Лабораторная работа №5

«Множественное наследование в языке С++»

Цель работы

Получение практических навыков при использовании множественного наследования в языке С++.

Закрепить знания по теме: Классы, наследование классов, виртуальные функции, абстрактные классы.

Выбор варианта задания

Определить вариант задания, равный порядковому номеру студента в журнале (взять свой порядковый номер по модулю количества вариантов).

Описание работы

В работе необходимо построить иерархию классов согласно схеме наследования, приведенной в варианте задания.

Каждый класс должен содержать:

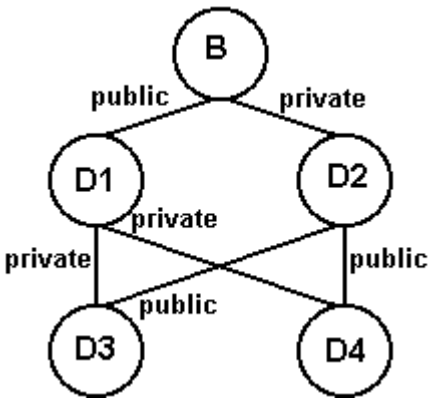
- инициализирующий конструктор

- функцию ***show*** для вывода значений.

- деструктор

Функция ***main*** должна иллюстрировать иерархию наследования.

Пусть задана следующая схема наследования классов:



1) Сначала создадим необходимую иерархию классов.

Базовый класс может быть задан только один раз в списке порождения нового класса. Однако базовый класс может встретиться несколько раз в иерархии порождения, тогда он (базовый класс) определяется как ***виртуальный*** заданием ключевого слова virtual в списке порождения перед именем базового класса или указанием типа наследования.

Следуя ей мы получим, например для класса D4:

```
class D4 : public D2, private D1
{
     int e;
public:
};
```

2) Создадим теперь во всех классах конструкторы, которые смогли бы по цепочке наследования инициализировать свои переменные и передавать остальные значения дальше вверх вплоть до конструктора базового класса. Также в конструкторы следует добавить вывод сообщения о том, что работает соответствующий конструктор.

Например:

```
D4(int x, int y, int z, int i, int j) : D1(y, z), D2(i, j)
{
   e = x;
   cout << "Работает конструктор D4: параметр = " << e << "\n";
}
```

Следует вопрос: «Если D1 и D2 совместно используют родительский класс B, то кто ответственный за его создание?» Оказывается класс D3 (при создании объекта класса D3) и класс D4 (при создании объекта класса D4). Конструктор D3 отвечает за создание объекта B. Это один из тех случает, когда дочернему классу разрешено вызывать конструктор родительского класса, который не является его непосредственным родителем. Конструкторы D1 и D2 по прежнему вызывают конструктор B. При создании объекта D3 эти вызовы конструктора просто игнорируются, так как именно D3 отвечает за создание B, а не D1 или D2. Однако если бы мы создавали объекты D1 или D2, то эти конструкторы вызывались бы и применялись бы обычные правила наследования. Точно также конструктор D4 отвечает за создание объекта B.
Поэтому следует предусмотреть способ передачи параметра классу B.

3) Теперь добавим в каждый класс функцию ***show***, которая бы выводила на экран переменную из секции ***private*** класса, которому принадлежит сама и вызывала бы функции ***show*** тех классов, которые стоят выше по иерархии наследования.

Например:

```
void show_D4()
{
    cout << "D4=  " << e << "\n"; show_D1(); show_D2();
}
```

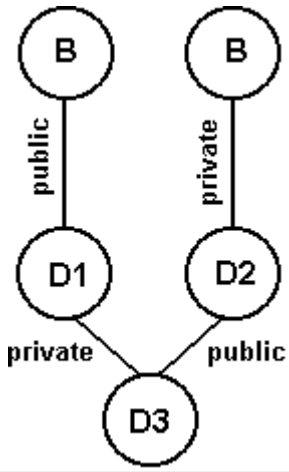
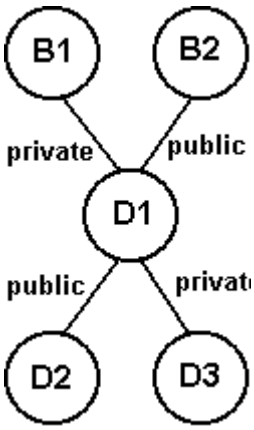
4) Не забываем про деструкторы для каждого класса.

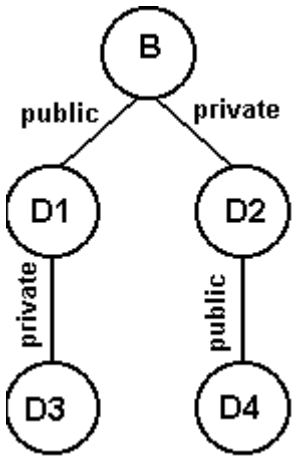
5) В функции main() создадим объекты производных классов, находящихся на нижней ступени иерархии.

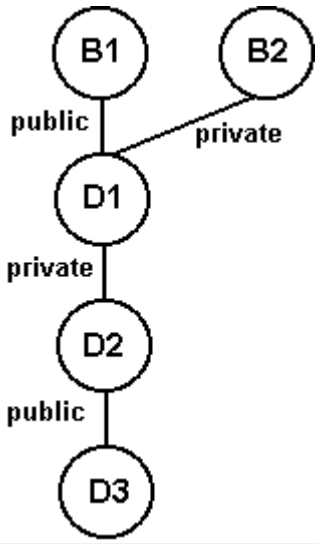
Например, для нашей иерархии:

```
int main() {

	D3 temp(100, 200, 300, 400, 500);
	D4 temp1(1, 2, 3, 4, 5);
	cout << "D3 temp(100,200,300,400,500);\n";
	cout << "\nСледуя иерархии класса D3: \n";
	temp.show_D3();

	cout << "D4 temp1(1,2,3,4,5);\n";
	cout << "\nСледуя иерархии класса D4\n";
	temp1.show_D4();
	cin.get();
}
```

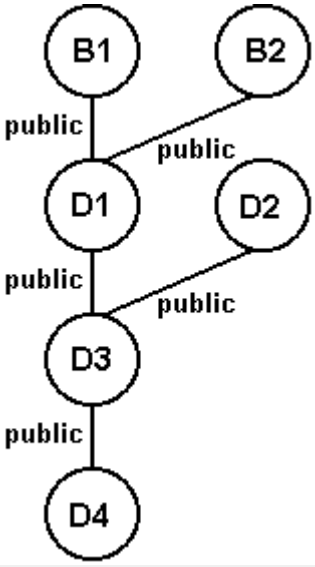
Варианты заданий

1  	2 
3	4

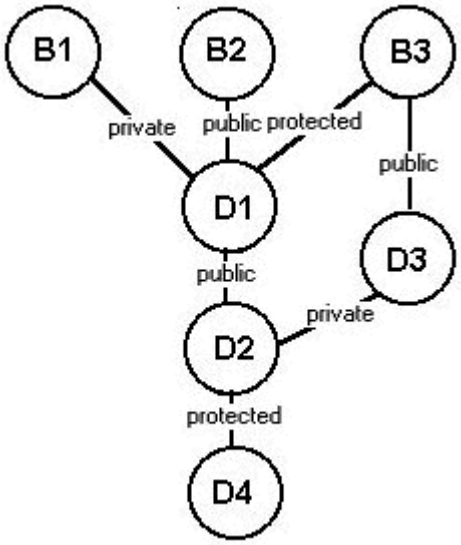

B
public
private
D1
D2
private
public
D3
D4


B1
B2
public
private
D1
private
D2
public
D3

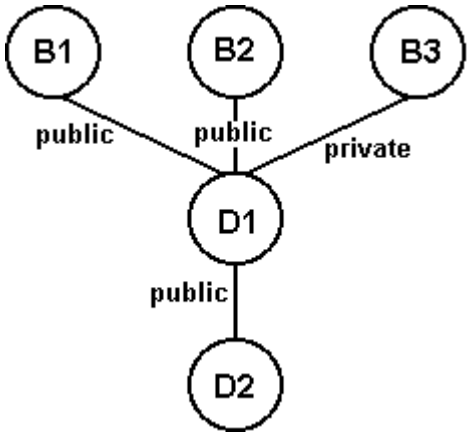
5


B1
B2
public
public
D1
D2
public
public
D3
public
D4

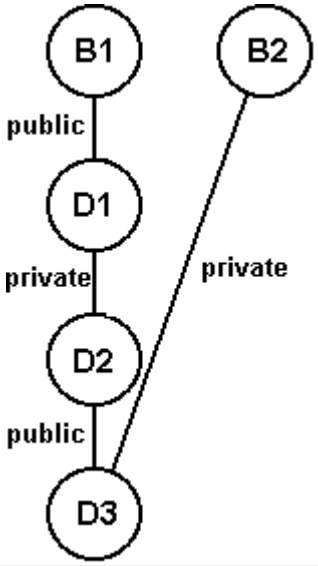
6


B1
B2
B3
private
public protected
public
D1
D3
public
private
D2
protected
D4

7

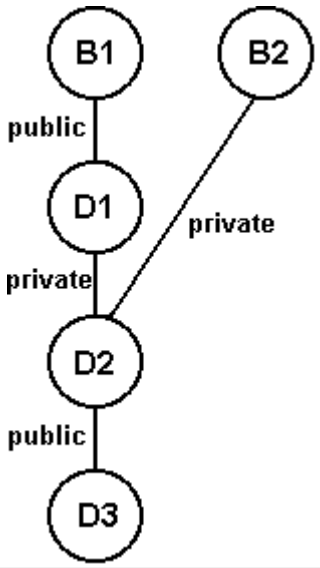

B1
B2
B3
public
public
private
D1
public
D2

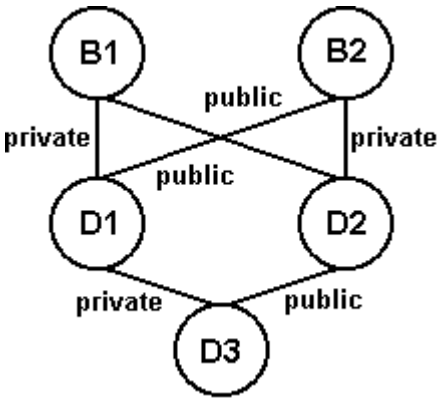
8


B1
B2
public
D1
private
private
D2
public
D3

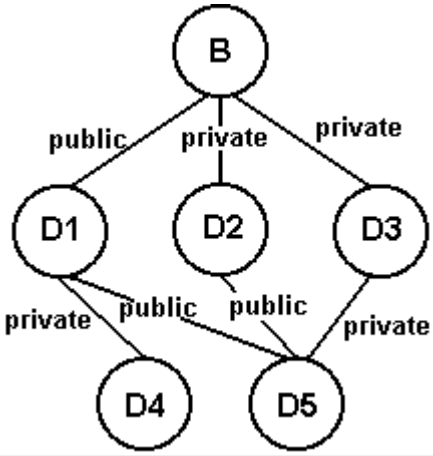
9

10

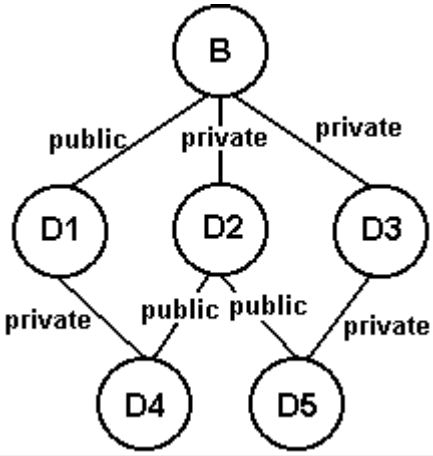




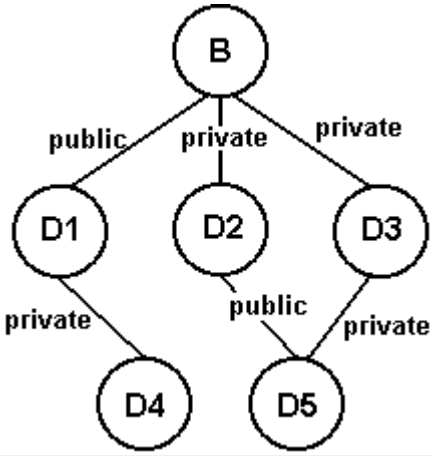
11



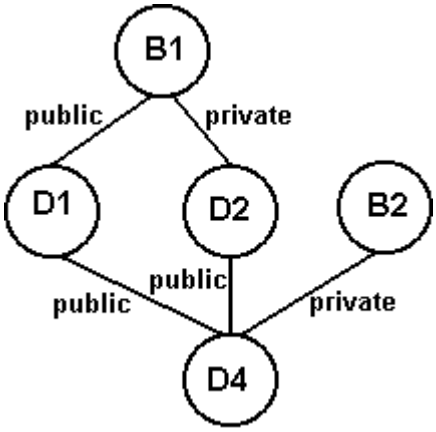
12



13

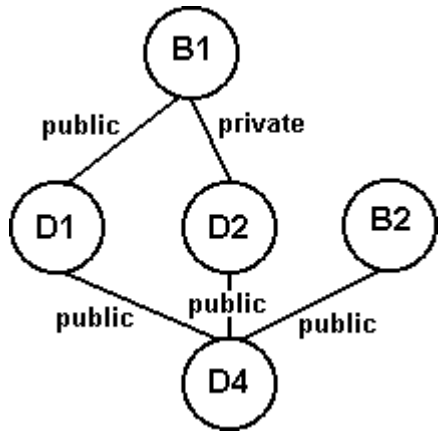


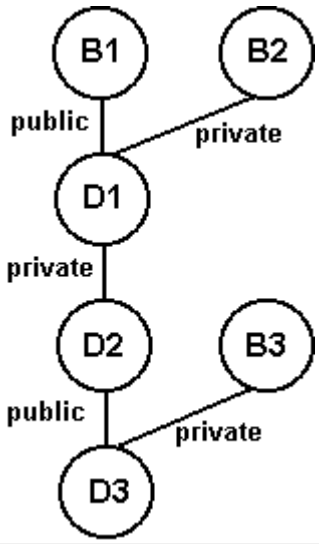
14



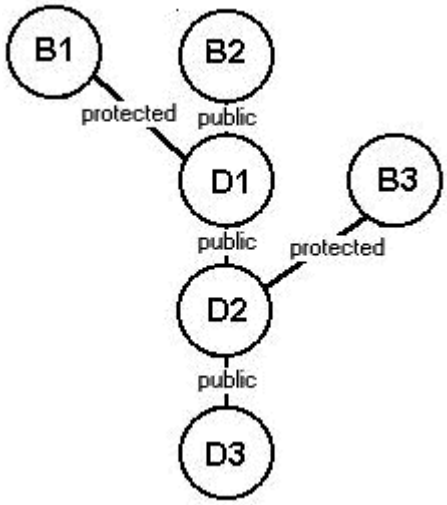
15

16

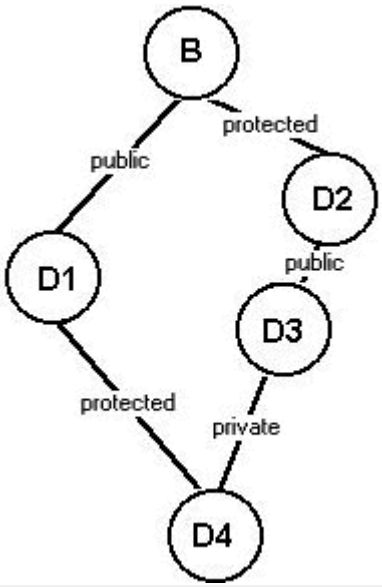

B1
public
private
D1
D2
B2
public
public
public
D4


B1
B2
public
private
D1
private
D2
B3
public
private
D3

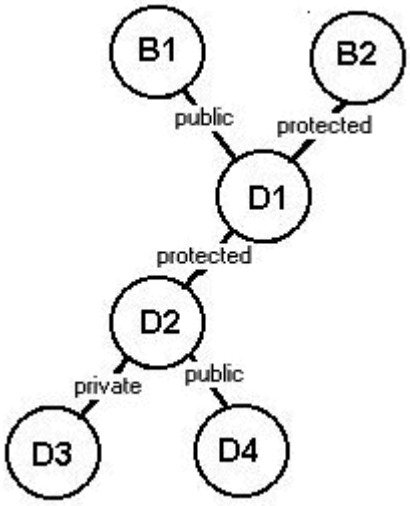
17


B1
B2
protected
public
D1
B3
public
protected
D2
public
D3

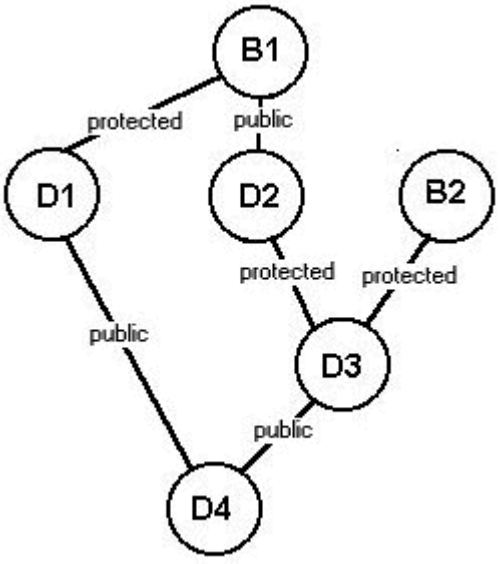
18

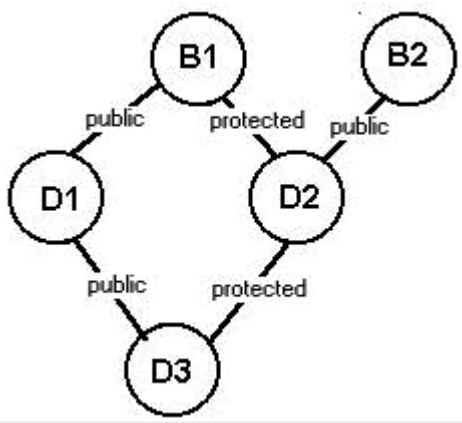
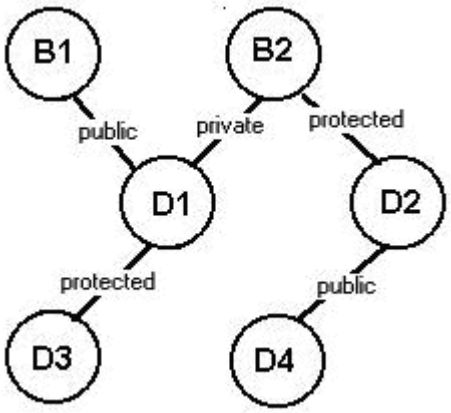
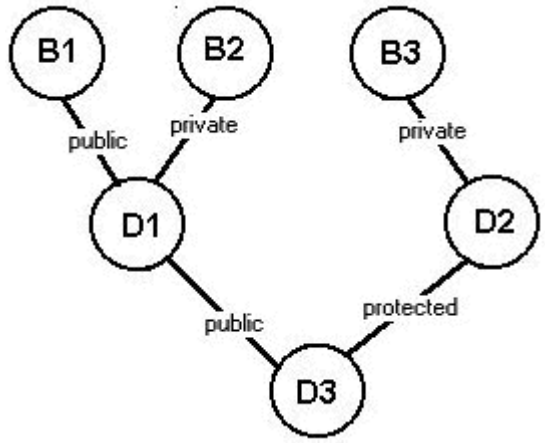
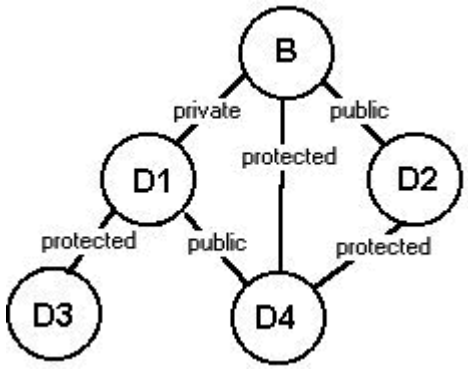

B
protected
public
D2
D1
public
D3
protected
private
D4

19


B1
B2
public
protected
D1
protected
D2
private
public
D3
D4

20


B1
protected
public
D1
D2
B2
protected
protected
public
D3
public
D4

21 	22 
23 	24 

Содержание отчета:

1. Титульный лист

2. Условие (с указанием номера варианта)

3. Полный текст (листинг) программы

4. Скриншоты с результатами (скриншоты должны демонстрировать все возможные ветви алгоритма решения).

5. Выводы