**Topic: Inheritance**

- Hierarchies of concepts
- Inheritance vs. class templates
- C++ syntax for inheritance
- Replacing `switch` logic
- Base and derived classes
- Generic and heterogeneous collections
- Protected and private members
- Public vs. private inheritance
- Multiple inheritance
- Polymorphism uses inheritance

---

**A taxonomy uses inheritance**

```
life forms
  /   \
/     /
animals  plants
  /     /
/     /     /
insects  birds  mammals  trees  seaweeds
  /   /   /
beetles  flies  
```

- *Example:* a tree is a *kind of* plant
Object-oriented design and inheritance

- A class that embodies a concept that is a subcategory of some other concept may use inheritance, representing that other concept as a class.
- The is-a or kind-of relationship is an inheritance relationship

Numeric subclasses

These three notations for inheritance are equivalent
Kinds of subclassing

- Class templates
- Ordinary inheritance
- Inheritance with polymorphism

Inheritance vs. class templates

- A template is not a class but a tool to create many classes
- Inheritance lets client programmer extend features of base class even without its source code
- Inheritance permits multiple levels of subclassing
- Inheritance is the basis for polymorphism
An *employees* example

- Problem: to calculate paychecks for an employee roster that includes salaried and hourly employees
- Some employees have a *salary*
- Others have an hourly *wage rate* and an *hours worked* value
Inheritance replaces *switch* logic

- Without inheritance, each employee would have extra members and a flag:
  ```
  struct employees {
    int ID, pay_category;
    double salary, hours, wage_rate;
  };
  ```
- In C, a loop would calculate weekly pay for an employee:
  ```
  switch(emp.pay_category) {
    case HOURS:
      pay = hours_worked * wage_rate;
      break;
    case SALARIED: pay = salary / 52;
      break;
  }
  ```

Inheritance with employee classes

```cpp
class employees
{
 public:
  employees(int I) : ID(I) { };
 private:
  int ID;
};

class hourly : public employees
{
 public:
  hourly(int I, double W, double H) :
    wage_rate(w), hours(H), employees(I)
  { };
 private:
  double wage_rate, hours;
};
```
Base and derived classes

- Derived class inherits members from base class
- Base class encapsulates a more general category
- Derived class = subclass = descendant
- Base class = superclass = ancestor

A base class may...

- have multiple derived classes

- be a derived class too

- have no instances
**Base-class constructor runs first**

```cpp
class games
{
public:
    games() { cout << "games\n"; }
};

class board_games : public games
{
public:
    board_games() { cout << "board_games\n"; }
};

void main()
{
    board_games g;
}
```

**Output:**

games
board_games

---

**Derived-class constructors**

- May initialize base-class members using member initializer with base-class name:
  ```cpp
  hourly(int I) : employees(I)
  {
      wage_rate = hours = 0;
  };
  ```
- To instantiate: `hourly emp(1234);`
- If derived-class constructor does not do this, then base class must have default constructor
- Constructors and friend status are not inherited
Base and derived class member names

- Duplicate member identifier in derived class overrides base-class member identifier
- To access base-class identifier, use scope resolution operator

Protected members

class moms
{
protected:
   char heirloom[80];
};

class daughters : public moms
{
public:
   daughters()
   { strcpy(heirloom,"armchair\n");
   }
   char* get_heirloom() { return heirloom; }
};

void main()
{
daughters sally;
}
A long-string is a kind of string

- Given the existence of a string class and the need for a type to store strings of arbitrary length, it’s reasonable to extend the `strings` class to handle a linked list of strings considered as one long string:

  ```cpp
  class lstrings : public strings {
  public:
    lstrings();
    lstrings(char* s);
    ...
  private:
    lstrings* next;
  };
  ```

Private members of a base class

```cpp
class dads {
    char secret[80];
 public:
    dads() { strcpy(secret,"I know DOS"); } }
}

class sons : public dads {
 public:
    sons() { cout "Secret is " secret; }
    }

void main()
{ sons junior; }
```
Generic arrays using inheritance

```cpp
class arrays {
protected:
    int max_size, size;
    void **element;
public:
    arrays() { max_size = size = 0; }
    void set_max_size(int sz) { max_size = sz; }
    void *get_element(int subscript) {
        if (subscript < size) return element[subscript];
        else return NULL;
    }
    void set_element(int subscript, void *value) {
        if (subscript < size) element[subscript] = value;
    }
    void insert(void *value) {
        if (size < max_size) element[size++] = value;
    }
};

class string_arrays : public arrays {
public:
    string_arrays(int sz) {
        set_max_size(sz);
        element = (void **)(new char_ptrs[sz][array.h, strarray.cpp])
    }
};
```
Public vs. private inheritance

- With private inheritance, all base-class members become private in the derived class
- A class $C1$ that declares its base class public enables $C1$’s descendants to have access to base class’s public and protected members
- Private inheritance may help a class designer restrict access to inappropriate base-class operations, e.g., list operations in a stack class

Multiple inheritance

![Diagram of Student, Employee, and Grad assistant with age common to both base classes](image.png)

**Problem:** What if Grad assistant stores age at Student, retrieves from Employee?
Polymorphism uses inheritance

• Application programmer who uses application-class library writes a derived class that redefines the base-class event handler
• Each element of a list of base-class pointers points to a derived-class object

Example: processing a mixed payroll of hourly and salaried employees

Discussion problems

1. Derive class Supervisor from class Employee. A supervisor has an office-key number.
2. Implement a stack class based on a linked list, with and without inheritance.
3. Implement a teaching-assistant class with multiple inheritance, if students and employees each have an age member.