```
{                                    函数（即非成员函数）或全局数据
}

animal::~animal()
{
}

void animal::eat()                   //注意：虽然我们在函数体中什么也没写，但仍然是实现了
                                     这个函数
{
}

void animal::sleep()
{
}

void animal::breathe()               //注意，在头文件（.h 文件）中加了 virtual 后，在源文
                                     件（.cpp 文件）中就不必再加 virtual 了
{

    cout<<"animal breathe"<<endl;
}
```

fish.h

```
#include "animal.h"                  //因 fish 类从 animal 类继承而来，要让编译器知道
                                     animal 是一种类的类型，就要包含 animal.h 头文件

class fish:public animal
{
public:
     void breathe();
};
```

fish.cpp

```
#include "fish.h"
#include <iostream.h>
void fish::breathe()
{
    cout<<"fish bubble"<<endl;
}
```

EX10.cpp

```
#include "animal.h"
#include "fish.h"
void fn(animal *pAn)
{
    pAn->breathe();
```

```
}
void main()
{
    animal *pAn;
    fish fh;
    pAn=&fh;
    fn(pAn);
}
```

现在我们就可以按下键盘上的 F7 功能键编译整个工程了，编译结果如下：

```
\animal.h(2) : error C2011: 'animal' : 'class' type redefinition
```

为什么会出现类重复定义的错误呢？请读者仔细查看 EX10.cpp 文件，在这个文件中包含了 animal.h 和 fish.h 这两个头文件。当编译器编译 EX10.cpp 文件时，因为在文件中包含了 animal.h 头文件，编译器展开这个头文件，知道 animal 这个类定义了，接着展开 fish.h 头文件，而在 fish.h 头文件中也包含了 animal.h，再次展开 animal.h，于是 animal 这个类就重复定义了。

读者可以测试一下，如果我们多次包含 iostream.h 这个头文件，也不会出现上面的错误。要解决头文件重复包含的问题，可以使用条件预处理指令。修改后的头文件如下：

animal.h

```
#ifndef ANIMAL_H_H
#define ANIMAL_H_H
class animal
{
public:
    animal();
    ~animal();
    void eat();
    void sleep();
    virtual void breathe();
};
#endif
```

//我们一般用#define 定义一个宏，是为了在程序中使用，使程序更加简洁，维护更加方便，然而在此处，我们只是为了判断 ANIMAL_H_H 是否定义，以此来避免类重复定义，因此我们没有为其定义某个具体的值。在选择宏名时，要选用一些不常用的名字，因为我们的程序经常会跟别人写的程序集成，如果选用一个很常用的名字（例如：X），有可能会造成一些不必要的错误

fish.h

```
#include "animal.h"
#ifndef FISH_H_H
#define FISH_H_H
class fish:public animal
{
public:
    void breathe();
```

```
};
#endif
```

我们再看 EX10.cpp 的编译过程。当编译器展开 animal.h 头文件时，条件预处理指令判断 ANIMAL_H_H 没有定义，于是就定义它，然后继续执行，定义了 animal 这个类；接着展开 fish.h 头文件，而在 fish.h 头文件中也包含了 animal.h，再次展开 animal.h，这个时候条件预处理指令发现 ANIMAL_H_H 已经定义，于是跳转到#endif，执行结束。

通过分析，我们发现在这次的编译过程中，animal 这个类只定义了一次。

提示：Windows 2000 初始安装完毕后，对于已知文件类型的扩展名是隐藏的，例如："test.txt" 这个文件，在资源浏览器中看到的是 "test"，在这种情况下，修改其文件名为 "test.cpp" 时，实际的文件名是 "test.cpp.txt"，仍然是文本文件。因此在 Win2000 下做开发时，要取消 "隐藏已知文件类型的扩展名" 这一选项。

操作步骤：在资源浏览器（或我的电脑）中，选择菜单中的 "工具->文件夹选项（O）..."，选择 "查看" 标签页，将滚动栏拖到底端，将 "隐藏已知文件类型的扩展名" 复选框中的对号（√）取消掉，单击 "确定" 按钮。

2.2.11 VC++程序编译链接的原理与过程

我们在 EX10 这个工程中，选择菜单中【Build】→【Rebuild All】，重新编译所有的工程文件，可以看到如下输出：

```
Deleting intermediate files and output files for project 'EX10 - Win32 Debug'.
--------------------Configuration: EX10 - Win32 Debug--------------------
Compiling...
EX10.CPP
fish.cpp
animal.cpp
Linking...

EX10.exe - 0 error(s), 0 warning(s)
```

从这个输出中，我们可以看到可执行程序 EX10.exe 的产生，经过了两个步骤：首先，C++编译器对工程中的三个源文件 EX10.cpp、fish.cpp、animal.cpp 单独进行编译（Compiling...）。在编译时，先由预处理器对预处理指令（#include、#define 和#if）进行处理，在内存中输出翻译单元（一种临时文件）。编译器接受预处理的输出，将源代码转换成包含机器语言指令的三个目标文件(扩展名为 obj 的文件)：EX10.obj、fish.obj、animal.obj。注意，在编译过程中，头文件不参与编译；在 EX10 工程的 Debug 目录下，我们可以看到编译生成的 obj 文件。接下来是链接过程（Linking...），链接器将目标文件和你所用到的 C++类库文件一起链接生成 EX10.exe。整个编译链接的过程如图 2.18 所示。

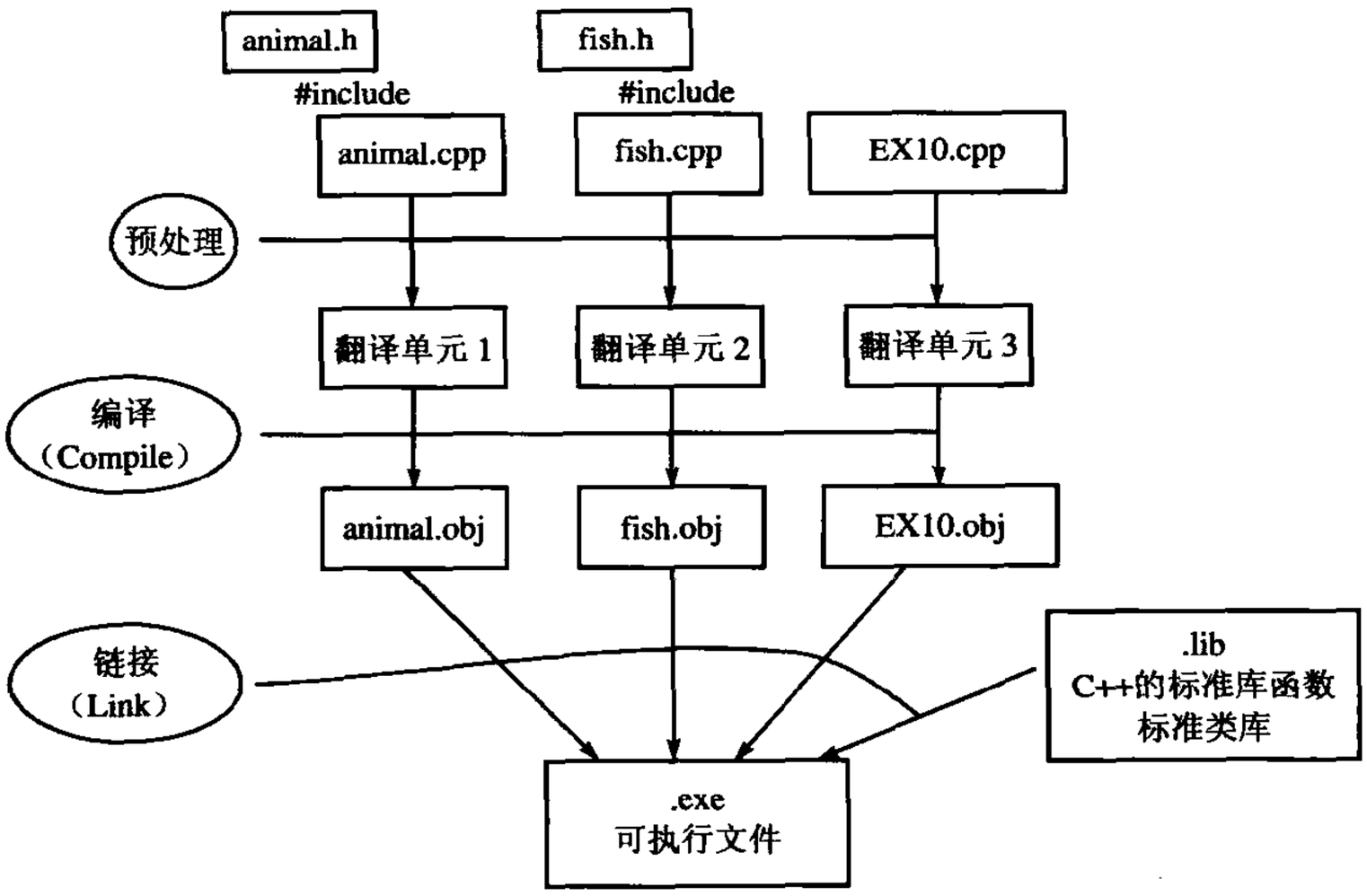

图 2.18　EX10 程序编译链接的过程

好了，到此 C++的知识就讲解完毕了。当然 C++的内容还有很多，但这一章的内容，对于我们从事 VC++开发已经足够了，还有部分 C++内容，会在后面的章节中讲解。休息一下，然后继续我们的 VC++之旅。