CHAPTER 16

Access Levels

Every class member has an accessibility level that determines where the member will be visible. There are three of them available in C++: `public`, `protected` and `private`. The default access level for class members is `private`. To change the access level for a section of code, an access modifier is used followed by a colon. Every field or method that comes after this label will have the specified access level, until another access level is set or the class declaration ends.

```
class MyClass
{
  int myPrivate;

 public:
  int myPublic;
  void publicMethod();
};
```

Private access

All members regardless of their access level are accessible in the class in which they are declared, the enclosing class. This is the only place where private members can be accessed.

```
class MyClass
{
  // Unrestricted access
  public: int myPublic;

  // Defining or derived class only
  protected: int myProtected;

  // Defining class only
  private: int myPrivate;
```

```
  void test()
  {
    myPublic    = 0; // allowed
    myProtected = 0; // allowed
    myPrivate   = 0; // allowed
  }
};
```

Protected access

A protected member can also be accessed from inside a derived class, but it cannot be reached from an unrelated class.

```
class MyChild : public MyClass
{
  void test()
  {
    myPublic    = 0; // allowed
    myProtected = 0; // allowed
    myPrivate   = 0; // inaccessible
  }
};
```

Public access

Public access gives unrestricted access from anywhere in the code.

```
class OtherClass
{
  void test(MyClass& c)
  {
    c.myPublic    = 0; // allowed
    c.myProtected = 0; // inaccessible
    c.myPrivate   = 0; // inaccessible
  }
};
```

Access level guideline

As a guideline, when choosing an access level it is generally best to use the most restrictive level possible. This is because the more places a member can be accessed, the more places it can be accessed incorrectly, which makes the code harder to debug. Using restrictive access levels will also make it easier to modify the class without breaking the code for any other programmers using that class.

Friend classes and functions

A class can be allowed to access the private and protected members of another class by declaring the class a friend. This is done by using the `friend` modifier. The friend is allowed to access all members in the class where the friend is defined, but not the other way around.

```
class MyClass
{
  int myPrivate;

  // Give OtherClass access
  friend class OtherClass;
};

class OtherClass
{
  void test(MyClass c) { c.myPrivate = 0; } // allowed
};
```

A global function can also be declared as a friend to a class in order to gain the same level of access.

```
class MyClass
{
  int myPrivate;

  // Give myFriend access
  friend void myFriend(MyClass c);
};

void myFriend(MyClass c) { c.myPrivate = 0; } // allowed
```

Public, protected and private inheritance

When a class is inherited in C++ it is possible to change the access level of the inherited members. Public inheritance allows all members to keep their original access level. Protected inheritance reduces the access of public members to protected. Private inheritance restricts all inherited members to private access.

```
class MyChild : private MyClass
{
  // myPublic is private
  // myProtected is private
  // myPrivate is private
};
```

Private is the default inheritance level, although public inheritance is the one that is nearly always used.