Back Next Top

C++ Concept Map

C++ Inheritance

Inheritance is a way of relating two classes so that one class may use another class's members without redefining them (another way is using the friend declaration). A class may be derived from a base class by using the inheritance syntax:

```
class base { ... };
```

```
class derived : base { ... };
```

In fact if this done as above, all the members of the class base become private in the class derived, so it is better to use public inheritance:

```
class base { ... };
```

```
class derived : public base { ... };
```

In this way, all the members, whether data or functions, of base retain their access control category: public members become public, private members remain private. The rules of scope still apply, however, so that function members of the derived class cannot access the inherited members directly, unless the base class declares them to *protected* (or public). In the example below, the class `derived` contains a member function `accessP` that accesses a data member `dm1` of the class `base`, from which it inherits `dm1`. If the access control of `dm1` was private then the function `accessP` would not compile. It would work for public access, but rather than open up access to users of the class, the protected keyword laves the member private, execept for derived classes: a useful convenience for programmers.

```
class base {
protected:
  int dm1;
...
};

class derived: public base {
...
public:
  void accessP() { ... dm1 ... }
...
};
```

Inheritance makes two new types which can be used separately, if necessary. However, more common is to use them as sub- and super-type.

Please mail any corrections and/or suggestions to Roger Hartley at rth@cs.nmsu.edu