西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计

Programming in C

1011014

主讲：姜学锋，计算机学院

编写规模化程序

1、多文件程序结构

2、工程项目

4.10 程序组织结构

- 学习了编写函数之后，那么，更大规模的程序编写的焦点开始从运算、语句，转向函数、程序组织方面。

4.10.1 内部函数

- 函数本质上是全局的，在多文件的程序中，在连接时会检查函数在全局作用域内是否名字唯一，如果不是则出现连接错误。

4.10.1 内部函数

- 在函数定义前加上static修饰，则函数称为内部函数，定义形式为：

```
static 返回类型 函数名(形式参数列表)
{
    函数体
}
```

4.10.1 内部函数

- 按照前面的实体可见规则，内部函数仅在包含它的文件中有效。

- 之所以使用内部函数的原因是该函数逻辑上仅限定在一个文件中使用，其他文件不会用到。而且希望连接检查时永远不可能出现该函数名不唯一的连接错误，这在多人编写同一个程序的软件开发模式中是常用的策略。

4.10.2 外部函数

- 在函数定义前加上extern声明，则函数称为外部函数，定义形式为：

```
extern 返回类型 函数名(形式参数列表)
{
   函数体
}
```

- C语言中所有的函数本质上都是外部函数。因此，上面的extern都可以省略。

4.10.2 外部函数

- 在调用另一个文件中的函数时，需要用extern声明此函数是外部函数，声明形式为：

```
extern 返回类型 函数名(类型1 参数名1,类型2 参数名2,.....);
```

```
extern 返回类型 函数名(类型1, 类型2, .....);
```

- C语言中所有的函数本质上都是外部函数。因此，上面的extern都可以省略。

4.10.3 多文件结构

- 一个C程序允许由多个源文件构成，称为多文件结构程序。

- 使用多文件结构与程序规模、多人协同开发模式有关，但并非全都如此。多文件结构本质上反映面向过程的结构化设计内在要求，同时多文件结构也是高效率的编程模式。

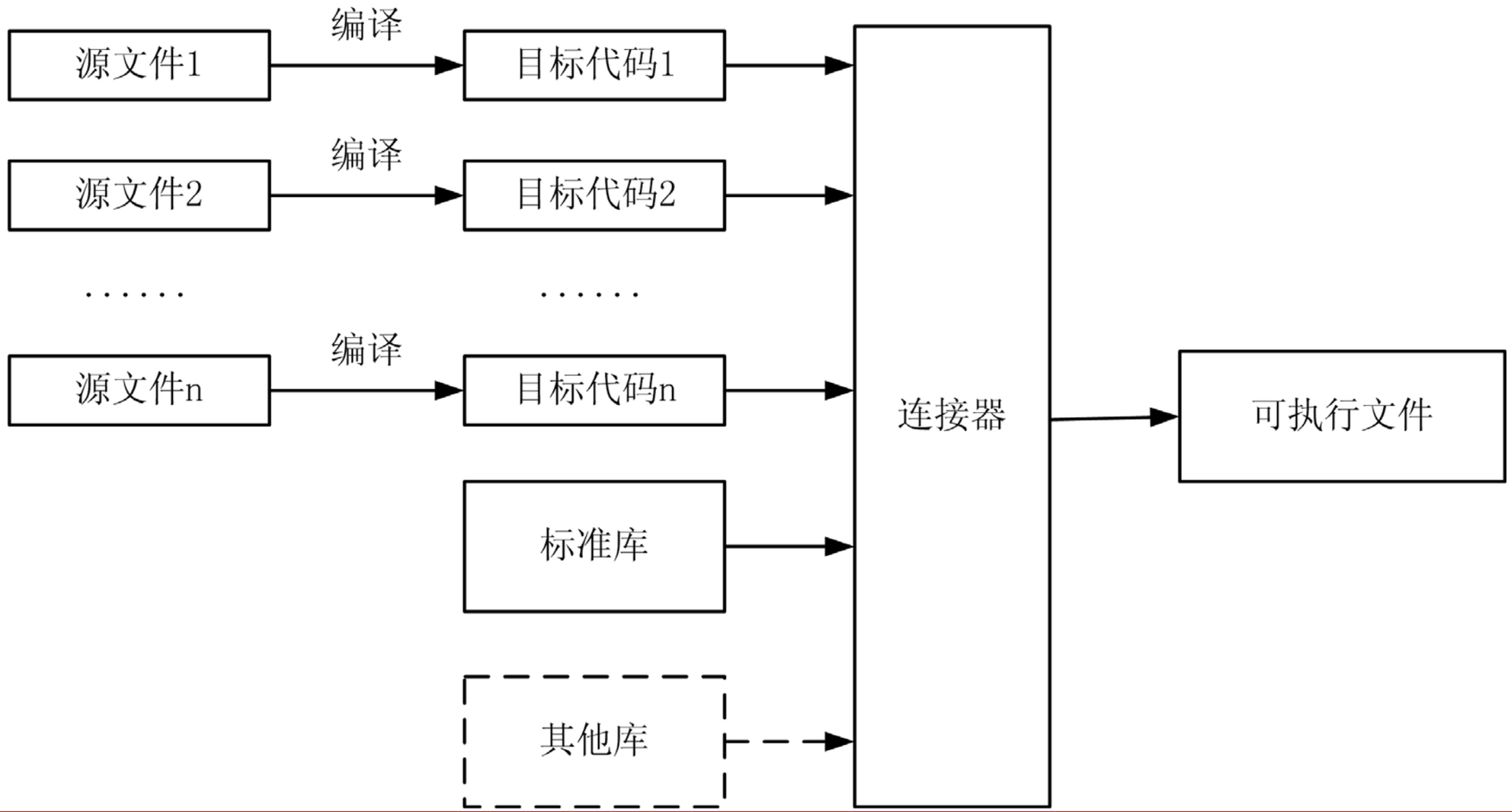
4.10.3 多文件结构

- 下面介绍编译器和连接器处理多文件结构程序的工作原理。

- 编译器编译时的基本单位是源文件。对一个文件的编译是独立的，所谓独立是指编译器不会把前面文件编译的信息自然传到下一个文件中，例如头文件包含、宏定义、各种类型声明等信息。

4.10.3 多文件结构

图4.11 多文件程序结构



4.10.3 多文件结构

- 例如如果所有的文件都用到了sin函数，那么必须在每个文件中都要包含math.h头文件，哪个文件缺少则哪个文件的sin函数调用就会因为没有函数声明而出现“未定义”的编译错误。同理，自定义的外部函数也是这样处理的。

- 但自定义的全局变量必须要加`extern`声明，因为“`extern int a;`”表示变量已经定义了，可以使用多次，而多个“`int a;`”的写法则是重复定义。

4.10.3 多文件结构

- 编译完成后得到目标代码文件，然后连接器开始工作。连接器将多个目标代码和使用到的库函数代码（通常都是二进制形式）逐个连接起来，这个过程中，它可以发现：
  - ①全局函数或全局变量是否在不同的文件中重复定义，或者在全局范围内是否有相同名字实体。
  - ②在多个目标代码和库函数中找不到全局函数或全局变量的定义。
- 若连接器成功处理完所有的连接，则产生可执行文件，一次编译连接的过程宣告完成。

4.10.3 多文件结构

- 不同的编译器用不同的扩展名来表示文件类型，表4-4是GCC和VC编译器的扩展名。

表4-4 GCC和VC文件类型及扩展名

	GCC	Visual C++
源文件	C语言：.c C++语言：.cpp、.cc、.cxx	C语言：.c C++语言：.cpp
头文件	.h	.h
目标代码文件	.o	.obj
链接库文件	.a	.lib

4.10.4 头文件与工程文件

- 1．头文件
- 头文件本质上是源文件，它是通过文件包含命令嵌入到源文件中去编译的，文件包含命令的写法格式如下：

```
①#include <头文件名>
②#include "头文件名"
```

- 一般情况下，使用系统函数时应用第①种写法，使用自定义头文件和附加库头文件时应用第②种写法。

4.10.4 头文件与工程文件

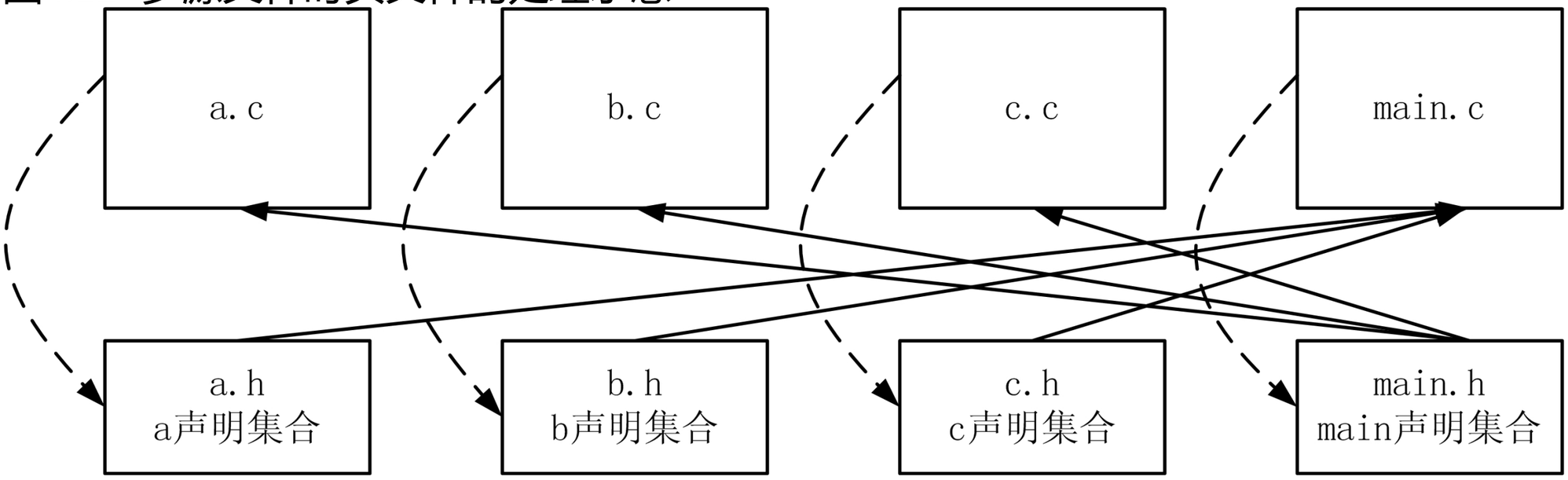
- 为什么要使用头文件呢？
- 如果是多文件结构程序，欲在文件中调用别的文件中的函数，需要有函数的声明，而且每个文件均是如此。如果是函数声明比较多的情况下，在每个文件中都写上函数声明不是好办法，很难管理。例如：某个函数定义有变动，那么所有含有这个函数声明的调用文件都需要找出来，逐一修改。

4.10.4 头文件与工程文件

- 使用头文件可以解决这个问题，其工作原理是通过将每个源文件中外部函数的函数声明等信息集中写到一个文件中，称为头文件（有别于源文件），而别的源文件只需用文件包含命令将这个头文件包含，则编译时编译器自然就有了函数声明。

4.10.4 头文件与工程文件

图4.12 多源文件时头文件的处理示意



4.10.4 头文件与工程文件

- 图中虚线表示将四个源文件（.c）各自的声明分离到头文件（.h），实线表示头文件被包含到源文件上，例如main.c文件有其他三个文件的声明集合，换言之，其他三个文件的外部函数声明等在main.c文件可见。

4.10.4 头文件与工程文件

- 头文件除了函数声明外，还经常包括全局性常量、宏定义等信息，但头文件一般不包括定义内容，例如函数定义、变量定义，只包含声明内容，这是因为头文件会被多次包含，如果是定义内容会导致重复定义，这是编写自定义头文件的重要原则：定义内容、程序实现代码等放在对应的源文件中。

4.10.4 头文件与工程文件

- 一般情况下，头文件与源文件是对应的，多个源文件就有多个头文件，文件名与源文件也是相同的，这样自成体系，便于管理。

4.10.4 头文件与工程文件

- 2．工程文件
- 多文件结构程序在编译时需要工程文件来管理，不同的编译器有不同的工程文件格式，但都支持命令行编译Makefile文件。
- Makefile文件定义了一系列的规则来指定：哪些文件需要先编译，哪些文件需要后编译，哪些文件需要重新编译，编译时使用什么参数，甚至进行更复杂的功能操作。
- Makefile文件使用脚本命令，可以执行操作系统所允许的命令。

4.10.4 头文件与工程文件

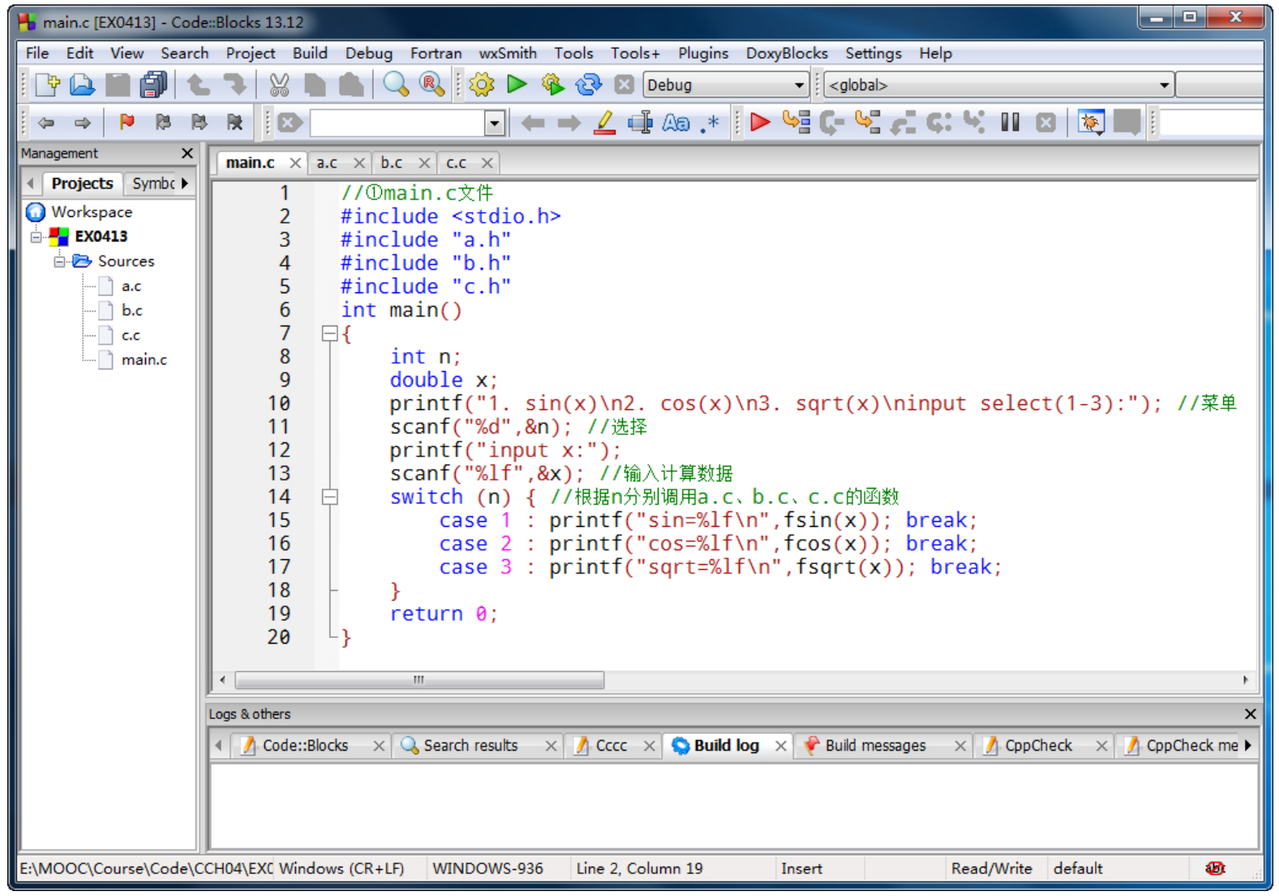
如下是GCC编译系统的Makefile文件示例，它指令编译连接四个文件。

```
CFLAGS = -nologo -W3 -Zi -MD -O1 -DWIN32
INCLUDES =
COMPILE_cpp = cl -Fo$@ $(CFLAGS) -EHsc $(INCLUDES) -c
COMPILE_c = cl -Fo$@ $(CFLAGS) $(INCLUDES) -c
a.o : a.c
    $(COMPILE_c) a.c
b.o : b.c
    $(COMPILE_c) b.c
c.o : c.c
    $(COMPILE_c) c.c
main.o : main.c
    $(COMPILE_c) main.c
```

4.10.4 头文件与工程文件

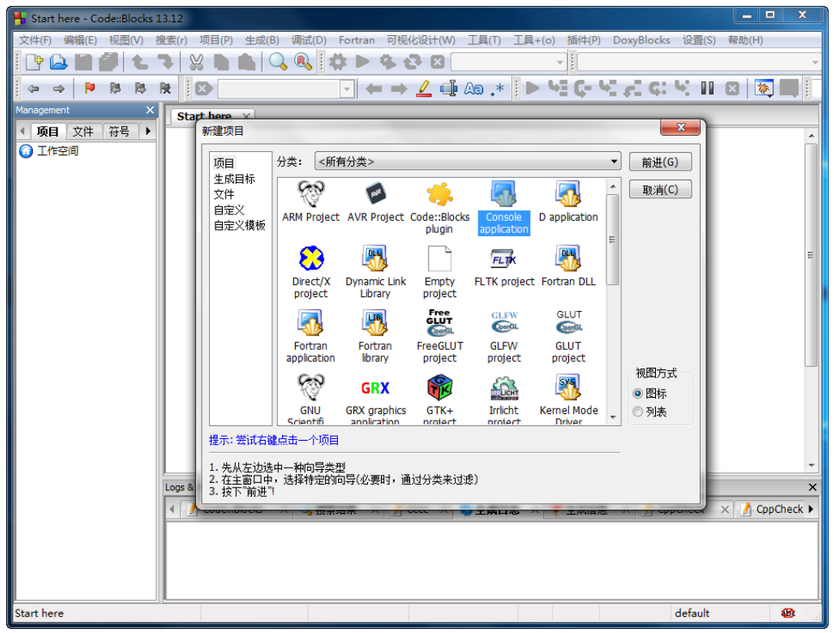
- 集成开发环境的编译器，可视化管理工程文件。如图所示，左边的树形结构为多文件列表。

CodeBlocks工程文件管理



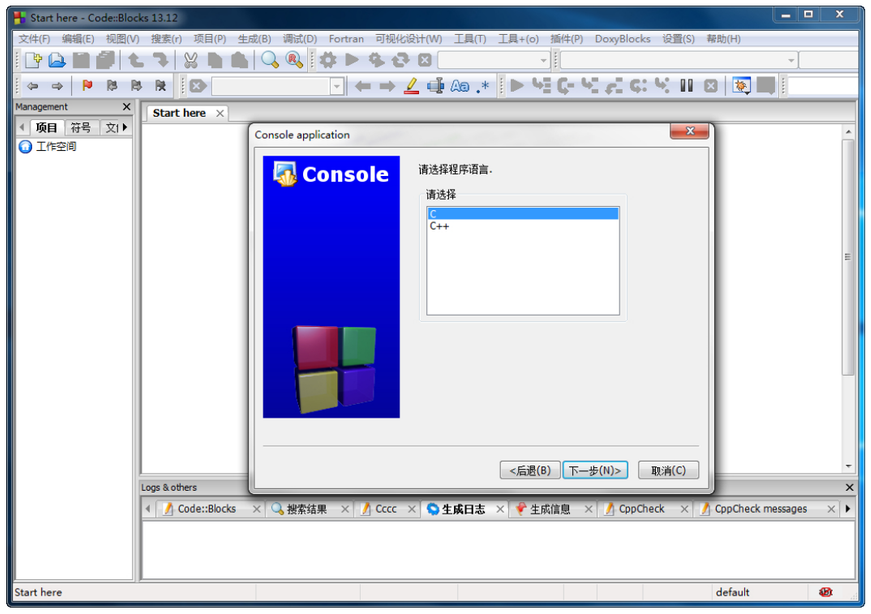
4.10.4 头文件与工程文件

- 在CodeBlocks中建立C语言多文件项目的方法
- 1.点击文件(F)-新建-项目...菜单命令，进入如下界面:



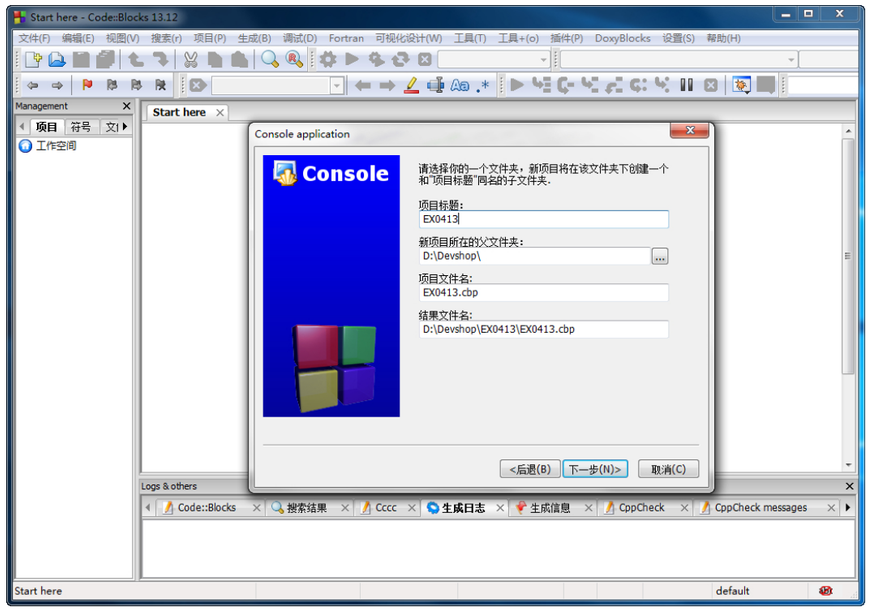
4.10.4 头文件与工程文件

- 在CodeBlocks中建立C语言多文件项目的方法
- 2. 选择Console application（控制台应用程序），然后单击“出发”按钮，进入Console application向导



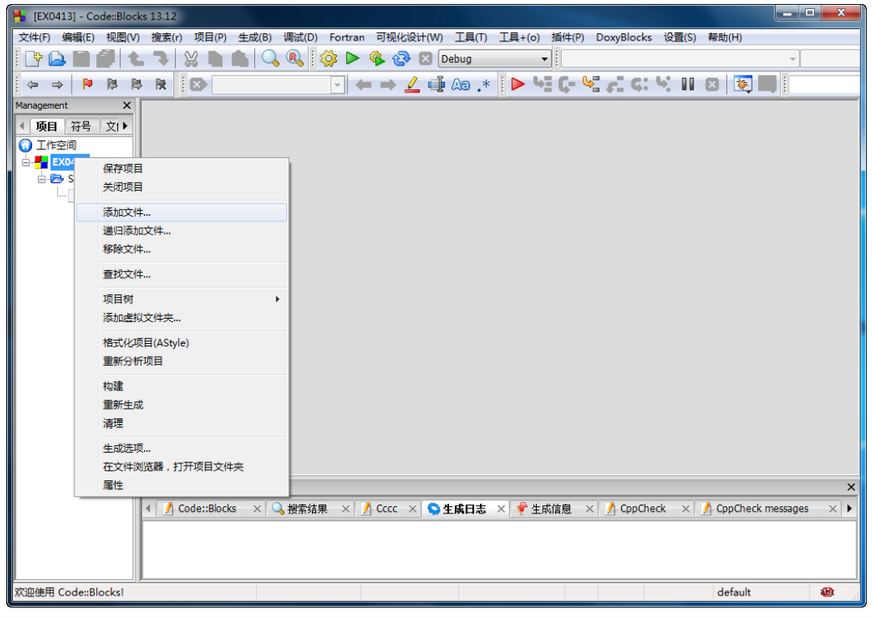
4.10.4 头文件与工程文件

- 在CodeBlocks中建立C语言多文件项目的方法
- 3. 单击“下一步”， 输入“项目标题”及“项目所在的父文件夹”等信息；



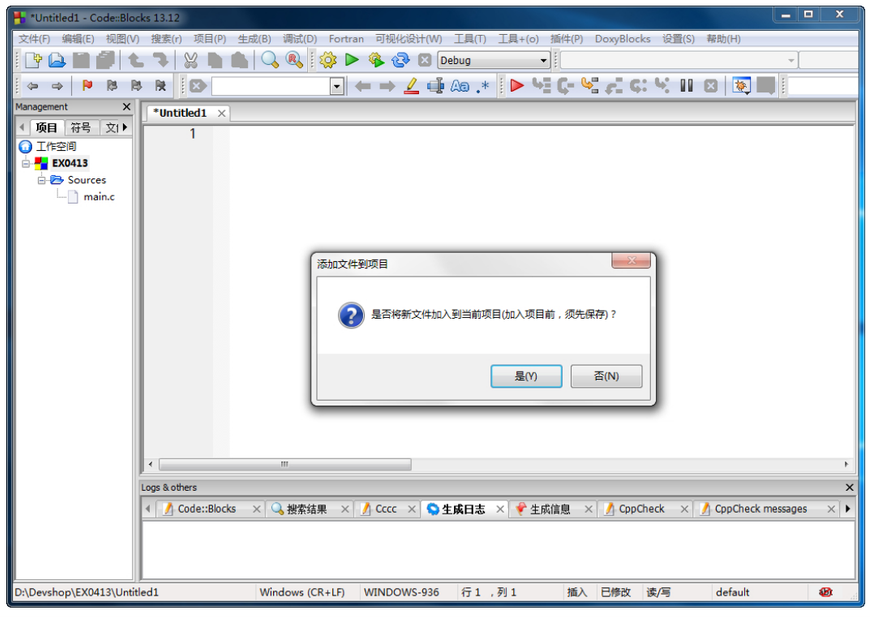
4.10.4 头文件与工程文件

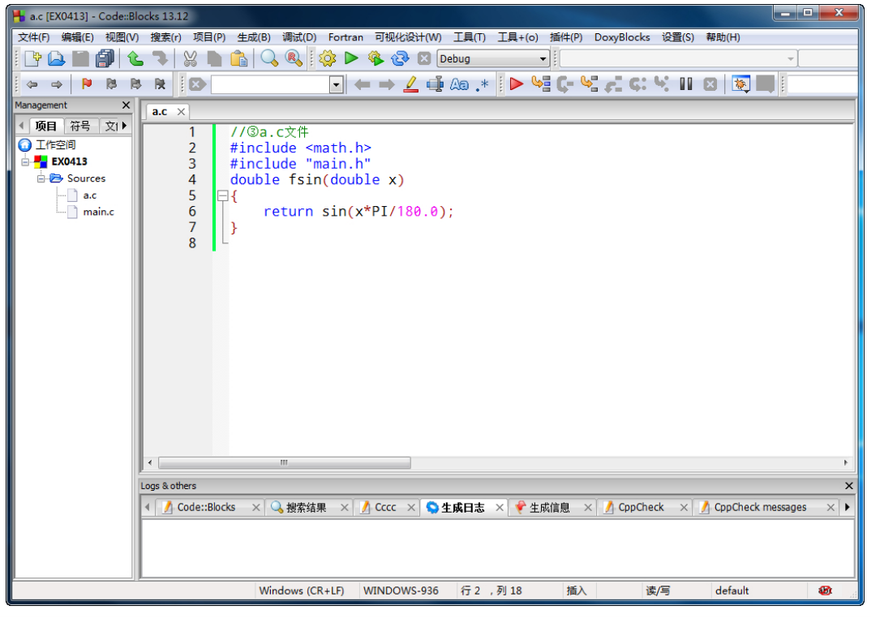
- 在CodeBlocks中建立C语言多文件项目的方法
- 4. 单击“下一步”，设置编译配置； 单击“完成”，进入编辑界面。可以在项目名称上右键“添加文件”。



4.10.4 头文件与工程文件

- 在CodeBlocks中建立C语言多文件项目的方法
- 5. 修改编辑main.c文件即可。可用“新建-文件”菜单新建多个C/C++源文件添加到这个项目中。






程序设计