ТЕСТОВЕ

ПО

ОБЕКТНО–ОРИЕНТИРАНО ПРОГРАМИРАНЕ

Магдалина Тодорова

март-юни 2007

Тест 1

Дефиниране на класове

Задача. Отбележете и обяснете грешките в програмата. Поправете ги, така че да се получи работеща програма. Намерете резултата от изпълнението й.

```
#include <iostream.h>
class A
{public:
     A(int, int = 1);
     void print();
     int f_Ax() const;
     int f_Ay() const;
  private:
      int x, y;
  };
  A::A(int a, int b)
  { x = a;
    y = b;
  }
  void A::print()
  {  cout << x << " " << y << endl;
  }
  int A::f_Ax() const
  {  return x;
  }
  int A::f_Ay() const
  {  return y;
  }
  class B
  {public:
       B(double, A);
       void print() const;
       double f_Bx();
       A f_Ba() const;
   private:
      double x;
      A a;
  };
  B::B(double d, A e)
  { x = d;
    a = e;
  }
  void B::print() const
  {  cout << x << endl;
    a.print();
  }
  double B::f_Bx() const
  { cout << a.f_Ax() << " " << a.f_Ay() << endl;
    return x;
```

```
}
A B::f_Ba()
{  return a;
}
void main()
{ A a(1), x;
  a.print();
  cout << x.x << " " << x.y << endl;
  B b(2.5, a);
  b.print();
}
```

Канонично представяне на клас

Задача. Намерете резултата от изпълнението на програмата:

```
#include <iostream.h>
#include <string.h>
class A
{public:
     A(char* ="AAA", double = 0.0);
     ~A();
      A(const A&);
      A& operator=(const A&);
      void print() const;
   private:
       char* st;
       double x;
};
A::A(char* s, double y)
{  cout  << "A(" << s << "," << y << ")\n";
   st = new char[strlen(s)+1];
   strcpy(st, s);
   x = y;
}
A::~A()
{  cout << "~A()\n";
   delete st;
}
A::A(const A& s)
{  cout << "A(const s)\n";
   st = new char[strlen(s.st)+1];
   strcpy(st, s.st);
   x = s.x;
}
A& A::operator=(const A& s)
{  cout << "A::operator=()\n";
   if(this != &s)
  { delete st;
     st = new char[strlen(s.st)+1];
     strcpy(st, s.st);
     x = s.x;
   }
   return *this;
}
```

```
void A::print() const
{  cout << st << " " << x << endl;
}

class B
{public:
      B(double, const A&);
      B(const B&);
      B& operator=(const B&);
      void print() const;
   private:
      double x;
      A a;
};
B::B(double d, const A& e) : a(e)
{  cout << "B::B(d, e)\n";
    x = d;
}
B::B(const B& p) : a(p.a)
{  cout << "B::B(const p)\n";
    x = p.x;
 }
B& B::operator=(const B& p)
{  cout << "B::operator=()\n";
    if(this != &p)
   {  x = p.x;
       a = p.a;
   }
   return *this;
}
void B::print() const
{  cout << x << endl;
    a.print();
}

void main()
{ A a1("***"), a2;
   B b(5, a1), c(10, a2), d(c);
   b.print();
   c.print();
   d.print();
   d = b;
   d.print();
}
```

Тест 3

Единично наследяване

Задача 1. За йерархията base->der1->der11->der111:

```
class base
{ private : int a1;
  protected: int a2;
  public: int a3();
 } b;
class der11 :  protected der1
{ private: int a7;
  protected: int a8;
  public: int a9();
 } d11;
```

```
class der1 :  public base
{ private: int a4;
  protected: int a5;
  public: int a6();
 } d1;
class der111 : der11
{ private: int a10;
  protected: int a11;
  public: int a12();
 } d111;
```

определете възможностите за достъп на обектите: b, d1, d11 и d111 до компонентите на класовете.

Задача 2. Посочете грешките в дефиницията на йерархията A -> DerA:

```
class A
{ int data1, data2;
  int F() const;
  double G(int, int);
 };
int A::F() const
{  cout << data1 << " " << data2 << endl;
}
double A::G(int x, int y)
{  data1 = x+y;
   data2 = x-y;
}
class DerA : protected A
{protected:
     int H() const;
  public:
     double R()
     {  cout << "data1, data 2=";
        cin >> data1 >> data2;
        cout << data1 << "  " << data2 << "\n";
      }
  };
 int DerA::H() const
{  cout << "DerA\n" << data1 << "\n" << data2 << "\n";
};
```

```
int main()
{ A a, b;
  DerA Da, Db;
  a.F();
   b.G(10, 20);
   Da.H();
   Db.R();
 }
```

Задача 3. Изберете подходящи спецификатори за достъп и атрибут за област в йерархията

base->der:

```
class base                          class der: …… base
{……: int b1;                        {……: int d1;
  ……: int b2;                        ……: int d2;
  ……: int b3();                      ……: int d3();
};                                   };
```

така че фрагментите:

```
int der::d3()
{ cout << b2 << "\n";
  cout << d1 << " " << d2 << "\n";
  return b2+d2;
};
…
der d1, d2;
d1.d2 = 15;
d1.b2 = 25;
d1.d3();
```

да не предизвикват синтактични грешки, а всяка линия на фрагмента:

```
d2.b1;
d2.d1 = 22;
d2.b3();
```

да предизвиква синтактични грешки.

Приятелски функции

Задача. Разгледайте “програмата”:

```
#include <iostream.h>
class first
{private:
     int f1;
  protected:
     int f2;
  public:
     friend void first_friend(first&, int, int);
     void first_read(int x, int y)
     {  f1 = x;
        f2 = y;
     }
     void first_print() const
     {  cout << "f1: " << f1 << endl
             << "f2: " << f2 << endl;
     }
   };
 class second : first
 {private: int s1;
   protected: int s2;
   public:
       friend void second_friend(second &d, first b, int x, int y);
       void readsecond(int x, int y, int z, int t)
       {  first_read(x, y);
          s1 = z;
          s2 = t;
        }
        void second_print() const
        {  cout << "first_print():" << endl;
           first_print();
           cout << "s1: " << s1 << endl
                << "s2: " << s2 << endl;
         }
    };
   void first_friend(first& f, int x, int y)
   {  f.f1 = x + y;
      f.f2 = x - y;
   };
   void second_friend(second& s, first f, int x, int y)
```

```
{  cout << "friend function second_friend(): " << endl;
    first_friend(f, x, y);
    s.s1 = f.f1 + x;
    s.s2 = f.f2 - x;
    s.f1 = x + y;
    s.f2 = f.f2 + f.f1 + 3*x;
    cout << "s.first_print(): " << endl;
    s.first_print();
    cout << "d.second_print(): " << endl;
    s.second_print();
 }
void main()
{   first f;
    f.first_read(7, 9);
    f.first_print();
    first_friend(f, 10, 20);
    f.first_print();
    second s;
    s.readsecond(2, 4, 3, 5);
    s.second_print();
    second_friend(s, f, 1, 2);
    cout << "f.first_print()\n";
    f.first_print();
    cout << "s.second_print()\n";
    s.first_print();
 }
```

1. Намерете и коментирайте грешките в нея.

2. Поправете грешките като промените единствено спецификаторите за достъп и атрибута за област на класа first.

3. Какъв е резултатът от изпълнението на поправената програма?

4. Посочете и коментирайте грешките в последната (поправената) програма, ако в нея промените first_friend и тя получава вида:

```
void first_friend(first& f, int x, int y)
{  second s;
   s.f1 = x + y;
   s.f2 = x - y;
   s.s1 = (s.f1 + s.f2)/2;
   s.s2 = (s.f1 - s.f2)/2;
}
```

Предефиниране на компоненти

Задача. Какъв е резултатът от изпълнението на програмата?

```
#include <iostream.h>
class A
{public:
     void init(int x, int y)
     {  bx = x;
        by = y;
     }
     void print() const
     {  cout << " A::bx= " << bx << endl
             << " A::by= " << by << endl;
      }
  protected:
     int bx;
  protected:
     int by;
 };
class B : public A
{public:
   void init(int x, int y, int z, int t)
   {  A::init(x, y);
      bx = z;
      by = t;
   }
   void print() const
   {  A::print();
      cout << " B::bx = " << bx << endl
           << " B::by = " << by << endl;
   }
 protected:
    int bx;
 private:
    int by;
};
class C :  public B
{public:
   void init(int x, int y, int z, int t, int p, int q)
   {  bx = p;
     by = q;
     A::init(x, y);
     B::init(x, y, z, t);
```

```
  }
  void print() const
  {  A::print();
     B::print();
     cout << " C::bx = " << bx << endl
          << " C::by = " << by << endl;
  }
protected:
    int bx;
private:
    int by;
};
void main()
{  A a;
   B b;
   C c;
   a.init(11, 12); b.init(13, 15, 17, 19);
   c.init(2, 4, 6, 8, 10, 12);
   a.print(); b.print(); c.print();
   b.A::init(2, 2);
   b.A::print();
   b.print();
   c.B::init(4, 3, 2, 1);
   c.A::init(7, 6);
   c.print();
}
```

Единично наследяване. Конструктори и деструктори

Задача. Подчертайте, коментирайте и поправете грешките в програмата:

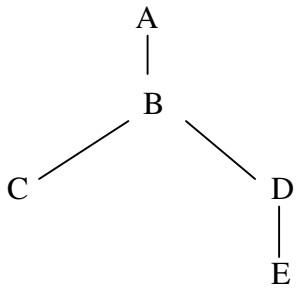
```
#include <iostream.h>
#include <string.h>
class first
{ public:
      first(char* x = "first")
      {  f = new char[strlen(x) + 1];
         strcpy(f, x);
      }
      ~first()
      { delete f;
      }
    protected:
        char* f;
};
class second1 :  public first
{public:
     second1(char* x = "second1") : first("fififi")
      {  s = new char[strlen(x) + 1];
         strcpy(s, x);
      }
      ~second1()
      {  ~first();
         delete s;
      }
      void Print()
      {  cout << "second1:: " << s << "  first:: " << f << endl;
      }
      private:
        char* s;
};
class second2 : public first
{ public:
       second2(char* x = "second2") : first();
       ~second2()
       {  delete s;
       }
       void Print()
       {  cout << "second2:: " << s << "  first:: " << f << endl;
       }
 private:
```

```
    char* s;
};
second2::second2(char* x = "second2") : first()
 {  s = new char[strlen(x) + 1];
    strcpy(s, x);
  }
class second3 : public first
{public:
     second3(char* x = "second3") : first(), first("ERROR"), second1()
      {  s = new char[strlen(x) + 1];
         strcpy(s, x);
      }
      ~second3()
      {  delete s;
         delete f;
      }
      void Print()
      {  cout << "second3:: " << s << "  first:: " << f << endl;
      }
 private:
    char* s;
};
void main()
{  second1 s11("s11");
   second2 s21("s21"), s22;
   second3 s31("s31");
   cout << "s11: "; s11.Print();
   cout << "s21: "; s21.Print();
   cout << "s22: "; s22.Print();
   cout << "s31: "; s31.Print();
}
```

Какъв е резултатът от изпълнението на поправената програма?

Единично наследяване. Конструктори

Задача. Подчертайте и коментирайте грешките в реализацията на следната йерархия:




```
#include <iostream.h>
class A
{private: int a;
    public:
    void read(int x = 0)
    {  a = x;
    }
    void print()
    {  cout << "a: " << a << endl;
    }
};
class B : public A
{private: int y;
 public:
     B(int x) : B(x = 0);
      void print()
      {  cout << "y: " << y << endl;
         cout << "A::print():" << endl;
         A::print();
      }
};
 B::B(int x) : A(x = 5)
 {  cout << "constructor B\n";
    y = x;
 }
class C :  public B
{private: int y;
 public:
 void print()
 {  cout << "C  member y: " << y << endl;
```

```
      cout << "B::print():" << endl;
      B::print();
  }
};
class D : public B
{private: int y;
 public:
     D()
     {y = 0;
     }
     D(int x) : B(x = 0);
     void print()
     {  cout << "D  member y: " << y << endl;
        cout << "B::print():" << endl;
        B::print();
     }
};
D::D(int x) : A(x)
{  cout << "constructor D\n";
   y = x;
}
class E : public D
{private: int y;
  public:
       void print()
       {  cout << "E  member y: " << y << endl;
          cout << "C::print():" << endl;
          C::print();
       }
};
void main()
{  B b(1); b.print();
   C c; c.print();
   D d(1); d.print();
   E e; e.print();
}
```

Поправете грешките.

Конструктор за присвояване, операторна функция =

Задача. Какъв е резултатът от изпълнението на програмата?

```
#include <iostream.h>
class A
{public:
      A(int x = 5)
      {  a = x;
       }
      A& operator=(const A &x)
      {  if(this!=&x) a = x.a + 1;
         return *this;
      }
      void Print() const
      {  cout << "A: " << a << endl;
      }
 private:
   int a;
};
class B : public A
{public:
      B(int x = 1) {y = x;}
      B& operator=(const B& x)
      {  if(this != &x)
         {  y = x.y + 2;
            A::operator=(x);
          }
          return *this;
      }
      void Print() const
      {  cout << "B: " << y << endl;
        A::Print();
      }
 private:
   int y;
};
class C : public A
{public:
     C(int x = 2) : A(x+3)
      {  y = x;
      }
     C(const C& p)
     {  y = p.y + 2;
```

```
      }
      C& operator=(const C& x)
      {  if(this !=&x)
         {  y = x.y + 3;
            A::operator=(x);
          }
         return *this;
      }
      void Print() const
      {  cout << "C: " << y << endl;
         A::Print();
      }
 private:
   int y;
};
class D : public C
{public:
      D(int x = 3)
      {  y = x;
      }
      D(const D& p) : C(p)
      {  y = p.y+3;
      }
      void Print()
      {  cout << "D: " << y << endl;
         C::Print();
      }
 private:
   int y;
};
void main()
{  B x(10), y = x;
   C z(20), u;
   D v(30), t = v;
   cout << "x: "; x.Print();
   cout << "y: "; y.Print();
   y = x; cout << "y: "; y.Print();
   cout << "z: "; z.Print();
   cout << "u: "; u.Print();
   u = z; cout << "u: "; u.Print();
   cout << "v: "; v.Print();
   cout << "t: "; t.Print();
   t = v; cout << "t: "; t.Print();
}
```

Преобразуване на типове

Задача. Дадена е йерархията от класове:

```
#include <iostream.h>
class base
{public:
     base(int x = 0)
     {  b = x;
     }
     int get_b() const
     {  return b;
     }
     void f() const
     {  cout << "b: " << b << endl;
     }
 private:
    int b;
 };
class der1 : public base
{public:
     der1(int x = 0) : base(x)
     {  d = 1;
     }
     int get_d() const
     {  return d;
     }
     void f_der1() const
     {  cout << "class der1: d: " << d
              << " b: " << get_b() << endl;
     }
 private:
    int d;
 };
class der2 : public base
{public:
     der2(int x = 0) : base(x)
     {  d = 2;
     }
     int get_d() const
     {  return d;
     }
     void f_der2() const
```

```
    {  cout << "class der2: d: " << d
               << " b: " << get_b() << endl;
    }
private:
   int d;
};
```

Кои от следните фрагменти са коректни и кои не са? Обяснете грешките. Всеки фрагмент да се разглежда като самостоятелен.

А)

```
der1 d1;  der2 d2;
base x = d2;
d1 = x;
der1 &d3 = d1;
base &y = d3;
d3 = y;
der2 *d4 = &d2;
(*d4).f_der2();
base *z = d4;
(*z).f();
d4 = z;
```

Б)

```
base x;
der2 y = (der2)x;
base *pb = new base;
der2* pd = pb;
pb -> f();
der1 *pc = pb;
(*pc).f_der1();
cout << pc->get_b() << endl;
```

В)

```
void (der1::*pd)() = der1::f_der1;
void (base::*pb)() = pd;
void (der2::*pdd)() const;
void (base::*pbb)() const;
pbb = base::get_b;
pdd = pbb;
```

Множествено наследяване

Задача. Намерете резултата от изпълнението на програмата.

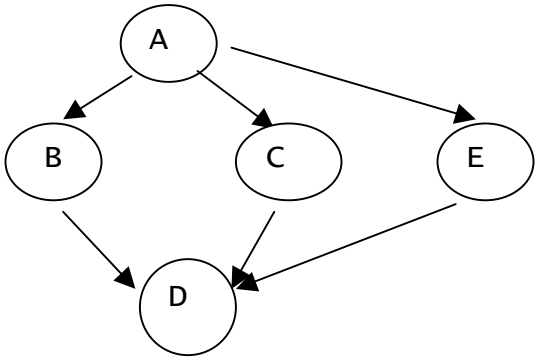
```
#include <iostream.h>
class base
{ public:
      base(int a = 50)
      { n = a;
        x = -31.6;
        cout << "base: " << n << "," << x << endl;
      }
      base(const base& p)
      { n = p.n;
        x = p.x;
        cout << "base.n: " << n << endl
             << "base.x: " << x << endl;
      }
      base& operator=(const base& p)
      { if(this!=&p)
        { n = p.n + 95;
          x = p.x + 43.2;
          cout << "base.n: " << n << endl
               << "base.x: " << x << endl;
         }
         return *this;
       }
    private:
       int n;
      double x;
};
class second
{ public:
      second(double b = 20)
      { n = 7;
        y = b;
        cout << "second: " << n << "," << y << endl;
       }
    private:
       int n;
      double y;
};
```

```
class tirth
{ public:
      tirth(double b = 10)
      { n = 3;
        x = b;
        cout << "tirth: " << n << "," << x << endl;
       }
       tirth(const tirth& p)
      { n = p.n + 67;
        x = p.x + 11.7;
        cout << "tirth.n: " << n << endl
             << "tirth.x: " << x << endl;
       }
    private:
       int n;
      double x;
};
class fourth : public second, base, protected tirth
{ public:
      fourth(int x=23, int y=21, int z=27): base(x), second(y), tirth(z)
     { n = z;
       m = x-y;
       cout << "fourth: " << n << "," << m << endl;
      }
      fourth& operator=(const fourth& p)
     { if(this!=&p)
       { base::operator =(p);
         n = p.n;
         m = p.m;
        }
        return *this;
     }
 private:
     int n, m;
};
void main()
{ fourth x, y(13,17,12), z;
  fourth t = x;
  z = y;
}
```

Тест 11

Виртуални класове

Задача. Да се изгради йерархията:



така, че A да е виртуален клас за класовете B и C и да не е виртуален за класа E. Класовете да съдържат голямата четворка като всеки клас да определя за член-данна символен низ, реализиран като динамичен масив.
Какво е разпределението на паметта за обект от клас D при направената реализация?

Виртуални функции

Задача. Разгледайте програмата:

```
#include <iostream.h>
class Base
{public:
    virtual void f()
    {  cout << "f() \n";
    }
    void fgh()
    {  cout << "fgh()\n";
       f();
       g();
       h();
     }
 private:
    virtual void g()
    {  cout << "g()\n";
    }
 protected:
   virtual void h()
  {  cout << "h()\n";
  }
};
class Der : public Base
{private:
     virtual void f()
     {  cout << "Der-class\n";
     }
 protected:
   virtual void g()
   {  cout << "Der-g()\n";
   }
 public:
    virtual void h()
   {  cout << "Der-h()\n";
   }
};
void main()
{  Base b; Der d;
   Base *p = &b;
```

```
    Base *q = &d;
    b.f();
    p->f();
    q->f();
    p->g();
    q->g();
    p->h();
    q->h();
    p->Base::f();
    Der *r  = new Der;
    r->f();
    r->g();
    p->fgh();
    q->fgh();
    r->fgh();
    delete r;
}
```

а) Намерете и обяснете грешките в процедурата main на горната програма.

б) Какъв е резултатът от изпълнението на програмата след отстраняване на неправилните обръщения към виртуланите функции?

Виртуални функции. Полиморфизъм. Статично и динамично свързване

Задача. Разгледайте програмата:

```
#include <iostream.h>
class Base
{public:
       virtual void virt1()
       {  cout << "Base::virt1() \n";
        }
        Base()
        {  cout << "Base()\n";
           virt1();
           virt2();
           virt3();
         }
   private:
         virtual void virt2()
         {  cout << "Base::virt2()\n";
         }
  protected:
        virtual void virt3()
       {  cout << "Base::virt3()\n";
       }
};
class Der1 : public Base
{  void virt1()
   {  cout << "Der1::virt1()\n";
    }
  protected:
       void virt2()
       {  cout << "Der1::virt2()\n";
        }
  public:
       void virt3()
        {  cout << "Der1::virt3()\n";
         }
 };
class Der2 : public Der1
{ protected:
      void virt1()
      {  cout << "Der2::virt1()\n";
```

```
        }
  public:
        void virt2()
        {  cout << "Der2::virt2()\n";
        }
  private:
         void virt3()
        {  cout << "Der2-virt3()\n";
         }
};
void main()
{  Base b;
   Der1 d1; Der2 d2;
   Base *p = &d1;
   Der1 *q = &d2;
   b.virt1();
   b.Base();
   p->virt1();
   p->virt2();
   p->virt3();
   q->Base();
   q->virt1();
   q->virt2();
   q->virt3();
  p = &d2;
   p->virt1();
   p->virt2();
   p->virt3();
   Der1 *r = new Der2;
   r->virt1();
   r->virt2();
   r->virt3();
   delete r;
}
```

а) Намерете и обяснете грешките в процедурата main на горната програма.
б) Кои връзки в нея се разрешават статично и кои динамично?
в) Какъв е резултатът от изпълнението на програмата след отстраняване на неправилните обръщения към виртуланите функции?

Виртуални класове. Виртуални деструктори

Задача. По време на изпълнение на програмата:

```
#include <iostream.h>
#include <string.h>
class A
{public:
      A(char* = "");
      ~A();
      A(const A&);
      A& operator=(const A &);
      virtual void print() const;
 private:
     char* x;
 };
A::A(char* s)
{  x = new char[strlen(s)+1];
   strcpy(x, s);
}
A::~A()
{  cout << "~A()\n";
   delete x;
}
A::A(const A& p)
{  x = new char[strlen(p.x)+1];
   strcpy(x, p.x);
}
A& A::operator=(const A& p)
{  if (this != &p)
   {  delete x;
      x = new char[strlen(p.x)+1];
      strcpy(x, p.x);
    }
    return *this;
}
void A::print() const
{  cout << "A:: x " << x << endl;
}
class B : virtual public A
{public:
      B(char* = "", char* = "");
      ~B();
      B(const B&);
      B& operator=(const B&);
      void print() const;
 private:
     char* x;
```

```
};
B::B(char* a, char* b) : A(a)
{  x = new char[strlen(b)+1];
   strcpy(x, b);
}
B::~B()
{  cout << "~B()\n";
   delete x;
}
B::B(const B& p) : A(p)
{  x = new char[strlen(p.x)+1];
   strcpy(x, p.x);
}
B& B::operator=(const B& p)
{  if (this != &p)
   {  A::operator=(p);
      delete x;
      x = new char[strlen(p.x)+1];
      strcpy(x, p.x);
    }
   return *this;
}
void B::print() const
{  A::print();
   cout << "B:: x " << x << endl;
}
class C : virtual public B
{public:
      C(char* = "", char* = "", char* = " ");
      ~C();
      C(const C&);
      C& operator=(const C&);
      void print() const;
 private:
     char* x;
};
C::C(char* a, char* b, char* c): B(a, b)
{  x = new char[strlen(c)+1];
   strcpy(x, c);
}
C::~C()
{  cout << "~C()\n";
   delete x;
}
C::C(const C& p) : B(p)
{  x = new char[strlen(p.x)+1];
   strcpy(x, p.x);
}
C& C::operator=(const C& p)
{  if (this != &p)
   {  B::operator=(p);
```

```
        delete x;
        x = new char[strlen(p.x)+1];
        strcpy(x, p.x);
      }
    return *this;
}
void C::print() const
{  B::print();
   cout << "C:: x " << x << endl;
}
void main()
{  A *ptr1 = new B("O", "K");
   ptr1->print();
   delete ptr1;
   ptr1 = new C("M", "A", "M");
   ptr1->print();
   delete ptr1;
}
```

възниква грешка. Поправете я. Какъв е резултатът от изпълнението на поправената програма?

Тест 15

Приложение – Входно/изходни операции. В/И оператори, дефинирани от потребителя. Форматиране

Задача 1. Да се предефинират операторите << и >> за да могат да извеждат и въвеждат стек от цели числа. Целите числа се въвеждат от клавиатурата като между числата може да има произволни знаци.

Задача 2. Какъв е резултатът от изпълнението на програмата?

```
#include <iostream.h>
#include <iomanip.h>
void main()
{cout.width(25); cout.precision(8);
 cout.setf(ios::right|ios::showpoint|ios::showpos);
 double y = 9876543219.7678;
 cout << y << endl
      << setiosflags(ios::scientific) << y << '\n'
      << resetiosflags(ios::scientific)
      << setiosflags(ios::fixed)
      << y << '\n' << resetiosflags(ios::fixed)
      << y << endl;
}
```

Задача 3. Напишете програмен фрагмент, който решава задачата:

а)Извежда 100 в 16 позиционна система с префикс 0x.

б) Извежда 1.234 в поле с широчина 9 и с предшестващи 0.

в) Извежда 100.4562738 закръглено до: десети, стотни, хилядни, десетохилядни и стохилядни.

г) Чрез член-функцията read() въвежда символен масив от 50 символа.

д) Чете 10 последователни символа от входния поток. Четенето продължава до разделителя ‘.’. Да се не прескочи символът ‘.’ от входния поток.

е) Чете 10 последователни символа от входния поток. Четенето продължава до разделителя ‘.’. Да се прескочи символът ‘.’ от входния поток.

ж) Като използва манипулатори извежда 1.92876, 1.925564 и 1.925845 с точност 3 в общ; в научен и във фиксиран формат.

з) Извежда 250 със и без знак.

Задача 4. Какъв е резултатът от изпълнението на фрагмента?

а) cout << “abcdefg\0hijklm\n”;

б) cout.write(“abcdefg\0hijklm\n”, 10);

в)
```
char c1, c2, c3;
   cin.get(c1).get(c2).get(c3);
   cout << c1 << c2 << c3 << endl;
```
ако входният поток съдържа: a b c d e

г)
```
char c1, c2, c3;
   cin >> c1 >> c2 >> c3;
   cout << c1 << c2 << c3 << endl;
```
ако входният поток съдържа: a b c d e

д)
```
char s1[10], s2[10];
  cin.getline(s1, 10, ‘?’).getline(s2, 10, ‘?’);
  cout << s1 << “  “ << s2 << endl;
```
ако входният поток съдържа: a+b=?2-4=-2 123 67