

## 2.1 熟悉 C 语言程序的开发环境实验

### (一) 实验目的

- (1) 熟悉 Code::Blocks 等 C 语言程序的开发环境。
- (2) 掌握 C 程序的编辑、编译、连接和运行方法。
- (3) 通过运行简单的 C 程序，学会在集成开发环境中调试程序的方法。

### (二) 实验内容

上机输入下面的 2 个源程序，然后编译、连接、运行程序以及调试程序，使程序最终输出正确结果。

1) **源程序 1 说明：**下面的程序可实现输入三角形三边  $a$ ,  $b$ ,  $c$  的值，计算并输出三角形的面积。三角形面积公式为：

$$area = \sqrt{s(s-a)(s-b)(s-c)}$$

其中， $s = (a + b + c) / 2$  为三角形的半周长。(函数  $\text{sqrt}(x)$  求平方根，见教材附录 3)。

**源程序 1：**

```
#include <stdio.h>
#include <math.h>

int main(void)
{
    float a,b,c,s,t;
    print ("Input three edge of the triangle\n");
    scanf ("%f%f%f", &a,&b,&c);
    s=(a+b+c)/2
    t=s*(s-a)*( s-b)*(s-c);
    area=sqrt(t);
    print ("area=%d\n", area);
    return 0;
}
```

2) **源程序 2 说明：**下面的程序可实现输入 10 个整数，计算并输出这 10 个整数的累加和。

**源程序 2：**

```
#include<stdio.h>
#define N 5;
int sum(int x[],int n);

int main(void)
{
    int i,a[N],total;
    printf("Input %d integers please!\n",N);
    for(i=0;i<N;i++)
        scanf ("%d ",a[i]);
```

```

        total=sum(a,N);
        printf("The sum is %d\n" , total);
        return 0;
    }

int sum(int x[],int n);
{
    int i,s;
    for (s=0, i=0; i<n; i++);
        s+=x[i];
    return s;
}

int sum(int x[],int n);

int main(void)
{
    int i,a[10],total;
    printf("Input 10 integers please!\n");
    for(i=0;i<10;i++)
        scanf ("%d",a[i]);
    total=sum(a,10);
    printf("The sum is %d\n" , total);
    return 0;
}

int sum(int x[],int n);
{
    int i,s;
    for (s=0, i=0; i<n; i++);
        s+=x[i];
    return s;
}

```

### (三) 实验步骤及要求

#### 1. 源程序 1 实验步骤及要求

整个实验步骤包括建立工程、编辑源程序、编译和运行程序、调试程序。

##### 1) 创建一个工程

同VC一样，在Code::Blocks中编写程序首先需要创建一个工程。即可用以下三种方法之一创建工程：

- (1) 选择File/New/Projec…菜单项。
- (2) 点击File下面的图标 (New file)，再选择Project…选项。
- (3) 从Code::Blocks主界面中点击图标 (Create a new project)。

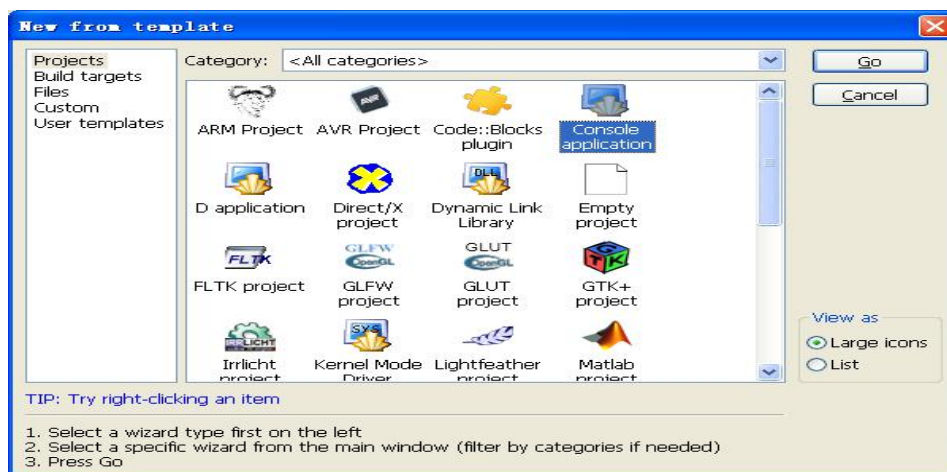


图2.1 New from template窗口

使用以上任一方法都会打开一个如图2.1所示的窗口，其中含有很多带有标签的图标，代表不同种类的工程。选择控制台应用(Console application)图标，再点击右侧的Go按钮。在弹出的对话框中点击“Next”按钮进入下一步。在弹出的对话框中有C和C++两个选项，选择“C”表示编写C控制台应用程序，然后点击下方的Next按钮进入下一步，又弹出一个对话框，如图2.2所示。图2.2所示的对话框中有4个文本框需要填写，一般填上前两个(工程名和工程文件夹路径)即可，后两个的内容会自动生成，然后再选择“Next”按钮进入下一步。

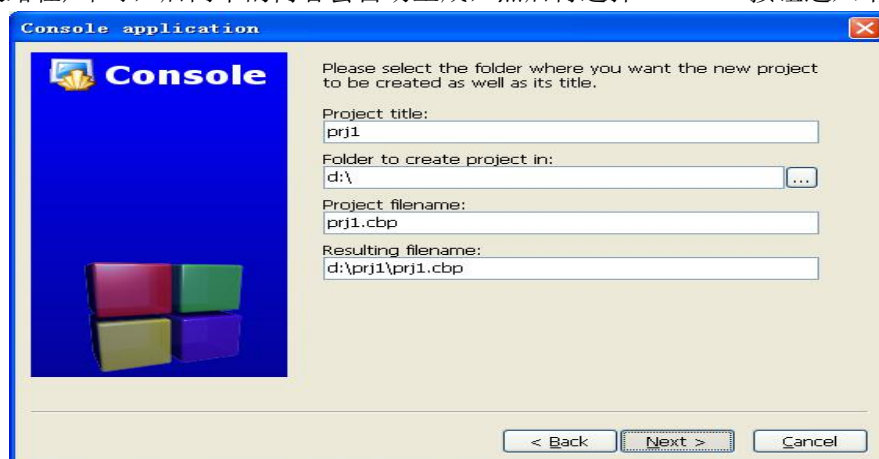


图2.2 Console application窗口

编译器选项仍旧选择默认的编译器，剩下的全部打勾，然后选择“Finish”按钮，此时即创建了一个名为“prj1”的工程（后缀为.cbp），创建结果如图2.3所示。

在图2.3所示的工作区上，逐级点击使之变成，依次展开左侧的prj1, Sources, main.c, 最后显示文件main.c的源代码。

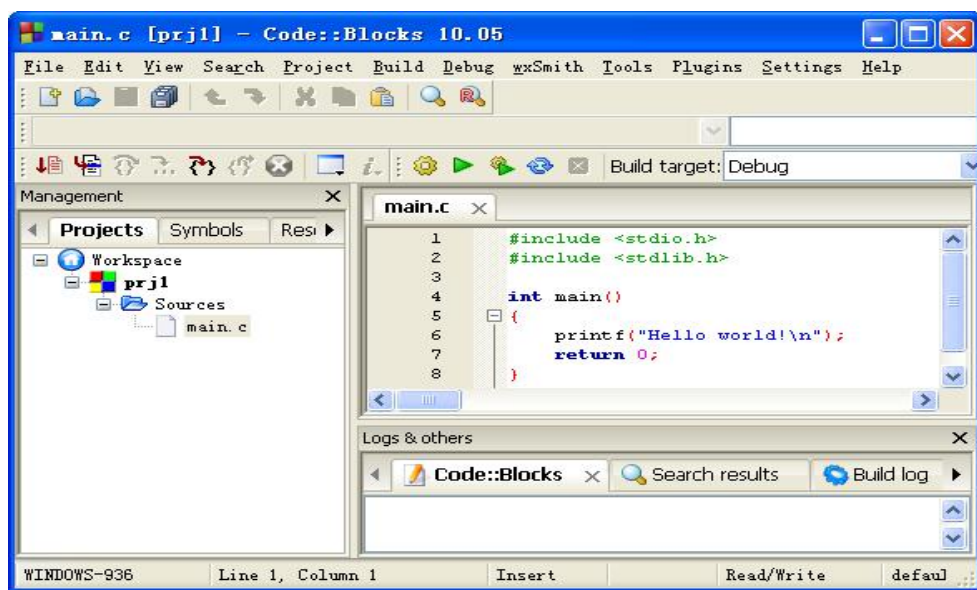


图 2.3 新建的 prj1 工程及其工作区

### (A) 了解有关概念

(1) 工程(Project): 在图 2.3 中可看到, code::blocks 创建了一个称为 prj1 的工程。左边树状结构中的“prj1”节点代表了该工程。该工程下面有 1 个逻辑文件夹 Sources, 它下面有 1 个预定义的 main.c 源文件。



一个工程就是一个或者多个源文件(包括头文件)的集合。创建一个工程可以很方便地把相关文件组织在一起。一个工程刚建立时, 一般仅仅包含一个源文件。

(2) 工作空间(Workspace): 在创建 prj1 工程的同时, Code::Blocks 也创建一个工作空间跟踪当前的工程。

### (B) 查看物理文件夹


打开 Windows 资源管理器, 选择 D:\prj1 目录, 可以看到两个文件 main.c 和 prj1.cbp, 其中 prj1.cbp 是工程文件, main.c 是该工程包含的源文件。

对于已经存在的工程, 可选择以下三种方法之一打开, 在随后打开的对话框中选择待打开的.cbp 文件。

- (1) 选择File/Open (Ctrl-o) 菜单项。
- (2) 点击File下面的  图标 (Open)。
- (3) 从Code::Blocks主界面中点击  图标 (Open an existing project)。

## 2) 编辑源文件和保存

选择工程中的源文件 main.c, 则显示该文件的源代码, 用 2.1.2 节第 1 题计算三角形面积的源程序替换它, 编辑完毕后, 保存当前源文件。

保存当前源文件方法很多, 常见的有两种, 一种是点击  按钮, 另一种是选择 File/Save file (Ctrl-S) 菜单项, 括号里是该菜单项的快捷键, 下同。

## 3) 编译和运行程序

### (A) 设置编译参数

选择Projec/Build options...菜单项, 会弹出一个关于工程prj1的Projec build options对话框, 有两个类别, debug和release, 配置debug选项, 一般将Compiler Flags菜单下的图2.4中两项打勾即可, 前者表示产生调试信息, 后者意味着给出标准的编译警告信息。

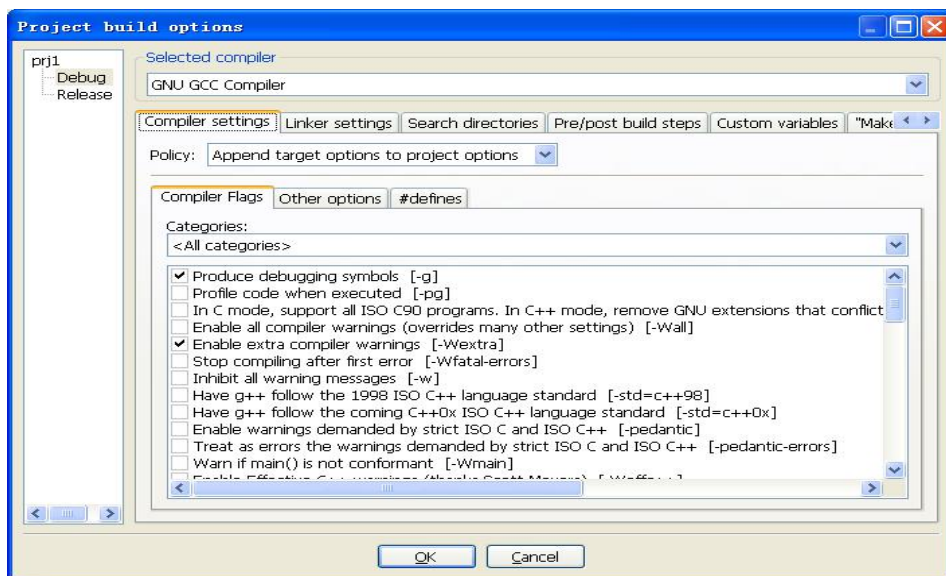




图 2.4 Project build options 对话框

编译后的目标文件可以有两个版本debug和release，debug版本的目标文件可以使用调试器对该文件进行调试。一般而言，debug版本的目标文件通常较大，因为它包含了一些用于调试的额外信息，release版本的目标文件一般较小，因为它不包含调试信息。

### (B) 编译

编译命令都在Build主菜单下，另外，还有编译工具栏： Build target: Debug ▾，它们的按钮图标相同，功能也相同。

首先选择编译目标文件(Build target)为debug版本，然后选择Build/Build(ctrl+F9)菜单项或点击工具栏的图标，则开始编译当前工程，编译器在日志窗口中给出error和warning信息，提示在哪个文件的哪一行有什么错误。

Error信息是语法错误，即程序的编写违反了C的语法规则，warning是警告信息，它虽然不影响程序执行，但有些警告常预示着隐藏较深的实际错误，必须认真弄清原因。



修改源程序中的错误技巧性非常强，能看懂哪些错误信息就先修改哪里的错误，一个实际错误有时会使编译程序产生许多出错信息行，这些错误信息中很大一部分可能没有任何帮助价值，因此需要找出有意义的错误信息。有时会发现，修改完一个错误重新编译后原来给出的很多错误信息突然变得少多了。

双击日志窗口的提示信息可跳转到错误源代码处进行修改，当用鼠标选中某个字符串(如print)时，所有和print相同的字符串都会变成红色。

分析出错原因并纠正，修改完后保存，然后重新编译，可能需要反复这个过程，直至编译成功。

**查看物理文件夹：**编译成功后，可得到 debug 版本的目标文件 main.o 和可执行文件 prjl.exe。打开 Windows 资源管理器，选择 D:\prjl 目录，查看这两个文件。

### (3) 运行

选择Build/Run(ctrl+F10)菜单命令或点击工具栏的图标，则运行编译成功的文件；表示编译并运行。

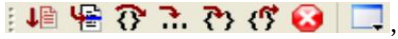
运行时会出现一个类似DOS操作系统的窗口，输入 3 4 5 后回车，屏幕上显示area=0。显然，执行的结果不对。

编译成功说明没有语法错误，但未必没有逻辑错误。尤其随着程序的愈来愈复杂，更难一次性编译成功并运行得到期望的结果，这时需要对程序进行调试以便定位错误。调试程序


有时需要在程序中的某些地方设置一些特殊“点”，让程序运行到该位置停下来，有时需要检查某些变量的值，以帮助我们检查程序中的逻辑错误。

#### 4) 调试程序——单步执行


单步执行即一次执行一行代码，相当于在每一代码行均设置了断点。一边执行，一边观测程序的流向以及查看变量、表达式的值，从中发现问题。

调试命令都在Debug主菜单下，另外，还有调试工具栏：, 要巧妙地用好调试工具栏上的这几个按钮。

##### (A) 启动调试器

把光标置于int main()之前，点击按钮(Run to cursor)，程序运行到光标处就停下来，则可以看到{前面有个黄色的箭头，而且还会出现一个没有任何输出信息的对话框。

##### (B) 打开观察窗口

为了查看程序运行中变量值的变化情况，需要打开观察变量的窗口，点击按钮，在其下拉菜单中选择Watches打开即可(也可以用Debug/Debugging windows/Watches菜单命令打开)。

为了方便观察，拖动Watches窗口到左下角，并展开各个变量，如图2.5所示。此时，从Watches窗口中可以看到定义的局部变量都是随机值，因为程序尚未执行到给这些变量赋值的语句。

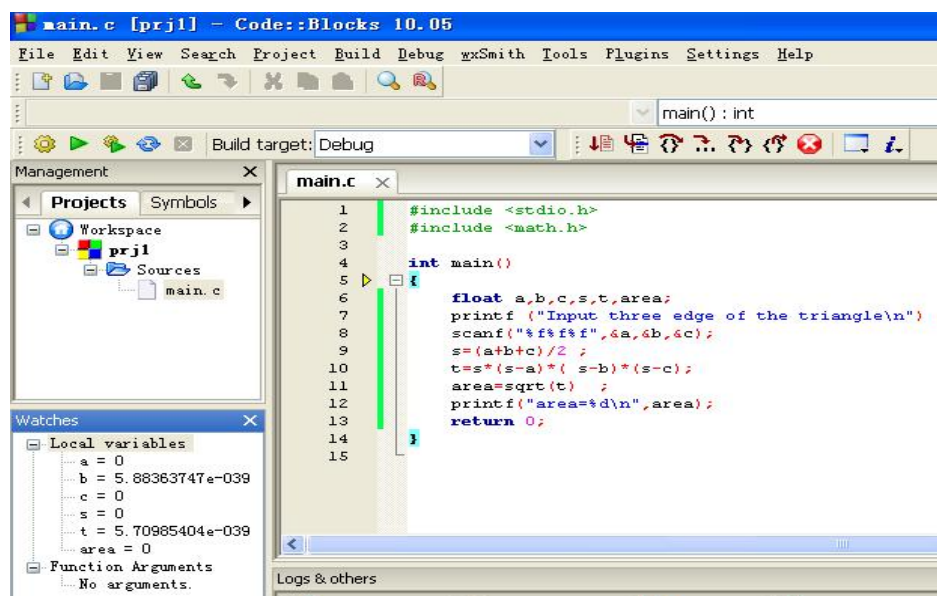








图 2.5 左下角是 Watches 窗口

##### (C) 单步执行

点击按钮(Next line)，继续运行到程序下一行前面，Watches窗口变量的值还是系统赋予的随机值。再点击2次按钮，输入3 4 5后回车，第9行有个黄色箭头(它指向即将执行的代码)，由于第8行语句已经被执行，从观察窗口可以看到局部变量a=3, b=4, c=5，此外日志窗口的调试器Debugger栏目也出现了一些文字，显示了main.c被执行的代码。



再点击，第9行语句被执行，黄色箭头停在第10行，观察窗口可以看到局部变量s=6(正确)；再点击，第10行语句被执行，黄色箭头停在第11行，观察窗口可以看到局部变量t=36(正确)；再点击，第11行语句被执行，黄色箭头停在第12行，观察窗口可以看到局部变量area=6(正确)。再点击按钮，第12行语句被执行，黄色箭头停在第13行，




程序输出屏幕上显示area=0。

**分析：**从观察窗口可以看到程序中aera的计算是正确的，但输出结果却显示aera=0，问题就在printf函数语句内，而且肯定是输出格式使用不恰当，导致未能正确输出一个数据。经仔细检查，发现用了输出整数的格式符%d，而area为float型，这就查出了出错的原因，应将%d改为%f。

#### (D) 终止调试器

点击  按钮 (Stop debugger)，终止调试，或者点击  按钮 (Debug/Continue)，继续运行到程序结束。

退出调试后，修改源程序，重新编译，然后运行，或者点击  按钮 (Build and run) 完成编译运行。

**观察：**运行过程中输入 1 2 6，观察输出结果。

屏幕上显示：area=-1.#IND00。

输出结果不是数字，表示计算中出现了错误情况，比如 0.0 除以 0.0 或者求负数的平方根等。下面用设置断点的方法找出原因。

#### 5) 调试程序——设置断点

对于一个较长的程序，常用的调试方法是在程序中设若干个断点，程序执行到断点时暂停，如果未发现错误，继续执行到下一个断点。这种方法可以将找错的范围从整个程序缩小到一个分区，然后集中精力检查有问题的分区。再在该分区内设若干个断点或单步跟踪，直到找到出错点。

##### (A) 设置断点

先将光标移动到需要设置断点的代码行上，再用Debug/Toggle breakpoint (F5) 菜单命令设置断点。在一个断点前再次Toggle breakpoint，则该段点就被删除，如果想删除所有断点，可以用Debug/ Remove all breakpoints菜单命令。

将光标置于第9行（即s=...行），按快捷键F5，这时，在这一行的左边会出现一个红色圆点，表示在该位置设立了一个断点，如图2.6所示。如要去掉此断点，只需在断点前再次按下F5。

将光标置于第11行（即srea=...行），按F5，在这一行也设立了一个断点。

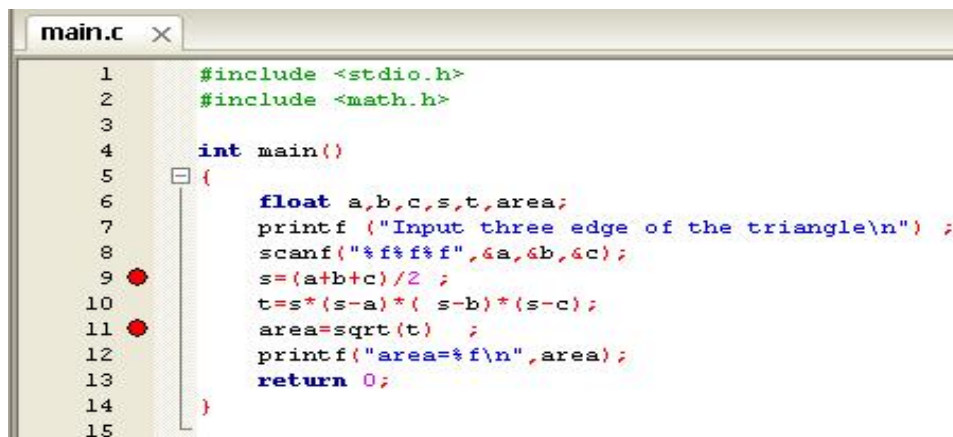



图2.6 设置断点


##### (B) 启动调试器

在设置断点后，可以通过另一种方法来启动调试器，就是选择Debug/ Start (F8) 菜单命令；或者点击  按钮；或者直接按F8。


程序将正常执行，运行到第一个断点时暂停程序。第一个断点前如果有输入语句，则需要输入数据，现在输入：1 2 6 后回车，程序暂停在第9行，并用一个黄色箭头来标记执行

到的代码行，该行并没有被执行，它是下一步要开始执行的行。


### (C) 打开观察窗口

点击  按钮，选择Watches打开观察窗口，拖动Watches窗口到左下角，并展开各个变量。此时，从观察窗口可以看到局部变量 $a = 1$ ,  $b = 2$ ,  $c = 6$ ,


### (D) 继续执行到下一断点

点击  按钮，程序继续运行到下一断点时暂停，从观察窗口可以看到局部变量 $s = 4.5$ ,  $t = -59.0625$ 。如果还有断点，可能需要反复这个过程，直至发现问题。

**分析：**跟踪到这已发现问题，可以看到  $t$  的值为  $-59.0625$ ， $t$  为负值在计算平方根时肯定会出现错误。这个错误是由于输入  $a$ ,  $b$ ,  $c$  的值不恰当而造成的，程序中应判断：输入的三角形三边的值是否满足三角形的性质：两边之和大于第三边，若满足，才计算面积。

发现问题后，不需要再调试了，点击 ，终止调试，修改源程序，重新编译运行。

### 6) 编译Release目标文件

当程序编译调试完毕，应该交付release目标文件。选择编译目标文件(Build target)为Release版本，点击 ，重新编译，可得到release目标文件和可执行文件。

**查看物理文件夹：**打开 Windows 资源管理器，选择 D:\prj1 目录，查看这两个文件，并比较 debug 和 release 版本目标文件的大小。

## 2. 源程序 2 实验步骤及要求

### 1) 创建一个工程

工程取名为prj2，该工程中有1个c文件main.c。建立的这个工程可以和前面建立的工程prj1共用一个工作空间Workspace，编译或者调试这个工程前需要激活它(让系统知道是对它操作，而不是对prj1工程操作)，激活的方法很简单，选择工程名prj2，在按下鼠标右键所弹出的菜单中选择Activate project，则该工程此时就处于激活状态，原来激活状态的prj1工程进入休眠。

建立prj2工程前，也可以选择File/close project菜单命令关闭当前工程，再建立工程。

### 2) 编辑源文件和保存

用源程序 2 替换 prj2 工程中 main.c 的源代码，编辑完毕后保存。


### 3) 编译和运行程序

纠正所有的error、warning，然后运行之。运行时输入“1 2 3 4 5”后回车，此时会发现输出结果为“The sum is 5”，显然结果不对。

下面组合应用断点、单步跟踪等调试方法对程序进行调试，找出错误。

### 4) 调试程序

#### (A) 启动调试器

在第8行(即第1个printf行)设置一个断点，点击  运行之，运行到第8行时程序暂停。然后打开Watches窗口进行跟踪，见图2.7。从图2.7的Watches窗口中可以看到局部变量都是随机值。



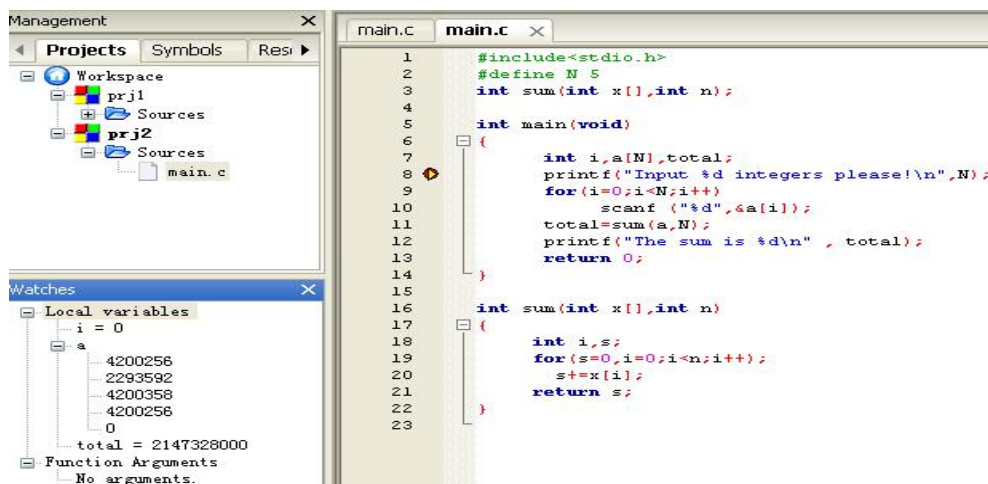





图2.7 调试prj2工程





### (B) 执行到第11行

把光标放到第11行代码前面，点击  (Run to cursor)，继续运行到第11行时暂停。由于期间有输入语句，现在输入：1 2 3 4 5后回车，从观察窗口可以看到i=5, a={1, 2, 3, 4, 5} (观察窗口红色文字部分)，total还是随机值，因为第11行语句还未执行。

### (C) 跟踪进sum函数

由于第11行语句中要调用sum函数，现在点击  按钮 (Step into)，则进入sum函数体，见图2.8。从图2.8可以看出，由于执行到函数sum的第19行 (黄色箭头标记)，函数参数x和n已经被赋值，但是i,s两个局部变量都是系统赋予的随机值 (因为第19行代码尚未执行)。

再点击  Step into，或者点击 ，则在函数体内单步跟踪，继续执行到下一行 (第20行代码)，从观察窗口可以看到i=5, s=0。

 和  的区别在于：在跟踪时，如果遇到自定义函数调用， 转到该函数的源程序中继续跟踪，而  则不转入被调用函数，直接执行下一行。

**分析与改错：**根据watch窗口看到的数据，得知for语句已经执行完，循环了5次 (因为i变成了5)，而语句s+=x[i]还未执行，因为其值是初值0，表明该赋值语句未进入循环，程序的流程有问题。此时再仔细分析源程序，发现问题出在了for后多了分号，变成了两条语句。

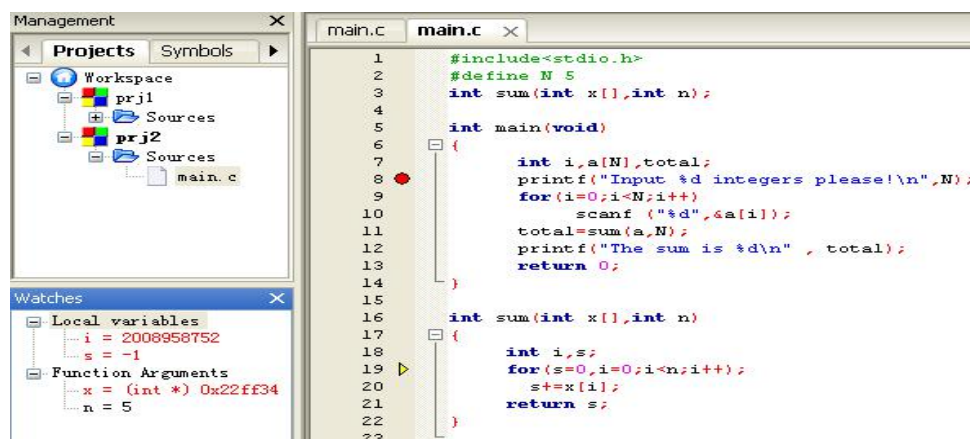




图2.8 跟踪进sum函数

### (D) 跳出sum函数

如果希望跳出sum函数，可以点击按钮（Step out），则跳出sum函数体。从日志Debugger栏目的信息可以看出，sum函数已经执行完毕，重新进入main.c函数体。

#### **(E) 继续运行到程序结束**

点击按钮继续运行，由于其后无断点，则程序运行完毕。

修改源程序，即将for后的分号去掉，重新编译运行，直到结果正确。

在编写较长的程序时，能够一次成功而不含任何错误是不容易的，这需要进行长期大量的练习。编写的程序若已没有编译错误，可以成功运行，但执行结果不对时，需要灵活的借助调试工具对程序进行跟踪调试，分析并查找出错原因。