Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

ПНИПУ

**Лабораторная работа**

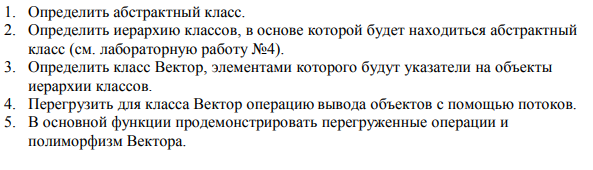
**"Классы №5”**

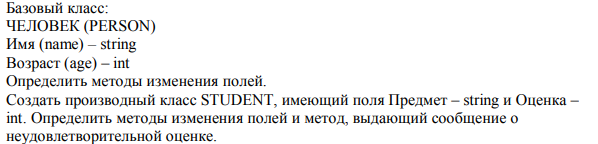
Выполнил:   
студент группы РИС-23-1б   
Агзамов Артур Альферович

Проверила:   
доцент кафедры ИТАС   
О.А. Полякова

2024 г.

**Задача:**





**Анализ задачи:**  
  
1. Определение абстрактного класса Person с полями Имя и Возраст, а также чисто виртуальными методами.

2. Создание производного класса Student, наследуемого от класса Person, с добавлением полей Предмет и Оценка.

3. Разработка класса Vector для хранения указателей на объекты классов Person и Student.

4. Перегрузка операции вывода объектов с использованием потоков для класса Vector.

5. Написание основной функции программы для демонстрации перегруженных операций и использования полиморфизма вектора.

6. Разработка программы для работы с классами Person, Student и Vector, включая демонстрацию перегруженных операций и полиморфизма вектора.

**Код:**

#include<iostream>

#include<string>

#include <vector>

#include "class-5.h"

using namespace std;

int main(){

system("chcp 1251>null")

Vector vec(4);

person per;

student stud;

cin >> per;

cout << per;

cin >> stud;

cout << stud;

Object\* p = &per;

vec.Add(p);

p = &stud;

vec.Add(p);

cout << vec;

return 0;

}  
  
#pragma once

#include <vector>

#include<iostream>

#include<string>

using namespace std;

class Object {

public:

virtual void print() = 0;

};

class person:public Object{

protected:

string name;

int age;

public:

person() {age = 0;name = "";}

person(string name, int age){

this->name= name;

this->age=age;

cout<<"Constructor person"<<endl<<endl;

}

person(const person& Person){

this-> name =Person.name;

this-> age= Person.age;

}

~person(){

cout<<"Constructor person"<<endl<<endl;

}

virtual void SetName(string name){

this->name=name;

}

virtual void SetAge(int age){

this->age=age;

}

virtual string get\_name() {

return name;

}

virtual int get\_age() {

return age;

}

void print() override {

cout << this->name << "name" << endl;

cout << this->age << " age" << endl;

}

friend istream& operator >>(istream& is, person& per) {

cout << "name: ";

getline(is, per.name);

cout << "enter age: ";

(is >> per.age).ignore();

return is;

}

friend ostream& operator <<(ostream& os, person& stud) {

cout << " name: " << stud.name << endl;

cout << " age: " << stud.age << endl;

return os;

}

};

class student:public person{

private:

string subject;

int grade;

friend ostream& operator <<(ostream& out, const student & Student);

friend istream& operator >>(istream& in, const student& Student);

public:

student() {subject = "";grade = 0;age = 0;name = "";}

student(string name, int age, string subject, int grade) : person(name, age){

this->subject= subject;

this->grade=grade;

this->name=name;

this->age=age;

cout<<"Constructor student"<<endl<<endl;

}

student(const student& Student): person(name, age){

this->name=Student.name;

this->age=Student.age;

this->subject=Student.subject;

this->grade=Student.grade;

}

void operator = (const student& Student){

this->name=Student.name;

this->age=Student.age;

this->subject=Student.subject;

this->grade=Student.grade;

}

~student(){

cout<<"destructor student"<<endl<<endl;

}

void SetSubject(string subject){

this->subject=subject;

}

void SetGrade(int grade){

this->grade=grade;

}

string get\_subject(){

return subject;

}

int get\_grade(){

return grade;

}

void check\_grade(){

if (grade<3){

cout<<"student "<<name<<" has unsatisfactory grade in " << subject << endl;

}

else{

cout << "Student " << name << " has passed " << subject << " with grade " << grade << endl;

}

}

};

ostream& operator <<(ostream& out, const student & Student){

out << endl <<"Person:"<< Student.name<<" Age: "<<Student.age<<" Have grade "<<Student.grade<<" in subject "<<Student.subject<<endl;

out<<endl;

return out;

}

istream& operator >> (istream& in, student& Student){

cout<<"Enter person, his age, subject and his grade in subject"<<endl;

string name;

int age;

string subject;

int grade;

in>>name>>age>>subject>>grade;

Student.SetName(name);

Student.SetAge(age);

Student.SetSubject(subject);

Student.SetGrade(grade);

return in;

}

class Vector

{

public:

friend ostream& operator<<(ostream& out, const Vector& v) {

if (v.size == 0) {

cout << "\nВектор пустой" << endl;

}

else {

Object\*\* ptr = v.beg;

for (int i = 0; i < v.cur; i++) {

(\*ptr)->print();

ptr++;

}

}

return out;

}

void Add(Object\* ptr) {

if (cur < size) {

beg[cur] = ptr;

cur++;

}

}

Vector() {

beg = 0;

size = 0;

cur = 0;

}

Vector(int lin) {

beg = new Object \* [lin];

size = lin;

cur = 0;

}

~Vector() {};

private:

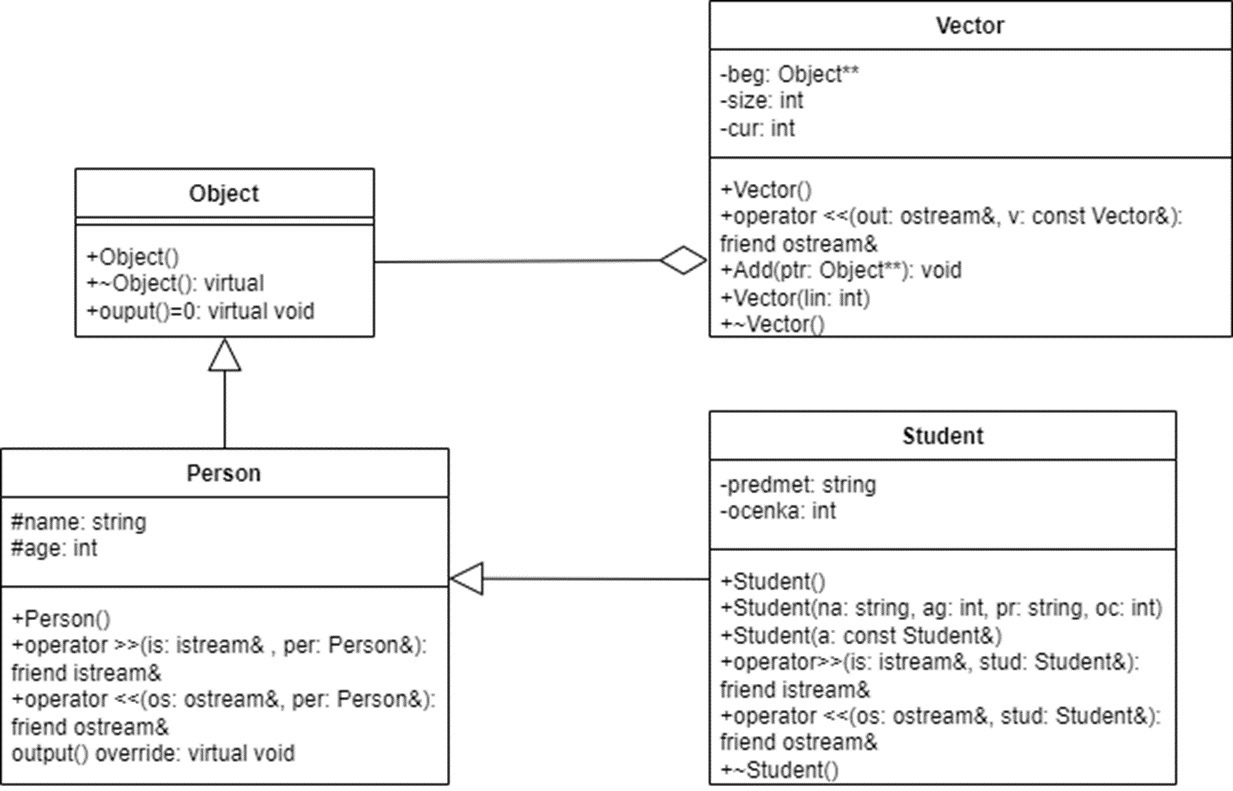
Object\*\* beg;

int size;

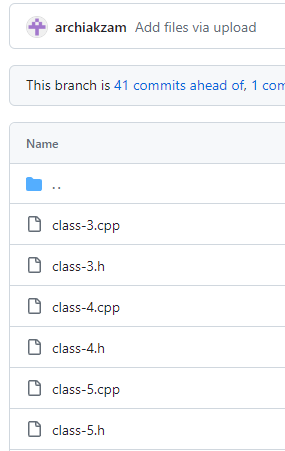
int cur;

};

**UML диаграмма:**



Скрины из git:



**Выводы:** программа сработала корректно и вывела все возможные решения.