Федеральное государственное автономное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

Лабораторная работа №4
«Простое наследование. Принцип подстановки»

Выполнил:
студент первого курса
ЭТФ группы РИС-23-3б
Акбашева Софья Руслановна

Проверила:
Доцент кафедры ИТАС О. А. Полякова

Пермь, 2024

Простое наследование. Принцип подстановки

Цель задания

1) Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.

2) Создание иерархии классов с использованием простого наследования.

3) Изучение принципа подстановки.

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.

Задание

Базовый класс:

ЧЕЛОВЕК (PERSON)

Имя (name) – string Возраст (age) – int

Определить методы изменения полей.

Создать производный класс STUDENT, имеющий поля Предмет – string и Оценка – int. Определить методы изменения полей и метод, выдающий сообщение о неудовлетворительной оценке.

Анализ задачи

1) Использование директивы #pragma once предотвращает повторную загрузку заголовочного файла, что может улучшить производительность программы, особенно при использовании множества включаемых файлов.

2) Необходимо реализовать класс Person и дочерний для него класс Student.

3) В классах надо прописать конструкторы: по умолчанию, с параметрами и копирования; а также деструктор, для предотвращения утечки памяти.

4) Класс Person содержит защищенные данные name и age, что предотвращает их изменение извне класса. Также определены конструктор по умолчанию, конструктор с параметрами, конструктор копирования и деструктор. Предоставлены функции для установки значений и вывода информации о персоне. Определён оператор присваивания и дружественные функции для работы с потоками ввода/вывода.

5) Класс Student является производным от Person. Он добавляет свои собственные защищенные данные subject и mark, а также предоставляет дополнительные функции для установки этих значений и вывода информации о студенте. Определены конструктор с параметрами, функции для установки предмета и оценки, а также дружественные функции для работы с потоками ввода/вывода.

6) Функция good_bad_mark() в классе Student предназначена для вывода сообщения о низкой оценке.

7) Перегруженные операторы +, -, = и << позволяют выполнять различные операции с объектами класса. Например, оператор + позволяет

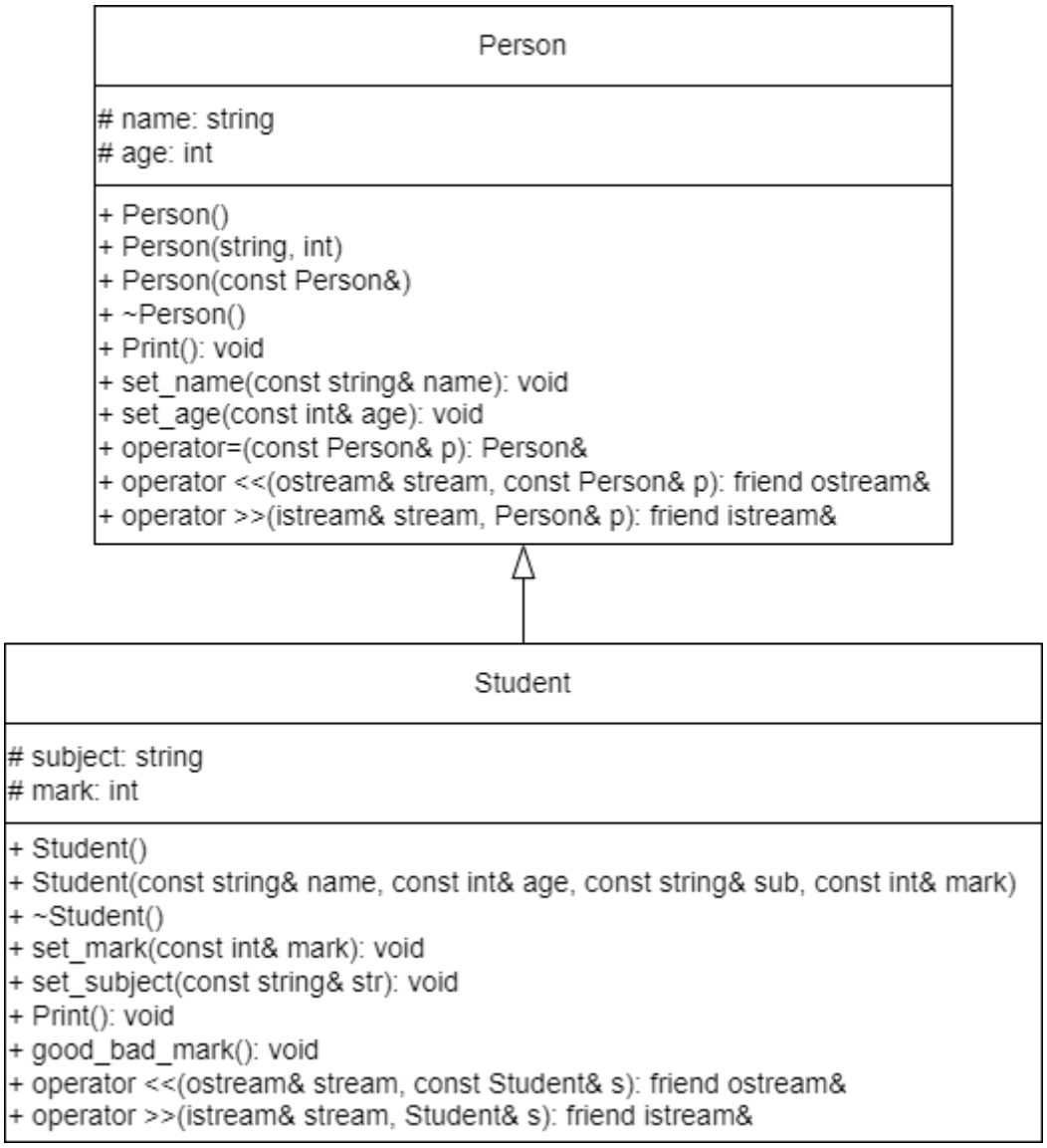
складывать целое число с первым полем объекта класса и вещественное число со вторым полем соответственно. Оператор << позволяет выводить объекты класса в поток вывода.

8) Методы set_name() и set_age() используются для установки имени и возраста соответственно.

9) Метод set_mark() и set_subject() используются для установки предмета и оценки соответственно.

10) В главной функции показаны операции, которые можно выполнить с объектом классов

Блок схема



Код

Файл Class_4.cpp

```cpp
#include <iostream>
#include <string>
#include "person.h"

int main(){
   system("chcp 1251 > Null");

   Person p1("Иванов Иван", 5);
   p1.Print();
   Person p3;
   p3.Print();
   Person p4(p1);
   p4.Print();

   cout << "------------------" << endl << endl;

   Student st1("Сергеев Сергей", 5, "math", 2);
   cout << st1;
   cout << endl;

   Student st2;
   cout << st2 << endl;
   cin >> st2;
   cout << endl;

   Student st3(st1);
   cout << st3;
   cout << "------------------" << endl << endl;

   st1.Print();
   st2.Print();
   st3.Print();
}
```

Файл person.cpp

```cpp
#include "person.h"
#include <iostream>
#include <string>
using namespace std;

Person::Person() {//конструктор по умолчанию
        this->name = "";
        this->age = 0;
        cout << "Конструктор по умолчанию для объекта " << this << endl;
}

Person::Person(string name, int age) {//конструктор с параметрами
        this->name = name; //сначала передается имя, потом возраст
        this->age = age;
        cout << "Конструктор с параметрами для объекта " << this << endl;
}

Person::Person(const Person& p) {
        this->name = p.name;
        this->age = p.age;
        cout << "Конструктор копирования для объекта " << this << endl;
}

Person::~Person() {//деструктор
        cout << "Деструктор для объекта " << this << endl;
}
```

```
void Person::Print() {//вывод параметров класса person
	cout << "ФИО: " << this->name << endl << "Возраст: " << this->age << endl << endl;
}

void Person::set_name(const string& name) {//установка имени
	this->name = name;
}

void Person::set_age(const int& age) {//установка возраста
	this->age = age;
}

Person& Person::operator=(const Person& p) {//оператор присваивания
	this->name = p.name;
	this->age = p.age;
	return *this;
};

ostream& operator <<(ostream& stream, const Person& p) {//оператор для потока вывода
	stream << "Имя " << p.name << endl;
	stream << "Возраст " << p.age << endl;
	return stream;
}

istream& operator >>(istream& stream, Person& p) {//оператор для потока ввода
	string temp;
	int a;

	cout << "Введите имя: ";
	stream.ignore();
	getline(stream, temp);
	p.set_name(temp);

	cout << "Введите возраст: ";
	stream >> a;
	p.set_age(a);

	return stream;
}
```

Файл student.cpp

```
#include "person.h"
#include <iostream>
#include <string>
using namespace std;

Student::Student() {
	this->mark = 0;
	this->subject = "";
	cout << "Конструктор по умолчанию для объекта " << this << endl;
}

Student::Student(const string& name, const int& age, const string& sub, const int& mark) {
	this->set_name(name);
	this->set_age(age);
	this->set_mark(mark);
	this->set_subject(sub);
	cout << "Конструктор с параметрами для объекта " << this << endl;
}

void Student::set_mark(const int& mark) {//установка предмета
```

```
	this->mark = mark;
}

void Student::set_subject(const string& str) {//установка оценки
	this->subject = str;
};

void Student::Print() {//вывод параметров класса student
	cout << "ФИО: " << this->name << endl << "Возраст: " << this->age << endl;
	cout << "Предмет: " << this->subject << endl << "Оценка: " << this->mark << endl;
	(*this).Student::good_bad_mark();
	cout << endl;
}

void Student::good_bad_mark() { //сообщение о низкой оценке
	if (this->mark < 3) {
		cout << "Отметка неудовлетворительна!" << endl;
	}
}

ostream& operator <<(ostream& stream, const Student& s) {//оператор для потока вывода
	const Person* human_stud = &s;
	stream << *human_stud;
	stream << "Предмет " << s.subject << endl;
	stream << "Оценка " << s.mark << endl;

	return stream;
}

istream& operator >>(istream& stream, Student& s) {//оператор для потока ввода
	string temp;
	int a;

	Person* const human_stud = &s;
	stream >> *human_stud;

	cout << "Предмет: ";
	stream >> temp;
	s.set_subject(temp);

	cout << "Оценка: ";
	stream >> a;
	s.set_mark(a);

	cout << endl << endl;

	return stream;
	}
```

Файл person.h

```
#pragma once //предотвращает повторную загрузку заголовочного файла, если он уже был включен
#include <iostream> //стандартные потоки ввода и вывода
#include <string>
using namespace std;

class Person {
protected:
	string name;
	int age;
public:
	Person();//конструктор по умолчанию
```

```
        Person(string, int);//сначала передается имя, потом возраст
        Person(const Person&);//конструктор копирования
        ~Person();//деструктор

        void Print(); //вывод
        void set_name(const string& name);//установка имени
        void set_age(const int& age);//установка возраста

        Person& operator=(const Person& p);//оператор присваивания

        friend ostream& operator <<(ostream& stream, const Person& p);//друж-я ф-я для потока вывода
        friend istream& operator >>(istream& stream, Person& p);//друж-я ф-я для потока ввода
};

class Student :public Person {
protected:
        string subject;
        int mark;
public:
        Student();//конструктор по умолчанию
        Student(const string& name, const int& age, const string& sub, const int& mark);
        void set_mark(const int& mark);//установка предмета
        void set_subject(const string& str); //установка оценки
        void Print();//вывод

        void good_bad_mark();//сообщение о низкой оценке

        friend ostream& operator <<(ostream& stream, const Student& s);//друж-я ф-я для потока вывода
        friend istream& operator >>(istream& stream, Student& s);//друж-я ф-я для потока ввода
};
```

Результат работы

```
Конструктор с параметрами для объекта 00000006FC9FF498
ФИО: Иванов Иван
Возраст: 5

Конструктор по умолчанию для объекта 00000006FC9FF4E8
ФИО:
Возраст: 0

Конструктор копирования для объекта 00000006FC9FF538
ФИО: Иванов Иван
Возраст: 5

------------------

Конструктор по умолчанию для объекта 00000006FC9FF590
Конструктор с параметрами для объекта 00000006FC9FF590
Имя Сергеев Сергей
Возраст 5
Предмет math
Оценка 2

Конструктор по умолчанию для объекта 00000006FC9FF610
Конструктор по умолчанию для объекта 00000006FC9FF610
Имя
Возраст 0
Предмет
Оценка 0

Введите имя: Андрей Смирнов
Введите возраст: 19
Предмет: inform
Оценка: 5



Конструктор копирования для объекта 00000006FC9FF690
Имя Сергеев Сергей
Возраст 5
Предмет math
Оценка 2
------------------
```

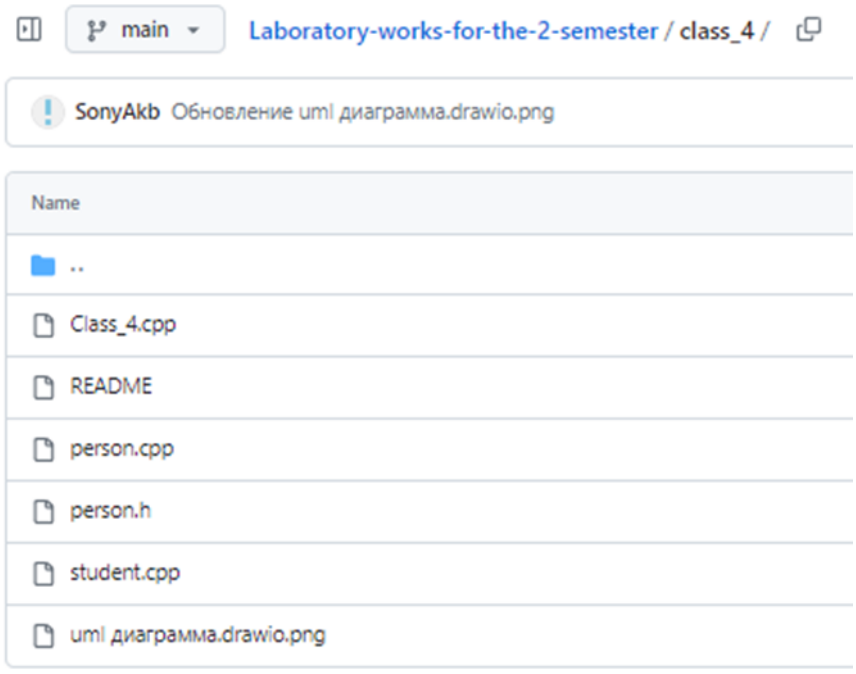
```
ФИО: Сергеев Сергей
Возраст: 5
Предмет: math
Оценка: 2
Отметка неудовлетворительна!

ФИО: ндрей Смирнов
Возраст: 19
Предмет: inform
Оценка: 5

ФИО: Сергеев Сергей
Возраст: 5
Предмет: math
Оценка: 2
Отметка неудовлетворительна!

Деструктор для объекта 00000006FC9FF690
Деструктор для объекта 00000006FC9FF610
Деструктор для объекта 00000006FC9FF590
Деструктор для объекта 00000006FC9FF538
Деструктор для объекта 00000006FC9FF4E8
Деструктор для объекта 00000006FC9FF498
```

Вывод

В ходе работы я применила знания о работе с классами. В частности, о простом наследовании с помощью принципа подстановки. По итогу работы была реализованная поставленная задача.

GitHub

Ссылка: https://github.com/SonyAkb/Laboratory-works-for-the-2-semester/tree/main/class_4



Контрольные вопросы

1. Для чего используется механизм наследования?

Механизм наследования в C++ используется для создания иерархий классов, позволяющих повторно использовать код и обеспечивающих возможность расширения функциональности классов без изменения их исходного кода. Это позволяет создавать более гибкие и мощные программы.

2. Каким образом наследуются компоненты класса, описанные со спецификатором public?

Компоненты класса, описанные со спецификатором public, наследуются открыто. Это означает, что они доступны для использования в производном классе так же, как и в базовом классе. Они могут быть переопределены в производном классе, если это необходимо.

3. Каким образом наследуются компоненты класса, описанные со спецификатором private?

Компоненты класса, описанные со спецификатором private, не наследуются. Они являются приватными членами базового класса и недоступны для прямого доступа или изменения в производном классе.

4. Каким образом наследуются компоненты класса, описанные со спецификатором protected?

Компоненты класса, описанные со спецификатором protected, наследуются защищенно. Они доступны для использования в производном классе, но не доступны напрямую извне производного класса. Защищенные члены могут быть переопределены в производном классе.

5. Каким образом описывается производный класс?

Производный класс описывается путем объявления его базовым классом.

```
class Person
{
public:
    void print() const
    {
        std::cout << "Name: " << name << "\tAge: " << age << std::endl;
    }
    std::string name;       //  имя
    unsigned age;           // возраст
};
class Employee : public Person
{
public:
    std::string company;    // компания
};
```

6. Наследуются ли конструкторы?

Конструкторы не наследуются, а создаются заново в каждом производном классе.

7. Наследуются ли деструкторы?

Не наследуются, но они вызываются, когда дочерний класс инициализирует свой объект.

8. В каком порядке конструируются объекты производных классов?

Объекты класса конструируются снизу вверх: сначала базовый, потом компоненты-объекты (если они имеются), а потом сам производный класс. Таким образом объект производного класса содержит в качестве подобъекта объект базового класса.

9. В каком порядке уничтожаются объекты производных классов?

Деструкторы вызываются в обратном порядке от конструкторов.

10. Что представляют собой виртуальные функции и механизм позднего связывания?

Виртуальные функции и механизм позднего связывания позволяют полиморфно вызывать функцию, определенную в базовом классе, через объект производного класса. Это означает, что во время выполнения программы будет выбрана правильная версия функции, соответствующая типу объекта, даже если вызывается функция базового класса.

11. Могут ли быть виртуальными конструкторы? Деструкторы?

Конструкторы нет, деструкторы да.

12. Наследуется ли спецификатор virtual?

Да.

13. Какое отношение устанавливает между классами открытое наследование?

Все члены базового класса становятся доступными в производном классе. Это позволяет производному классу полностью использовать функциональность базового класса.

14. Какое отношение устанавливает между классами закрытое наследование?

Производный класс не имеет доступа к членам базового класса, описанным как private. Это ограничивает возможности производного класса по использованию функциональности базового класса. Закрытое (также как и защищенное) наследование не создает иерархии типов.

15. В чем заключается принцип подстановки?

Принцип подстановки гласит, что если два выражения имеют одно и то же значение, они могут быть взаимозаменяемыми в любом контексте. В контексте наследования это означает, что если объект производного класса может быть использован там, где ожидается объект базового класса, благодаря полиморфизму, это не должно привести к ошибкам во время выполнения программы.

16. Имеется иерархия классов: class Student

{

int age;

public:

};

string name;

class Employee : public Student

{

protected:

};

string post;

class Teacher public Employee

{

protected: int stage;

};

Teacher x;

Какие компонентные данные будет иметь объект x?

Name и Stage, private age не наследуется, post наследуется в private секции.

17. Для классов Student, Employee и Teacher написать конструкторы без параметров.

```
public: string name;
    Student() {
        age = 0;
        name = "none";
    }
```

```
public:
    Employee() {
        post = "none";
    }
};

public:
    Teacher() {
        stage = 0;
    }
```

18. Для классов Student, Employee и Teacher написать конструкторы с параметрами.

```
Student(int a, string n) {
    this->age = a;
    this->name = n;
}


Employee(string p) {
    this->post = p;
}

Teacher(int s) {
    this->stage = s;
}
```

19. Для классов Student, Employee и Teacher написать конструкторы копирования.

```
Student(const Student& t) {
    this->age = t.age;
    this->name = t.name;
}


Employee(const Employee& t) {
    this->post = t.post;
}


Teacher(const Teacher& t) {
    this->stage = t.stage;
}
```

20. Для классов Student, Employee и Teacher определить операцию присваивания.

```
Student& operator=(const Student& t) {
    this->age = t.age;
    this->name = t.name;
    return *this;
}
```

```cpp
Employee& operator=(const Employee& t) {
    this->post = t.post;
    return *this;
}


Teacher& operator=(const Teacher& t) {
    this->stage = t.stage;
    return *this;
}
```