4 лабораторная работа.

Теоретические сведения.

Наследование – один из основополагающих принципов ООП. Благодаря наследованию класс может использовать переменные и методы другого класса. Наследование позволяет избежать постоянного переписывания одного и того же кода, что ускоряет процесс разработки.

Класс, от которого наследуют, называется родительским (или базовый), а класс, который наследует данные от родительского класса, называется дочерним (или наследник).

В некоторых языках программирования, в том числе и в C++, возможно множественное наследование – наследование происходит от двух и более родительских классов и дочерний класс обладает всеми свойствами родительских классов.

Важным фактом при наследовании является то, что при изменении родительского класса, изменения будут транслироваться и в дочерние классы.

При наследовании классов тоже используются модификаторы доступа. Поведение модификаторов доступа для полей и методов класса происходит абсолютно одинаково.

Виды наследования (модификаторы наследования/доступа):

– публичный (**public**)- доступ к полям и методам класса возможен как в наследном классе, так и в объекте наследного класса.

– защищенный (**protected**) — доступ к полям и методам класса возможен в наследном классе, но не в объекте наследного класса.

– приватный (**private**) — все унаследованные данные становятся приватными, доступа у класса-наследника к полям и методам родительского класса с таким модификатором доступа нет. С полями и методами данного модификатора можно взаимодействовать только из класса-родителя, где было создано поле/метод с этим типом доступа.

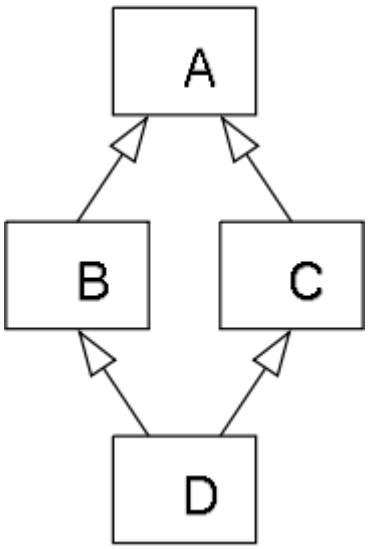
Модификаторы доступа и их влияние при наследовании классов приведены в таблице 1.

Модификатор доступа	Модификатор в родительском классе	Модификатор в дочернем классе	Комментарий
Public	Public	Public	Без изменений
	Protected	Protected	
	Private	Private	
Protected	Public	*Protected*	Public -> protected, Private без изменений
	Protected	Protected	
	Private	Private	
Private	Public	*Private*	Все private
	Protected	*Private*	
	Private	Private	

Множественное наследование требует тщательного проектирования, так как может привести к непредвиденным последствиям. Большинство таких последствий вызваны неоднозначностью в наследовании.

Несмотря на то, что приватные данные не наследуются, разрешить неоднозначное наследование изменением уровня доступа к данным на приватный невозможно. При компиляции, сначала происходит поиск метода или переменной, а уже после — проверка уровня доступа к ним.

Проблема ромба.



Проблема ромба *(Diamond problem)*- классическая проблема в языках, которые поддерживают возможность множественного наследования. Эта проблема возникает, когда классы **B** и **C** наследуют **A**, а класс **D** наследует **B** и **C**.

К примеру, из класса **A** в классы **B** и **C** наследуется метод **PrntSmth**(), где каждый класс переопределяет этот метод. Если **PrntSmth**() будет вызываться классом **D** (наследник классов **B** и C), не понятно, какой метод должен быть вызван — метод класса **A**, **B** или **C**.

Разные языки по-разному подходят к решению ромбовидной проблем, однако в C ++ решение проблемы оставлено на усмотрение программиста.

Проблема ромба — прежде всего проблема дизайна, и должна быть предусмотрена на этапе проектирования.

Но если все-таки произошла данная ситуация, то возможные разрешения проблемы ромба выглядят следующим образом:

- вызвать метод конкретного родительского класса;
- обратиться к объекту подкласса как к объекту определенного родительского класса;
- переопределить проблематичный метод в последнем дочернем классе.

Задание.

I.

---

1. Создайте класс Parent

```
class Parent {
    public:
    string msg1 = "A";
    private:
    string msg2 = "B";
    protected:
    string msg3 = "C";
};
```

2. Создайте наследный класс Child и добавьте методы, которые выводят в консоль поля msg1 – msg3. Попробуйте поменять модификаторы доступа при наследовании родительского класса. Попробуйте вывести все поля и посмотрите, что получится. Объясните полученные результаты.

II.

---

1. Создать классы Table, Chair, Mouse, Computer, Monitor, которые наследуются из класса Inventorization.
2. Добавить поля, характерные только для дочернего класса.
3. Добавить новый метод в каждый класс, который взаимодействует с добавленными полями.
4. Добавить новый метод в родительский класс и переопределить его в дочерних.
5. Проиллюстрировать работу.