

第 1 章 初识 C 语言——编程探索之旅的崭新起点

在科技日新月异的当代社会,编程已成为连接现实与数字世界的神奇纽带,它不仅推动了科技的进步,更深刻地改变了我们的生活方式。而在众多编程语言中,C 语言如同一颗璀璨的明珠,以其简洁、高效、灵活的特性,赢得了无数编程爱好者的青睐,成为他们踏入编程世界的首选语言。

从本章开始,我们将正式踏上 C 语言的学习之旅,这不仅是一次对编程技能的探索,而且是一场关于逻辑思维、问题解决能力的全面提升。C 语言作为计算机科学领域的基础语言之一,自其诞生以来,就以其强大的功能和广泛的应用领域,在系统级编程、嵌入式开发、游戏开发等多个领域发挥着举足轻重的作用。它不仅是学习其他高级编程语言的桥梁,更是深入理解计算机底层工作原理的钥匙。这种对计算机底层原理的深刻认知,将使我们未来面对复杂系统时具备独特的调试和优化视野。让我们携手并进,共同踏上这段充满探索与发现的 C 语言学习之旅。

1.1 计算机语言

计算机语言是指用于人与计算机之间通信的语言,是人与计算机之间传递信息的媒介,因为它它是用来进行程序设计的,所以又称程序设计语言或者编程语言。

计算机语言在诞生的短短几十年里,经过了一个从低级到高级的演变过程。具体地说,它经历了机器语言、汇编语言、高级语言 3 个阶段。

1. 机器语言

机器语言是一种用二进制代码表示的、计算机能直接识别和执行的机器指令的集合,它是计算机的设计者通过计算机的硬件结构赋予计算机的操作功能。机器语言是第一代计算机语言。

计算机使用的是由 0 和 1 组成的二进制数,在计算机诞生之初,人们只能用计算机语言对计算机发出指令,即写出一串串由 0 和 1 组成的指令序列交由计算机执行,这种计算机认识的语言,就是机器语言。

一段表示两个整数相加的机器指令如下所示:

```
0001 1111 1110 1111  
0010 0100 0000 1111
```

```
0001 1111 1110 1111
0010 0100 0001 1111
0001 0000 0100 0000
0001 0001 0100 0001
0011 0010 0000 0001
0010 0100 0010 0010
0001 1111 0100 0010
0010 1111 1111 1111
0000 0000 0000 0000
```

用机器语言编写程序,编程人员要首先熟记所用计算机的全部指令代码和代码的含义。编写程序时,程序员需要自己处理每条指令以及每一数据的存储分配和输入/输出,还要记住编程过程中每步所使用的工作单元处在何种状态,这是一件十分烦琐的工作。而且,编出的程序全是二进制的指令代码,直观性差又容易出错,并且修改起来也比较困难。此外,不同型号的计算机的机器语言是不相通的,按一种计算机的机器指令编制的程序不能在另一种计算机上执行,所以,在一台计算机上执行的程序要想在另一台计算机上执行,必须另编程序,造成重复工作。但机器语言由于可以被计算机直接识别而不需要进行任何翻译,其运算效率是所有语言中最高的。

2. 汇编语言

为了克服机器语言难读、难编、难记和易出错的缺点,人们就用与代码指令实际含义相近的英文缩写词、字母和数字等符号来取代指令代码(如用 ADD 表示运算符“+”的机器代码),于是就产生了汇编语言。所以说,汇编语言是一种用助记符表示的仍然面向机器的计算机语言。汇编语言亦称符号语言。汇编语言由于采用了助记符来编写程序,比用机器语言的二进制代码编程要方便些,在一定程度上简化了编程过程。汇编语言的特点是用符号代替了机器指令代码。而且助记符与指令代码一一对应,基本保留了机器语言的灵活性。使用汇编语言能面向机器并较好地发挥机器的特性,得到质量较高的程序。

使用汇编语言编写实现两个整数相加的程序,具体代码与说明如表 1.1 所示。

表 1.1 汇编具体代码与说明

代 码	说 明
LOAD RF Keyboard	从键盘获取数据,存到寄存器 F 中
STORE Number1 RF	把寄存器 F 中的数据存入 Number1
LOAD RF Keyboard	从键盘获取数据,存到寄存器 F 中
STORE Number2 RF	把寄存器 F 中的数据存入 Number2
LOAD R0 Number1	把 Number1 中的内容存入寄存器 0
LOAD R1 Number2	把 Number2 中的内容存入寄存器 1

续表

代 码	说 明
ADD1 R2 R0 R1	把寄存器 0 和寄存器 1 中内容相加,结果存入寄存器 2
STORE Result R2	把寄存器 2 中的内容存入 Result
LOAD RF Result	把 Result 中的值存入寄存器 F
STORE Monitor RF	把寄存器 F 中的值输出到显示器
HALT	停止

由于汇编语言中使用了助记符,将用汇编语言编制的程序送入计算机后,计算机不能像对用机器语言编写的程序一样直接识别和执行,必须通过预先放入计算机的“汇编程序”的加工和翻译,汇编语言才能变成能够被计算机识别和处理的二进制代码程序。用汇编语言等非机器语言书写好的符号程序称为源程序,运行时汇编程序要将源程序翻译成目标程序。目标程序是机器语言程序,它一经被安置在内存的预定位置上,就能被计算机的 CPU 处理和执行。

汇编语言像机器指令一样,是硬件操作的控制信息,因而仍然是面向机器的语言,使用起来比较烦琐费时,通用性也差。汇编语言是低级语言。但是,汇编语言可以用来编制系统软件和过程控制软件,其目标程序占用内存空间少,运行速度快,有着高级语言不可替代的优势。

3. 高级语言

无论是汇编语言还是机器语言都是面向硬件的具体操作,语言对机器的过分依赖要求使用者必须对硬件结构及其工作原理都十分熟悉,这对于非计算机专业人员是难以做到的,对计算机的推广应用是不利的。计算机事业的发展,促使人们去寻求一些与人类自然语言相接近且能为计算机所接受的语意确定、规则明确、自然直观和通用易学的计算机语言。这种与自然语言相近并为计算机所接受和执行的计算机语言称为高级语言。高级语言是面向用户的语言。无论是何种机型的计算机,只要配备上相应的高级语言的编译或解释程序,则用该高级语言编写的程序就可以通用。常见的高级语言有 C、C++、Java、Python、JavaScript、PHP、Basic、C# 等。例如,C 语言中,实现两个整数相加的代码具体如下:

```
#include <stdio.h>
int main( )
{
    int a,b,c;
    scanf("%d,%d",&a,&b);
    c=a+b;
    printf("%d",c);
    return 0;
}
```

1.2 C 语言发展历史

C 语言的原型为 ALGOL 60 语言(也称为 A 语言)。

1963 年,剑桥大学将 ALGOL 60 语言发展成为 CPL 语言。

1967 年,剑桥大学的马丁·理查兹(Martin Richards)对 CPL 语言进行了简化,出现了 BCPL 语言。

1970 年,美国贝尔实验室的肯·汤普森(Ken Thompson)对 BCPL 进行了修改,并为它起了一个有趣的名字——“B 语言”。意思是将 CPL 语言“煮干”,提炼出它的精华,并且他用 B 语言写了第一个 UNIX 操作系统。

1972 年,美国贝尔实验室的丹尼斯·麦卡利斯泰尔·里奇(D. M. Ritchie)在 B 语言的基础上设计出了一种新的语言,他取了 BCPL 的第二字母作为这种语言的名字,这就是 C 语言。

1978 年由美国贝尔实验室正式发表了 C 语言。同时由布莱恩·W. 克尼汉(B. W. Kernighan)和丹尼斯·麦卡利斯泰尔·里奇合著了 *The C Programming Language* 一书,并在附录中提供了 C 语言参考手册,这本书成为以后广泛使用的 C 语言的基础,被人们称作非官方的 C 语言标准。

1983 年由美国国家标准协会(American National Standards Institute)在此基础上制定了一个 C 语言标准,通常称为 ANSI C。

1989 年,ANSI 发布了第一个完整的 C 语言标准——ANSI X3.159-1989,简称 C89,不过人们也习惯称其为 ANSI C。

1990 年,国际标准化组织 ISO(International Organization for Standards)接受了 89 ANSI C 为 ISO C 的标准(ISO 9899-1990)。

1994 年,ISO 修订了 C 语言的标准。

1995 年,ISO 对 C90 做了一些修订,即“1995 基准增补 1(ISO/IEC/9899/AMD1:1995)”。

1999 年,ISO 又对 C 语言标准进行修订,在基本保留原来 C 语言特征的基础上,针对新增的需要,增加了一些功能,命名为 ISO/IEC9899:1999。

2001 年和 2004 年先后进行了两次技术修正。

2011 年 12 月 8 日,ISO 正式公布 C 语言新的国际标准草案——ISO/IEC 9899:2011,即 C11。

2018 年 6 月,ISO 发布了 ISO/IEC9899:2018,简称 C18(或 C17)。C18 标准没有引入新的语言特性,只对 C11 进行了补充与修正。

2022 年 9 月 3 日,ISO 于 Open Standards(计算机标准开放组织)网站上发布了新的 C 语言标准定稿,称为 ISO/IEC 9899:2023,简称 C23。

目前流行的 C 语言编译系统大多是以 ANSI C 为基础进行开发的,但不同版本的 C 编译系统所实现的语言功能和语法规则又略有差别。

1.3 C 语言的特点

C 语言是作为描述系统的语言而设计的,但随着其日益广泛的应用,特别是 20 世纪 80 年代以后各种计算机 C 语言的普及,它已经成为众多程序员最喜爱的语言,它的使用几乎覆盖了计算机的所有领域,包括操作系统、编译程序、数据库管理程序、CAD、过程控制、图形图像处理等。

随着人工智能技术的快速发展,C 语言在这一领域展现了其独特的价值。虽然 Python 等高级语言在人工智能开发中更为常见,但 C 语言凭借其高效的性能和底层控制能力,在人工智能算法的底层实现、嵌入式系统开发以及高性能计算中仍然发挥着重要作用。例如,许多深度学习框架的核心模块和性能关键部分都使用 C 语言进行优化,以提升计算效率。此外,C 语言在机器人控制、自动驾驶系统以及边缘计算等人工智能相关领域中也得到了广泛应用。因此,C 语言不仅在传统领域占据重要地位,在人工智能这一前沿技术领域同样具有不可替代的优势。

C 语言之所以如此成功是有原因的。

(1) C 语言简洁、紧凑,使用方便、灵活。ANSI C 一共只有 32 个关键字,如下所示。

auto	break	case	char	const	continue	default
do	double	else	enum	extern	float	for
goto	if	int	long	register	return	short
signed	static	sizeof	struct	switch	typedef	union
unsigned	void	volatile	while			

(2) 运算符丰富。共有 34 种。C 语言把括号、赋值、逗号等都作为运算符处理。从而使 C 语言的运算类型极为丰富,可以实现其他高级语言难以实现的运算。

(3) 数据结构类型丰富。C 语言的数据类型有整型、实型、字符型、数组类型、指针类型、结构体类型、共用体(联合)类型等,能用来实现复杂的数据结构(链表、栈、队列、树、图)的运算。

(4) 具有结构化的控制语句。9 种控制语句可以实现结构化的程序设计,C 程序由若干程序文件组成,一个程序文件由若干函数构成。用函数作为程序的模块,便于按模块化的方式组织程序,层次清晰,易于调试和维护。

(5) 语法限制不太严格,程序设计自由度大。一般的高级语言语法检查比较严,能检查出几乎所有的语法错误,而 C 语言允许程序员有较大的自由度,因此放宽了语法检查。

(6) C 语言允许直接访问物理地址,能进行位(bit)操作,能实现汇编语言的大部分功能,可以直接对硬件进行操作。因此有人把它称为中级语言。

C 语言允许直接访问物理地址的特性,在人工智能领域具有重要的应用价值。这一特性使得 C 语言能够直接与硬件交互,从而在需要高效资源管理和实时性能的场景中发挥关键作用。例如,在嵌入式人工智能和边缘计算中,C 语言可以直接操作硬件资源,优

化内存和计算单元的使用,确保 AI 算法在资源受限的设备(如智能传感器、自动驾驶系统)上高效运行。此外,C 语言通过直接访问物理地址,能够更好地支持硬件加速器(如 GPU、FPGA)的底层编程,从而加速深度学习模型的训练和推理过程。在实时性要求较高的应用(如机器人控制、计算机视觉)中,C 语言的这一特性确保了算法能够快速响应并处理数据。因此,C 语言直接访问物理地址的能力,使其在人工智能的底层实现、性能优化和硬件交互中扮演了不可替代的角色。

(7) 生成目标代码质量高,程序执行效率高,可移植性好。C 语言在不同机器上的编译程序,86%的代码是公共的,所以 C 语言的编译程序便于移植。在一个环境中用 C 语言编写的程序,不改动或稍加改动就可移植到另一个完全不同的环境中运行。

C 语言的高效性使其成为深度学习框架底层优化的首选语言,许多主流框架(如 TensorFlow、PyTorch)的核心计算模块使用 C 语言编写,以实现高效的矩阵运算和并行计算,从而加速神经网络的训练和推理过程。这种高效性不仅体现在计算速度上,还体现在对硬件资源的精细控制上,使得 C 语言能够充分发挥现代计算设备的性能潜力。此外,C 语言的可移植性使得基于其开发的 AI 算法能够轻松移植到不同硬件平台和操作系统中,这为人工智能技术的广泛应用提供了极大的灵活性。特别是在嵌入式人工智能和边缘计算领域,C 语言编写的轻量级 AI 模型能够在资源受限的设备(如智能传感器、自动驾驶系统)上高效运行,满足实时性和低功耗的需求。

1.4 第一个 C 语言程序

【例】 在屏幕上显示如下内容:

欢迎你走进 C 语言的世界!

```
#include <stdio.h>                //编译预处理命令
int main()                        //主函数
{
    printf("欢迎你走进 C 语言的世界!\n"); //输出语句
    return 0;
}
```



第一个 C 语言程序

运行结果:

欢迎你走进C语言的世界!

那么,怎样来实现这个程序呢?

下面先演示如何使用 Microsoft Visual C++ 6.0 实现。

(1) 首先启动 Microsoft Visual C++ 6.0(简称 VC 6.0),如图 1.1 所示。

(2) 选择“新建”→“工程”→Win32 Console Application 命令,设置工程的名字和位置,单击“确定”按钮,如图 1.2 所示。

(3) 选择“An empty project.”单选按钮,然后单击“完成”按钮,如图 1.3 所示。

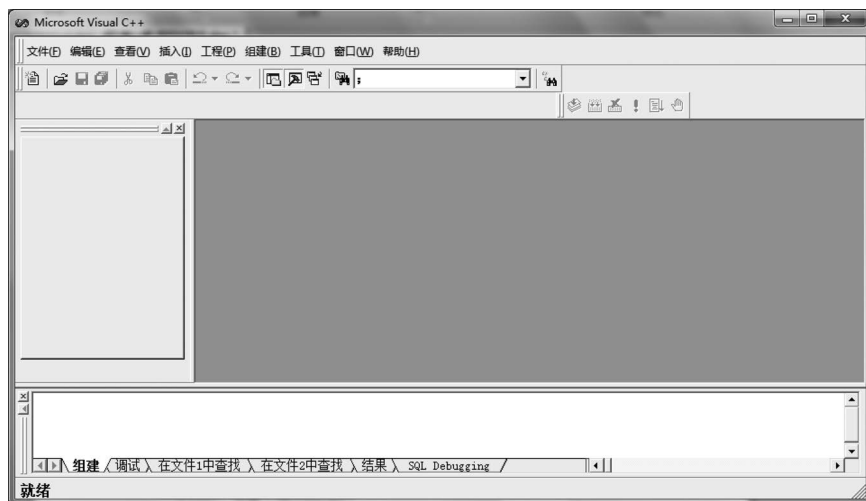


图 1.1 启动 VC 6.0



图 1.2 新建工程

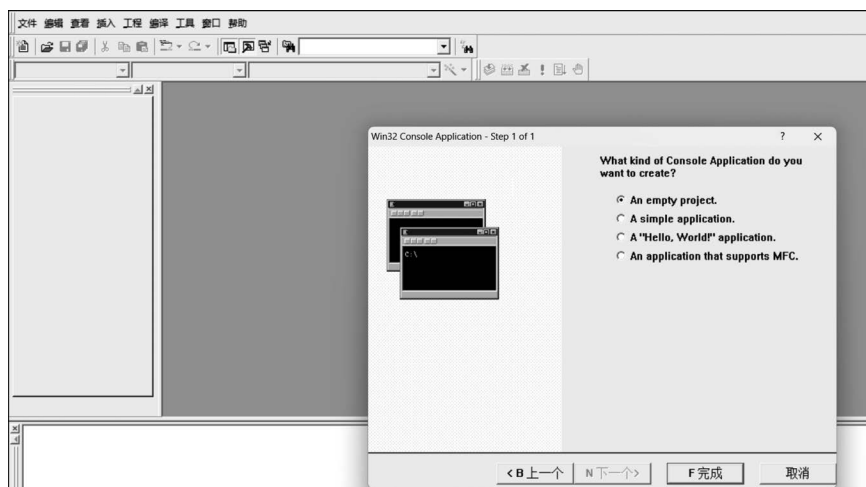


图 1.3 选择“An empty project.”单选按钮

(4) 选择“新建”→“文件”→C++ Source File 命令,将文件命名为 1_1,如图 1.4 所示。

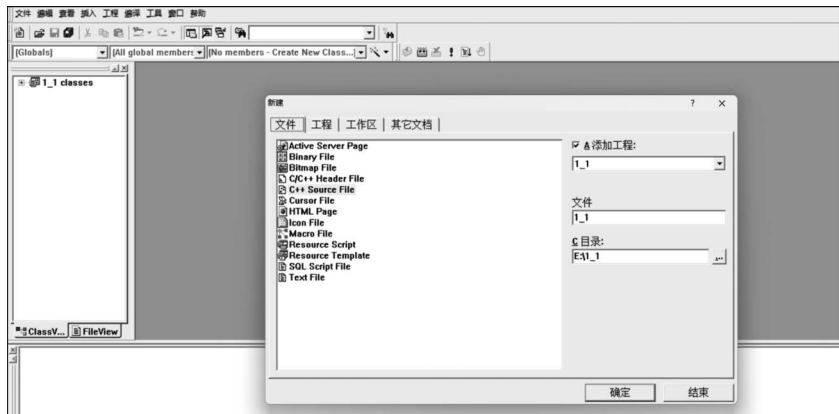


图 1.4 新建源文件

(5) 在新建文件里输入源程序,如图 1.5 所示。

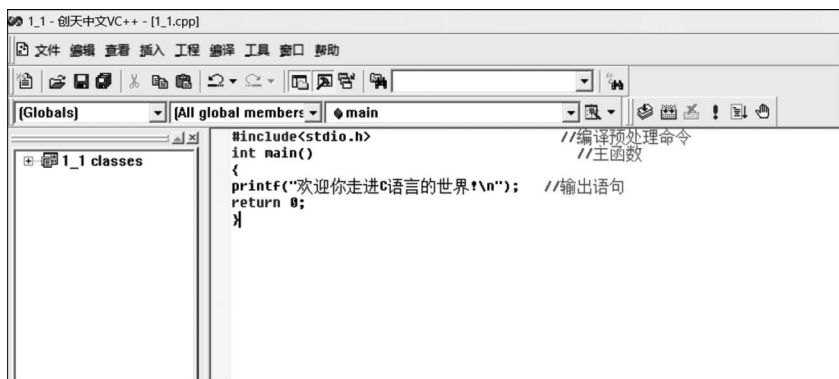


图 1.5 输入源程序

(6) 编译源程序,生成目标文件 1_1.obj,如图 1.6 所示。

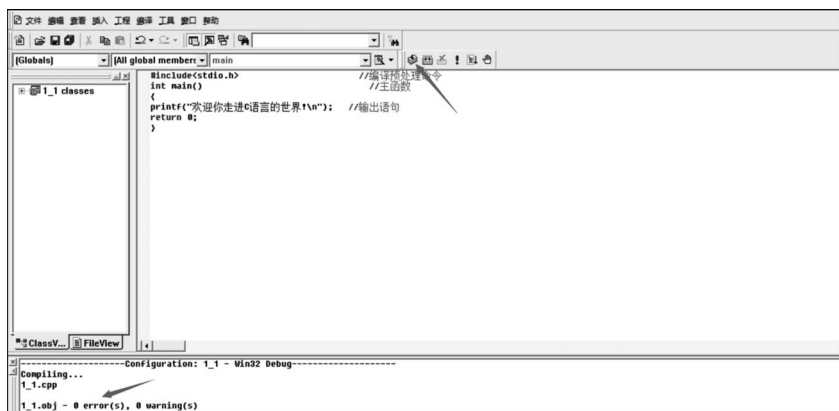


图 1.6 编译源程序

(7) 生成 exe 文件,如图 1.7 所示。

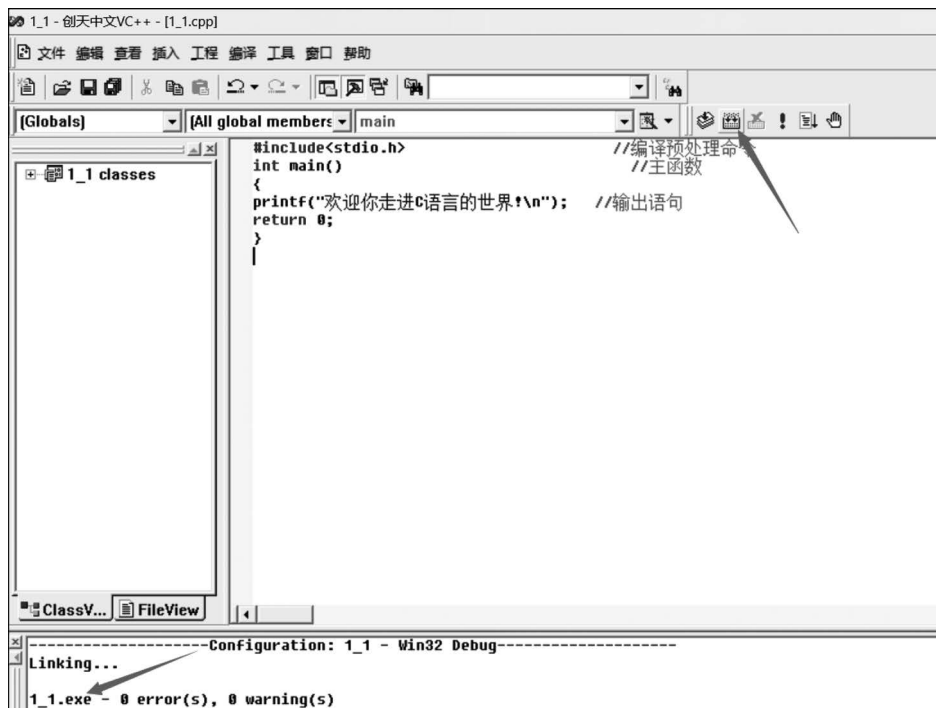


图 1.7 生成 exe 文件

(8) 运行程序,查看运行结果,如图 1.8、图 1.9 所示。

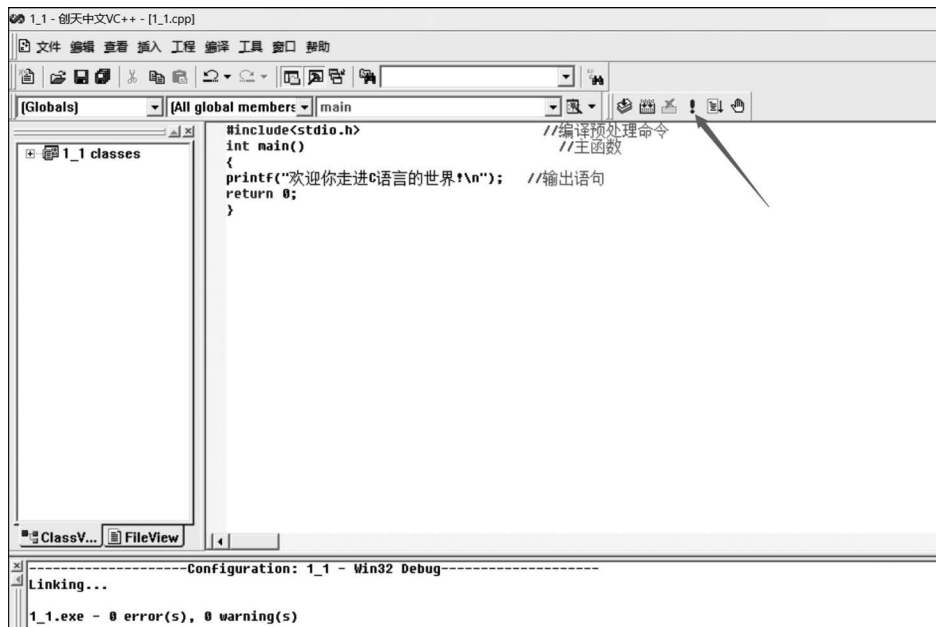


图 1.8 运行程序



图 1.9 程序运行结果

目前,很多学生使用 Dev C++来学习 C 语言,实现过程如下。

(1) 创建 C 语言源代码文件。打开 Dev C++,在上方菜单栏中选择“文件”→“新建”→“源代码”命令,如图 1.10 所示。

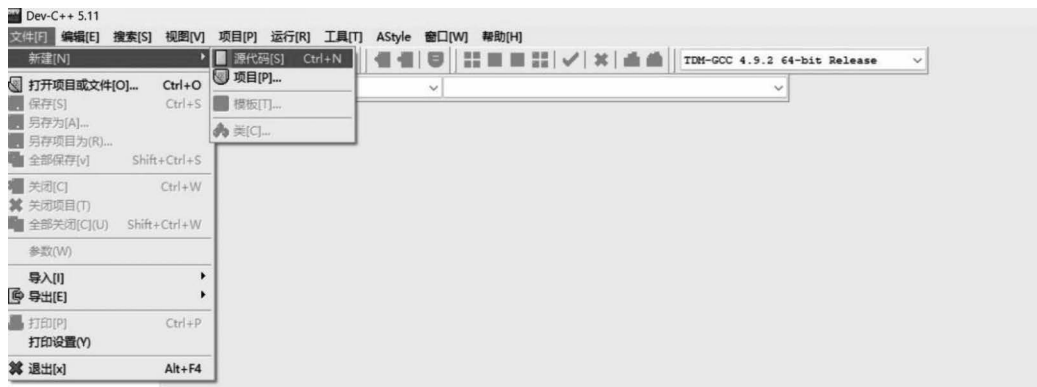


图 1.10 创建源代码文件

(2) 输入一段代码。用键盘输入以下代码,如图 1.11 所示。

```
#include <stdio.h>
int main()
{
    printf("hello, world");
    return 0;
}
```



使用 Dev C++ 来
调试 C 语言程序