

## 第 1 章 开发环境搭建及调试窗口设置

(视频讲解: 2 小时)

在学习 C 语言之前, 我们需要了解 C 语言的发展历史, 弄清 C 语言运行在哪些主流的操作系统之上, 知道什么是跨平台语言, 什么是可移植性强语言, 目前在 Windows 下大家使用的主要开发环境是 Visual Studio (后面我们简称 VS), 因此我们采用 VS 作为我们的开发工具, 对 C 语言进行讲解, 学完本章, 你将掌握以下内容:

- 开发环境的搭建与使用
- 新建项目及编译运行
- 基本的调试方法
- 学习 C 语言需要达到的目标和境界

### 1.1 开发环境搭建

#### 1.1.1 C 语言的那些事

在搭建环境之前讲一个故事, C 语言为什么叫 C 语言呢? 其实是因为先有高级语言 ALGOL 60, 简称 A 语言, 后来经过简化, 变为 BCPL 语言, 简称 B 语言, 而 C 语言是在 B 语言的基础之上发展而来的, 所以就称为 C 语言。所以世界上第一个 C 语言的编译器是用 B 语言编写的。目前主流的编译器是微软的 **masm**(Microsoft Visual Studio 使用的), 还有就是 Linux 使用的 **gcc** 编译器(Mac 电脑也使用该编译器), 学习一堆 C 的版本及标准意义不大, 关键是掌握编译器支持的语言特性, 接下来的八章我们所讲的 C 语言内容无论是在 Windows, 还是 Linux 及 Mac, 都可以编译通过。

##### UNIX

1965 年之前的时候, 电脑并不像现在一样普遍, 它可不是一般人能碰的起的, 除非是军事或者学院的研究机构, 而且当时大型主机至多能提供 30 台终端 (30 个键盘、显示器), 连接一台电脑, 如图 1.1.1-1 所示

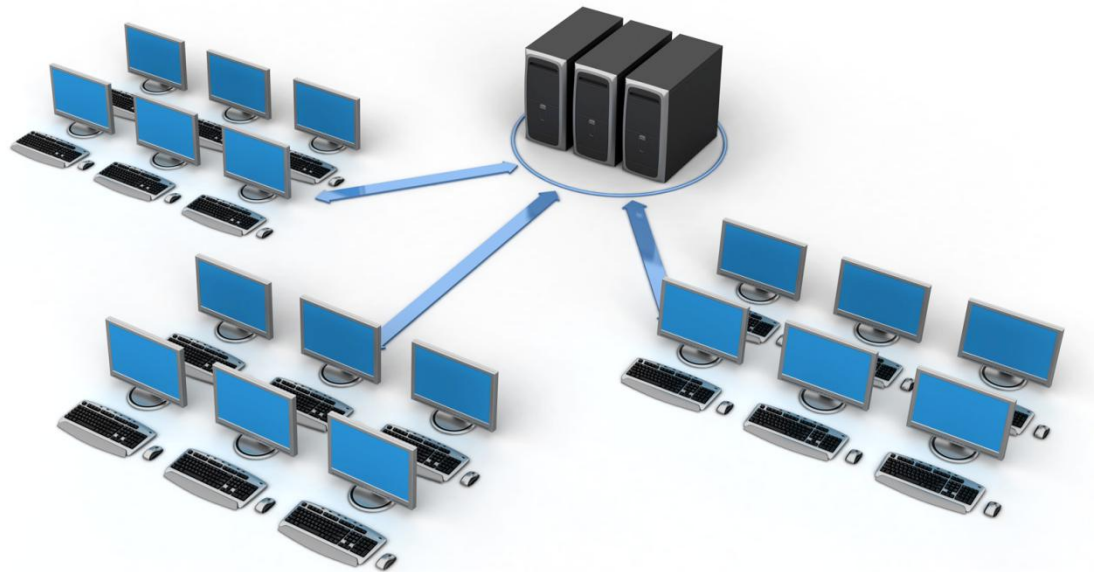


图 1.1.1-1

为了解决数量不够用的问题

- 1965 年后由 贝尔实验室 加入了 麻省理工学院 以及 通用电气 合作的计划 —— 该计划要建立一套 多使用者(multi-user)、多任务(multi-processor)、多层次(multi-level) 的 **MULTICS** 操作系统，想让大型主机支持 300 台终端
- 1969 年前后这个项目进度缓慢，资金短缺，贝尔实验室退出了研究
- 1969 年从这个项目中退出的 **Ken Thompson** 当时在实验室无聊时，为了让一台空闲的电脑上能够运行 "星际旅行 (Space Travel)" 游行，在 8 月份左右趁着其妻子探亲的时间，用了 1 个月的时间，使用汇编写出了 Unix 操作系统的原型
- 1970 年，美国贝尔实验室的 **Ken Thompson**，以 **BCPL** 语言为基础，设计出很简单且很接近硬件的 **B 语言**（取 **BCPL** 的首字母），并且他用 **B 语言** 写了第一个 **UNIX** 操作系统
- 1971 年，同样酷爱 "星际旅行 (Space Travel)" 的 **Dennis M.Ritchie** 为了能早点儿玩上游戏，加入了 **Thompson** 的开发项目，合作开发 **UNIX**，他的主要工作是改造 **B 语言**，因为 **B 语言** 的跨平台性较差
- 1972 年，**Dennis M.Ritchie** 在 **B 语言** 的基础上最终设计出了一种新的语言，他取了 **BCPL** 的第二个字母作为这种语言的名字，这就是 **C 语言**
- 1973 年初，**C 语言** 的主体完成，**Thompson** 和 **Ritchie**（图 1.1.1-2）迫不及待地开始用它完全重写了现在大名鼎鼎的 **Unix** 操作系统

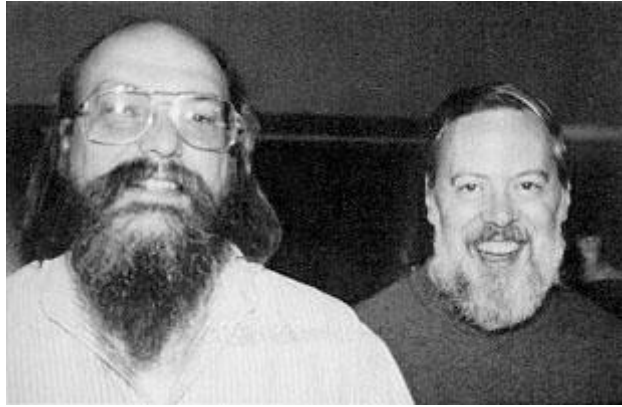


图 1.1.1-2 肯·汤普逊（左）和丹尼斯·里奇（右）

### C 语言

- 在把 **UNIX** 移植到其他类型的计算机上使用时，**C 语言**强大的移植性（Portability）在此显现
  - 机器语言和汇编语言都不具有移植性，为 **x86** 开发的程序，不可能在 **Alpha**，**SPARC** 和 **ARM** 等机器上运行
- 而 **C 语言**程序则可以使用在任意架构的处理器上，只要那种架构的处理器**具有对应的 C 语言编译器和库**，然后将 **C 源代码**编译、连接成目标二进制文件之后即可运行

有些小伙伴可能有疑问啦，那 **Java** 的跨平台是指怎么一回事呢，其实 **Java** 你编写任何代码，无序修改即可在任何一个平台上运行，所以称之为**跨平台**语言。而 **C 语言**我们称之为**可移植性强**，并不能跨平台，为什么呢？我们所讲的 **C 标准**，当然可以通过不同的编译器编译后在任何一台平台上运行，但是 **C 标准**除了文件操作之外，是没有涉及到操作系统硬件资源的接口的，比如进程调度，网络通信等，这些接口均为每一个操作系统独有的，**windows** 与 **Linux** 这些接口有差异，一旦你的 **C 程序**中使用了这些接口，代码放到另外一个平台就无法编译通过了，那是不是 **C** 不如 **Java** 呢，其实不是，**Windows** 操作系统本身用 **C** 和 **C++** 开发(少量汇编)，**Linux** 全部用 **C** 开发(少量汇编)，**C 语言**的执行效率一直是高级语言中的第一！另外 **Java** 及其他脚本语言中没有指针，无法访问物理地址，所以系统中的驱动都需要用 **C** 或 **C++** 进行编写。

## 1.1.2 开发环境安装

接下来的八章我们使用的开发环境为 **Microsoft Visual Studio 2012**(后面简称 **VS2012**)，也可以用 **VS** 的其他版本，**VS2013** 到 **VS2017** 都可以，你可以到网上下载安装包，当然也可以加入前言的 **QQ 群**，**QQ 群**里有安装包的分享链接，为什么使用 **VS2012** 讲解 **C 语言**呢，因为相对于 **Linux** 下的 **gcc**，**VS2012** 的图形化的调试界面非常清晰直观，对于初学者很友好。那为什么不用 **VS** 的最新版本呢，因为最新版本添加了我们学习 **C 语言**不相关的其他功能，我们学习 **C 语言**的目的能够到 **Linux** 下进行系统开发，所以不使用最新版本完全没有影响。针对 **windows 10** 的同学可以选择安装 **VS2017**，具体使用方法可以加入 **QQ 群（339198307）** 获取使用视频，接下来我们展示一下安装步骤

第一步：针对 **windows 7** 无法直接打开 **ISO 镜像**的同学，先安装一下 **DTLite** 虚拟光驱（当然电脑已经安装其他虚拟光驱也可以使用对应的虚拟光驱），**DTLite** 安装包可以到 **QQ 群**下载，

或者自己到百度自行下载，如图 1.1.2-1 所示，选择免费许可即可。



图 1.1.2-1

第二步，安装完成以后，如 1.1.2-2 所示，点击红框的加号，添加 Microsoft.VisualStudio.2012.Ultimate.CHS 镜像。

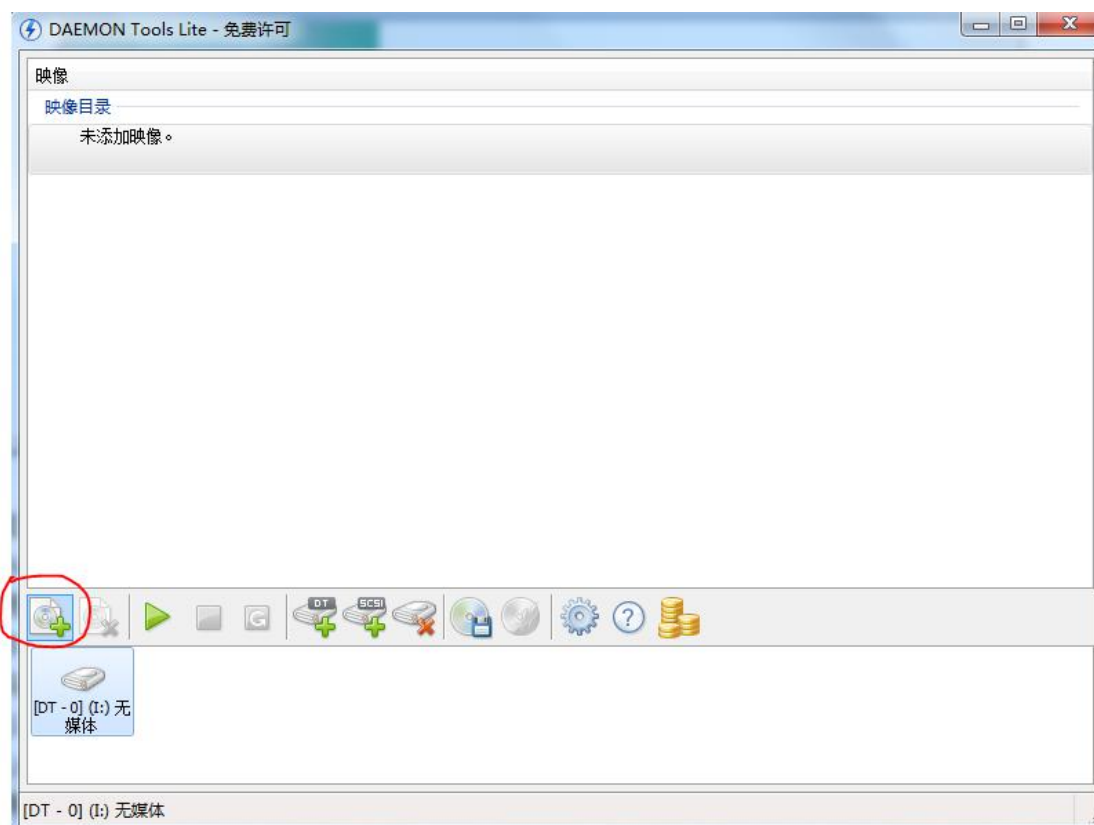


图 1.1.2-2

第三步，如图 1.1.2-3 所示，点击绿色三角形按钮载入镜像，载入后，双击箭头指向，即可打开 iso，如图 1.1.2-4 所示，双击 vs\_ultimate 即可开始安装。对于无法安装 DAEMON 虚拟光驱的同学，可以试一下其他的虚拟光驱，另外 windows 10 可以直接解压 iso 安装包进行安装，不安装虚拟光驱 DAEMON 也可以。

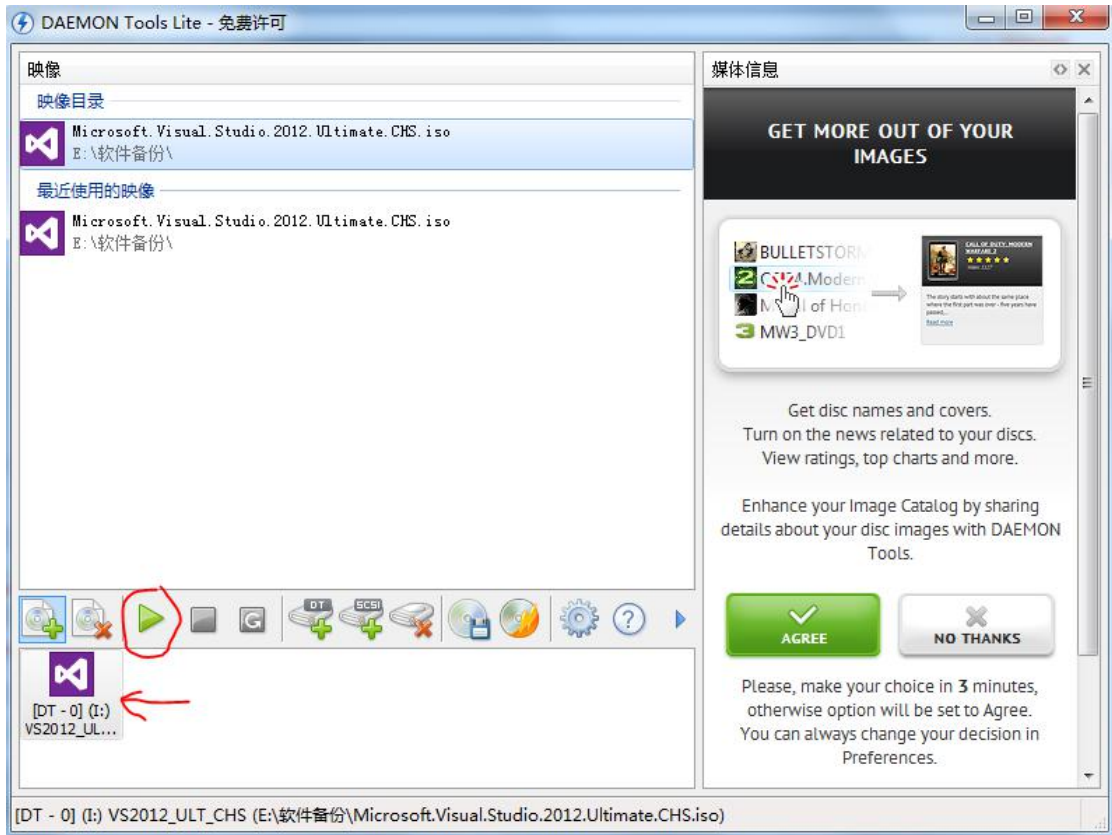


图 1.1.2-3

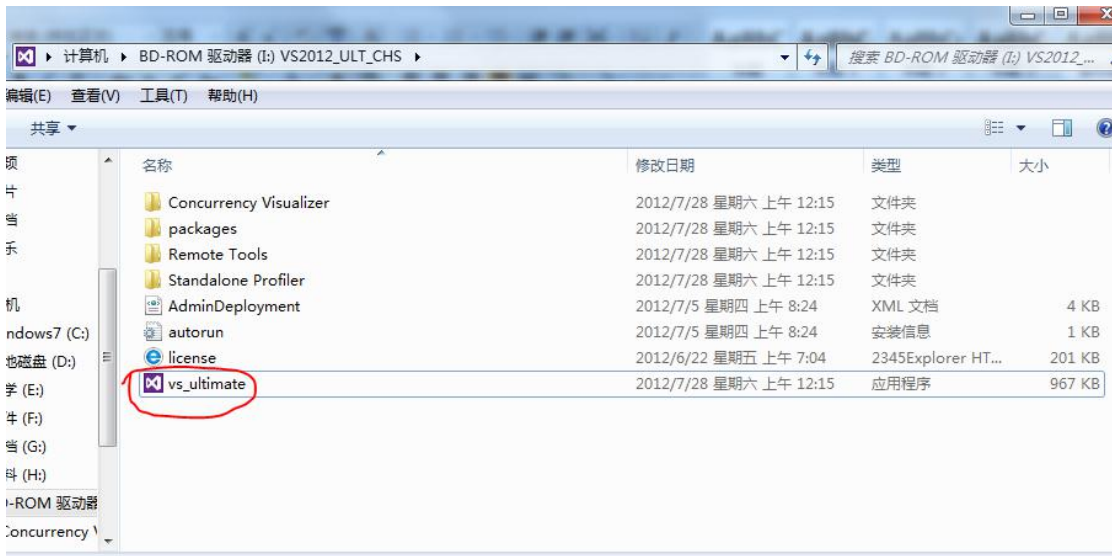


图 1.1.2-4

第四步，如图 1.1.2-5 所示，点击同意后，选择下一步，如果 C 盘空间小于 10G，建议选择其他盘，进行安装，如图 1.1.2-6 所示，全选，然后点击安装，安装就开始啦，建议在安装过程中，不要安装其他软件，避免安装失败（有分区是固态硬盘的同学，可以选择安装的固态硬盘分区，这样启动速度会非常快）。安装完毕后，如图 1.1.2-7，点击启动，如图

1.1.2-8, 提示输入产品密钥, 由于微软社区版对个人免费, 所以直接百度 Visual Studio Ultimate 2012 密钥, 搜到后复制输入即可, 激活成功, 显示如图 1.1.2-9, 然后点击关闭。点击关闭后, 显示如图 1.1.2-10, 选择 Visual C++开发设置, 然后点击启动即可。点击启动后会需要几分钟的时间配置, 耐心等待。



图 1.1.2-5



图 1.1.2-6





图 1.1.2-7





图 1.1.2-8



图 1.1.2-9

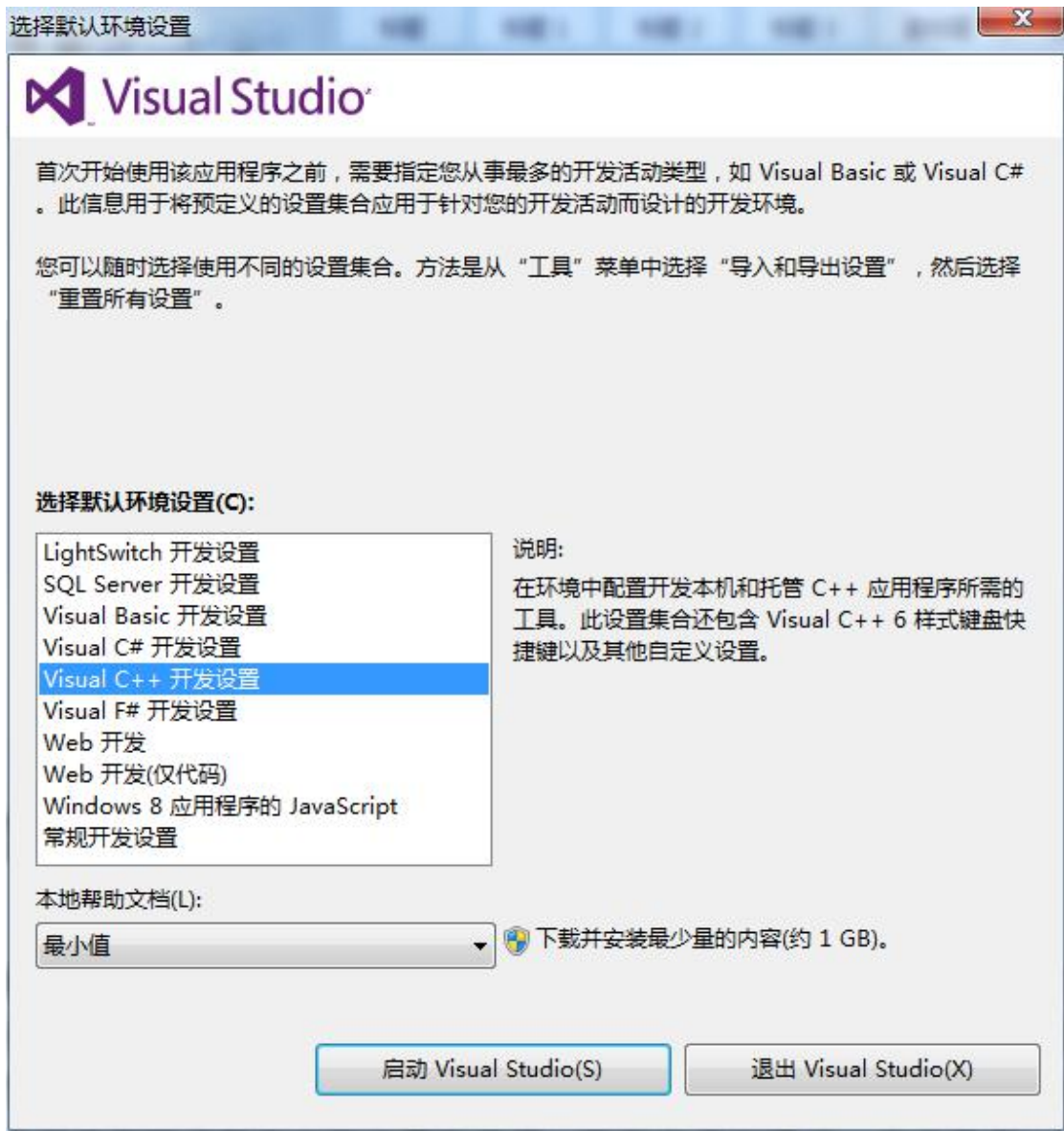


图 1.1.2-10

针对想使用较新的 Visual Studio 2017 的同学，安装过程中会有如图 1.1.2-11 的提示，如图 1.1.2-11 中所示，勾选第二项“使用 C++ 的桌面开发”和第三项“通用 Windows 平台开发”即可。

如果电脑有固态硬盘，而且空间足够大，建议将开发环境安装在固态硬盘上，这样可以提升开发环境的启动速度，其实对于任何集成开发环境都是如此（无论是 Java，Python 等语言的集成开发环境，安装在固态硬盘有助于提升启动速度）

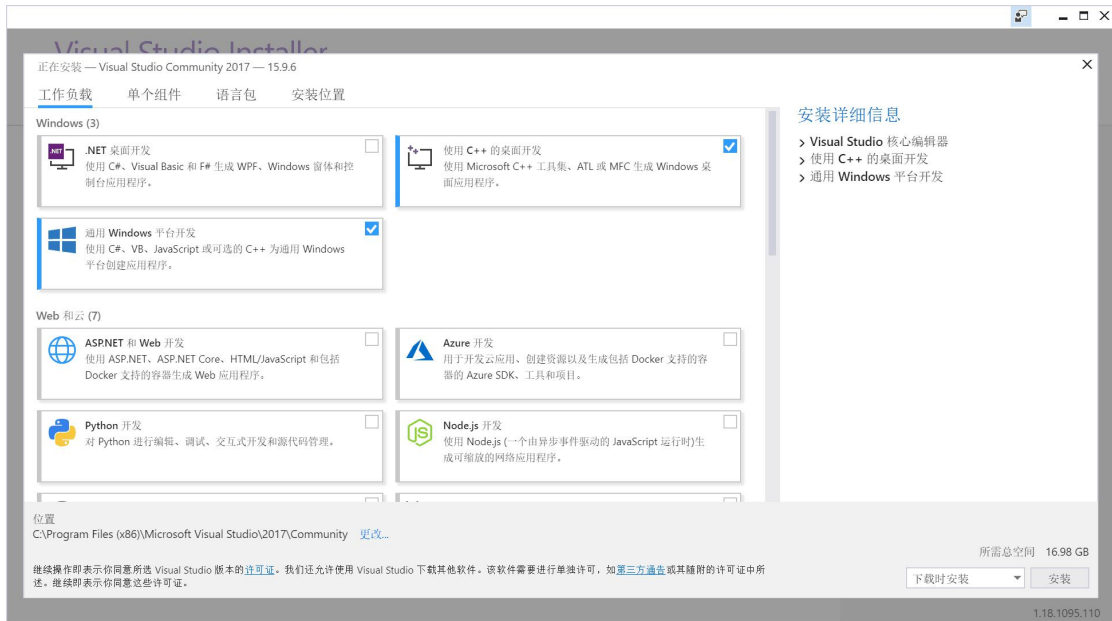


图 1.1.2-11

## 1.2 新建项目

环境安装完毕后，我们来看下如何新建项目，同时并**测试环境是否正常**，首先选择菜单里的文件->新建->项目，如图 1.2-1 所示，然后弹出如图 1.2-2 窗口，我们在右边栏选择 Visual C++，然后中间选择 win32 控制台应用程序(切记不要选择其他的，否则会导致编译不通)，首先点击浏览按钮选择将代码放置的目录，我在 D 盘创建了一个文件夹，BOOK，后面本书所有章节代码均会放置在该目录下，大家也可以创建一个目录，在学习过程中编写的代码都可以放在对应目录。接着填写解决方案名称，《开发环境搭建及调试窗口设置》，我们以每章的名字作为解决方案名字，然后再填写名称，名称即项目名称，我们填写 1.Hello\_world，这是我们的第一个项目，然后点击完成，显示如图 1.2-3，然后点击下一步，得到图 1.2-4，这时我们勾选空项目，即项目里没有任何文件及代码，同时勾掉安全开发生命周期（SDL）检查，因为这个检查是微软的安全规则，并非 C 标准的安全规则，同时不被业界所接受。得到图 1.2-5，然后点击完成。

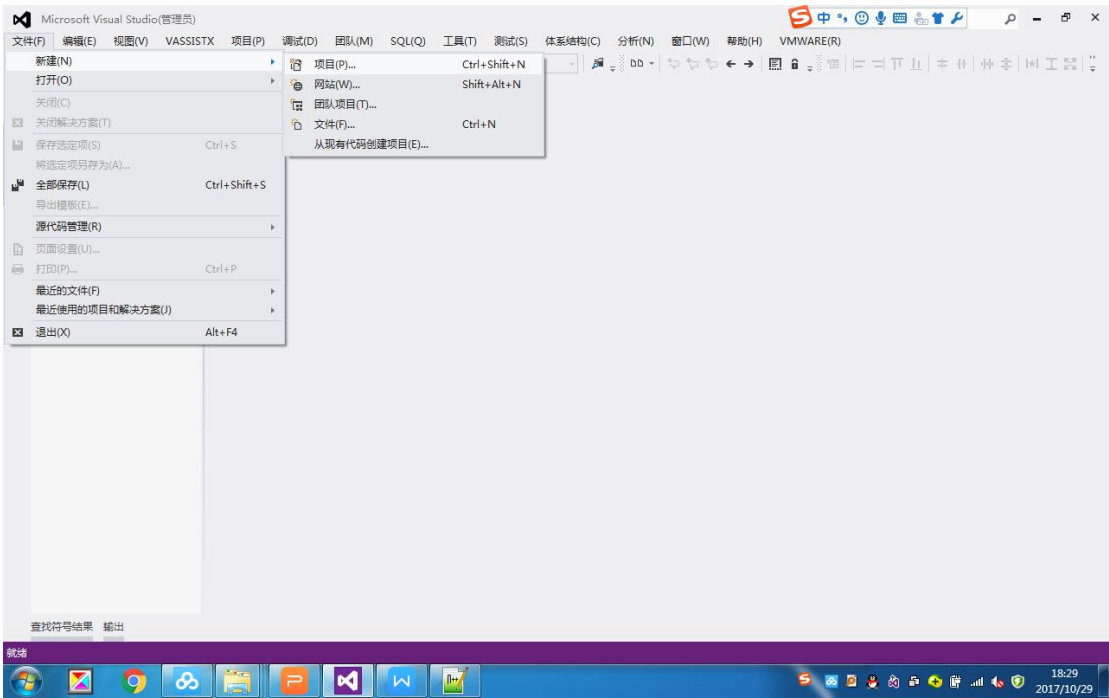


图 1.2-1 新建项目

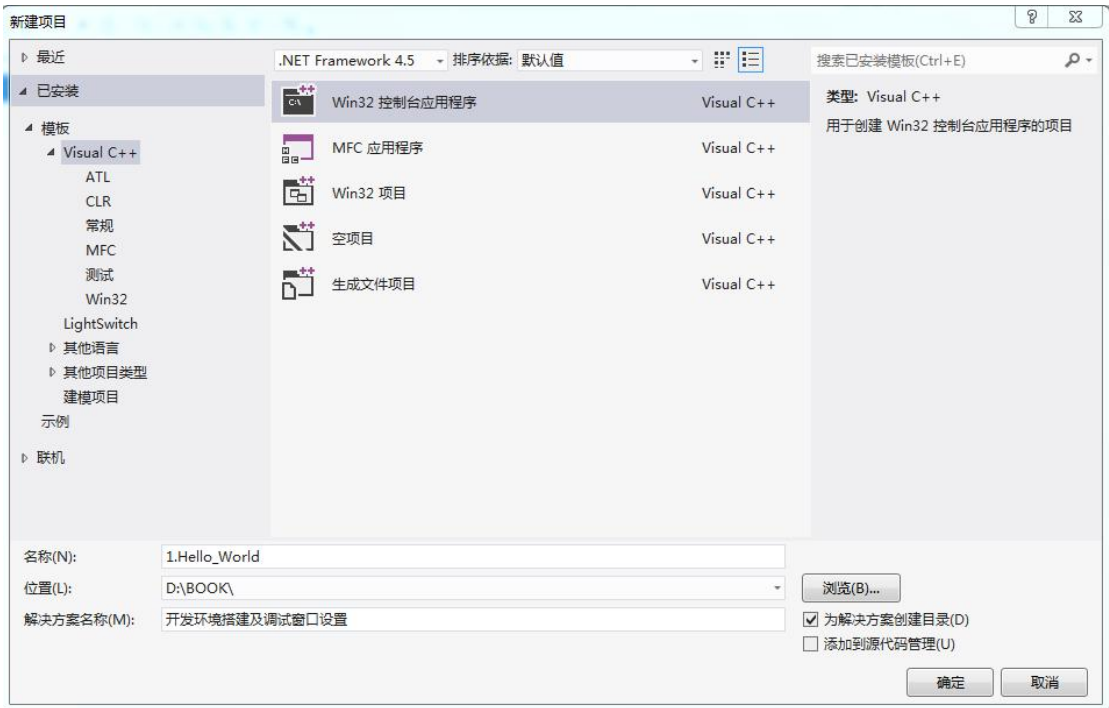


图 1.2-2 选择控制台应用程序



图 1.2-3



图 1.2-4



图 1.2-5

### 1.3 新建代码及编译运行

新建项目完成后，会得到如图 1.3-1 所示界面，右键点击源文件文件夹，选择添加，点击新建项，如图 1.3-2，在图 1.3-3 中，我们选择 C++ 文件，可能你会疑惑为什么选择 C++ 呢，因为微软的 `masm` 编译器是 C 与 C++ 混合的编译器，所以我们选择 C++ 文件，但是我们新建代码文件时，后缀命名为 `.c`，比如这次我们以 `main.c` 命名，那么 `masm` 编译时，就会按照 C 规则进行编译，然后点击添加按钮，得到窗口图 1.3-4



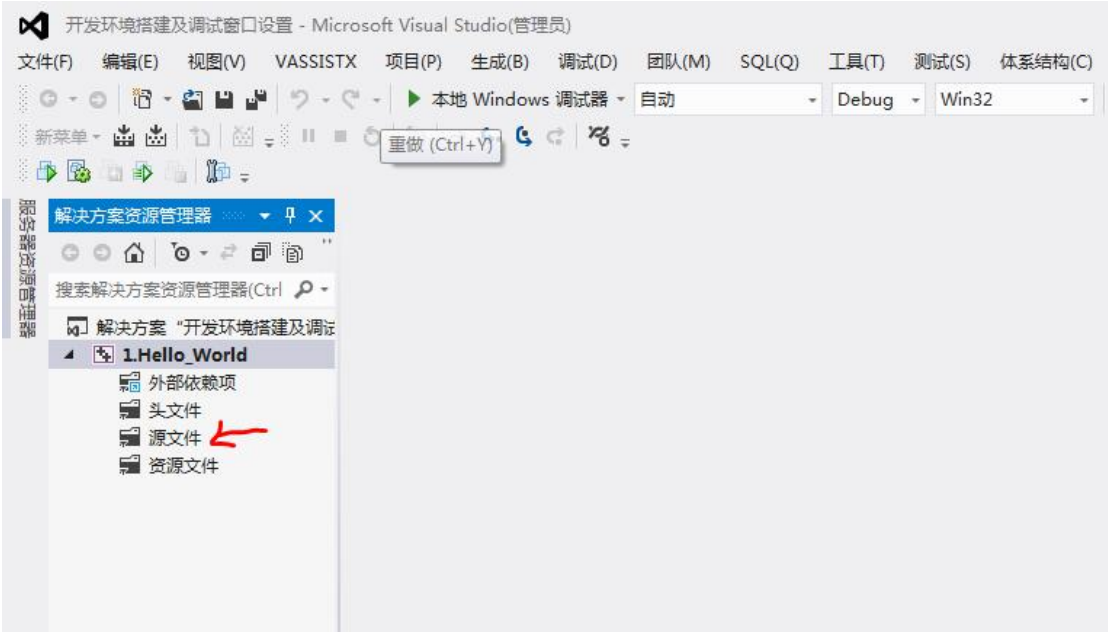


图 1.3-1

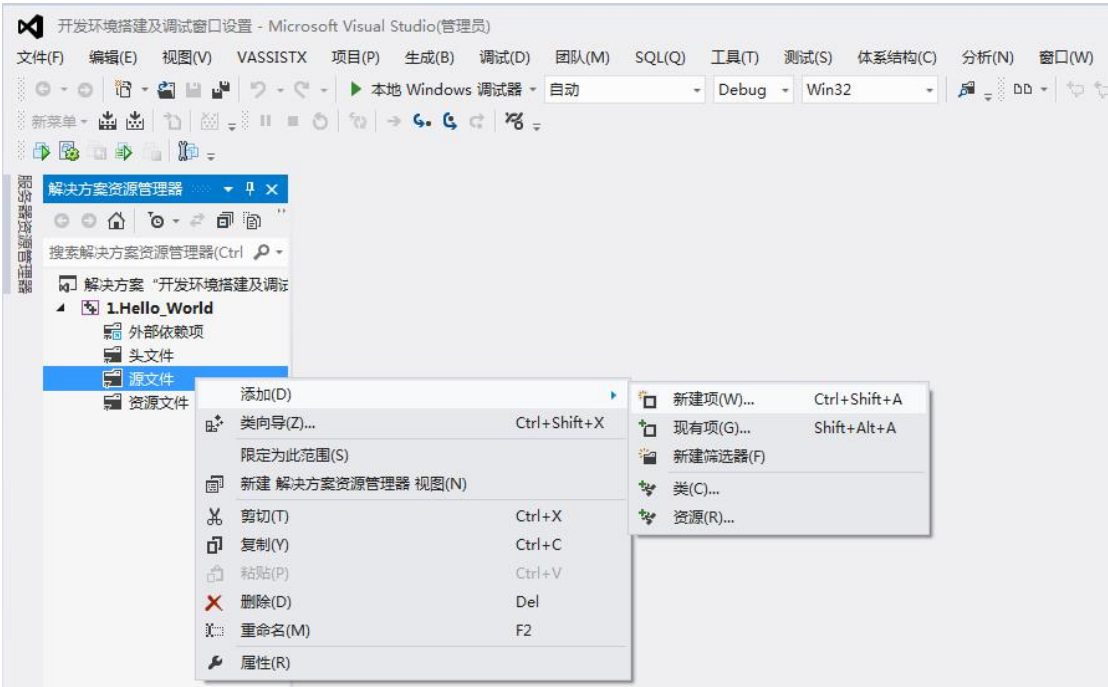


图 1.3-2

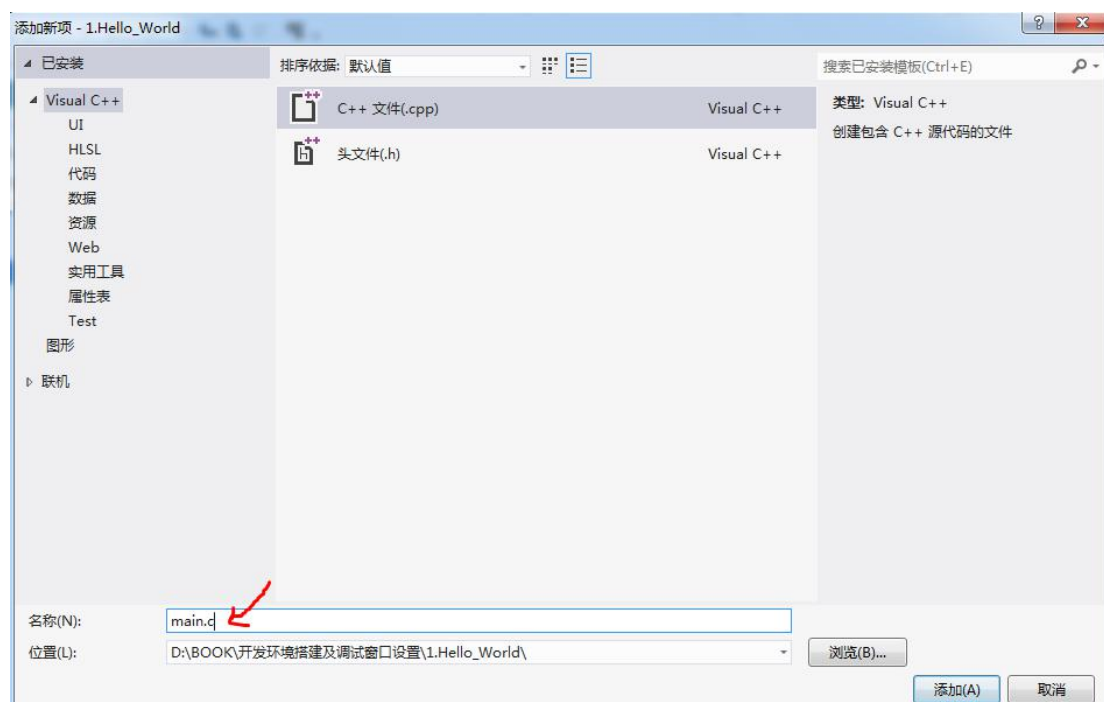


图 1.3-3

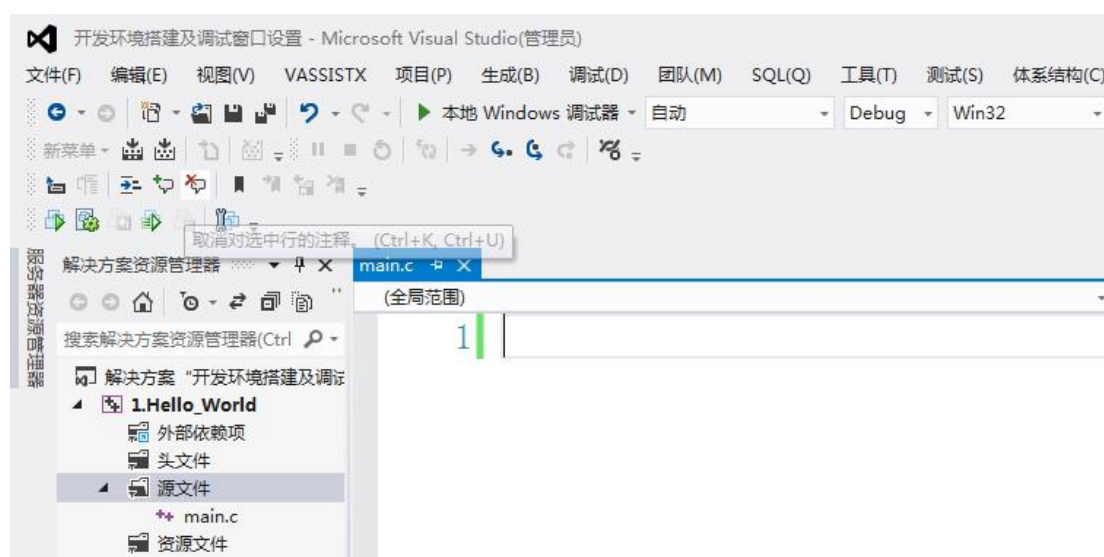


图 1.3-4

然后往 main.c 中添加代码，得到如图 1.3-5 的效果，然后点击图中绿色三角形按钮（后面我们简称执行按钮），该按钮的作用是编译并执行程序，得到如图 1.3-6 的控制台窗口，得到 Hello world 字符串输出

说明： main-主函数名， int-函数返回值类型

每个 C 程序必须且只能有一个主函数 main，程序从 main 函数开始执行

{ } 大括号是函数开始和结束的标志，不可省

每个 C 语句以分号结束

使用标准库函数时应在程序开头一行写：

#include <stdio.h> 函数 printf 需要使用该头文件

#include <stdlib.h> 函数 system 需要使用该头文件

//两个斜杠后面内容为代码注释，程序编译时并不会被编译到程序中

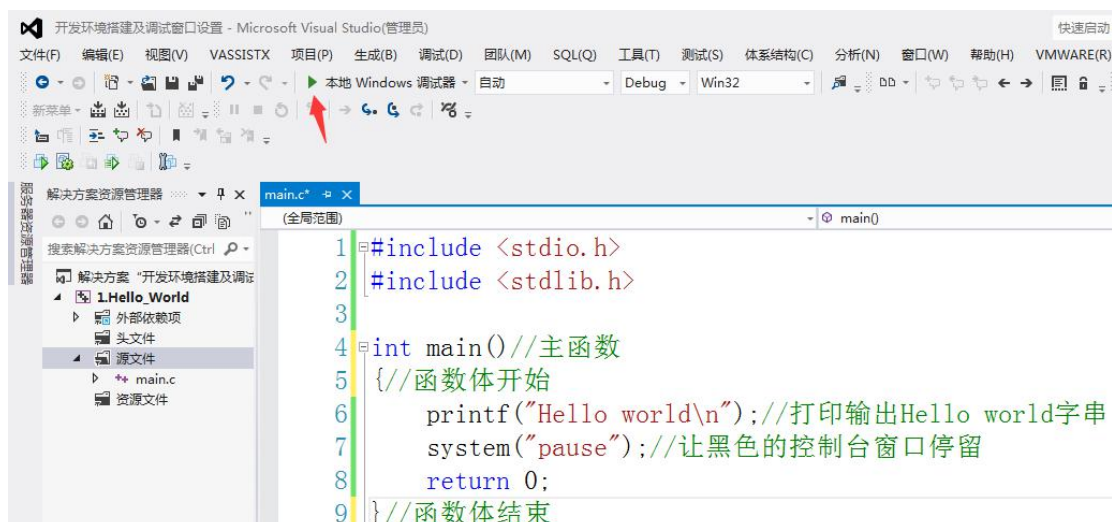


图 1.3-5

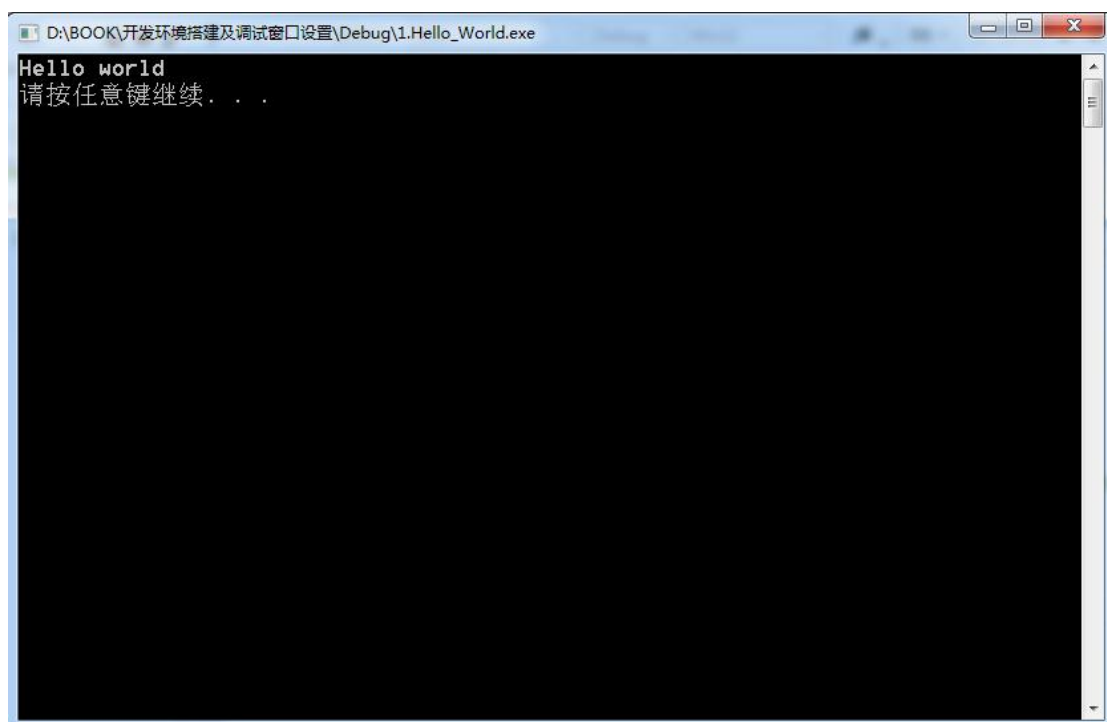


图 1.3-6

printf 函数是打印输出的作用，将双引号中的 Hello world 字符串打印到屏幕上，pause 实际是 windows 的一个批处理命令，通过 win+r 快捷方式打开运行窗口，输入 cmd，如图 1.3-7 所示，然后回车

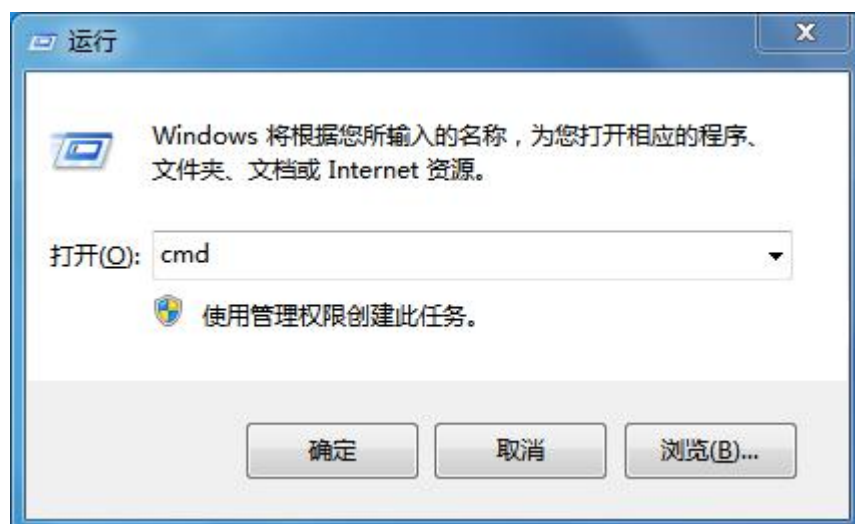


图 1.3-7

然后我们在图 1.3-8 所示的窗口中, 输入 `pause`, 就可以看到 请按任意键继续的打印, 然后我们再次按回车, 就会回到命令行, `pause` 命令的作用就是让 `cmd` 窗口 (我们又称为控制台窗口), 停留一下。我们的 `system` 函数, 是 C 语言调用其他语言的一个接口, `system("pause")`, 是我们通过 C 语言调用执行了 windows 的批处理命令 `pause`。

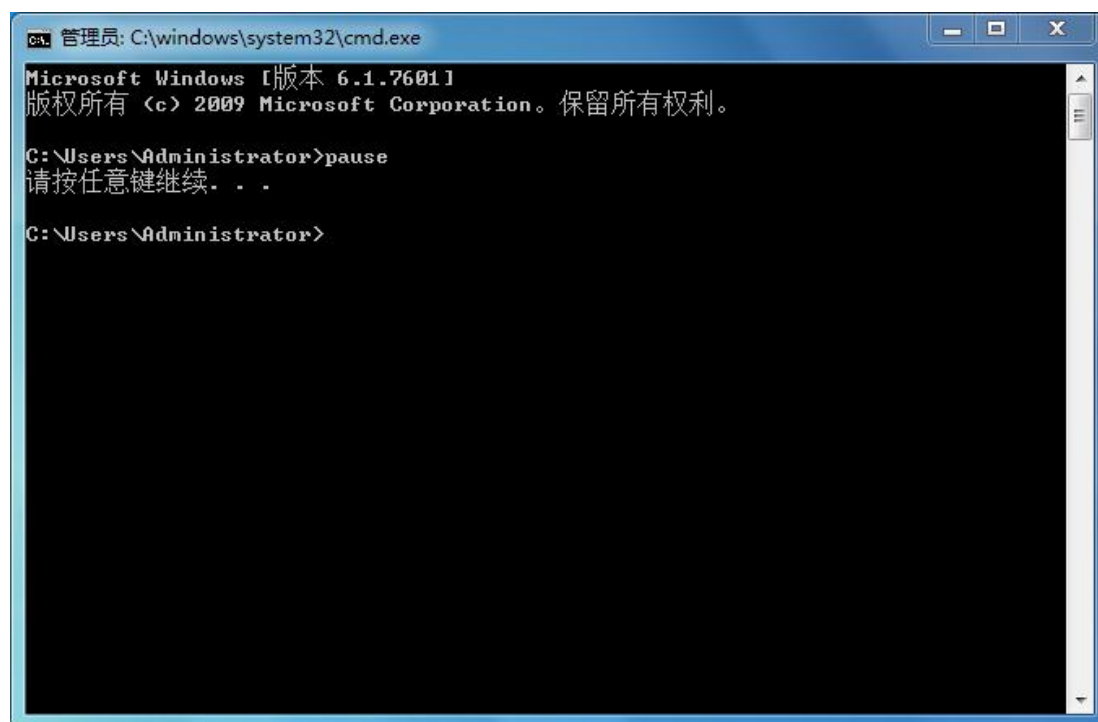


图 1.3-8

## 1.4 程序的编译过程

程序的编译过程如下图 1.4-1 所示, 首先我们编写源代码 `f.c` 文件, 编写完毕后, 通过编译器进行编译, 这里的编译包含预处理, 编译, 汇编, 详细过程会到 Linux 系统编程讲解,

当然大家有兴趣可以看编译原理，f.c 经过编译后，我们会得到 f.obj 文件，f.obj 文件中均为 0101 类型的机器码，也就是 cpu 能够识别的微指令（英特尔的机器指令）去运行，f.obj 文件并不能执行，因为我们调用的标准库函数的代码并不在 f.obj 文件中，例如上面 main.c 的 printf 函数，printf 函数的代码并不在 main.obj 中，这时经过链接，就得到可执行文件 f.exe，了解这个编译过程，对于我们后面章节编写程序遇到编译错误后，分析编译错误，我们可以区分清楚是**编译错误**，还是**链接错误**。

如图 1.4-2，右键点击我们的 Hello\_World 项目，然后选择在文件资源管理器中打开文件夹，得到图 1.4-3，我们可以看到一个解决方案，其实是一个大的文件夹，然后我们新建 Hello\_world 项目时，Visual studio 会在我们的解决方案目录《开发环境搭建及调试窗口设置》文件夹下，新建文件夹 1.Hello\_World，然后我们添加 main.c 时，就会在该目录下新建文本文件 main.c，在 Debug 文件夹中可以看到 main.obj 文件，在上级目录下的 Debug 文件夹下，可以看到 1.Hello\_World.exe 文件

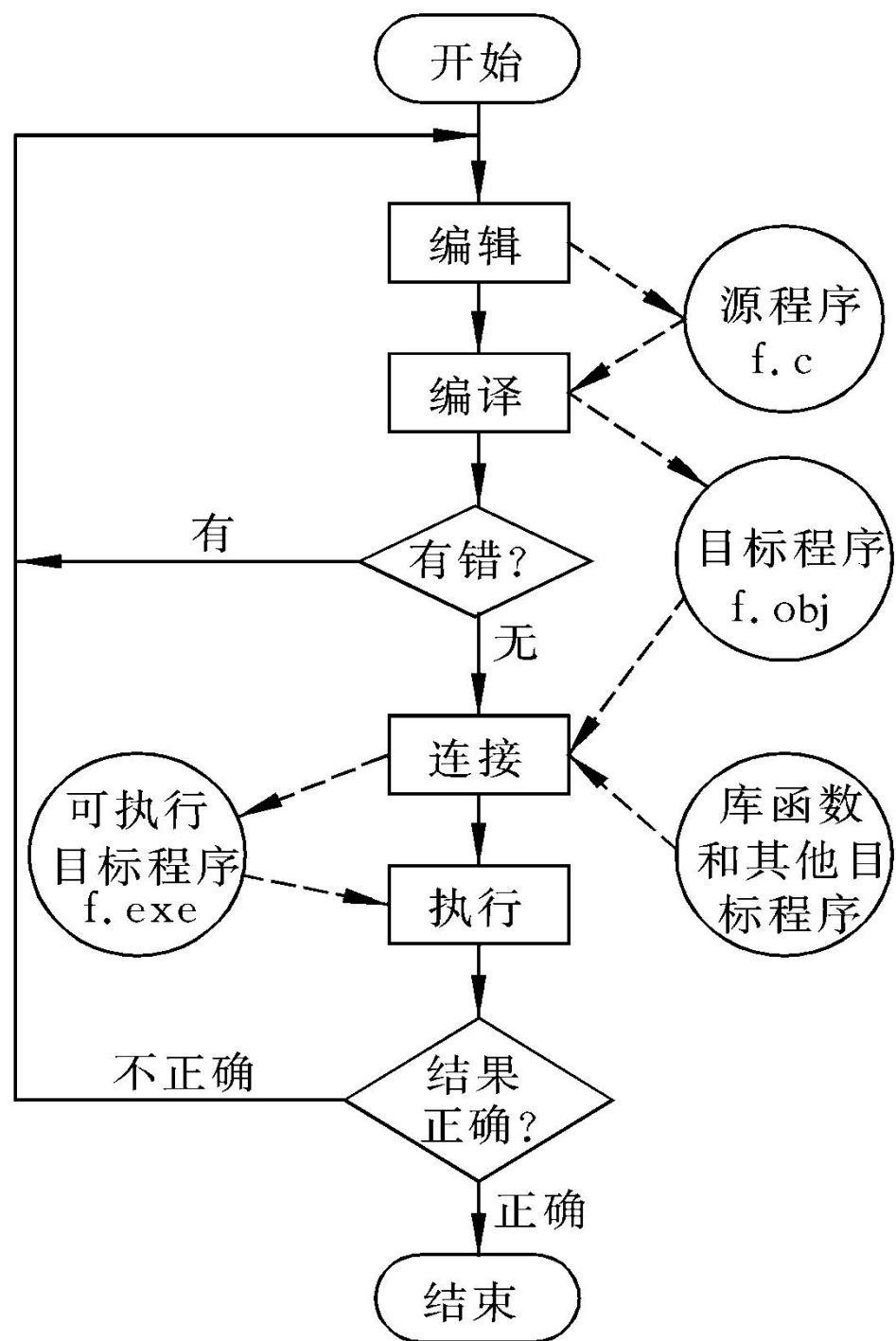


图 1.4-1 编译过程

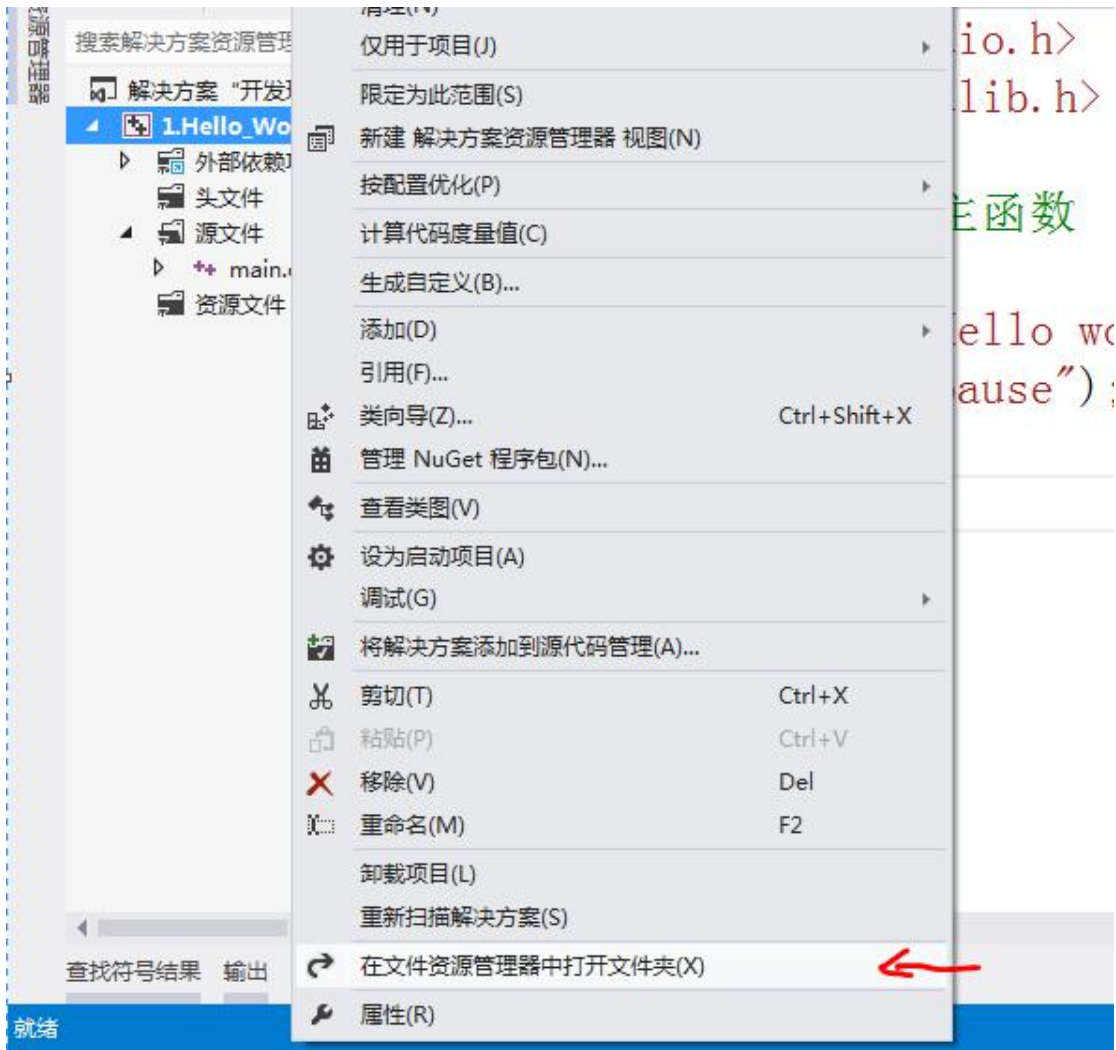


图 1.4-2



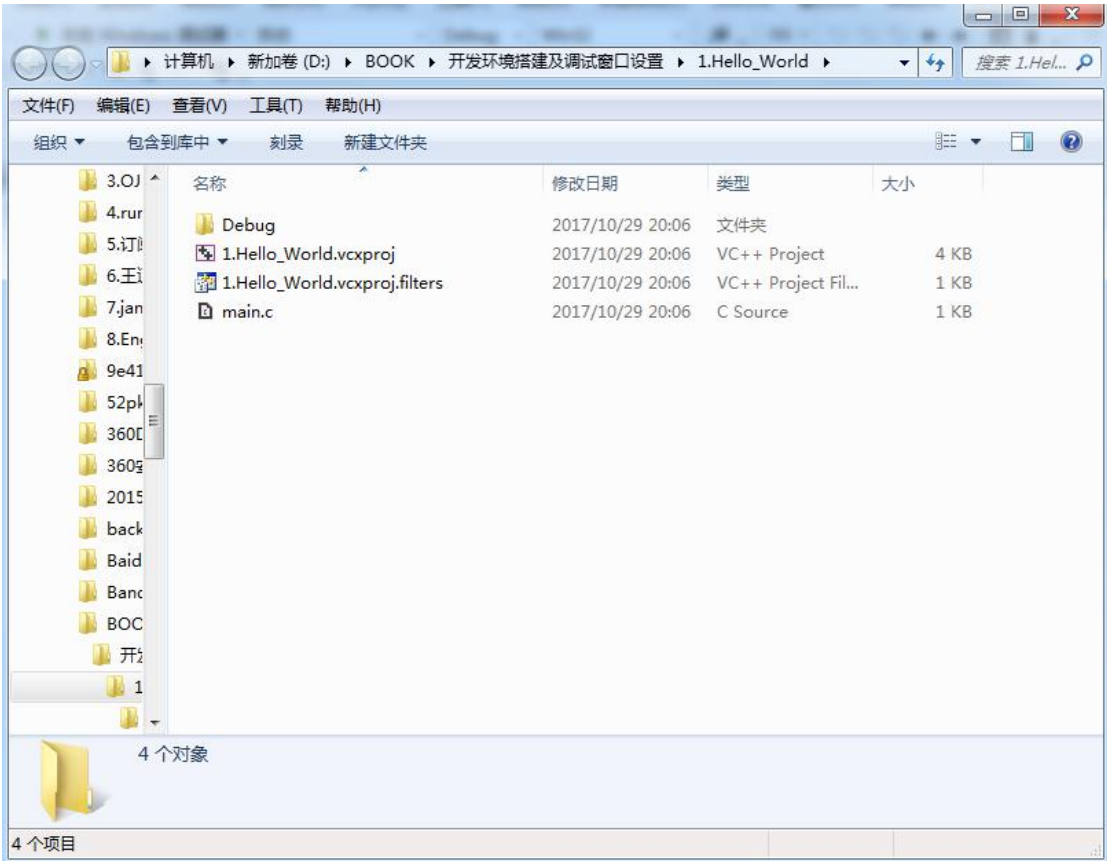


图 1.4-3

## 1.5 断点及调试窗口设置

首先如何把代码的行号显示出来，如图 1.5-1，选择菜单栏工具中的选项，然后显示如图 1.5-2 所示，点击文本编辑器，选择 C/C++，然后勾选行号，再点击确定，这时代码里的行号就显示出来了。

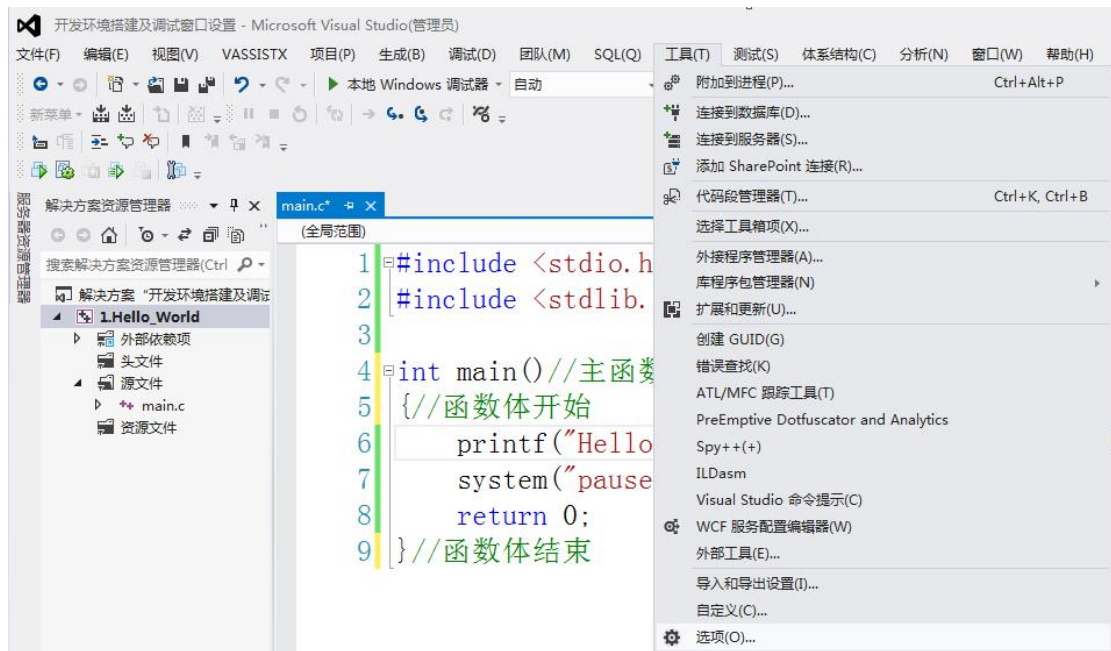


图 1.5-1

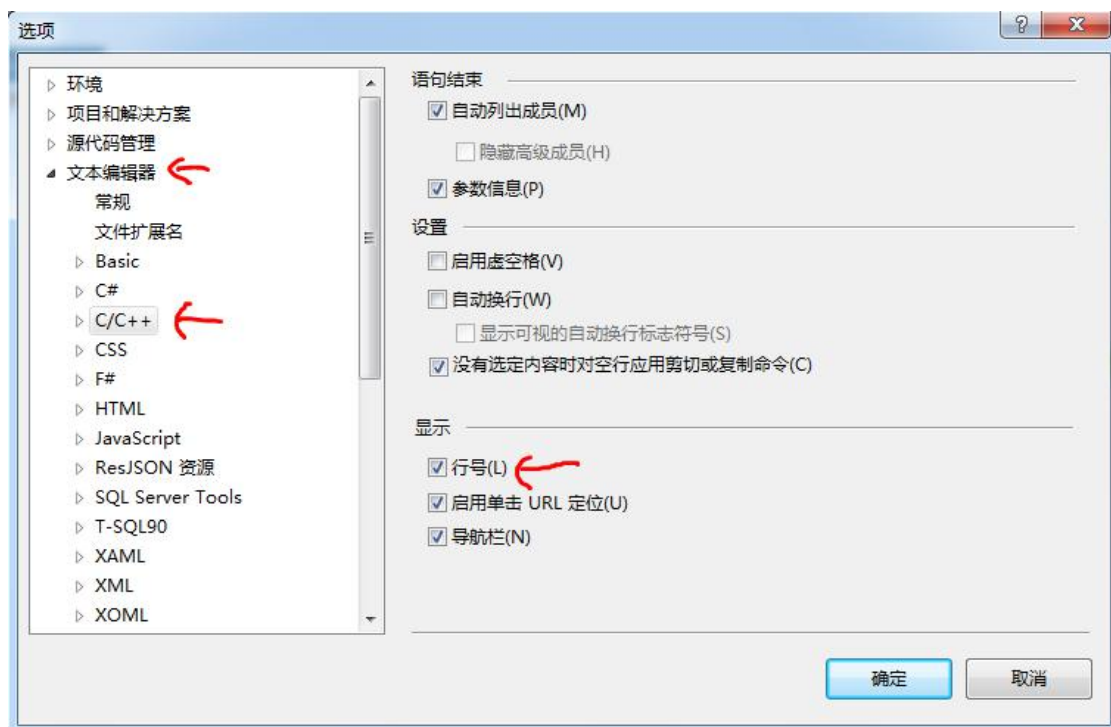


图 1.5-2

如图 1.5-3，我们在第 6 行的左边灰色区域左键点击，打上断点，然后我们点击执行按钮（绿色三角形，汉字注明的本地 windows 调试器即为执行按钮），启动运行后，显示效果如图 1.5-3，按钮 1 是单步执行按钮，快捷键是 F10，点击该按钮一次，程序会向下执行一步，按钮 2 是继续执行按钮，点击后，程序会执行到最后，或者执行到下一个断点，按钮 3 是停止执行按钮，点击后程序直接停止运行。（左键点击断点就可以取消断点）

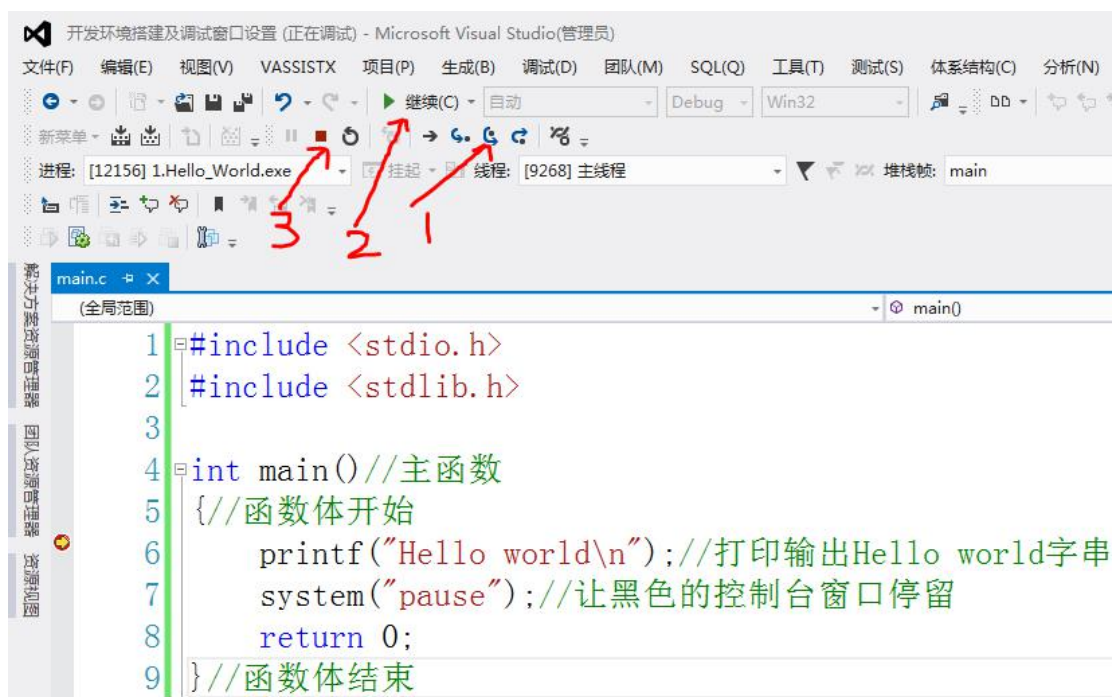


图 1.5-3

如图 1.5-4 所示，在断点调试状态下，我们点击菜单栏的**调试**，选择**窗口**，然后依次点击**监视**，**调用堆栈**，**内存**，三个调试窗口，让其显示效果如图 1.5-5，调用堆栈，监视，内存窗口对于后面我们调试程序及理解程序执行原理都至关重要，这次调出以后，后面每次调试程序都会自动弹出。

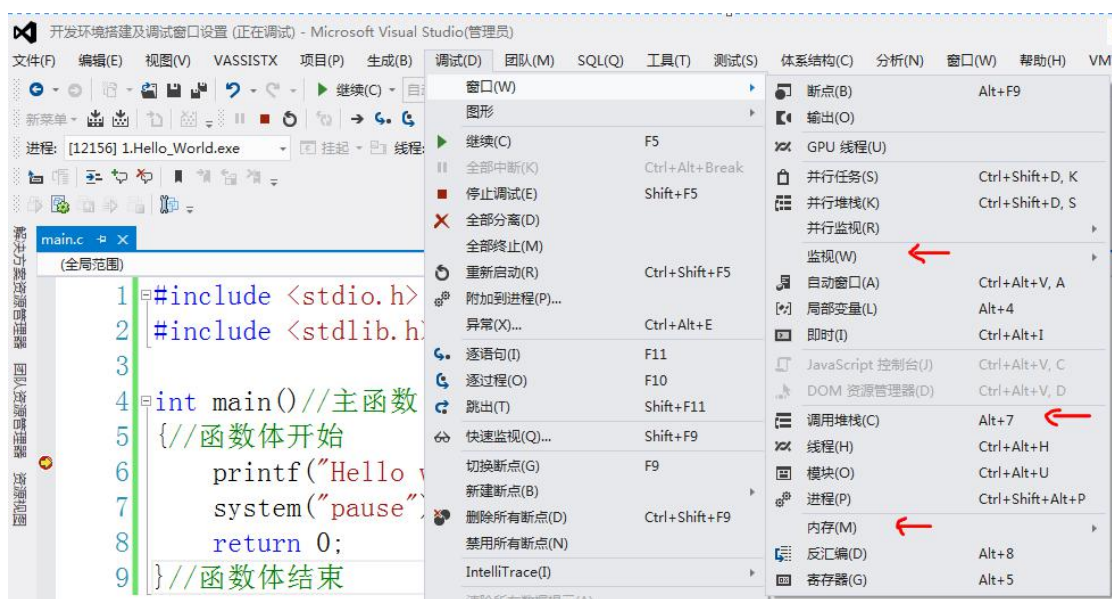


图 1.5-4

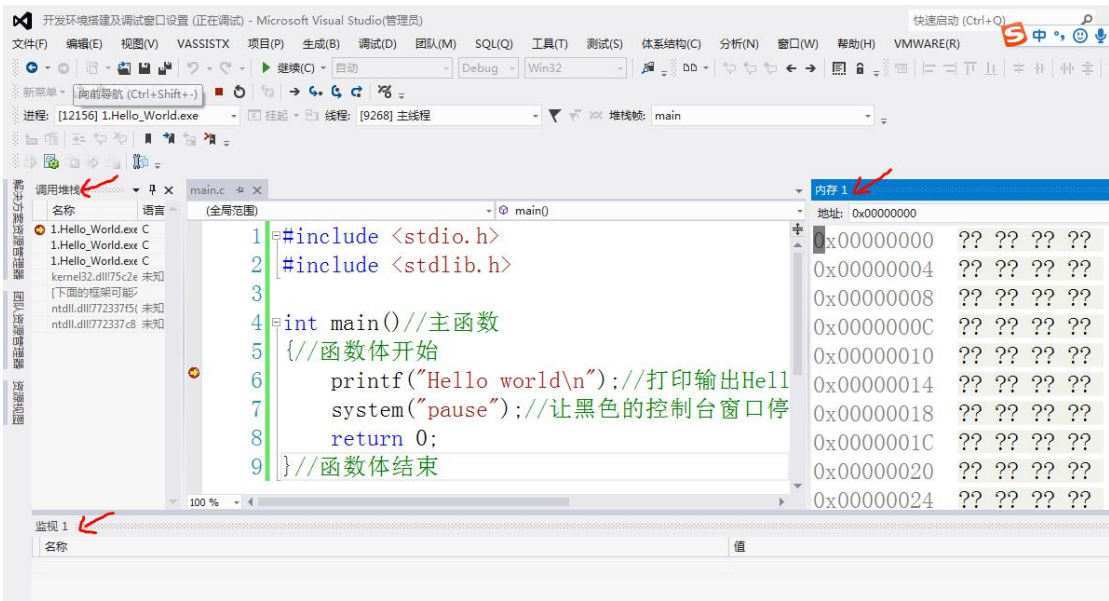


图 1.5-5

图 1.5-5 中，最左边放的调用堆栈窗口，下方放的监视窗口，右边放的内存窗口，大家可以按照我的布局进行放置，这样在后面的学习过程中会感觉非常自然。

## 1.6 学完 C 以后的境界



学完 C 语言对于计算机的理解要达到一种什么境界呢？龙哥认为学完 C 语言要达到的效果是你**理解了程序的执行过程**，对于理解程序的执行过程，C 语言是当之无愧的最佳选择，因为其他高级语言封装的太多，程序执行的过程是什么，其实**程序执行的过程**简单来说就是**内存的变化过程**，所以后面每个章节我们都会去关注内存的变化。如果你不是很理解，那么打个比方，内存的变化过程就像你的衣柜的变化过程，我们通过内存来存储我们的数据，通过衣柜来存储我们的衣服，数据存储需要有规律，便于我们高效取出，衣服存放也要有次序，否则你要花很久时间去找你想要的那件衣服。当然程序员的级别更高级一些，更像裁缝，你的衣柜里放了原始布料，相当于你的原始数据，裁缝通过剪刀，尺子等去加工布料，程序员用运算符，选择循环等去处理数据，裁缝把做好的衣服有规律放在衣柜，便于客户需要时给客户，程序员也需要把处理好的数据存放好，在用户需要时显示给用户！重点来了，准备好做裁缝了么

另外要掌握的能力就是**程序的调试能力**，灵活掌握单步调试，判断打印等调试手段，能够在清晰理解程序执行过程的基础上，准确分析数据的变化过程，从而定位程序的问题点，然后解决一切 bug，语言并不难，学完 C 语言，对于你以后学习其他任何语言，再调试对应程序的 bug，将会事半功倍！

### 练习题：

1、启动程序点击什么按钮？程序一步一步往下走点击什么按钮？终止程序点击什么按钮？

如何打断点，如何取消断点？

答案：按绿色小三角按钮或者快捷键 **F5** 启动程序，按 **F10** 或者  使得程序一步一步往下走，终止程序按钮是  或者按 **shift+F5**，左键点击侧边灰色区域打断点，再次左键点击断点来取消断点。