Inheritance

Single Inheritance

```
// inheritance using English Distances
#include <iostream>
using namespace std;
enum posneg { pos, neg };          //for sign in DistSign
////////////////////////////////////////////////////////////////
class Distance                     //English Distance class
   {
   protected:                      //NOTE: can't be private
      int feet;
      float inches;
   public:                         //no-arg constructor
      Distance() : feet(0), inches(0.0)
         {  }                      //2-arg constructor)
      Distance(int ft, float in) : feet(ft), inches(in)
         {  }
      void getdist()               //get length from user
         {
         cout << "\nEnter feet: ";  cin >> feet;
         cout << "Enter inches: ";  cin >> inches;
         }
      void showdist() const        //display distance
         { cout << feet << "\'-" << inches << '\"'; }
   };
////////////////////////////////////////////////////////////////
class DistSign : public Distance  //adds sign to Distance
   {
   private:
      posneg sign;                 //sign is pos or neg
   public:
                                   //no-arg constructor
      DistSign() : Distance()      //call base constructor
         { sign = pos; }           //set the sign to +

                                   //2- or 3-arg constructor
      DistSign(int ft, float in, posneg sg=pos) :
              Distance(ft, in)     //call base constructor
         { sign = sg; }            //set the sign

      void getdist()               //get length from user
              {
              Distance::getdist();     //call base getdist()
              char ch;                 //get sign from user
              cout << "Enter sign (+ or -): ";  cin >> ch;
              sign = (ch=='+') ? pos : neg;
              }
      void showdist() const         //display distance
              {
              cout << ( (sign==pos) ? "(+)" : "(-)" );  //show sign
              Distance::showdist();                     //ft and in
              }
   };
////////////////////////////////////////////////////////////////
int main()
   {
```

```
   DistSign alpha;                  //no-arg constructor
   alpha.getdist();                 //get alpha from user

   DistSign beta(11, 6.25);         //2-arg constructor

   DistSign gamma(100, 5.5, neg);   //3-arg constructor

                               //display all distances
   cout << "\nalpha = ";  alpha.showdist();
   cout << "\nbeta = ";  beta.showdist();
   cout << "\ngamma = ";  gamma.showdist();
   cout << endl;
   return 0;
   }
```

Overriding functions in the subclasses

```
// models employee database using inheritance
#include <iostream>
using namespace std;
const int LEN = 80;                 //maximum length of names
////////////////////////////////////////////////////////////
class employee                      //employee class
   {
   private:
      char name[LEN];               //employee name
      unsigned long number;         //employee number
   public:
      void getdata()
         {
         cout << "\n   Enter last name: "; cin >> name;
         cout << "   Enter number: ";      cin >> number;
         }
      void putdata() const
         {
         cout << "\n   Name: " << name;
         cout << "\n   Number: " << number;
         }
   };
////////////////////////////////////////////////////////////
class manager : public employee     //management class
   {
   private:
      char title[LEN];              //"vice-president" etc.
      double dues;                  //golf club dues
   public:
      void getdata()
         {
         employee::getdata();
         cout << "   Enter title: ";          cin >> title;
         cout << "   Enter golf club dues: "; cin >> dues;
         }
      void putdata() const
         {
         employee::putdata();
```

```
        cout << "\n   Title: " << title;
        cout << "\n   Golf club dues: " << dues;
        }
   };
////////////////////////////////////////////////////////////
class scientist : public employee  //scientist class
   {
   private:
      int pubs;                    //number of publications
   public:
      void getdata()
         {
         employee::getdata();
         cout << "   Enter number of pubs: "; cin >> pubs;
         }
      void putdata() const
         {
         employee::putdata();
         cout << "\n   Number of publications: " << pubs;
         }
   };
////////////////////////////////////////////////////////////
class laborer : public employee    //laborer class
   {
   };
////////////////////////////////////////////////////////////
int main()
   {
   manager m1, m2;
   scientist s1;
   laborer l1;

   cout << endl;           //get data for several employees
   cout << "\nEnter data for manager 1";
   m1.getdata();

   cout << "\nEnter data for manager 2";
   m2.getdata();

   cout << "\nEnter data for scientist 1";
   s1.getdata();

   cout << "\nEnter data for laborer 1";
   l1.getdata();
                           //display data for several employees
   cout << "\nData on manager 1";
   m1.putdata();

   cout << "\nData on manager 2";
   m2.putdata();

   cout << "\nData on scientist 1";
   s1.putdata();

   cout << "\nData on laborer 1";
   l1.putdata();
   cout << endl;
   return 0;
```

```
   }
```

Public and private inheritance

```
// tests publicly- and privately-derived classes

#include <iostream>
using namespace std;
////////////////////////////////////////////////////////////////
class A                  //base class
   {
   private:
      int privdataA;     //(functions have the same access
   protected:            //rules as the data shown here)
      int protdataA;
   public:
      int pubdataA;
   };
////////////////////////////////////////////////////////////////
class B : public A       //publicly-derived class
   {
   public:
      void funct()
         {
         int a;
         a = privdataA;  //error: not accessible
         a = protdataA;  //OK
         a = pubdataA;   //OK
         }
   };
////////////////////////////////////////////////////////////////
class C : private A      //privately-derived class
   {
   public:
      void funct()
         {
         int a;
         a = privdataA;  //error: not accessible
         a = protdataA;  //OK
         a = pubdataA;   //OK
         }
   };
////////////////////////////////////////////////////////////////
int main()
   {
   int a;

   B objB;
   a = objB.privdataA;   //error: not accessible
   a = objB.protdataA;   //error: not accessible
   a = objB.pubdataA;    //OK (A public to B)

   C objC;
   a = objC.privdataA;   //error: not accessible
   a = objC.protdataA;   //error: not accessible
   a = objC.pubdataA;    //error: not accessible (A private to C)
   return 0;
   }
```

Levels of inheritance

```
// multiple levels of inheritance
#include <iostream>
using namespace std;
const int LEN = 80;                 //maximum length of names
////////////////////////////////////////////////////////////////
class employee
   {
   private:
      char name[LEN];               //employee name
      unsigned long number;         //employee number
   public:
      void getdata()
         {
         cout << "\n   Enter last name: "; cin >> name;
         cout << "   Enter number: ";      cin >> number;
         }
      void putdata() const
         {
         cout << "\n   Name: " << name;
         cout << "\n   Number: " << number;
         }
   };
////////////////////////////////////////////////////////////////
class manager : public employee     //manager class
   {
   private:
      char title[LEN];              //"vice-president" etc.
      double dues;                  //golf club dues
   public:
      void getdata()
         {
         employee::getdata();
         cout << "   Enter title: ";          cin >> title;
         cout << "   Enter golf club dues: "; cin >> dues;
         }
      void putdata() const
         {
         employee::putdata();
         cout << "\n   Title: " << title;
         cout << "\n   Golf club dues: " << dues;
         }
   };
////////////////////////////////////////////////////////////////
class scientist : public employee  //scientist class
   {
   private:
      int pubs;                     //number of publications
   public:
      void getdata()
         {
         employee::getdata();
         cout << "   Enter number of pubs: "; cin >> pubs;
         }
      void putdata() const
         {
         employee::putdata();
         cout << "\n   Number of publications: " << pubs;
         }
```

```
   };
////////////////////////////////////////////////////////////
class laborer : public employee    //laborer class
   {
   };
////////////////////////////////////////////////////////////
class foreman : public laborer     //foreman class
   {
   private:
      float quotas;  //percent of quotas met successfully
   public:
      void getdata()
         {
         laborer::getdata();
         cout << "   Enter quotas: "; cin >> quotas;
         }
      void putdata() const
         {
         laborer::putdata();
         cout << "\n   Quotas: " << quotas;
         }
   };
////////////////////////////////////////////////////////////
int main()
   {
   laborer l1;
   foreman f1;

   cout << endl;
   cout << "\nEnter data for laborer 1";
   l1.getdata();
   cout << "\nEnter data for foreman 1";
   f1.getdata();

   cout << endl;
   cout << "\nData on laborer 1";
   l1.putdata();
   cout << "\nData on foreman 1";
   f1.putdata();
   cout << endl;
   }
```

Member functions in multiple inheritance

```
// englmult.cpp
// multiple inheritance with English Distances
#include <iostream>
#include <string>
using namespace std;
////////////////////////////////////////////////////////////
class Type                          //type of lumber
   {
   private:
      string dimensions;
      string grade;
   public:                          //no-arg constructor
      Type() : dimensions("N/A"), grade("N/A")
         {  }
```

```
                                    //2-arg constructor
      Type(string di, string gr) : dimensions(di), grade(gr)
         {  }
      void gettype()                //get type from user
         {
         cout << "   Enter nominal dimensions (2x4 etc.): ";
         cin >> dimensions;
         cout << "   Enter grade (rough, const, etc.): ";
         cin >> grade;
         }
      void showtype() const         //display type
         {
         cout << "\n   Dimensions: " << dimensions;
         cout << "\n   Grade: " << grade;
         }
   };
////////////////////////////////////////////////////////////////
class Distance                      //English Distance class
   {
   private:
      int feet;
      float inches;
   public:                          //no-arg constructor
      Distance() : feet(0), inches(0.0)
         {  }                       //constructor (two args)
      Distance(int ft, float in) : feet(ft), inches(in)
         {  }
      void getdist()                //get length from user
         {
         cout << "   Enter feet: ";  cin >> feet;
         cout << "   Enter inches: ";  cin >> inches;
         }
      void showdist() const         //display distance
         { cout  << feet << "\'-" << inches << '\"'; }
   };
////////////////////////////////////////////////////////////////
class Lumber : public Type, public Distance
   {
   private:
      int quantity;                       //number of pieces
      double price;                       //price of each piece
   public:                                //constructor (no args)
      Lumber() : Type(), Distance(), quantity(0), price(0.0)
         {  }
                                     //constructor (6 args)
      Lumber( string di, string gr,      //args for Type
              int ft, float in,          //args for Distance
              int qu, float prc ) :      //args for our data
              Type(di, gr),              //call Type ctor
              Distance(ft, in),          //call Distance ctor
              quantity(qu), price(prc)   //initialize our data
          {  }
      void getlumber()
         {
         Type::gettype();
         Distance::getdist();
         cout << "   Enter quantity: "; cin >> quantity;
         cout << "   Enter price per piece: "; cin >> price;
```

```
        }
      void showlumber() const
        {
        Type::showtype();
        cout << "\n   Length: ";
        Distance::showdist();
        cout << "\n   Price for " << quantity
               << " pieces: $" << price * quantity;
        }
   };
////////////////////////////////////////////////////////////////
int main()
   {
   Lumber siding;                    //constructor (no args)

   cout << "\nSiding data:\n";
   siding.getlumber();               //get siding from user

                                     //constructor (6 args)
   Lumber studs( "2x4", "const", 8, 0.0, 200, 4.45F );

                                     //display lumber data
   cout << "\nSiding";  siding.showlumber();
   cout << "\nStuds";     studs.showlumber();
   cout << endl;
   return 0;
   }
```

Constructors in multiple inheritance

```
// multiple inheritance with English Distances
#include <iostream>
#include <string>
using namespace std;
////////////////////////////////////////////////////////////////
class Type                          //type of lumber
   {
   private:
      string dimensions;
      string grade;
   public:                          //no-arg constructor
      Type() : dimensions("N/A"), grade("N/A")
        {  }
                                    //2-arg constructor
      Type(string di, string gr) : dimensions(di), grade(gr)
        {  }
      void gettype()                //get type from user
        {
        cout << "   Enter nominal dimensions (2x4 etc.): ";
        cin >> dimensions;
        cout << "   Enter grade (rough, const, etc.): ";
        cin >> grade;
        }
      void showtype() const         //display type
        {
        cout << "\n   Dimensions: " << dimensions;
        cout << "\n   Grade: " << grade;
```

```
        }
   };
////////////////////////////////////////////////////////////////
class Distance                      //English Distance class
   {
   private:
      int feet;
      float inches;
   public:                          //no-arg constructor
      Distance() : feet(0), inches(0.0)
         {  }                       //constructor (two args)
      Distance(int ft, float in) : feet(ft), inches(in)
         {  }
      void getdist()                //get length from user
         {
         cout << "   Enter feet: ";  cin >> feet;
         cout << "   Enter inches: ";  cin >> inches;
         }
      void showdist() const         //display distance
         { cout  << feet << "\'-" << inches << '\"'; }
   };
////////////////////////////////////////////////////////////////
class Lumber : public Type, public Distance
   {
   private:
      int quantity;                       //number of pieces
      double price;                       //price of each piece
   public:                                //constructor (no args)
      Lumber() : Type(), Distance(), quantity(0), price(0.0)
         {  }
                                    //constructor (6 args)
      Lumber( string di, string gr,       //args for Type
              int ft, float in,           //args for Distance
              int qu, float prc ) :       //args for our data
              Type(di, gr),               //call Type ctor
              Distance(ft, in),           //call Distance ctor
              quantity(qu), price(prc)    //initialize our data
          {  }
      void getlumber()
         {
         Type::gettype();
         Distance::getdist();
         cout << "   Enter quantity: "; cin >> quantity;
         cout << "   Enter price per piece: "; cin >> price;
         }
      void showlumber() const
         {
         Type::showtype();
         cout << "\n   Length: ";
         Distance::showdist();
         cout << "\n   Price for " << quantity
                << " pieces: $" << price * quantity;
         }
   };
////////////////////////////////////////////////////////////////
int main()
   {
   Lumber siding;                    //constructor (no args)
```

```
   cout << "\nSiding data:\n";
   siding.getlumber();              //get siding from user

                                    //constructor (6 args)
   Lumber studs( "2x4", "const", 8, 0.0, 200, 4.45F );

                                    //display lumber data
   cout << "\nSiding";  siding.showlumber();
   cout << "\nStuds";     studs.showlumber();
   cout << endl;
   return 0;
   }
```

Ambiguity in multiple inheritance

```
// ambigu.cpp
// demonstrates ambiguity in multiple inheritance
#include <iostream>
using namespace std;
////////////////////////////////////////////////////////////////
class A
   {
   public:
      void show()  { cout << "Class A\n"; }
   };
class B
   {
   public:
      void show()  { cout << "Class B\n"; }
   };
class C : public A, public B
   {
   };
////////////////////////////////////////////////////////////////
int main()
   {
   C objC;             //object of class C
// objC.show();        //ambiguous--will not compile
   objC.A::show();     //OK
   objC.B::show();     //OK
   return 0;
   }
```

Aggregation: Classes within classes

```
// containership with employees and degrees
#include <iostream>
#include <string>
using namespace std;
////////////////////////////////////////////////////////////////
class student                   //educational background
   {
   private:
      string school;           //name of school or university
      string degree;           //highest degree earned
   public:
      void getedu()
```

```
         {
         cout << "   Enter name of school or university: ";
         cin >> school;
         cout << "   Enter highest degree earned \n";
         cout << "   (Highschool, Bachelor's, Master's, PhD): ";
         cin >> degree;
         }
      void putedu() const
         {
         cout << "\n   School or university: " << school;
         cout << "\n   Highest degree earned: " << degree;
         }
   };
////////////////////////////////////////////////////////////////
class employee
   {
   private:
      string name;            //employee name
      unsigned long number;   //employee number
   public:
      void getdata()
         {
         cout << "\n   Enter last name: "; cin >> name;
         cout << "   Enter number: ";      cin >> number;
         }
      void putdata() const
         {
         cout << "\n   Name: " << name;
         cout << "\n   Number: " << number;
         }
   };
////////////////////////////////////////////////////////////////
class manager                  //management
   {
   private:
      string title;            //"vice-president" etc.
      double dues;             //golf club dues
      employee emp;            //object of class employee
      student stu;             //object of class student
   public:
      void getdata()
         {
         emp.getdata();
         cout << "   Enter title: ";          cin >> title;
         cout << "   Enter golf club dues: "; cin >> dues;
         stu.getedu();
         }
      void putdata() const
         {
         emp.putdata();
         cout << "\n   Title: " << title;
         cout << "\n   Golf club dues: " << dues;
         stu.putedu();
         }
   };
////////////////////////////////////////////////////////////////
class scientist                //scientist
   {
```

```
   private:
      int pubs;               //number of publications
      employee emp;           //object of class employee
      student stu;            //object of class student
   public:
      void getdata()
         {
         emp.getdata();
         cout << "   Enter number of pubs: "; cin >> pubs;
         stu.getedu();
         }
      void putdata() const
         {
         emp.putdata();
         cout << "\n   Number of publications: " << pubs;
         stu.putedu();
         }
   };
////////////////////////////////////////////////////////////////
class laborer                  //laborer
   {
   private:
      employee emp;           //object of class employee
   public:
      void getdata()
         { emp.getdata(); }
      void putdata() const
         { emp.putdata(); }
   };
////////////////////////////////////////////////////////////////
int main()
   {
   manager m1;
   scientist s1, s2;
   laborer l1;

   cout << endl;
   cout << "\nEnter data for manager 1";    //get data for
   m1.getdata();                            //several employees

   cout << "\nEnter data for scientist 1";
   s1.getdata();

   cout << "\nEnter data for scientist 2";
   s2.getdata();

   cout << "\nEnter data for laborer 1";
   l1.getdata();

   cout << "\nData on manager 1";           //display data for
   m1.putdata();                            //several employees

   cout << "\nData on scientist 1";
   s1.putdata();

   cout << "\nData on scientist 2";
   s2.putdata();
```

```
cout << "\nData on laborer 1";
l1.putdata();
cout << endl;
return 0;
}
```