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 ИВТ-23-2Б  
Чудинов Данил Николаевич

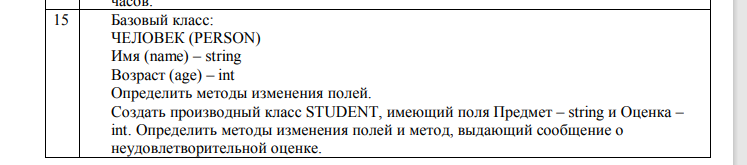
Проверила:   
доцент кафедры ИТАС   
О.А. Полякова

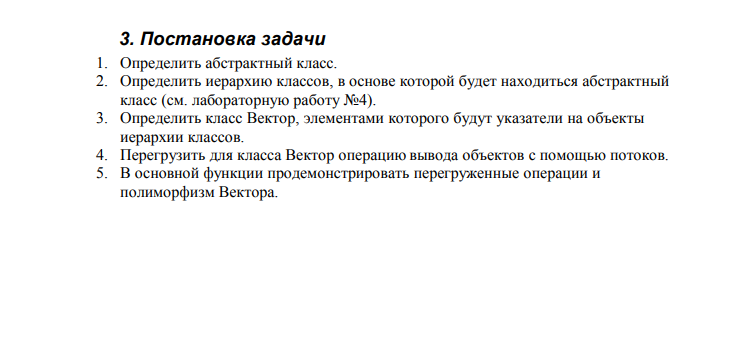
Пермь, 2024 г.

**«Классы и объекты. Виртуальные функции. Полиморфизм»**

**Анализ задачи:**

***Вариант 15:***

****



**Код на языке C++:**

#include <iostream>

#include <string>

using namespace std;

class Object {

public:

Object() {

}

virtual ~Object() {

}

virtual void output() = 0;

};

class Person : public Object {

protected:

string name;

int age;

public:

Person() {

this->name = "noname";

this->age = 777;

};

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

cout << "Введите имя: ";

is >> per.name;

cout << "Введите возраст: ";

is >> per.age;

return is;

}

friend ostream& operator <<(ostream& os, Person& per) {

cout << "Имя: ";

os << per.name;

cout << endl;

cout << "Возраст: ";

os << per.age;

cout << endl;

return os;

}

virtual void output() override {

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

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

}

};

class Student : public Person {

string predmet;

int ocenka;

public:

Student() {

this->name = "noname";

this->age = 777;

this->predmet = "Все виды спорта мира";

this->ocenka = 777;

cout << "Вызван конструктор без параметров " << this << endl;

cout << "Имя: " << this->name << endl << "Возраст: " << this->age << endl << "Предмет: " << this->predmet << endl << "Оценка по предмету: " << this->ocenka << endl;

}

Student(string na, int ag, string pr, int oc) {

this->name = na;

this->age = ag;

this->predmet = pr;

this->ocenka = oc;

cout << "Вызван конструктор с параметрами " << this << endl;

cout << "Имя: " << this->name << endl << "Возраст: " << this->age << endl << "Предмет: " << this->predmet << endl << "Оценка по предмету: " << this->ocenka << endl;

if (ocenka < 3) {

cout << "Оценка: неудовлетворительно";

}

}

Student(const Student& a) {

this->name = a.name;

this->age = a.age;

this->predmet = a.predmet;

this->ocenka = a.ocenka;

cout << "Вызван конструктор копирования " << this << endl;

cout << "Имя: " << this->name << endl << "Возраст: " << this->age << endl << "Предмет: " << this->predmet << endl << "Оценка по предмету: " << this->ocenka << endl;

}

friend istream& operator >> (istream& is, Student& stud) {

cout << "Введите имя: ";

is >> stud.name;

cout << "Введите возраст: ";

is >> stud.age;

cout << "Введите предмет: ";

is >> stud.predmet;

cout << "Введите оценку: ";

is >> stud.ocenka;

return is;

}

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

cout << "Имя: " << stud.name << endl;

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

cout << "Предмет: " << stud.predmet << endl;

cout << "Оценка: " << stud.ocenka << endl;

return os;

}

~Student() {

cout << "Вызван деструктор: " << this;

cout << endl;

}

};

class Vector

{

private:

Object\*\* beg;

int size;

int cur;

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)->output();

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() {};

};

int main() {

system("chcp 1251>NULL");

Vector vec(4);

Person p1;

cin >> p1;

cout << endl << endl;

cout << p1;

cout << endl << endl;

Student s1;

cout << endl << endl;

Student s2("Василий", 18, "Математика", 2);

cout << endl << endl;

Student s3(s2);

cout << endl << endl;

Student s4;

cout << endl << endl;

cin >> s4;

cout << endl << endl;

cout << s4;

cout << endl << endl;

Object\* p = &p1;

vec.Add(p);

p = &s4;

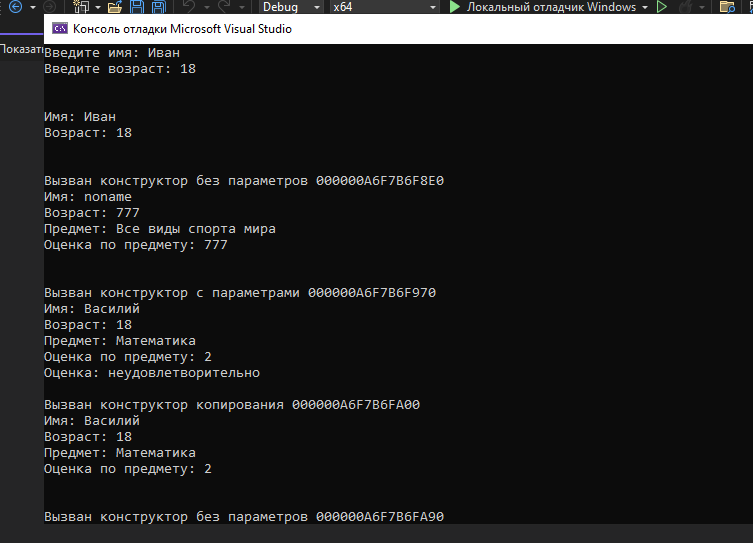
vec.Add(p);

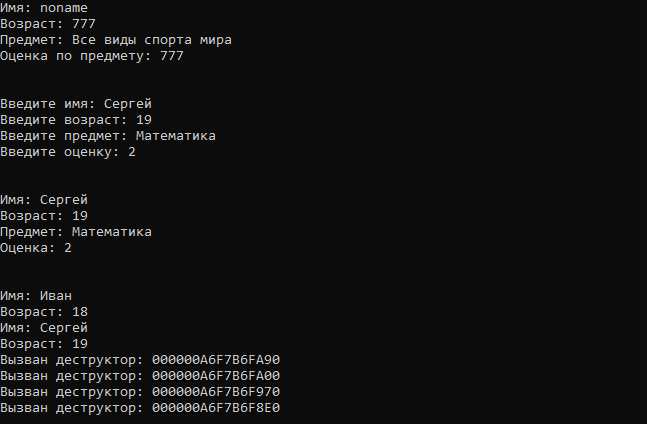
cout << vec;

return 0;

}

**Работа программы:**

****



**Ответ на вопросы:**

1. Чисто виртуальный метод - это метод в абстрактном классе, который не имеет своей реализации в этом классе, а лишь объявлен (чисто виртуальный метод обозначается в C++ с помощью "= 0"). Отличие чисто виртуального метода от виртуального заключается в том, что виртуальные методы имеют реализацию по умолчанию, которую можно переопределить в наследуемых классах, в то время как чисто виртуальные методы требуют обязательной реализации в классах-наследниках.

2. Абстрактный класс - это класс, который содержит хотя бы один чисто виртуальный метод и не может быть создан напрямую, он служит только для наследования. В C++ абстрактные классы используются для создания интерфейсов и определения общего поведения для подклассов.

3. Абстрактные классы предназначены для определения общего интерфейса для классов-наследников, обеспечивая единообразие взаимодействия с различными классами. Они помогают абстрагировать общие черты из различных классов для более удобного использования и структурирования кода.

4. Полиморфные функции - это функции, которые могут принимать объекты разных типов и вести себя по-разному в зависимости от типа объекта. Это один из аспектов полиморфизма в объектно-ориентированном программировании.

5. Полиморфизм позволяет вызывать одно и то же имя функции для разных классов, что приводит к выполнению различной логики в зависимости от конкретного класса. Принцип подстановки (Liskov Substitution Principle) заключается в том, что объекты подклассов могут быть заменены объектами суперкласса без изменения желаемого поведения программы.

6. Примеры иерархий с использованием абстрактных классов:

- Абстрактный класс `Shape`, унаследованный классами `Circle`, `Rectangle`, `Triangle`.

- Абстрактный класс `Animal`, унаследованный классами `Dog`, `Cat`, `Bird`.

7. Примеры полиморфных функций:

- Функция `draw()` в классе `Shape`, которая отрисовывает различные фигуры в зависимости от конкретного подкласса (`Circle`, `Rectangle`, `Triangle`).

- Функция `makeSound()` в классе `Animal`, которая издает звуки различных животных (`Dog`, `Cat`, `Bird`).

8. Механизм позднего связывания используется в тех случаях, когда необходимо определить время выполнения вызываемой функции. Этот механизм позволяет определить конкретную функцию во время выполнения программы, что особенно полезно при использовании полиморфизма и виртуальных

**Вывод:** Задача была выполнена. Всё получилось.

