Inheritance

1. Introduction (15)

Inheritance allows a new class to be based on an existing class. The new class automatically inherits all the member variables and functions, except the constructors and destructor, or the class it is based on.

In the real world, you can find many objects that are specialized versions of other more general objects. When one object is a specialized version of another object, there is an "*is a*" relationship between them.

- A student *is a* person.
- A teacher *is a* person.
- A bumblebee *is an* insect.
- A grasshopper *is an* insect.
- A coconut tree *is a* plant.
- A rose bush *is a* plant.

In object-oriented programming, *inheritance* is used to create an "is a" relationship between classes. Inheritance involves a base class and a derived class. The **base class** is the general class and the **derived class** is the specialized class. The derived class inherits all the member variables and member functions of the base class. Furthermore, new member variables and functions may be added to the derived class to make it more specialized than the base class.

2. Base and Derived Class

`Person.h`

```
#include <string>
using namespace std;

class Person {
protected:
    string name;
    string id;
public:
    Person();
    Person(string n, string i);
    string getName();
    void setName(string n);
};
```

Note the use of the keyword **protected**.

- **public** – public members are accessible from outside the class.
- **private** – private members are not accessible from outside the class.
- **protected** – protected members are not accessible from outside the class, but they are accessible from a derived class.

Person.cpp

```cpp
#include <iostream>
#include "Person.h"
using namespace std;

Person::Person() {
    name = "";
    id = "";
}

Person::Person(string n, string i) {
    name = n;
    id = i;
}

string Person::getName() {
    return name;
}

void Person::setName(string n) {
    name = n;
}
```

Student.h

```cpp
#include "Person.h"

class Student : public Person {     // derived class
private:
    double score;
public:
    Student();
    Student(string n, string i, double s);
    void setScore(double s);
    double getScore();
};
```

Student.cpp

```cpp
#include "Student.h"

Student::Student() {
    name = "";
    id = "";
    score = 0;
}

Student::Student(string n, string i, double s) : Person(n, i) {
    score = s;
}

void Student::setScore(double s) {
    score = s;
}

double Student::getScore() {
```

```
    return score;
}
```

Teacher.h

```
#include "Person.h"

class Teacher : public Person {     // derived class
private:
    string education;
public:
    Teacher(string n, string i, string e);
};
```

Teacher.cpp

```
#include "Teacher.h"

Teacher::Teacher(string n, string i, string e) : Person(n, i) {
    education = e;
}
```

main.cpp

```
#include <iostream>
#include "Student.h"
using namespace std;

int main() {
    Student p1("Andy", "1234", 89);
    Teacher p2("Dr Hwang", "1234", "Ph.D");
}
```

3. *Problems* (Problems with an asterisk are more difficult)

1. Write the insect base class with some meaningful data members and member functions.
2. Write the bumblebee derived class with some additional meaningful data members and member functions.
3. Write the grasshopper derived class with some additional meaningful data members and member functions.
4. Write the plant base class with some meaningful data members and member functions.
5. Write the coconut tree derived class with some additional meaningful data members and member functions.
6. Write the rose bush derived class with some additional meaningful data members and member functions.

