Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Пермский национальный исследовательский политехнический университет
Электротехнический факультет
Кафедра информационных технологий и автоматизированных систем

Лабораторная работа №4

" Простое наследование. Принцип подстановки "

Вариант: 12

Выполнил студент ИВТ-24-2б:
Шишкин Максим Григорьевич

_____
(дата, подпись)

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

_____
(дата, подпись)

Пермь 2025

Содержание

1 Постановка задачи 3

2 Код на C++ 4-6

3 Результаты работы 7

4 UML-диаграмма классов 7

5 Ответы на контрольные вопросы 8-10

6 Ссылка на github..... 10

1 Постановка задачи

1. Определить пользовательский класс.
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).
5. Перегрузить операцию присваивания.
6. Перегрузить операции ввода и вывода объектов с помощью потоков.
7. Определить производный класс.
8. Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.
9. Реализовать функции, получающие и возвращающие объект базового класса. Продемонстрировать принцип подстановки.

Вариант:

Базовый класс:

ЧЕЛОВЕК (PERSON)

Имя (name) – string

Возраст (age) – int

Определить методы изменения полей. Создать производный класс STUDENT, имеющий поле год обучения. Определить методы изменения и увеличения года обучения.

2 Код на C++

Person.h | Person.cpp | Student.h | Student.cpp | main.cpp

лаба4переделка | (Глобальная область)

```cpp
#include <iostream>
#include <clocale>
#include <windows.h>
#include "Person.h"
#include "Student.h"

using namespace std;

Person processPerson(Person p) {
    cout << "Обработка человека: " << p << endl;
    p.setAge(p.getAge() + 1);
    return p;
}

int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    setlocale(LC_ALL, "Russian");

    cout << "=== Демонстрация класса Person ===" << endl;
    Person p1;
    cin >> p1;
    cout << "p1: " << p1 << endl;

    Person p2("Алиса", 25);
    cout << "p2: " << p2 << endl;

    Person p3 = p2;
    cout << "p3 (копия p2): " << p3 << endl;

    p1 = p3;
    cout << "p1 после присваивания: " << p1 << endl;

    cout << "\n=== Демонстрация класса Student ===" << endl;
    Student s1;
    cin >> s1;
    cout << "s1: " << s1 << endl;

    Student s2("Боб", 20, 2);
    cout << "s2: " << s2 << endl;

    s2.incrementStudyYear();
    cout << "s2 после увеличения года обучения: " << s2 << endl;

    Student s3 = s2;
    cout << "s3 (копия s2): " << s3 << endl;

    s1 = s3;
    cout << "s1 после присваивания: " << s1 << endl;

    cout << "\n=== Демонстрация принципа подстановки ===" << endl;
    Person* personPtr = new Student("Чарли", 22, 3);
    cout << *personPtr << endl;
    delete personPtr;

    cout << "\n=== Демонстрация работы функции ===" << endl;
    Person p4 = processPerson(p2);
    cout << "После обработки: " << p4 << endl;

    Student s4("Диана", 19, 1);
    Person p5 = processPerson(s4);
    cout << "После обработки студента как человека: " << p5 << endl;

    return 0;
}
```

Person.h Person.cpp Student.h **Student.cpp** main.cpp

лаба4переделка (Глобальная область)

```cpp
#include "Student.h"

Student::Student() : Person(), studyYear(1) {}
Student::Student(const string& n, int a, int year) : Person(n, a), studyYear(year) {}
Student::Student(const Student& s) : Person(s), studyYear(s.studyYear) {}
Student::~Student() {}

int Student::getStudyYear() const { return studyYear; }
void Student::setStudyYear(int year) { studyYear = year; }
void Student::incrementStudyYear() { studyYear++; }

Student& Student::operator=(const Student& s) {
    if (this != &s) {
        Person::operator=(s);
        studyYear = s.studyYear;
    }
    return *this;
}

ostream& operator<<(ostream& os, const Student& s) {
    os << static_cast<const Person&>(s) << ", Год обучения: " << s.studyYear;
    return os;
}

istream& operator>>(istream& is, Student& s) {
    is >> static_cast<Person&>(s);
    cout << "Введите год обучения: ";
    is >> s.studyYear;
    is.ignore();
    return is;
}
```

Person.h Person.cpp **Student.h** Student.cpp main.cpp

лаба4переделка (Глобальная облас

```cpp
#pragma once
#include "Person.h"

class Student : public Person {
private:
    int studyYear;

public:
    Student();
    Student(const string& n, int a, int year);
    Student(const Student& s);
    ~Student() override;

    int getStudyYear() const;
    void setStudyYear(int year);
    void incrementStudyYear();

    Student& operator=(const Student& s);

    friend ostream& operator<<(ostream& os, const Student& s);
    friend istream& operator>>(istream& is, Student& s);
};
```

Person.h | Person.cpp | Student.h | Student.cpp | main.cpp

лаба4переделка

```cpp
#include "Person.h"

Person::Person() : name(""), age(0) {}
Person::Person(const string& n, int a) : name(n), age(a) {}
Person::Person(const Person& p) : name(p.name), age(p.age) {}
Person::~Person() {}

string Person::getName() const { return name; }
int Person::getAge() const { return age; }
void Person::setName(const string& n) { name = n; }
void Person::setAge(int a) { age = a; }

Person& Person::operator=(const Person& p) {
    if (this != &p) {
        name = p.name;
        age = p.age;
    }
    return *this;
}

ostream& operator<<(ostream& os, const Person& p) {
    os << "Имя: " << p.name << ", Возраст: " << p.age;
    return os;
}

istream& operator>>(istream& is, Person& p) {
    cout << "Введите имя: ";
    getline(is, p.name);
    cout << "Введите возраст: ";
    is >> p.age;
    is.ignore();
    return is;
}
```

Person.h | Person.cpp | Student.h | Student.cpp | main.cpp

лаба4переделка

```cpp
#pragma once
#include <iostream>
#include <string>
using namespace std;

class Person {
protected:
    string name;
    int age;

public:
    Person();
    Person(const string& n, int a);
    Person(const Person& p);
    virtual ~Person();

    string getName() const;
    int getAge() const;

    void setName(const string& n);
    void setAge(int a);

    Person& operator=(const Person& p);

    friend ostream& operator<<(ostream& os, const Person& p);
    friend istream& operator>>(istream& is, Person& p);
};
```

3 Результаты работы

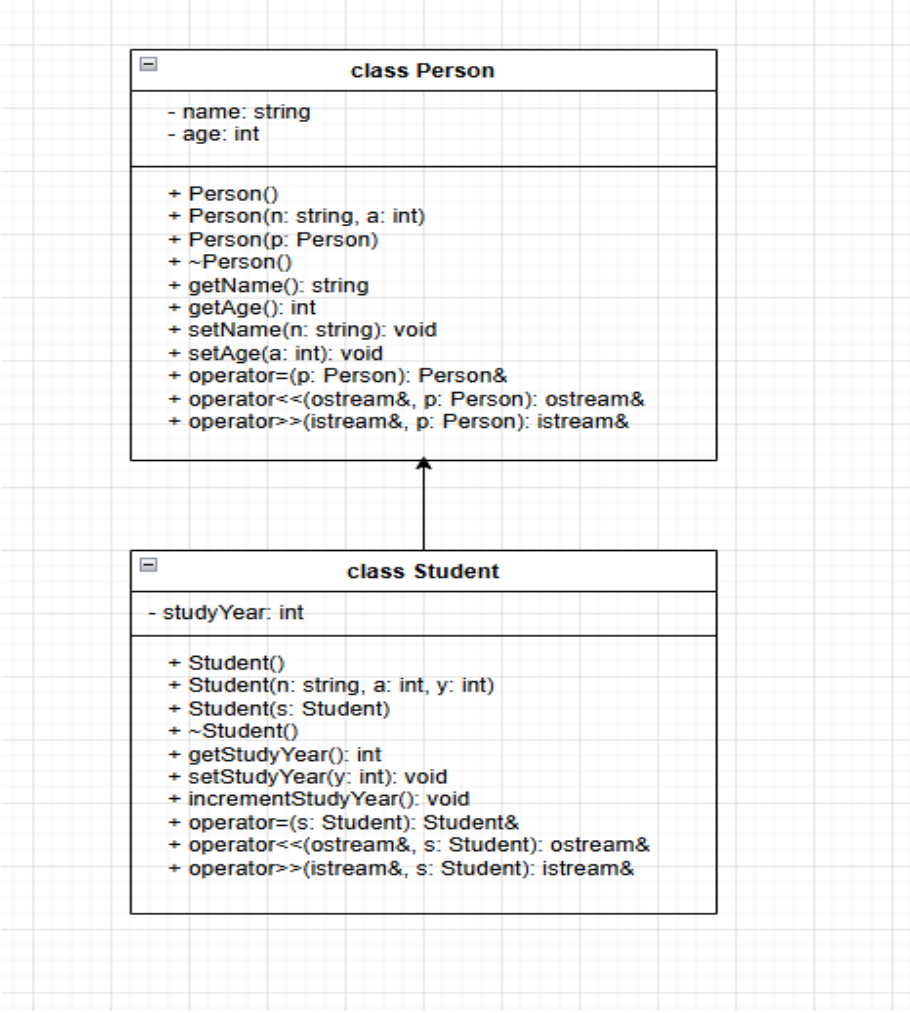
```
Введите имя: bob
Введите возраст: 12
p1: Имя: bob, Возраст: 12
p2: Имя: Алиса, Возраст: 25
p3 (копия p2): Имя: Алиса, Возраст: 25
p1 после присваивания: Имя: Алиса, Возраст: 25

=== Демонстрация класса Student ===
Введите имя: Den
Введите возраст: 13
Введите год обучения: 2025
s1: Имя: Den, Возраст: 13, Год обучения: 2025
s2: Имя: Боб, Возраст: 20, Год обучения: 2
s2 после увеличения года обучения: Имя: Боб, Возраст: 20, Год обучения: 3
s3 (копия s2): Имя: Боб, Возраст: 20, Год обучения: 3
s1 после присваивания: Имя: Боб, Возраст: 20, Год обучения: 3

=== Демонстрация принципа подстановки ===
Имя: Чарли, Возраст: 22

=== Демонстрация работы функции ===
Обработка человека: Имя: Алиса, Возраст: 25
После обработки: Имя: Алиса, Возраст: 26
Обработка человека: Имя: Диана, Возраст: 19
После обработки студента как человека: Имя: Диана, Возраст: 20
```

4 UML-диаграмма классов



5 Ответы на контрольные вопросы

1 -Для повторного использования кода и реализации полиморфизма.

2 - Наследуются как public.

3 - Не наследуются.

4 - Наследуются как protected.

5 - class Derived : public Base { ... };

6 - Нет, но могут быть унаследованы через using.

7 - Да, особенно при использовании virtual.

8 - Сначала базовый, потом производный.

9 - Сначала производный, потом базовый.

10 -Это механизм вызова переопределённой функции через указатель/ссылку на базовый класс.

11 - Конструкторы — нет, деструкторы — да.

12 - Да.

13 - Отношение "является" (is-a).

14 - Отношение "реализовано через" (has-a), доступ ограничен.

15 - Объект производного класса может использоваться вместо объекта базового.

16 - age, name, post, stage.

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

Employee() : Student(), post("") {}

Teacher() : Employee(), stage(0) {}

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

Employee() : Student(), post("") {}

Teacher() : Employee(), stage(0) {}

18 - Student(int a, const string& n) : age(a), name(n) {}

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

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

19 - Student(const Student& s) : age(s.age), name(s.name) {}

Employee(const Employee& e) : Student(e), post(e.post) {}

Teacher(const Teacher& t) : Employee(t), stage(t.stage) {}

```
20 - Student& operator=(const Student& s) {
   if (this != &s) { age = s.age; name = s.name; }
   return *this;
}
Employee& operator=(const Employee& e) {
   if (this != &e) {
      Student::operator=(e);
      post = e.post;
   }
   return *this;
}
Teacher& operator=(const Teacher& t) {
   if (this != &t) {
      Employee::operator=(t);
      stage = t.stage;
 }
   return *this;
}
```

6 Ссылка на github

ссылка на github - https://github.com/MAKSPOWERO/mas1