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

ПНИПУ

**Лабораторная работа  
“Классы”. №4**

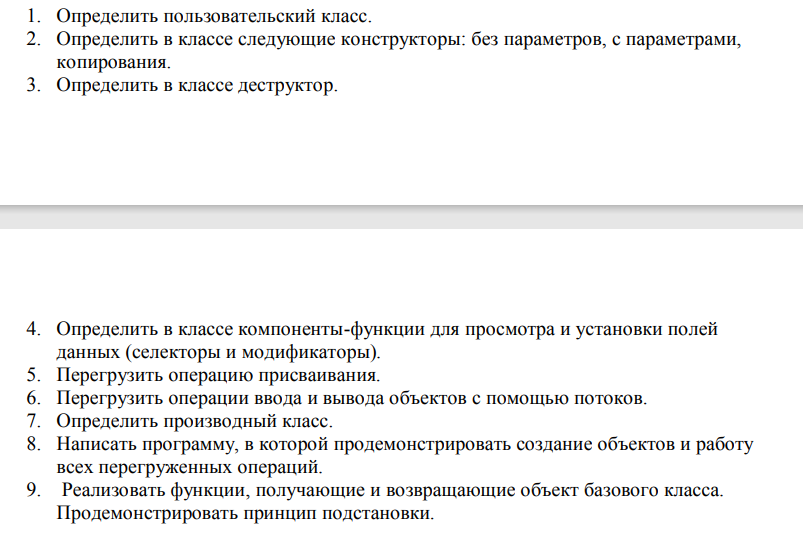
Выполнил:   
студент группы ИВТ-23-2Б   
Чудинов Данил Николаевич

Проверила:   
доцент кафедры ИТАС   
О.А. Полякова

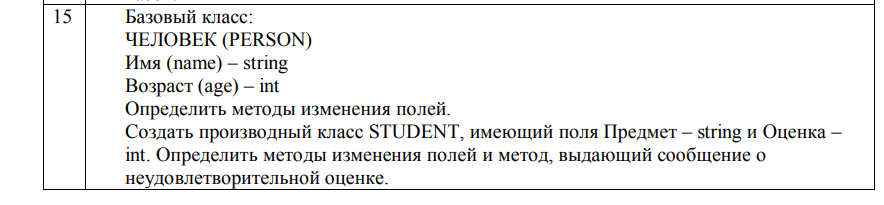
Пермь, 2024 г.

**«Классы и объекты. Принцип подстановки»**

**Анализ задачи:**

****

***Вариант 15:***

****

**Код на языке C++:**

#include <iostream>

#include <string>

using namespace std;

class Person {

protected:

string name;

int age;

};

class Student : private 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;

}

Student& operator = (const Student& a) {

cout << "Вызвана перегрузка оператора = : " << endl;

this->name = a.name;

this->age = a.age;

this->predmet = a.predmet;

this->ocenka = a.ocenka;

return \*this;

}

void SetStr(string name) {

this->name = name;

}

void SetInt(int age) {

this->age = age;

}

void SetStr1(string predmet) {

this->predmet = predmet;

}

void SetInt1(int ocenka) {

this->ocenka = ocenka;

}

string GetName() {

return this->name;

}

int GetAge() {

return this->age;

}

string GetPredmet() {

return this->predmet;

}

int GetOcenka() {

if (ocenka < 3) {

cout << "Оценка: неудовлетворительно";

cout << endl;

}

return this->ocenka;

}

~Student() {

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

cout << endl;

}

};

ostream& operator << (ostream& os, Student& b) {

cout << "Вызвана перегрузка оператора « " << endl;

os << b.GetName() << endl << b.GetAge() << endl << b.GetPredmet() << endl << b.GetOcenka() << endl;

return os;

}

istream& operator >> (istream& is, Student& a) {

cout << "Вызвана перегрузка оператора » " << endl;

string name;

int age;

string predmet;

int ocenka;

cout << "Введите имя: ";

is >> name;

cout << "Введите возраст: ";

is >> age;

cout << "Введите предмет: ";

is >> predmet;

cout << "Введите оценку по предмету: ";

is >> ocenka;

a.SetStr(name);

a.SetInt(age);

a.SetStr1(predmet);

a.SetInt1(ocenka);

return is;

}

int main() {

system("chcp 1251>NULL");

Student G1;

cout << endl << endl;

Student G2("Василий", 18, "Математика", 2);

cout << endl << endl;

Student G3(G2);

cout << endl << endl;

cin >> G1;

cout << endl << endl;

cout << G1;

cout << endl << endl;

Student G4;

cout << endl << endl;

G4 = G3;

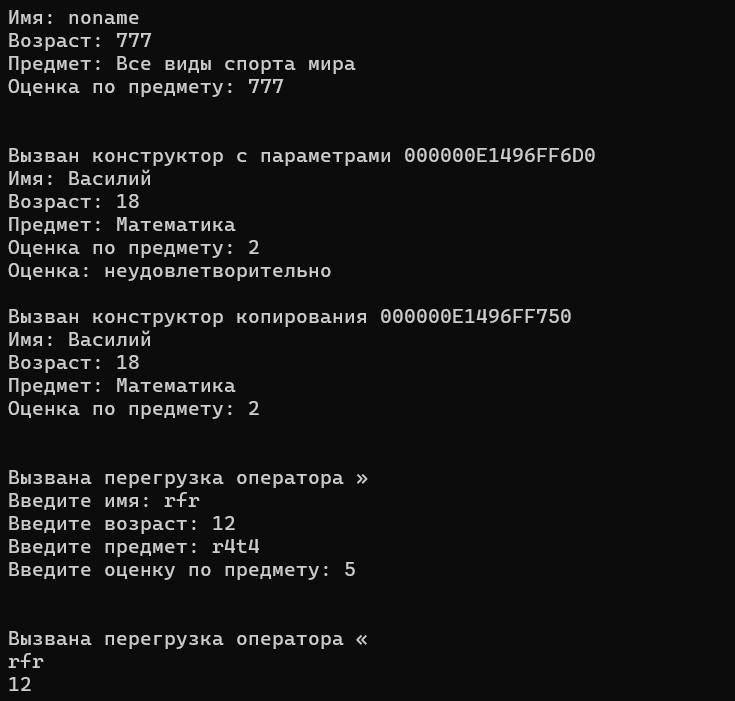
cout << G4;

cout << endl << endl;

return 0;

}

**Работа программы:**



**Ответ на вопросы:**

1. Механизм наследования используется в объектно-ориентированном программировании для создания иерархий классов, когда один класс (производный) может наследовать свойства и методы другого класса (базового).

2. Компоненты класса, описанные со спецификатором `public`, наследуются производным классом также со спецификатором `public`. Это значит, что эти компоненты будут доступны и снаружи производного класса.

3. Компоненты класса, описанные как `private`, не наследуются производным классом и остаются закрытыми для доступа извне.

4. Компоненты класса, описанные как `protected`, наследуются производным классом с таким же спецификатором `protected`.

5. Производный класс описывается с использованием ключевого слова `class`, за которым идет имя производного класса, двоеточие и ключевое слово `public` или `private`, затем имя базового класса.

6. Конструкторы наследуются, но они должны быть вызваны явно в конструкторе производного класса.

7. Деструкторы также наследуются, но если они виртуальные, то вызывается сначала деструктор производного класса, а затем базового.

8. Объекты производных классов конструируются в следующем порядке: сначала конструируется базовый класс, затем производный класс.

9. Объекты производных классов уничтожаются в обратном порядке: сначала вызывается деструктор производного, затем базового класса.

10. Виртуальные функции обеспечивают механизм позднего связывания, что позволяет использовать методы производного класса через указатель или ссылку на базовый класс.

11. Виртуальные конструкторы не могут быть, но деструкторы могут быть виртуальными. Наследуемый лицензионный спецификатор `virtual`.

12. Открытое наследование устанавливает "есть-как" отношение, где производный класс является расширением базового класса.

13. Закрытое наследование устанавливает "есть-часть" отношение, где производный класс является частным случаем базового класса.

14. Принцип подстановки заключается в том, что объект базового класса может заменить объект производного класса, если поведение базового класса соответствует поведению производного.

15. Объект `x` класса `Teacher` будет иметь следующие компоненты данных: `age` (унаследовано от `Student`), `name` (унаследовано от `Student`), `post` (унаследовано от `Employee`) и `stage`.

16. Конструкторы без параметров:

class Student {

public:

Student() : age(0), name("") {}

};

class Employee : public Student {

protected:

Employee() : post("") {}

};

class Teacher : public Employee {

protected:

Teacher() : stage(0) {}

};

17. Конструкторы с параметрами:

class Student {

public:

Student(int a, string n) : age(a), name(n) {}

};

class Employee : public Student {

protected:

Employee(int a, string n, string p) : post(p) {}

};

class Teacher : public Employee {

protected:

Teacher(int a, string n, string p, int s) : stage(s) {}

};

18. Конструкторы копирования:

class Student {

public:

Student(const Student& other) : age(other.age), name(other.name) {}

};

class Employee : public Student {

protected:

Employee(const Employee& other) : post(other.post) {}

};

class Teacher : public Employee {

protected:

Teacher(const Teacher& other) : stage(other.stage) {}

};

19. Операция присваивания:

class Student {

public:

Student& operator=(const Student& other) {

if (this != &other) {

age = other.age;

name = other.name;

}

return \*this;

}

};

class Employee : public Student {

protected:

Employee& operator=(const Employee& other) {

if (this != &other) {

static\_cast<Student&>(\*this) = other;

post = other.post;

}

return \*this;

}

};

class Teacher : public Employee {

protected:

Teacher& operator=(const Teacher& other) {

if (this != &other) {

static\_cast<Employee&>(\*this) = other;

stage = other.stage;

}

return \*this;

}

};

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

