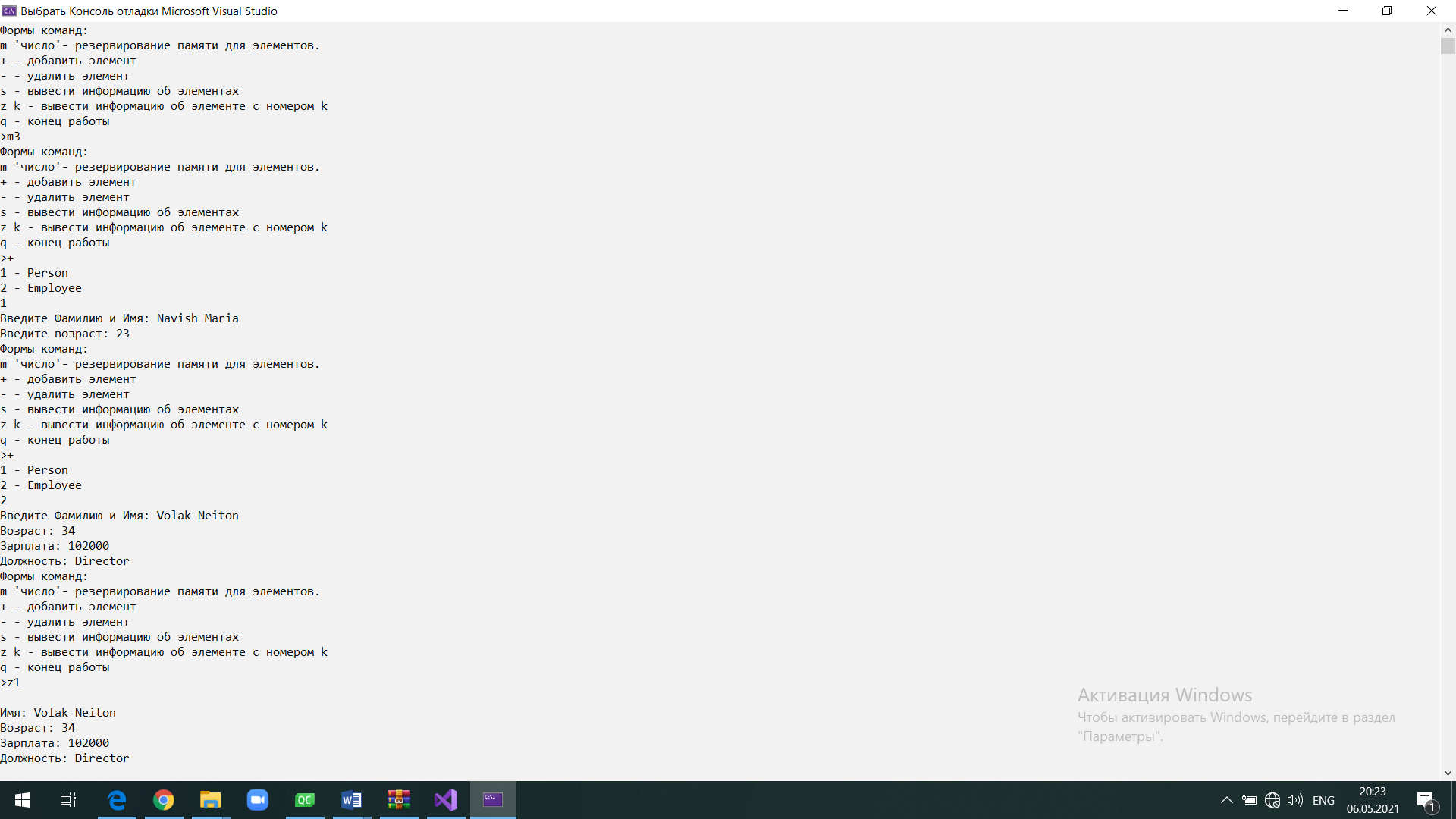
void Dialog::EndExec()

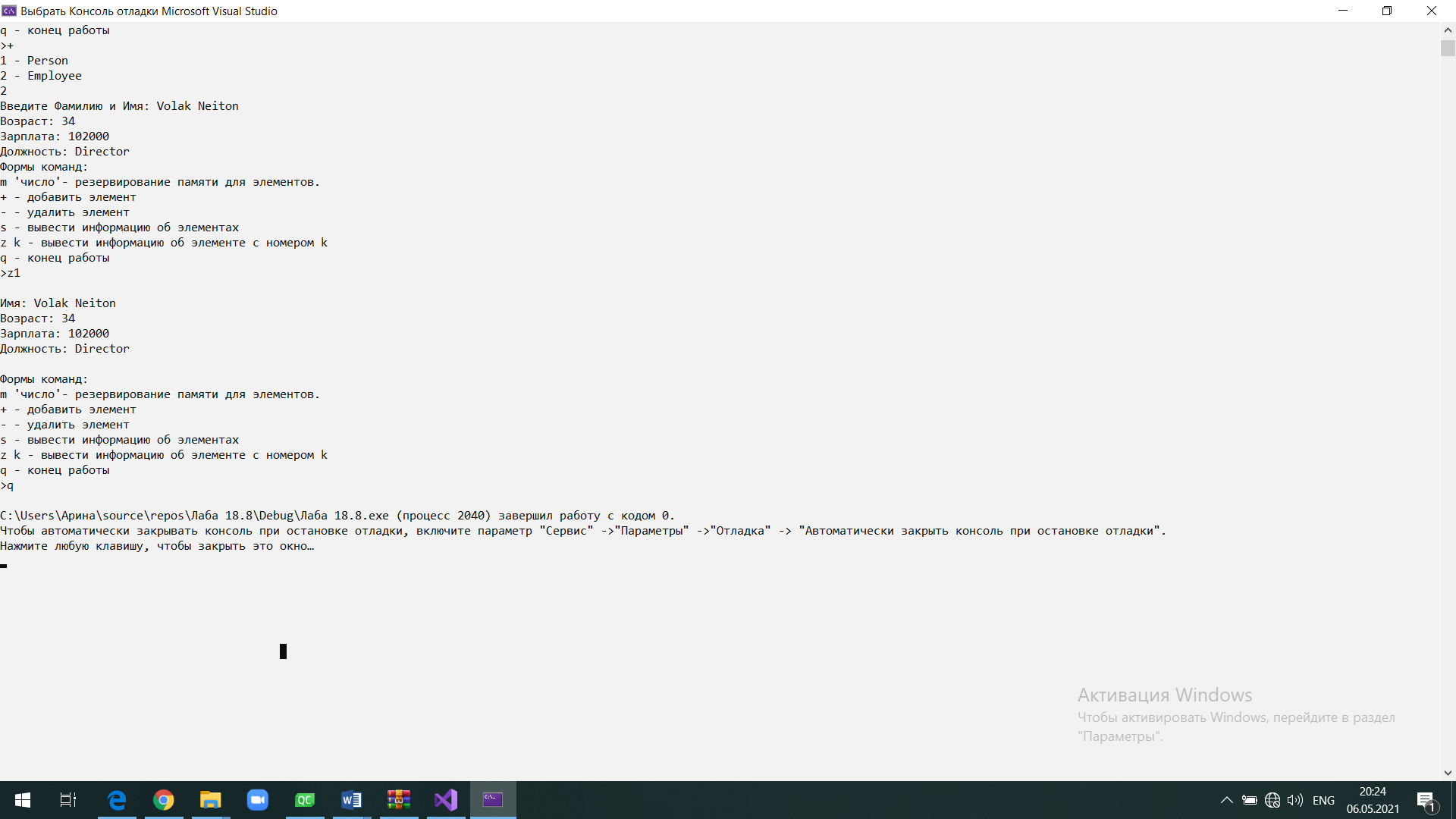
{

EndState = 1;

}

1. **Выполнение программы**





1. **Блок-схема**

