**Министерство высшего образования и науки Российской Федерации**

**ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (ПНИПУ)**

**Электротехнический факультет**

**Кафедра Информационные технологии и автоматизированные системы**

**Отчёт по теме**

**«ООП. Классы и объекты»**

**«Виртуальные функции. Полиморфизм»**

**Выполнил:**

**Студент 1 курса**

**Группы ИВТ-22-2б**

**Корючкин Савелий**

**Научный руководитель:**

**Доцент кафедры ИТАС**

**Полякова Ольга Андреевна**

**Пермь 2022**

**Постановка задачи**

1. Определить абстрактный класс.
2. Определить иерархию классов, в основе которой будет находиться абстрактный класс.
3. Определить класс Вектор, элементами которого будут указатели на объекты иерархии классов.
4. Перегрузить для класса Вектор операцию вывода объектов с помощью потоков.
5. В основной функции продемонстрировать перегруженные операции и полиморфизм Вектора.

**Вариант 14**

Базовый класс: ЧЕЛОВЕК (PERSON)

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

Определить методы изменения полей.

Создать производный класс TEACHER, имеющий поля Предмет – string и Количество часов – int. Определить методы изменения полей, а также увеличения и уменьшения часов.

**Описание классов**

Абстрактный класс Object. Public-методы: конструктор и деструктор; чисто виртуальный метод Show().

Базовый класс Person. Protected-поля: string name, int age.

Public-методы: 3 вида конструкторов и деструктор; геттеры и сеттеры для полей; переопределенный метод Show() для просмотра атрибутов; перегрузка оператора присваивания, перегрузка операторов ввода/вывода.

Производный класс Student. Protected-поля: string subject, int hours.

Public-методы: конструкторы и деструктор; геттеры и сеттеры; метод Change\_Hours(), переопределение методы Show(); перегрузка операторов присваивания и ввода/вывода.

Класс Vector для хранения указателей на объекты иерархии классов. Private-поля: Object\*\* beg, int size, int cur.

Public-методы: конструкторы и деструктор; метод Add() для добавления элементов в вектор; перегрузка оператора вывода.



Рис. 1 – UML-диаграмма

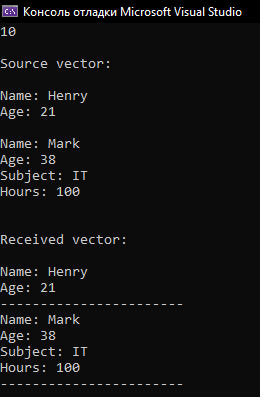


Рис. 2 – Результат работы программы

**Контрольные вопросы**

*1. Какой метод называется чисто виртуальным? Чем он отличается от виртуального метода?*

Чисто виртуальный метод – это функция без определения или реализации. Виртуальный метод (виртуальная функция) — это метод класса, который может быть переопределён в классах-наследниках так, что конкретная реализация метода для вызова будет определяться во время исполнения.

*Пример чисто виртуального метода:*

virtual void Show() = 0;

*2. Какой класс называется абстрактным?*

Абстрактный класс – базовый класс, который не предполагает создания экземпляров. Класс, содержащий хотя бы один Чисто виртуальный метод, будет [абстрактным](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BB%D0%B0%D1%81%D1%81).

*Пример:*

class Object

{

public:

Object();

~Object();

virtual void Show() = 0; //Чисто виртуальная функция

};

*3. Для чего предназначены абстрактные классы?*

Абстрактные классы предназначены для создания обобщенных  сущностей, на основе которых в дальнейшем предполагается создавать более конкретные производные классы.

*Пример:*

Нужно создать для класса TextFile базовый класс File, от которого будет унаследован еще один класс RTFFile. Однако, в такой ситуации неизвестно как реализовать метод read() класса File, т.к. класс File не реализует поведение какого-то конкретного типа файлов, а представляет интерфейс для работы с различными файлами. Для этого используется чисто виртуальная функция.

class File {

virtual string read(int count) = 0;

};

Метод read(...) должен быть определен в классах наследниках. Теперь класс File стал абстрактным, и его экземпляры невозможно создать. Но можно работать через указатель на абстрактный класс с объектами производных классов.

File\* f = new TextFile("text.txt");

//различные действия с файлом text.txt

delete f;

f = new RTFFile("rich\_text.rtf");

//различные действия с файлом rich\_text.rtf

delete f;

*4. Что такое полиморфные функции?*

При работе с абстрактными классами можно создать функцию, параметром которой будет являться указатель на абстрактный класс. Туда может передаваться указатель на объект любого производного класса, что позволяет создать полиморфные функции, то есть функции, работающие с объектом любого типа в пределах одной иерархии.

class Base {

virtual void work() = 0;

};

class Derived : public Base { ... };

void print(Base\* a) { cout << "Hello!"; }

int main() {

Derived\* t = new Derived;

print(t); //Функция работает, хотя в качестве параметра передается

//указатель на объект не абстрактного класса

return 0;

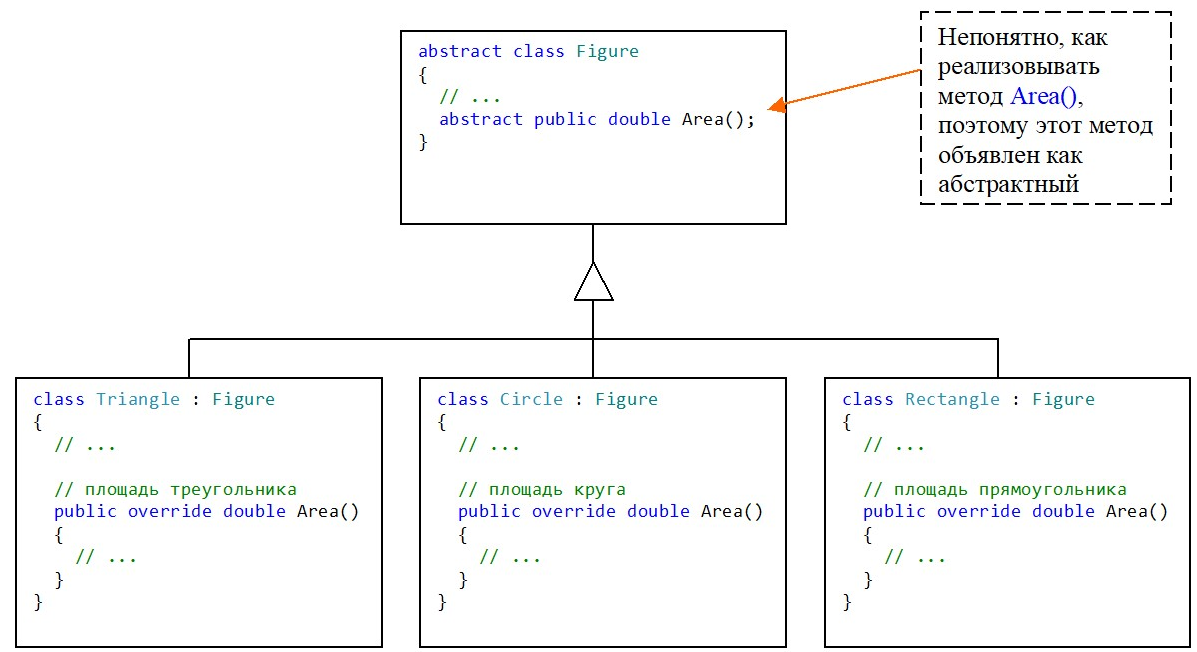
*5. Чем полиморфизм отличается от принципа подстановки?*

Полиморфизм - возможность иметь разные формы для одной и той же сущности.

Принцип подстановки: функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа не зная об этом.

Например, если у нас есть класс A и унаследованный от него класс B, то если мы заменим все использования класса A на B, ничего не должно измениться в работе программы. Ведь класс B всего лишь расширяет функционал класса A.

*6. Привести примеры иерархий с использованием абстрактных классов.*



*7. Привести примеры полиморфных функций.*

Вопрос 4: функция void print(Base\* a) { cout << "Hello!"; }

*8. В каких случаях используется механизм позднего связывания?*

В случаях вызова полиморфной функции с типом параметра класса-наследника.