**Задача 2.** Реализирайте базов клас *автомобил,* характеризиращ се с модел (например ”Aston-Martin DB9”) и разход на гориво. Реализирайте производен клас *тунингован\_автомобил,* допълващ базовия с име на компанията, извършила тунинга. Реализирайте и про­изво­ден клас *амортизиран\_автомобил,* имащ допълнителна информация за това с колко се е уве­личил разходът на гориво.

За класовете реализирайте подходящи конструктори, селектори и метод за извеждане на информацията за модела и специфичния му разход на гориво. Ако автомобилът е тунингован, към името на модела се добавя и името на тунинговащата фирма.

За йерархията

class base class der1 : public base

{ private: int a1; { private: int a4;

protected: int a2; protected: int a5;

public: int a3(); public: int a6();

}; };

class der2 : der1 class der3 : protected der2

{ private: int a7; { private: int a10;

protected: int a8; protected: int a11;

public: int a9(); public: int a12();

} ; } ;

class der4 : protected der1 class der5 : public der2

{ private: int a13; { private: int a16;

protected: int a14; protected: int a17;

public: int a15(); public: int a18();

} ; } ;

base b; der4 d4;

der1 d1; der5 d5;

der2 d2;

der3 d3;

определете възможностите за достъп на обектите: b, d1, d2, d3, d4 и d5 до компонентите на класовете.

|  |  |
| --- | --- |
| **13. зад.** (10 т.) Намерете резултата от изпълнението на програмата.  #include <iostream>  using namespace std;  class base1  { public:  base1(int a, double b = 0)  { n = a; x = b;  cout << "base1: " << n << ", " << x << endl;  }  base1(const base1& p)  { n = p.n+1;  x = p.x+2;  cout << "base1(const base1&)\n";  cout << "base1.n: " << n << endl  << "base1.x: " << x << endl;  }  base1& operator=(const base1& p)  { cout << "base1::operator=()\n";  if (this != &p)  { n = p.n + 1;  x = p.x + 1.5;  cout << "base1.n: " << n << endl  << "base1.x: " << x << endl;  }  return \*this;  }  private:  int n;  double x;  };  class base2  { public:  base2(int a = 1, double b = 0)  { n = a;  x = b;  cout << "base2: " << n << ", " << x << endl;  }  private:  int n;  double x;  }; | class base3  { public:  base3(int a = 2, double b = 0)  { n = a;  x = b;  cout << "base3: " << n << ", " << x << endl;  }  base3(const base3& p)  { n = p.n + 1;  x = p.x + 1.5;  cout << "base3(const base3&)\n";  cout << "base3.n: " << n << endl  << "base3.x: " << x << endl;  }  private:  int n;  double x;  };  class der : public base1, protected base3, base2  { public:  der(int x = 0, int y = 0, int z = 0) : base1(x),  base2(y, z), base3(z, x), a(x+y)  { n = z;  m = x-y;  cout << "der: " << n << ", " << m << endl;  }  der(const der& p) : base3(p), base1(p), a(p)  { cout << "der(const der&)\n";  n = p.n + 1;  m = p.m + 1;  cout << "der.n: " << n << endl  << "der.m: " << m << endl;  }  private:  int n, m;  base1 a;  base2 b;  base3 c;  };  int main()  { der p, q(1, 2, 3);  der r = p;  p = q;  return 0;  } |