Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Кафедра **«**Информационные технологии и автоматизированные системы**»**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

Дисциплина: «Основы алгоритмизации и программирования»

Семестр 2

Тема: Простое наследование. Принцип подстановки.

Вариант 12.

Выполнил работу

Студент группы РИС-22-1Б

Головин М.А.

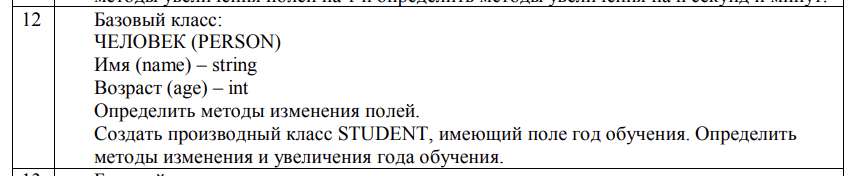
Проверил

Доцент кафедры ИТАС

Полякова О.А.

г. Пермь-2023

**Постановка задачи**



**Код программы**

**#include** <iostream>

#include "Student.h"

using namespace std;

int main() {

Student Chmoha("kek",20,2);

Chmoha.outg();

return 0;

**}**

**#include** <iostream>

#include <string>

using namespace std;

class Person {

protected:

string name;

int age;

public:

Person(void);

Person(string, int);

Person(const Person&);

~Person(void);

int ageo(void);

string nameo(void);

**};**

#include "Person.h"

#include <iostream>

#include <string>

using namespace std;

Person::Person(void) {

name = "Ivan";

age = 1;

}

Person::Person(string namei, int agei)

{

name = namei;

age = agei;

}

Person::Person(const Person&P)

{

name = P.name;

age = P.age;

}

Person::~Person(void){}

int Person::ageo(void)

{

return age;

}

string Person::nameo(void)

{

return name;

}

#pragma once

#include <iostream>

#include <string>

#include "Person.h"

class Student:

public Person {

protected:

int eduage;

public:

Student(void);

Student(string, int, int);

Student(const Student&);

~Student(void);

int eduageo(void);

void ch\_eduage(int);

void outg(void);

// friend ostream& operator<<(ostream&, Student& S);

// friend istream& operator>>(istream&, Student&);

**};**

#include "Student.h"

Student::Student() {

name = "Ivan";

age = 18;

eduage = 1;

}

Student::Student(string namei, int agei, int eduagei):Person(namei,agei)

{

eduage = eduagei;

}

Student::Student(const Student& S) {

name = S.name;

age = S.age;

eduage = S.eduage;

}

Student::~Student(void) {}

int Student::eduageo(void)

{

return eduage;

}

void Student::ch\_eduage(int eduagei) {

eduage = eduagei;

}

void Student::outg(void) {

cout << "--Student Information--" << endl;

cout << "Name:" << name << endl;

cout << "Age:" << age << endl;

cout << "Years of education:" << eduage << endl;

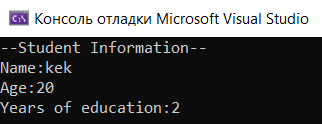
}

//istream& operator<<(ostream& out, const Student& S) {

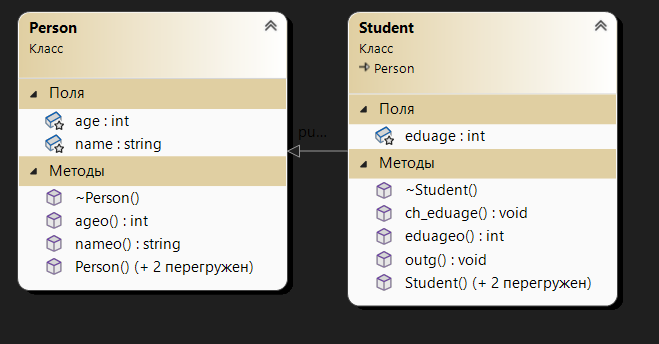
// return(out<< "--Student Information--" << endl<<"Name:"<<S.nameo()<<endl<<"Age:"<<S.age<<)

//}

**Вывод программы**

****

**UML диаграмма**

****