Лабораторная работа №6

Множественное наследование

### 

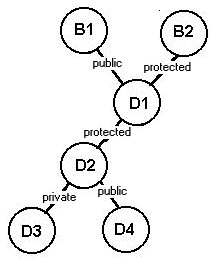
**Цель работы:** Целью лабораторной работы является получение практических навыков при использовании множественного наследования в языке С++.

**Задание:**

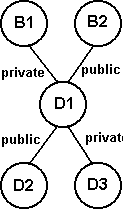
Необходимо построить иерархию классов согласно схеме наследования, приведенной в варианте задания.

Каждый класс должен содержать инициализирующий конструктор и функцию ***show*** для вывода значений.  
Функция ***main*** должна иллюстрировать иерархию наследования.

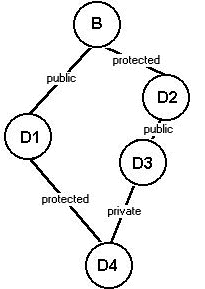
Вариант 1.



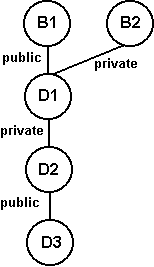
Вариант 2.



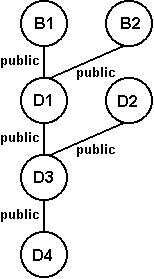
Вариант 3.



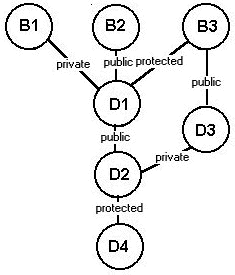
Вариант 4.



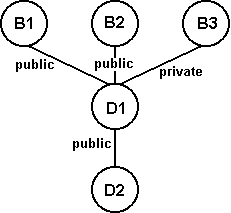
Вариант 5.



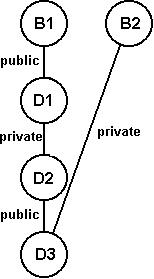
Вариант 6.



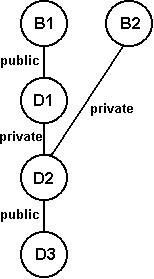
Вариант 7.



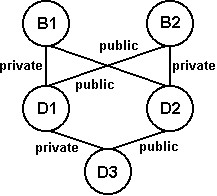
Вариант 8.



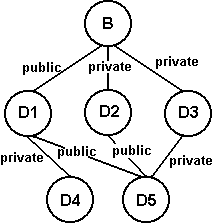
Вариант 9.



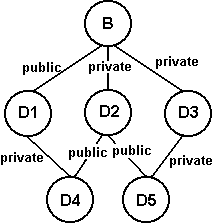
Вариант 10.



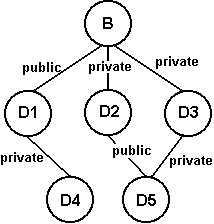
Вариант 11.



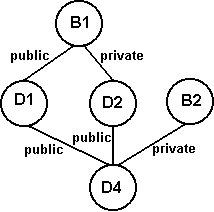
Вариант 12.



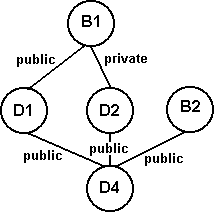
Вариант 13.



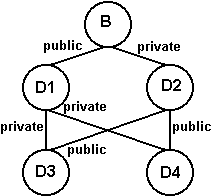
Вариант 14.



Вариант 15.



Пример



Сначала создадим необходимую иерархию классов. Следуя ей, мы получим: 

class B{

int a;

public:

};

class D1: public B {

int b;

public:

};

class D2: private B{

int c;

public:

};

class D3: private D1, public D2 {

int d;

public:

};

class D4: public D2, private D1 {

int e;

public:

};

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. 

B(int x) { a=x; }

D1(int x, int y) : B(y) { b=x;};

D2(int x, int y) : B(y) { c=x;};

D3(int x, int y, int z, int I, int j) : D1(y,z), D2(I,j) { d=x;}

D4(int x, int y, int z, int I, int j) : D1(y,z), D2(I,j) { e=x;}

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.

void show\_B() { cout <<”B= “<< a << “\n”; }

void show\_D1() { cout << “D1= “<< b << “\n”; show\_B();}

void show\_D2() { cout <<”D2= “<< c << “\n”; show\_B();}

void show\_D3() { cout << “D3= “<<d << “\n”; show\_D1(); show\_D2();}

void show\_D4() { cout <<”D4= “<< e << “\n”; show\_D1(); show\_D2();}

В итоге получаем программу:

class B{

int a;

public:

B() { };

B(int x) { a=x; }

void show\_B() { cout <<"B= "<< a << "\n"; }

};

class D1: public B {

int b;

public:

D1(int x, int y) : B(y) { b=x;};

void show\_D1() { cout <<"D1= "<< b << "\n"; show\_B();}

};

class D2: private B{

int c;

public:

D2(int x, int y) : B(y) { c=x;};

void show\_D2() { cout <<"D2= "<< c << "\n"; show\_B();}

};

class D3: private D1, public D2 {

int d;

public:

D3(int x, int y, int z, int i, int j) : D1(y,z), D2(i,j) { d=x;}

void show\_D3() { cout << "D3= "<<d << "\n"; show\_D1(); show\_D2();}

};

class D4: public D2, private D1 {

int e;

public:

D4(int x, int y, int z, int i, int j) : D1(y,z), D2(i,j) { e=x;}

void show\_D4() { cout <<"D4= "<< e << "\n"; show\_D1(); show\_D2();}

};

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 << "D4 temp1(1,2,3,4,5);\n";

cout<< "\nСледуя иерархии класса D3: \n";

temp.show\_D3();

cout<< "\nСледуя иерархии класса D4\n";

temp1.show\_D4();

return 0;

}