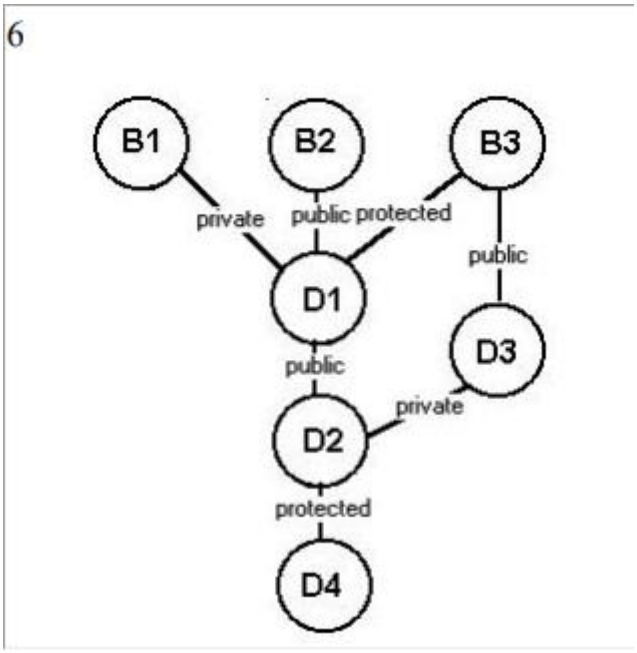
1. **Цель работы**

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

2. **Задание**

   Согласно варианту №6:

   Создать иерархию классов:



3. **Листинг программы**

```
#include <iostream>

using namespace std;

// БАЗОВЫЙ КЛАСС 1

class B1
{
    int x;
protected:
    void ShowB1();
    B1(int);
    ~B1();
};

void B1::ShowB1()
{
    cout << x << endl;
}

B1::B1(int x)
{
    this->x = x;
    cout << "Работает конструктор B1: параметр = " << this->x << endl;
}

B1::~B1()
{
```

```
    cout << "Работает деструктор B1" << endl;
}

// БАЗОВЫЙ КЛАСС 2

class B2
{
    int x;
protected:
    void ShowB2();
    B2(int);
    ~B2();
};

void B2::ShowB2()
{
    cout << x << endl;
}

B2::B2(int x)
{
    this->x = x;
    cout << "Работает конструктор B2: параметр = " << this->x << endl;
}

B2::~B2()
{
    cout << "Работает деструктор B2" << endl;
}

// БАЗОВЫЙ КЛАСС 3

class B3
{
    int x;
protected:
    void ShowB3();
    void SetB3(int);
    ~B3();
};

void B3::SetB3(int x)
{
    this->x = x;
}

void B3::ShowB3()
{
    cout << x << endl;
}

B3::~B3()
{
    cout << "Работает деструктор B3" << endl;
}

// ПРОИЗВОДНЫЙ КЛАСС 1

class D1 : private B1, public B2, virtual protected B3
{
    int x;
protected:
    void ShowD1();
    D1(int, int, int, int);
    ~D1();
```

```
};

void D1::ShowD1()
{
    cout << x << endl;
    ShowB1();
    ShowB2();
    ShowB3();
}

D1::D1(int x, int y, int z, int i): B1(x), B2(y)
{
    SetB3(z);
    cout << "Работает конструктор B3: параметр = " << z << endl;
    this->x = i;
    cout << "Работает конструктор D1: параметр = " << this->x << endl;
}

D1::~D1()
{
    cout << "Работает деструктор D1" << endl;
}

// ПРОИЗВОДНЫЙ КЛАСС 3

class D3 : virtual public B3
{
    int x;
public:
    void ShowD3();
    D3(int);
    ~D3();
};

void D3::ShowD3()
{
    cout << x << endl;
    ShowB3();
}

D3::D3(int y)
{
    this->x = y;
    cout << "Работает конструктор D3: параметр = " << this->x << endl;
}

D3::~D3()
{
    cout << "Работает деструктор D3" << endl;
}

// ПРОИЗВОДНЫЙ КЛАСС 2

class D2 : public D1, private D3
{
    int x;
protected:
    void ShowD2();
    D2(int, int, int, int, int, int);
    ~D2();
};

void D2::ShowD2()
{
    cout << x << endl;
```

```
    ShowD1();
    ShowD3();
}

D2::D2(int x, int y, int z, int i, int j, int k) : D1(x, y, z, i), D3(j)
{
    this->x = k;
    cout << "Работает конструктор D2: параметр = " << this->x << endl;
}

D2::~D2()
{
    cout << "Работает деструктор D2" << endl;
}

// ПРОИЗВОДНЫЙ КЛАСС 4

class D4 : protected D2
{
    int x;
public:
    void ShowD4();
    D4(int, int, int, int, int, int, int);
    ~D4();
};

void D4::ShowD4()
{
    cout << x << endl;
    ShowD2();
}

D4::D4(int x, int y, int z, int i, int j, int k, int g) : D2(x, y, z, i, j, k)
{
    this->x = g;
    cout << "Работает конструктор D4: параметр = " << this->x << endl;
}

D4::~D4()
{
    cout << "Работает деструктор D4" << endl;
}

int main()
{
    setlocale(LC_ALL,"RUS");

    D4 temp(1, 2, 3, 4, 5, 6, 7);
    cout << "D4 temp(1, 2, 3, 4, 5, 6, 7)" << endl;
    cout << "Следуя иерархии класса D4: " << endl;;
    temp.ShowD4();


}
```

4. Пример работы программы

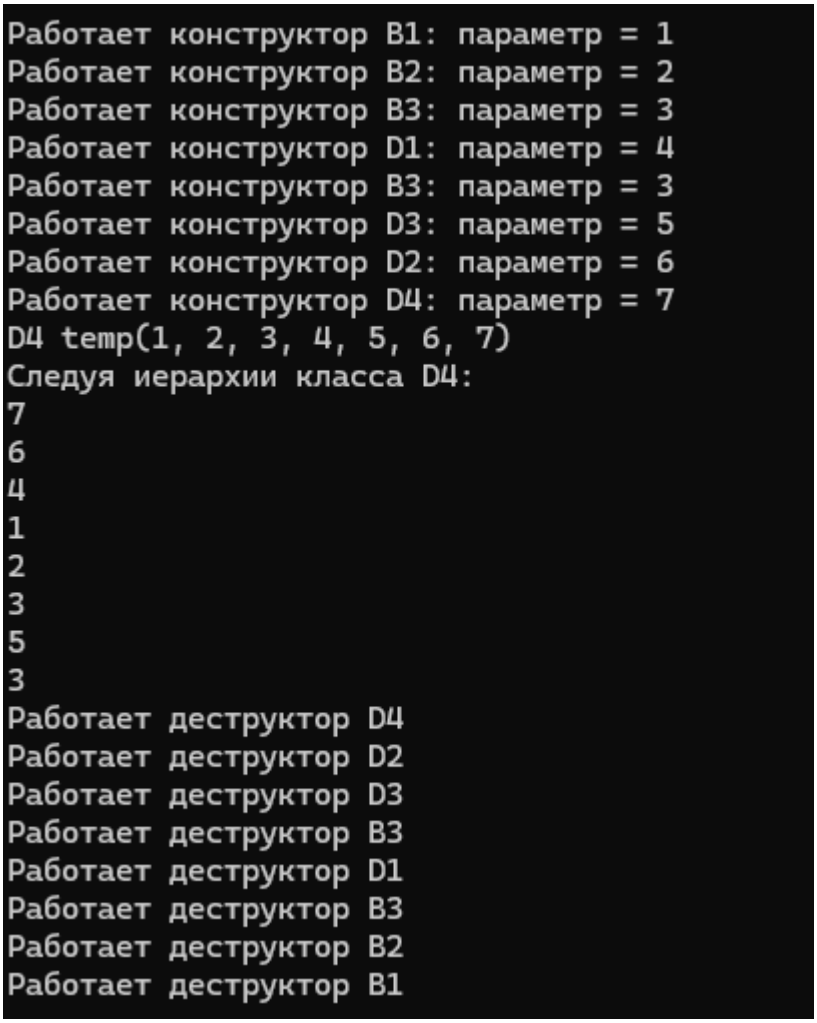


Рисунок 1 – Пример работы

5. Выводы

В процессе лабораторной работы были изучены принципы множественного наследования, а так же получены практические навыки реализации такого наследования.